AQA A-Level Computer science NEA project:

School register system for lower income countries

By Majdi Jaigirdar
Centre number: 34679
Candidate Number: 0164

Contents

Research and analysis

Introduction

What is a school register system?

A school register system is an essential tool utilised by educational institutions across the United Kingdom and indeed, the world. Fundamentally, it is a system used for tracking student attendance, necessary not only for internal administration but also for governmental reporting.

This system traditionally was done on paper; however, with the advent of technology, most school register systems now operate in digital formats. These electronic register systems offer benefits unsurpassed by their paper counterparts, such as streamlined efficiency, accessibility, and reliability of data.

School register systems frequently capture more than just daily attendance levels. They may include information on student tardiness, levels of unauthorised absence, and reasons for non-attendance. This data becomes a vital tool in monitoring patterns of absenteeism, enabling educators to address issues before they escalate to concerning heights. It can help in identifying students who may be facing academic challenges, health issues, or even socio-economic difficulties, allowing the school to provide timely support.

The information captured by the school register system is also essential to meet statutory reporting requirements. In the UK, schools are obliged to report attendance levels to the Department for Education. This data forms part of national statistics on school attendance, which helps in identifying trends, challenges, and inequalities in the country's education system.

However, the application of school register systems is not without its challenges. Managing such systems can be time-consuming for teachers. It can take away precious instructional time at the beginning of lessons. Thus, ensuring that register systems are user-friendly and efficient is critical for them to be effectively incorporated into everyday school practice without hindering teaching and learning time.

A school register system serves as a valuable asset within the education environment. It is a comprehensive tool, capturing critical data that goes beyond simple attendance, offering insight into student wellbeing and school performance. As technology continues to evolve, it is anticipated that these systems will further enhance their capabilities, becoming an even more integral part of school management and education policy planning in the UK and beyond.

How does a typical school register system work?

A typical school register system consists of multiple parts and systems that come together to create a whole solution. It operates through a streamlined set of processes. These processes are often automated through the use of digital technology to increase the system's overall efficiency.

At its core, a school register system begins with the fundamental task of student enrolment. Enrolment data can be manually entered or imported from a school's management system. This data usually includes the student's name, unique student ID, year group, parental/guardian contacts and other necessary details. It forms the students' master list that the system will use to track attendance.

Most digital systems work through a client based system on the teacher’s computer, tablets or other device. New modern systems use a web-based interface, allowing it to be accessed from any device with an internet connection. The functions that a teacher needs, such as marking a student present or absent, noting lateness, or specifying reason for absences are typically just a click away. In real-time, each student's attendance status is recorded.

These systems are usually designed to generate daily, weekly or monthly attendance reports as required. The data can be sorted by student, class, grade level or other criteria, providing the school with useful insights into overall attendance patterns. In addition, systems can often be configured to send automatic notifications to parents or guardians in case of unreported absences.

In the background, the attendance data collected by the register system can be linked to the school's management and reporting systems. This integration allows for seamless reporting on student progress and achievements, behaviour issues, and can even be used to identify potential learning difficulties or academic concerns.

Proper security measures are vital to a school register system. They ensure that student data is securely stored and accessed only by authorised individuals. Encryption and secure protocols are used to protect sensitive data. Besides, the system's administrative features usually include controls for managing user access, ensuring that data is visible only to those who need it.

The technical overview the general aspects of the system & on-site and off-site computing

As mentioned earlier, a typical school register system consists of multiple parts and systems that come together to create a whole solution. These parts are:

- **Client:** The client side usually consists of a local program, or more recently, a web interface accessible through browsers on computers, tablets, or smartphones. This is where users interact with the system, such as teachers taking attendance or students checking their schedules.

- **Server:** The server is the central hub that processes requests from the client side. It runs server-side scripts to handle the logic of the application, manage user sessions, and communicate with the database. Servers are typically powerful computers that can handle multiple requests simultaneously.

- **Database:** The database is where all the data is stored. It could be a relational database management system (RDBMS) like MySQL, PostgreSQL, or a NoSQL database like MongoDB, depending on the system's requirements. The database stores student records, attendance data, staff information, and more.

- **Application Layer:** This includes the application servers and the code that processes the logic for the application. It's written in programming languages like PHP, Java, or Python and frameworks like Laravel, Spring, or Django.

- **Data Layer:** This layer is responsible for storing and retrieving data. It includes the database servers and the actual databases.

- **Network:** The client and server communicate over a network, which could be the internet or a local intranet. Security measures like firewalls and encryption are used to protect data as it travels across the network.

- **Middleware:** Middleware is software that lies between the operating system and the applications on each side of a distributed computing system in a network. It enables communication and data management for distributed applications.

Here are some exemplar diagrams representing this structure and system:

Figure 1: Example scenario of school register system 1

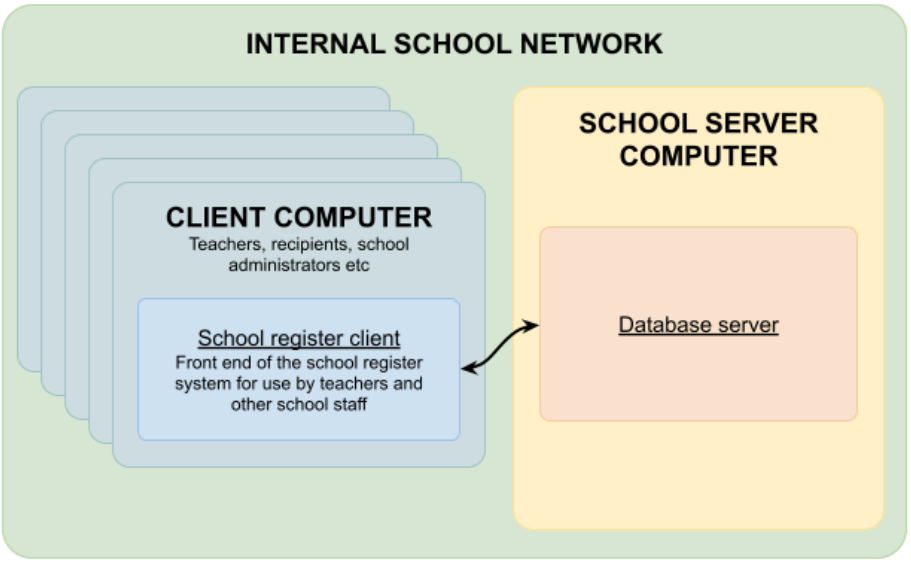
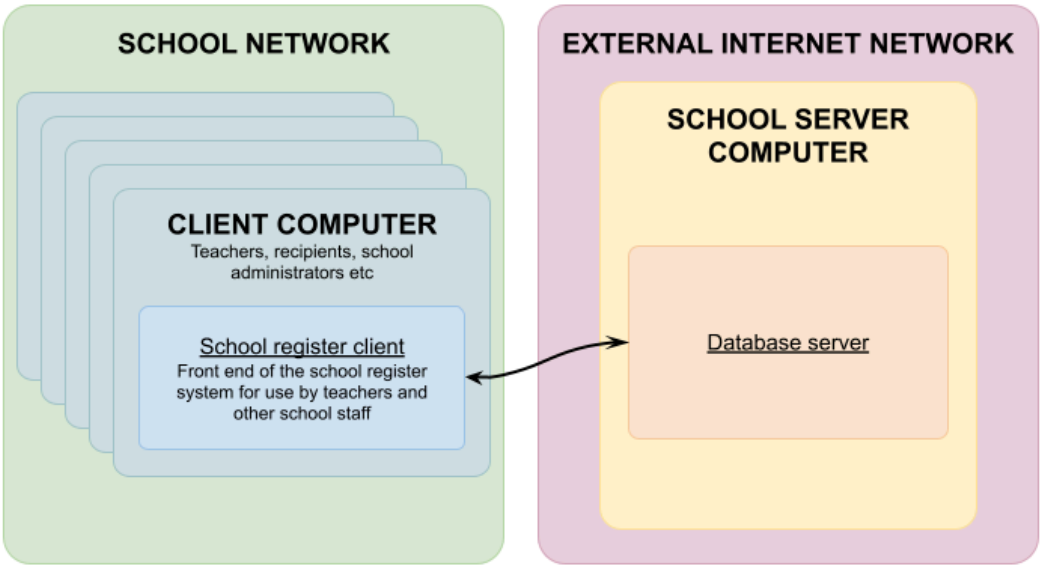


Figure 2: Example scenario of school register system 2



These two diagrams represent how a typical school register system works on a system/computer level. With multiple clients connecting to a main server hosting the school database. The clients and server can be on the same network on premises, or external to the network and off premises. Educational institutions use different setups based on their needs and current technologies implemented in their schools.

Previously, 5-15 years ago, the majority of institutions would have used an onsite local network setup for their school systems, which would contain the vast amount of the schools digital needs on premises on a central computer managed by IT staff of that institution. Email servers, web servers, students database, staff records, class materials, and other digital resources and data would all be hosted on that server on a central computer in site.

However, more recently, many schools and broadly other businesses and organisations are opting for a fully offsite server approach, where the central main computer is not within their school, but rather 'in the *cloud*' (The *cloud* refers to remote servers, typical in server farms, accessed over the internet to store, manage, and process data, rather than using a local server or a personal computer. It allows users to access files and applications from any device with an internet connection). This movement towards the cloud and away from onsite computing has been a gradual move, starting with the movement of web servers and email servers to the cloud, then movement to cloud storage, staff management tools and finally school register systems.

There's many reasons for this transition to the cloud, these being…

- **Expansion of Cloud Infrastructure:**
  The proliferation of cloud storage and the exponential growth of server farms have led to a reduction in costs, courtesy of major cloud service providers like Microsoft, Google, and Amazon. This downward price trend has not only been catalysed by economies of scale but has also spurred competition within the industry. Consequently, the affordability of cloud solutions has engendered a burgeoning ecosystem of sub-companies specialising in tailor-made services for schools and businesses, leveraging the infrastructure of major cloud platforms.

- **Empowerment through Client-Side Advancements:**
  Despite the increasing abilities of client-side computing, which may seem counterintuitive to the adoption of cloud-based solutions, its significance lies in optimising resource allocation. The evolving landscape of powerful yet cost-effective client-side hardware mitigates the necessity for extensive onsite server infrastructure and networking equipment. Consequently, the bulk of computational tasks that require more powerful systems can be offloaded to external servers, minimising data transmission overheads and making it cost effective to use the cloud.

- **Cost-Effectiveness of Cloud Infrastructure:**
  The pivotal driver behind the affordability of cloud computing stems from economies of scale. Rather than individual schools bearing the burden of setting up, maintaining, and upgrading costly server infrastructure, large-scale cloud providers specialise in these operations, offering computing access to diverse clients. For schools, the prospect of managing dedicated servers and network infrastructure on site entails exorbitant costs and resource allocations. Thus, leveraging cloud computing presents a financially better alternative.

- **Flexibility Offered by Cloud Infrastructure:**
  Beyond cost considerations, the adoption of cloud infrastructure affords unparalleled flexibility to schools and organisations. The scalability inherent in cloud-based solutions enables seamless expansion or contraction of computing resources in sync with evolving operational requirements. Moreover, the agility to access data and applications from any internet-enabled device empowers stakeholders with unparalleled mobility and productivity.

This migration in the direction of cloud computing is still continuing, and in my own observations I've seen a continuous casting off on local central servers in my schools. In my previous secondary school, Christ The King, in the first years my school was on a nearly all local computing structure. Everything was on site except email which used Google Workspace. They relied on local storage for class resources and files but then switched to Google Drive. They used to use a locally hosted website but then also switched to a google provided website solution. They used to use a locally hosted, application school register system then switched to a cloud based web solution in later years.

This switch to cloud infrastructure isn't necessarily bad or good, schools will use tools that are more affordable and flexible for their needs, as long as they use the tools that are useful for them, it's a positive. The switch to the web/cloud school register system was beneficial to my school as it allowed teachers to take attendance more easily for students, for example, when teachers were outside their classrooms without a computer, they could use an iPad and web browser to take attendance while in assembly in hall or in the middle of the field for PE. However, this new push to cloud infrastructure is not accessible to all schools and neither are local systems for some…

Lack of accessibility to technology for some schools

It is alarming to note that not all educational institutions worldwide have the technology or the resources to effectively utilise this critical tool. Despite the crucial role that a comprehensive school register system plays in improving school management and supporting student success, numerous factors hinder its widespread implementation, predominantly being the shortage of technology or appropriate technology.

The problem the education sector faces due to the lack of technology is far-reaching and multifaceted. Regions with limited access to technology often face challenges in maintaining a robust, efficient, and reliable school register system. Where modern, digitised register systems are not accessible, institutions typically revert to traditional, paper-based systems. While these are capable of collecting vital attendance data, they are inefficient, vulnerable to errors, and loss. There is also the added burden of additional time and physical resources spent on tracking, collating, and reporting attendance.

Additionally, regions with intermittent or unreliable access to electricity and internet services further compound the technology access issue. Without a consistent and reliable power supply or internet service, even schools with access to digital register systems may experience challenges in their effective operation. This scenario creates room for inaccuracies and inconsistencies in data collection and management.

A lack of technological literacy amidst teachers and administrators represent another barrier. It is one thing to have access to the technology required for a digital school register system, but it is another entirely to have the requisite skills to use such a system effectively. In many cases, teachers and administrators may not have the training needed to use these systems properly, leading to a significant barrier to efficient technology use.

In addressing these issues, it becomes crystal clear that developing an all-encompassing strategy is no simple task. Renewed focus needs to be placed on strengthening infrastructure in deprived regions, ensuring stable and reliable access to electricity and the internet. Governments, non-governmental organisations, and private sector technology providers all have a role to play in this.

Simultaneously, there needs to be a strong emphasis on technological literacy. Providing teachers and administrators with the necessary training to use school register systems effectively is as important as the systems themselves. Offering ongoing support is equally important to ensure that new software updates or problems do not hinder the continuous and efficient use of these systems.

Furthermore, there is a need for more affordable versions of digital register systems. Cost is often a prohibiting factor for many schools, particularly in under-resourced regions. By developing cost-effective, user-friendly systems, technology providers can play a crucial role in making digital school registers more accessible.

However, it is pertinent to remember that the benefits of a reliable register system extend beyond efficient administration. Insights from accurate attendance tracking can support targeted interventions, boosting academic achievement and reducing school dropout rates.

Access to effective school register systems is an ongoing issue worldwide. It demands concerted efforts from key stakeholders, including governments, educators, technology providers, and the broader community.

Focus on a specific region; Bangladesh

Across the globe, disparities in educational technology have been highlighted by this current era of advanced digital facilities and the pressing need for digital tools in education. This issue is especially pronounced in lower-income countries such as Bangladesh, where access to education technology is significantly limited, hindering the efficient delivery of education and impairing students' future prospects.

The advent of information and communication technologies (ICTs) has undoubtedly revolutionised the nature of education, offering digital tools and resources that can enhance teaching and learning processes. However, introducing these modern pedagogical tools in a developing country like Bangladesh, where a significant portion of the population lives in rural areas and below the poverty line, presents numerous challenges.

The foundational issue to start with in countries like Bangladesh is a lack of basic infrastructural amenities. A significant proportion of schools in Bangladesh, especially in rural areas, lack consistent access to electricity, a prerequisite for most educational technology. Moreover, the internet, a fundamental service for digital pedagogies, and e-learning platforms, is inconsistent, expensive or entirely absent in many regions. Schools simply cannot afford to provide students with reliable internet access, leaving them detached from the global digital learning community.

Further compounding these issues are limited financial resources. Schools in lower-income countries often operate on limited budgets with some even struggling to cover the basics, such as textbooks and teachers' salaries, leaving little or no room for investments in technology. Consequently, these schools are unable to afford digital devices, e-learning platforms, and technological infrastructures that schools in higher-income countries now consider standard, such as a school register system.

Additionally, issues of technological literacy amongst teachers and students present further roadblocks. Many educators in Bangladesh and similar lower-income countries do not possess the necessary digital skills to effectively utilise ICTs as teaching tools or to guide students in their use. Students, too, often enter school with varying levels of digital literacy, which can significantly impact their engagement and success with technology-based initiatives, if any.

However, the narrative is not entirely grim. Despite challenges, Bangladesh has demonstrated significant resolve to incorporate technology in education. This commitment can be seen through significant public and private sector investments in education technology, national ICT policies aimed at fostering digital literacy, and a proliferation of mobile learning initiatives. However, much work remains to ensure these initiatives reach all students in Bangladesh.

Despite the vast digital divide in Bangladesh, it's important to put into context that many teachers and schools, particularly in urban regions, do have computers. Unfortunately, these computers are often outdated, slow, and incapable of supporting contemporary software, thereby mitigating their

academic utility. These technological hurdles inevitably impact learning outcomes and present a significant obstacle to the incorporation of digital education tools, including digital register systems.

One procedural aspect these outdated computers compromise is the process of record keeping, from maintaining student records to taking attendance. Many schools in Bangladesh still use traditional pen-and-paper techniques for taking attendance, which, although simple, pose a significant risk for error due to human mistakes and issues, such as lost or damaged data.

Furthermore, relying solely on a manual system poses difficulties in tracking long-term attendance trends or identifying patterns, which could support early intervention strategies for students at risk. The data from attendance can shed light on a variety of school realities, such as student engagement, potential dropouts, academic performance, and even factors affecting mental health. Without a digital register system, analysing this data becomes time-consuming and cumbersome, if not impossible, which may forestall the timely application of necessary interventions.

A digital register system provides a solution to these issues, offering a streamlined approach to attendance tracking, data management, and analysis. With a digital system, data is stored securely, can be accessed easily by authorised personnel, and is not susceptible to physical damage.

However, the inadequate technological infrastructure of many schools in Bangladesh poses considerable challenges to the implementation of such systems. Slow computers cannot support modern software required for digital registers. Moreover, lack of server resources in schools prevents comprehensive networking required for dynamic data sharing and access.

Equally problematic is unreliable or completely lacking Internet access, precluding any cloud-based solutions from being feasible. Cloud-based attendance systems provide an expansive range of functionalities and benefits, including real-time data access, secure data backup, and easy collaboration among institution stakeholders. However, without the required Internet availability, these benefits remain frustratingly out of reach.

In sum, while the need and potential benefits of implementing a digital register system in schools across Bangladesh are clear, its realisation is substantially hampered by existing technological limitations. Improving the country's technological infrastructure is a necessary first step towards digital transformation. Furthermore, refurbishing the existing computer systems in schools, providing secure and adequate server resources, and ensuring reliable Internet access should be quick wins. In the context of Bangladesh, a long journey still awaits, but the destination ensures a brighter, more digitally inclusive future for the nation's education.

My school register system project

My project is to make a school register system that is accessible and easy to use for all education institutions, with a primary focus on lower income countries, *such as Bangladesh,* but also more generally for all schools where access to an online based school system or powerful central server isn't financially feasible or possible with current infrastructure. My solution will provide a low cost, efficient and intuitive solution to digitally track students attendance, class records, staff and other related school data without the user having to have advanced technology literacy or an extensive setup . This project will be done as a program for my cousin Jeni, who’s a primary school teacher

in Bangladesh in the city of Sylhet. She's a Bengali Teacher but is also incharge of student attendances at her primary school.

Discussing with Jeni on what she needs from a school register system

To understand what a good school register system would be for someone like Jeni, I conducted an unstructured informal interview over the phone. Speaking in Bengali, I translated our conversation and extracted the main dialogue and points.

Start of conversation

Me

Hi Jeni, I'm making a digital school register system to track, manage and see students attendance, manage classes, teachers and other school related data. It's made for schools who have lightweight computers and non-powerful central computers or advanced networking capabilities. As a school teacher and being in charge of students attendance, what would you want out of a school register system like this?

Jeni

Hi Majdi, that sounds like a fantastic project! I'm excited about the prospect of a digital school register system that can cater to the needs of schools like ours in Bangladesh. As a primary school teacher and someone responsible for tracking student attendance, here are a few key features that I think would be extremely beneficial in a school register system:

1. **Ease of Use:** One of the most important aspects for me would be the system's user-friendliness. It should be intuitive and easy to navigate, even for someone with limited technical skills. It should not require extensive training to be able to operate and manage.
2. **Attendance Tracking:** The system should have a feature that allows me to easily mark and track student attendance. It would be great if this could be done in real-time.
3. **Class Management:** It would be helpful to have a class management feature where I can schedule classes, assign tasks, and manage classroom activities efficiently.
4. **Data Security:** I would like the system to ensure the security and privacy of student data. It should have robust security measures in place to protect sensitive information.
5. **Offline Capability:** Given the unreliable internet access in many parts of Bangladesh, having the ability to use the system offline and then sync data locally would be a huge plus.
6. **Customization:** The system should be flexible and allow for customization based on the specific needs of our school. This could include features like generating reports, sending notifications to parents, or integrating with other school systems.
7. **Affordability:** Since budget constraints are a significant issue for many

schools in Bangladesh, a cost-effective solution that provides value for money would be highly appreciated.

I believe that a school register system with these features would greatly help in streamlining attendance tracking, data management, and overall school operations.

Me

What would be the best way to save these student data for you?

Jeni

Well, I have a basic understanding of data storage, and for me, it would be essential to have a system that is open and accessible by other programs for future developments. I would prefer a system that allows for easy integration with other educational tools and software, enhancing its functionality and usability.

In terms of keeping student data secure, it is paramount that the system incorporates robust security measures to protect sensitive information. Encryption methods and access controls would be vital in ensuring the privacy and confidentiality of student data.

Considering the limited resources and infrastructure challenges we face, it would be ideal to use a data storage solution that is not resource-intensive. Opting for a lightweight database management system that can efficiently handle the data without overwhelming the system's capabilities would be preferable.

However, we'd like to avoid cloud storage. It's a great solution to store data and backup, however with limited internet access it poses issues here.

Overall, a balance between data security, accessibility, resource efficiency, and future scalability would be crucial considerations in determining the best way to save student data for our school register system.

Me

When you say "sending notifications to parents", can you explain further what you mean?

Jeni

Sending notifications to parents is a crucial aspect of communication between the school and the students' families. In the context of student absences, this feature would involve the system automatically sending SMS text messages to the contacts of the absent students to inform them of the absence.

This notification system serves multiple important purposes. Firstly, it keeps parents informed about their child's attendance and any missed lessons. Consistent absences can negatively impact a student's academic performance, so notifying parents allows them to address any potential issues early on.

Moreover, in terms of safety, if a student is absent and the school is unaware of their whereabouts, it could raise concerns about their well-being. Sending notifications to parents ensures that they are aware of their child's absence and can take appropriate action if needed.

The notifications can also include information on whether the absence is authorised by the school and, if known, the reason for the absence. This transparency helps parents understand the circumstances surrounding their child's absence and enables them to support their child in maintaining good attendance.

Overall, integrating a notification system into the school register system that communicates student absences to parents via SMS text messages plays a vital role in promoting academic success, ensuring student safety, and fostering effective communication between the school and families.

Me

What's your current system of tracking students, if any at all?

Jeni

In our school, like many others in Bangladesh, we primarily rely on a manual system of tracking student attendance. This traditional method involves using pen and paper to take attendance, record class information, and manage other school-related data. While this system has been the standard practice, it comes with various limitations and challenges.

The manual pen-and-paper system can be quite time-consuming and labour-intensive, especially when it comes to compiling attendance data or generating reports. Additionally, there is a risk of errors such as misplaced or damaged records, which can compromise the integrity and accuracy of the data, and jeopardise security of student data.

In our school, most teachers have laptops, albeit low-powered ones, which they use for various tasks. However, stable internet access for the school is not guaranteed, which poses a significant obstacle to implementing online-based solutions. Given these circumstances, a digital school register system designed to run on low-end hardware and not be dependent on a powerful central computer would be highly beneficial.

I believe that transitioning from the manual pen-and-paper system to a digital solution like the one you are developing would greatly streamline our data management processes, enhance accuracy, and improve overall efficiency. I am excited about the potential of a more reliable and effective system that can meet our school's specific needs and technological constraints. Thank you for considering these aspects in your project.

Me

Why would you say an improved school register method would be beneficial for your school, other schools in Bangladesh and in general, the world?

Jeni

Improving the school register method would bring significant benefits not only to our school but also to other educational institutions in Bangladesh and beyond. Here are some reasons why an enhanced school register system would be highly advantageous:

For our School:

1. **Efficient Data Management:** Transitioning to a digital school register system would streamline attendance tracking, class management, and overall data management processes in our school. This would reduce paperwork, minimise errors, and save time for teachers like me.

2. **Enhanced Accountability:** A digital school register system would increase transparency and accountability in monitoring student attendance and academic performance. It would provide accurate and real-time data for school administrators, teachers, and parents to assess student progress.

For Schools in Bangladesh:

1. **Government Initiatives:** In Bangladesh, the government is actively promoting education by providing free primary education to all students up to the 8th grade. An improved school register system would help the government track attendance rates and ensure that all eligible students are enrolled and attending school.

2. **Empowering Girls' Education:** The government's efforts to provide free secondary education to rural girls in Bangladesh aim to empower them to pursue higher education and delay marriage. A digital register system would help monitor attendance, identify trends, and measure the impact of these initiatives on girls' education.

On a Global Scale:

1. **Universal Education Goals:** Education is a fundamental human right and a key driver of sustainable development. Improving education systems, including attendance tracking, is crucial to achieving the United Nations' Sustainable Development Goal of ensuring inclusive and equitable quality education for all.

2. **Data-Driven Decision-Making:** In an increasingly interconnected world, data on student attendance and educational outcomes are vital for informed decision-making at the national and global levels. A digital school register system can provide valuable insights and support evidence-based policy-making in education.

Overall, an enhanced school register system not only benefits individual schools like ours by improving efficiency and accountability but also contributes to larger educational goals in Bangladesh and worldwide. By promoting access to education, monitoring attendance, and empowering students, such a system plays a vital role in advancing educational opportunities and promoting social progress.

Me

Thank you Jeni for doing this interview. I'll make some system design mock ups and show you the project as I go along to get continuous feedback.

Jeni

You're very welcome, Majdi! I appreciate the opportunity to provide input on your project, and I look forward to seeing the system design mock-ups as you progress. Please feel free to reach out whenever you need further input or feedback. I'm excited to see how the school register system project evolves, and I'm here to support you along the way. Good luck with your project, and thank you for

considering the needs of schools like ours!

End of conversation

Conversing with Jeni as my client for this project proved very useful, she clearly explained what she needs from a school register system, why they're beneficial for schools, teachers, parents and students and the restraints that exist for these current systems to be implemented into schools of LICs (Lower income countries). She explained the current methods of tracking school attendance and how a new digital method, specially designed, would benefit her school, other schools in Bangladesh, and the globe. From this I'm able to draw up goals and objectives of what my project should consist of...

Project goals and objectives (MoSCoW)

This section of my project will outline and explain what my school register system should achieve and explain which objectives are must haves, should haves, could have, and won't have. This is the MoSCoW method; it's a prioritisation technique used in project management, particularly in software development, to categorise requirements based on their importance.

The acronym "MoSCoW" stands for:

- **Must Have:** Requirements that are critical for the project's success and must be implemented in the current phase.
- **Should Have:** Important requirements that are not critical for the project's success in the current phase but should be included if possible.
- **Could Have:** Requirements that are desirable but not critical, and their inclusion would be nice to have if resources allow.
- **Won't Have (or Would Have):** Requirements that are considered out of scope for the current project phase and won't be addressed.

Each objective / goal will be listed 1 - 4 corresponding to their level of importance by the MoSCoW method. The following objectives have been revisited multiple times throughout the project, are dynamic and subject to change as my project progresses. Flexibility and adaptability are essential in responding to new insights, feedback, challenges, and learning experiences that arise during the project. This ensures that my approach remains responsive to evolving circumstances and goals.

An evaluation and review of these objectives will be in the evaluation section of this document. The labelling of these objectives using MoSCoW / 1 - 4 doesn't represent if these goals have been achieved in the final solution, rather goals I believe the project/programm should or should not have started this project.

The total objectives are broken down into multiple parts. The number indicates the main overall objective, while the alphabetical list being the sub objectives of that main objective which come together to make the aims of this project.

1 - Must Have 2 - Should have 3 - Could have 4 - Won't have

Priority	Objectives
1	1. Use a database system appropriate for my client and software.
1	a. Integrate a system that can easily be accessed by my school register system client software
1	b. Use a database system that can be accessed by other programs and built up on by schools independently to make maintenance and data processing easier
1	c. A system that is lightweight on lower end hardware and can be setup on most various systems with ease
1	d. Use a system that can be externally managed outside my own program so schools have more control and flexibility with their data
1	e. Secure system that will protect sensitive student, teachers and other school related records

2 2	f. A solution that will allow for multiple concurrent users to access the database. g. Create a backup system of that database to secure backup data from the database locally and on the cloud.
1 1 2 1 3	2. Create a lightweight client program for the school register system for staff a. Make the client be able to run on lower end, older devices on various platforms and various environments b. A client side computer that doesn't relay on a powerful competitive central computer to do must of the processing c. A secure system that requires users to enter the program with a log in d. A secure logon with multiple accounts giving access / privileges to different users
3 3 3 3 3 3 3	3. Create a lightweight client portal for students / guardians use a. Create a secure login system for students/guardians to access the system b. Allow students/guardians to view their own student details c. Show students/guardians their classes d. Show students/guardians their schedules for their classes e. Allow students/guardians to create requests for authorised absence with authorisation for school and guardians f. Allow students / guardians edit their respective records
1 1 1 1 1 3	4. General ability of a functioning school register system a. Be able to mark a student present / absent for a specific class b. Be able to see the statistics of a student's presence / absences for a class c. Be able to see the statistics of a specific student across multiple classes d. Be able to see attendance statistics of students for a singular class e. Remind teachers to take attendance with notifications
1 1 2 1 1 1 1	5. Student records a. Create records of students: name, address, DOB, etc b. Creation of multiple parental / emergency / guardian contacts for students storing address, contact, names etc c. View all student records in the system d. Edit student records e. Delete parental / emergency / guardian contacts for student f. Delete student records and associated data
1 1 1 1 1	6. Teacher records a. Create records of teachers: Name: address, DOB, etc b. View all teacher records c. Edit a teachers record d. Delete a teacher record and associated data
1 1 1 1 1 2 1	7. Class records a. Create a class in the system b. Assign a teacher to it c. Enrol students into that class d. Schedule times for that class e. Prevent schedule overlaps f. Delete schedules for that class

1	g. Delete class and associated data
2 2 2 2 2 3 2	8. Exam records a. Keep track of students exams for various classes and subjects b. Create exam records c. View exam records d. Show overview of exam results e. Use advanced algorithm to estimate students final grade/results f. Delete exam records
1 2 3 3 3 2 3 3 3	9. Create a user interface for system a. Make UI that is intuitive and easy to understand for users b. Don't rely on too many visual effects / transitions as program is intended for lower powered hardware c. The UI should provide instructions, tips and hints on how to use it for the user d. Provide user with possible inputs and suggestions e. Provide user with information on input format f. The UI should not be cultured and should have a primary focus on the main subject g. Use a universal UI language to keep it all consistent h. Show statistical information a clean and understandable format
3 2 2 2 2 2 2 3 4	10. Create simple/advanced algorithms and use AI/machine learning to predict student stats a. Use simple/advanced algorithms to find out the day a student misses the most b. Use simple/advanced algorithms to find out the times a student misses the most c. Use simple/advanced algorithms to find out the classes a student misses the most d. Analyse the general attendance of the whole school e. Analyse the general attendance of the whole class f. Analyse the general attendance of the whole year/grade g. Using advanced algorithm and/or machine learning, and using existing attendance/exam data in the system, estimate a students final grade by comparing their exam results and attendance to other students previous them h. Using advanced algorithm and/or machine learning, suggest potential escalating attendance issues that might occur with a student by comparing their attendance patterns to that of other students who are having poor attendance
1 1 2 1 3 3 2	11. Notify parental / emergency / guardian contacts of their child's absences a. Send SMS messages to contacts when their child misses a lesson b. Make the SMS messaging simple and cheap c. Make the transport of data between devices secure and private d. Inform contact of their students' absence, if it's been authorised and if a reason for absence has been given. e. Make SMS sending process simple f. Keep the functionality lightweight and non-resource intensive
3 3	12. Create algorithm to intelligently figure out the best school time schedule a. The program will look at the current free time slots and figure out the best lesson schedules

3 3	b. It will use advanced algorithm to find free slots to put lessons in c. It will provide the user with multiple options of school timetables/schedules
1 1 1 3 1 2	13. Maintain student, teacher and other personal privacy, security & safety a. Require logon access b. Use a secure database method c. Hide user password input d. Prevent SQL injection with parameters e. Validate inputs to ensure security error prevention
3 2 3 3 1 2	14. Use of object oriented programing (OOP) a. When appropriate use OOP b. Create custom OOP using VB classes for functionality c. Use OOP rather than repeating code for following best practice, maintenance and decrease in storage d. Create meaningful and appropriate methods e. Use custom data classes when appropriate
1 2 2 2 2 2 1	15. Code in a uniform manner in which allows other 3rd parties to understand my code, program, solution and data processing a. Use appropriate and understandable variable names b. Use appropriate and understandable module, function & subroutine names c. Use appropriate and understandable class and method names d. Comment throughout the program explaining what code does and the current expected process e. Comments should be in a uniform / standardised manner f. Appropriate uses of custom classes

Jeni's thoughts on the project objectives so far

Start of conversation

Me

Hi Jeni, how do you feel about the current project objectives?

Jeni

The project objectives outlined for the development of the school register system are comprehensive and highly relevant to the needs of schools like ours in Bangladesh. Here's how I feel about the proposed objectives:

1. Database System and Integration: Using an appropriate database system that can easily integrate with the client software is crucial for efficient data management and accessibility.

2. Lightweight System: Building a system that is lightweight and can run on lower-end hardware is essential for schools with limited resources and older devices, ensuring wider accessibility.

3. Secure System and User Management: Prioritising data security and user authentication is paramount to protect sensitive school-related records and

ensure the system's integrity.

4. Student and Teacher Records Management: Creating and managing student, teacher, class, and exam records efficiently will streamline administrative tasks and enhance data accuracy.

5. User Interface Design: Developing an intuitive and user-friendly interface with clear instructions and minimal visual effects will make the system accessible and easy to navigate for all users.

6. Advanced Algorithms and Predictive Analysis: Implementing algorithms and AI/machine learning for predictive analysis can provide valuable insights into student attendance patterns and academic performance.

7. Parental Notification System: Enabling automated SMS notifications to parents/guardians about student absences will improve communication and ensure student safety and well-being.

8. School Time Schedule Optimization: Using advanced algorithms to intelligently optimise school timetables based on available time slots will enhance operational efficiency and classroom organisation.

Overall, the project objectives align well with the specific needs and challenges faced by many schools, including our own. I am excited about the potential impact this school register system could have in improving data management, communication, and overall educational outcomes for both students and teachers.

End of conversation

Design and technical solution of school register system

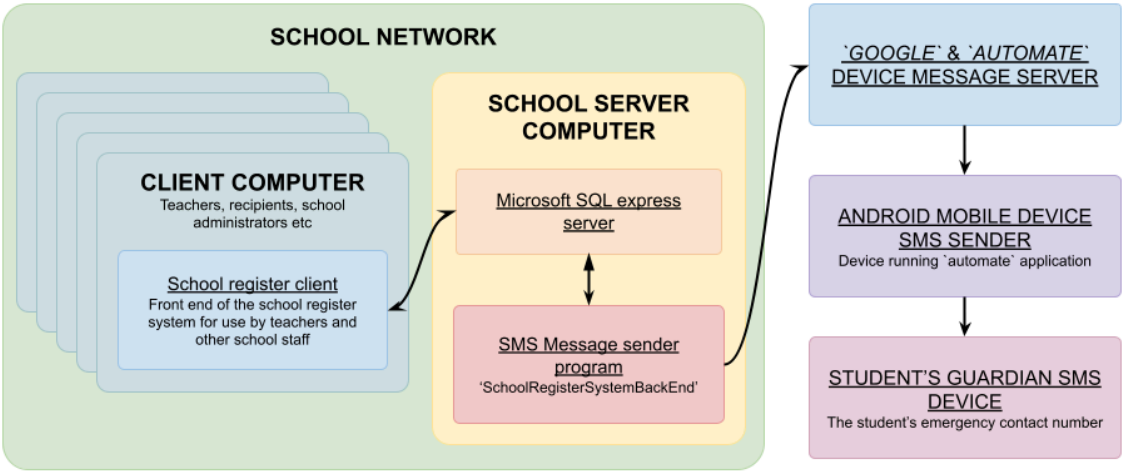
This section will outline the way the system will be designed and how its intended functions will work. The physical and application structure, the flow of data and the design of code, etc.

General system design

My general system design will replicate that of most school register systems. Consisting of multiple parts; The main parts are client computers running the school register client program on them, and then the school server computer running the database on it.

Here's a brief diagram:

Figure 3: Setup of my school register system



(Arrows represent flow of data and connectivity. Lines with one arrowhead showcase the flow of data in one direction. Lines with two arrowheads showcase the flow of data in both direction)

Programing language: Visual basic / VB.net

For my project, I'll be using Visual Basic as the primary programming language. This choice is primarily influenced by my familiarity with Visual Basic due to my studies and experience in using it in class. Additionally, Visual Basic provides a user-friendly and intuitive development environment that aligns well with the objective of creating an easy-to-use school register system.

Another reason for choosing Visual Basic is its cross-platform compatibility, allowing the system to run seamlessly on various operating systems such as Windows, Mac, and Linux. This cross-platform capability ensures that the school register system can be accessed and utilised by a wide range of users, regardless of their preferred operating system.

Moreover, creating the school register system as a console application in Visual Basic offers a lightweight and efficient solution that can run smoothly on lower-end hardware, meeting the project's objective of providing a system that is accessible and functional on a variety of devices.

By leveraging the capabilities of Visual Basic, I aim to develop a robust and user-friendly school register system that addresses the specific needs and challenges faced by many schools, ultimately contributing to improved data management, communication, and educational outcomes.

Database design

Microsoft SQL server express database

Microsoft SQL Server Express is a free version of Microsoft's SQL Server relational database management system (RDBMS). It's designed for small-scale deployments such as schools and for small-scale applications such as my register system, providing a powerful and reliable data storage solution without the cost associated with the full version of SQL Server.

SQL Server Express comes with certain limitations compared to the full version, such as limits on database size, memory usage, and the number of processors utilised, making it suitable for smaller projects or for learning purposes. However, it still offers many of the core features and functionality of the full SQL Server, including support for SQL queries, stored procedures, triggers, and more.

However the limitations of express won't be a hindrance to my project, I'm making a relatively small and basic system that will be used by a small number of teachers per school. The product being lightweight and free is a benefit to my case as it's made for my client who has lower powered hardware and can't afford to pay high amounts for software.

Overall, Microsoft SQL Server Express is a popular choice for developers, small businesses, and educational institutions looking for a robust and cost-effective database solution, and so, in my scenario it'll be fantastic for the job.

Database table and their fields (normalisation of data)

Address				
Field name	**Description**	**Data type**	**Example data**	**Validation**
ID	Address ID number - Used to reference the address in other tables and for user selection	Integer	12	Auto increment from 1 by 1
HouseNumberName	The house number or house name of a person's address	String	32	Can't be blank and Must be a string (or string number)
StreetName	The street address of a person's address	String	Scarisbrick New road	Can't be blank and Must be a string
LocalityName	The locality of a person's address	String	Meols Cop	Optional but Must be a string
CityTownName	City or town name of a person's address	String	Southport	Can't be blank and Must be a string
Postcode	Postcode of a person's address	String	PR86LR	Can't be blank and Must be a string
CountryName	Country location of a person's address	String	United kingdom	Optional but Must be a string - Default "United

				kingdom”

Student				
Field name	**Description**	**Data type**	**Example data**	**Validation**
ID	Student ID number - Used to reference the student in other tables and for user selection	Integer	5	Auto increment from 1 by 1
FirstName	The student's first name	String	John	Can’t be blank and Must be a string
MIddleName	Student’s middle name - optional	String	Daniels	Optional
LastName	Student’s last name	String	Smith	Can’t be blank and Must be a string
DateOfBirth	Student’s date of birth	Date	13/12/2005	Must contain and be a valid date format (DD/MM/YYYY)
AddressID	Related Student’s address ID row	Integer	12	Must be an ID that already exists in Address table - Is auto inserted by program

EmergencyGurdianContact				
Field name	**Description**	**Data type**	**Example data**	**Validation**
ID	EmergencyGurdianContact ID number - Not referenced in other tables but is used by user in the program for user selection	Integer	13	Auto increment from 1 by 1
StudentID	The student of which the parent / guardian / emergency contact is for. Referenced by the studentID	Integer	5	Must be an ID that already exists in Student table - Is auto inserted by program
Title	The parent / guardian / emergency contact title	String	Mrs	Can’t be blank and Must be a string
FirstName	The parent / guardian / emergency contact first name	String	Eva	Can’t be blank and Must be a string
MIddleName	parent / guardian / emergency contact middle name - optional	String	Rosie	Optional
LastName	parent / guardian / emergency contact last name	String	Smith	Can’t be blank and Must be a string

StudentRelationship	The relationship to the student in focus	String	Mother	Can't be blank and Must be a string
PhoneNumber	The mobile/telephone number of the student's parent / guardian / emergency contact	string	0754095214	Can't be blank and Must be a string (phone numbers should be treated like strings)
AddressID	Related parent / guardian / emergency contact address ID row	Integer	13	Must be an ID that already exists in Address table - Is auto inserted by program

Teacher				
Field name	**Description**	**Data type**	**Example data**	**Validation**
ID	Teacher ID number - Used to reference the teacher in other tables and for user selection	Integer	5	Auto increment from 1 by 1
Title	Teacher's title	String	Mr	Can't be blank and Must be a string
FirstName	The Teacher's first name	String	Michael	Can't be blank and Must be a string
MIddleName	Teacher's middle name - optional	String	Young	Optional
LastName	Teacher's last name	String	Davies	Can't be blank and Must be a string
DateOfBirth	Teacher's date of birth	Date	1/07/1985	Must contain and be a valid date format (DD/MM/YYYY)
AddressID	Related Teacher's address ID row	Integer	12	Must be an ID that already exists in Address table - Is auto inserted by program

ClassCourcse				
Field name	**Description**	**Data type**	**Example data**	**Validation**
ID	ClassCourcse ID number - Used to reference the ClassCourcse / class in other tables and for user selection	Integer	6	Auto increment from 1 by 1
Classname	The name of the class	String	Computer science	Can't be blank and Must be a string

			CS01	
FirstName	The Teacher's first name	String	Michael	Can't be blank and Must be a string
MIddleName	Teacher's middle name - optional	String	Young	Optional
LastName	Teacher's last name	String	Davies	Can't be blank and Must be a string
DateOfBirth	Teacher's date of birth	Date	1/07/1985	Must contain and be a valid date format (DD/MM/YYYY)
AddressID	Related Teacher's address ID row	Integer	12	Must be an ID that already exists in Address table - Is auto inserted by program

ClassEnrollments				
Field name	**Description**	**Data type**	**Example data**	**Validation**
ID	ClassEnrollments ID number	Integer	9	Auto increment from 1 by 1
StudentID	The ID of the student who is being enrolled to a class	Integer	5	Must be an ID that already exists in Student table - Is auto inserted by program
ClassCourcseID	The ID of the class where the student is going to enrolled to	Integer	6	Must be an ID that already exists in ClassCourcse table - Is auto inserted by program

ClassSchedule				
Field name	**Description**	**Data type**	**Example data**	**Validation**
ID	ClassSchedule ID number	Integer	20	Auto increment from 1 by 1
ClassCourcseID	The ID of the class of which to make a lesson schedule for	Integer	6	Must be an ID that already exists in ClassCourcse table - Is auto inserted by program
DayOfWeek	The day of the week for the scheduled lesson, saved as an	Integer	2 *(Tuesday)*	Must not be empty and must be an integer

	integer 1 being Monday to 7 being sunday.			ranging from 1-7. Program will auto convert day to integer.
StartTime	The the class lesson start time of the scheduled lesson	Time	14:00	Must contain and be a valid time format (HH:MM) / (HH:MM:SS)
EndTime	The the class lesson end time of the scheduled lesson	Time	15:00	Must contain and be a valid time format (HH:MM) / (HH:MM:SS)

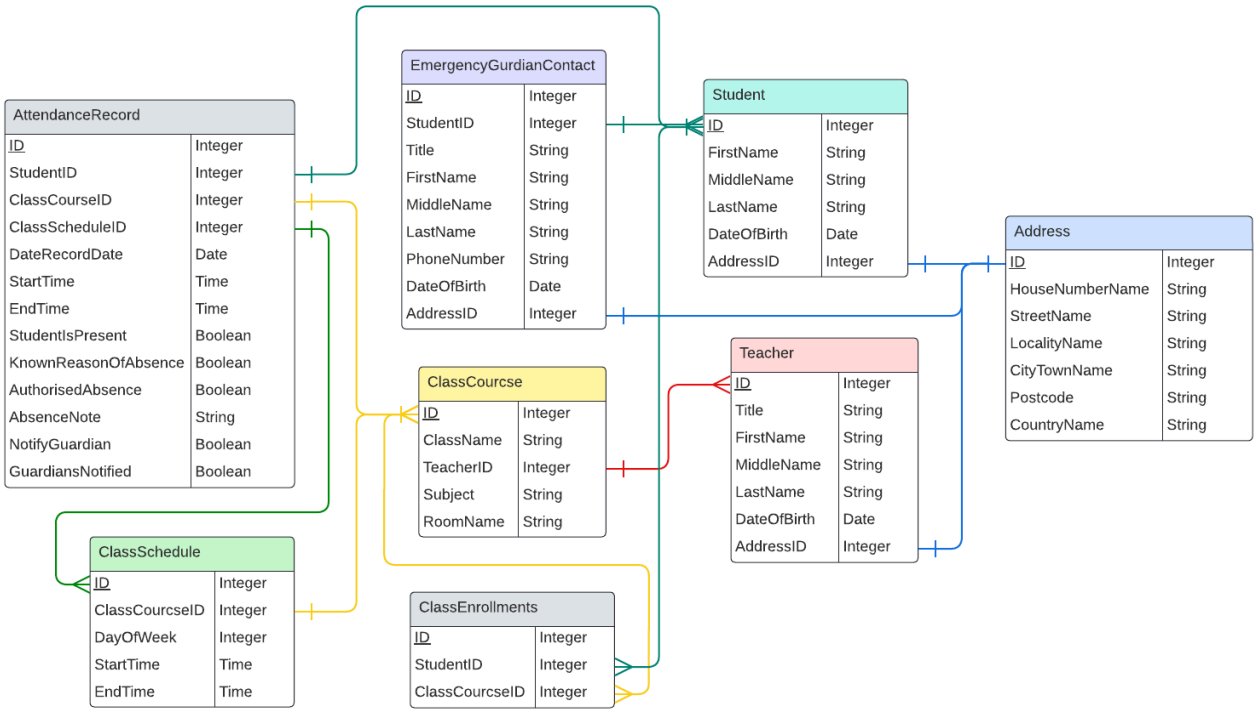
AttendanceRecord				
Field name	**Description**	**Data type**	**Example data**	**Validation**
ID	AttendanceRecord ID number - Not referenced in other tables but is used by program to show student absences	Integer	18	Auto increment from 1 by 1
StudentID	The ID of the student who is absent/present for a lesson	Integer	5	Must be an ID that already exists in Student table - Is auto inserted by program
ClassCourseID	The the ClassCourseID / classID of the class the student has been marked absent/present for	Integer	6	Must be an ID that already exists in ClassCourse table - Is auto inserted by program
ClassScheduleID	The the ClassScheduleID of the scheduled lesson the student has been marked absent/present for	Integer	20	Must be an ID that already exists in ClassSchedule table - Is auto inserted by program
DateRecordDate	The date of which the student has been marked absent/present for	Date	13/11/2023	Must contain and be a valid date format (DD/MM/YYYY)
StartTime	The the class lesson start time of the scheduled lesson the student has been marked absent/present for	Time	14:00	Must contain and be a valid time format (HH:MM) / (HH:MM:SS)
EndTime	The the class lesson end time of the scheduled lesson the student has been marked absent/present for	Time	15:00	Must contain and be a valid time format (HH:MM) / (HH:MM:SS)
StudentIsPresent	The status of the student for attendance record: Is the student absent/present for a scheduled lesson	Boolean	False	Must not be blank and must be true (present) or false (absent)

KnownReasonOfAbsence	If the status of the student is absent, is the reason for the student's absence known	Boolean	False	Optional, but must be true (known reason for absence) or false (No known reason for absence)
AuthorisedAbsence	If the status of the student is absent, was the absence authorised by the school	Boolean	False	Optional, but must be true (authorised absence) or false (unauthorised absence)
AbsenceNote	If the status of the student is absent, provide an optional notes of their absence	String	No reason for absence given - missed several lessons	Optional, but must be a string
NotifyGuardian	If the status of the student is absent, should their guardian be notified of their absence	Boolean	True	Optional, but must be true (notify guardians) or false (do not notify guardians)
GuardiansNotified	Used by the back end program - has this record of the student's absence been notice to the guardians after attendance	Boolean	Flase	Andeled by back end - Optional, could be true or false.

Database table connections

Thee database will consist of multiple multi relationship tables, here's a diagram:

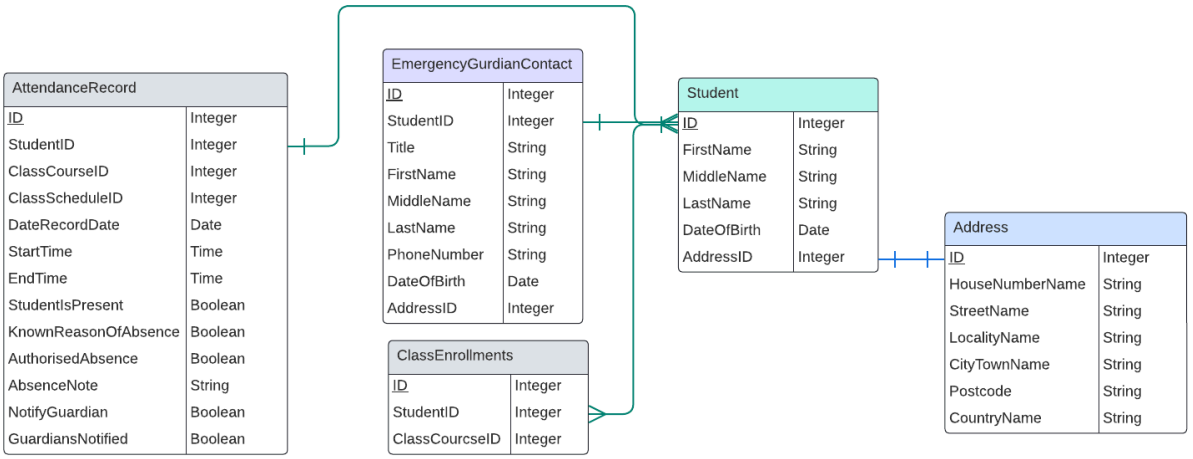
Figure 4: SQL multi relational database overview



This is a complex overview of the relationships between each table and their primary/foreign keys, to make it understandable I'll break it down into some small sections and focus on some tables then put the whole database together.

Student table

Figure 5: SQL multi relational database, student table



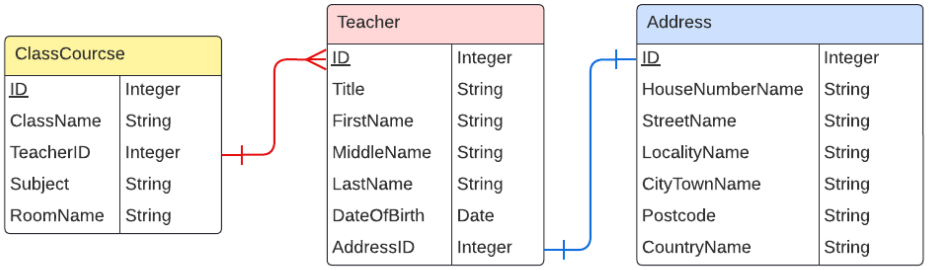
Here's the `Student` table (cyan colour). The student's record will hold information such as their First name, last name, date of birth and their address. Their address will be stored on a separate table called `Address` which will be referenced in the student table by its address ID. This is a one to one relationship. There will be one address for one student and vice versa.

The student table will have its own ID, this ID will be referenced by other tables, as shown in the diagram. The student is referenced in 3 other tabless, first in `EmergencyGurdianContact` table, second in `ClassEnrolments` and lastly in `AttendanceRecord`. All these are one to many relationships, where the StudentID is referenced in all 3 of them.

The `EmergencyGurdianContact` is a table that will store the guardian contacts of the student, such as their parents, grandparents, adult siblings, etc. It'll also store their phone number, this number will be used to send SMS notifications to when their student is absent by the backend server program.

Teacher table

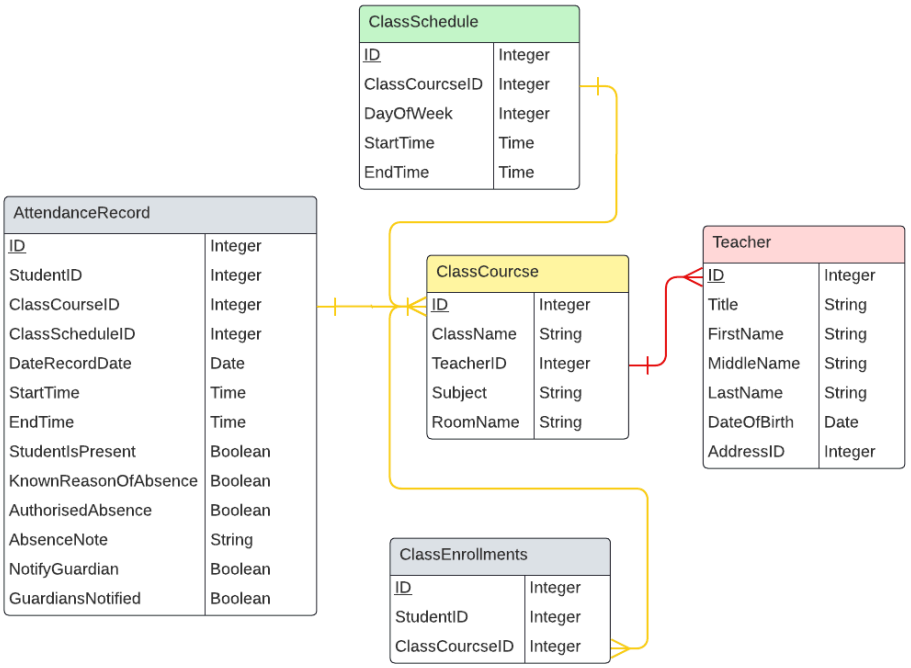
Figure 6: SQL multi relational database, teacher table



The `Teacher` table (red colour) is similar to the students table, additionally it contains the title field (Mr, Mrs, Miss, Dr, etc…). The address of the teacher is also in a different table, and is referenced by the AddressID. This separation of the Address and rest of the details is done as it allows all the addresses to be available in one area of the database. Like students, it's a one to one relationship as a teacher can have only one address and vice versa.

Class Course table

Figure 7: SQL multi relational database, class course table

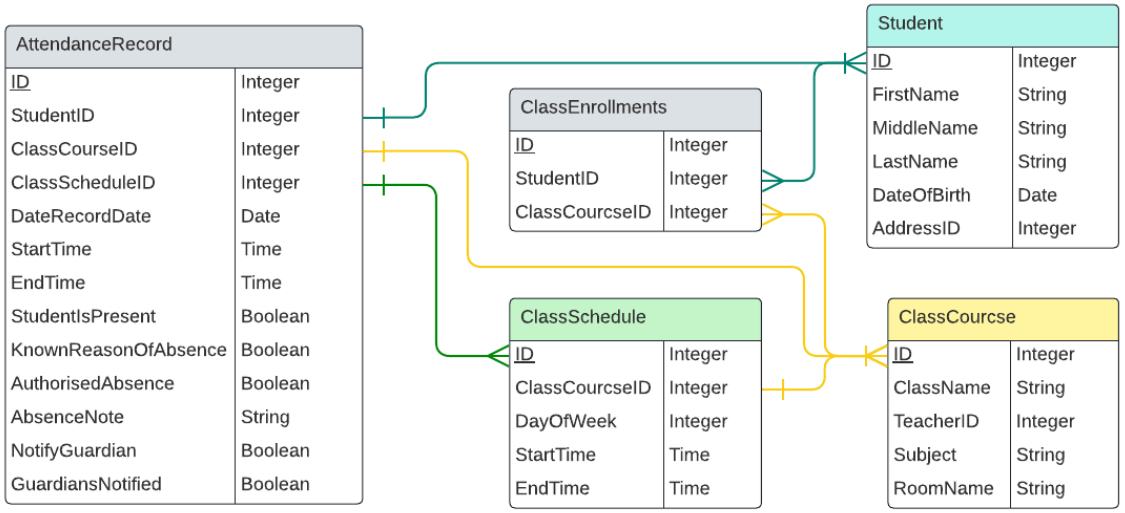


The `ClassCourse` table is the table that holds the information on who teaches the class, its subject, the room, etc. A class course can only have 1 teacher, but a teacher can be a teacher of many class courses. The ClassCourse ID is referenced by three other tables. One of these being the `AttendanceRecord`.

The attendance record table holds the record of each student attendance for each class at each scheduled session. It stores if the student was present/absent for a lesson as a boolean, it records the lesson time absent, and it holds the boolean question on if the student's guardian should be notified, if they have been notified yet, and the reason for absence.

Class schedule table & Class enrolments table

Figure 8: SQL multi relational database, Class schedule table & Class enrolments table



The `ClassSchedule` (green colour) table & `ClassEnrolments` (middle, grey colour) table hold information on when a class is and what students are in a class. ClassEnrolemnts will have the student's ID and the Class ID they're in. This is done as it's best practice not to have lists inside of other tables, meaning not having a list/array of students in the classCourse row, or have a list/array of class courses inside the students row. Insead, having a separate table with rows of students and classes to associate them together.

`ClassSchedule` consists of the lesson start time, end time and day of the week for that particular lesson schedule. Coded into the SQL server and the School register system program itself, it makes sure that there can only be one lesson at a time on that particular day of week.

In this school setup, we have a weekly based schedule, and we only have one schedule group, this means that multiple lessons can not happen at the same time. In real school environments with a large student population you'd have students be in multiple different classes at the same time, but in mine for simplicity of testing we have one schedule group. If i was to actually implement this in a school with a large population of students, like a college, I'd add another field called `ScheduleGroup` or `ScheduleBlock`that subdivides the students into groups for scheduling.

Together, all these tables come together to make the school register database.

Custom class definitions for school register system

Custom data classes are user-defined data types created in Visual Basic to represent specific entities or concepts within the system. These classes encapsulate both data (attributes) and behaviour (methods) related to the various components of the register system, such as students, teachers, class courses, and address. By using custom classes, I can organise and structure the data in a logical manner, making it easier to manage and manipulate my application and its behaviours. Each class serves as a blueprint for creating objects that mimic real-world entities, allowing me to efficiently model the complexity of the school register system and implement functionalities tailored specific requirements.

USE / DESCRIPTION:
Custom created data types used to process and handle data more effectively and easily throughout the program. These classes serve as blueprints for organising and managing various components of the school register system, such as students, teachers, courses, and attendance records.

NOTE:
The custom data types defined below mirror the tables in the SQL database. This approach facilitates a better understanding of how data is handled and processed on both ends. While some fields in these data type classes may not exist in the SQL database, and vice versa, aligning them allows for seamless integration and comprehension. For instance, foreign keys in SQL tables are represented by other class data types in the program, enhancing flexibility and usability. Additionally, arrays within these classes are utilised to represent multiple occurrences of similar data, which is not directly reflected in the SQL tables but aids in the program's functionality and clarity.

"(optional user input)" indicates that certain data is not mandatory for user entry and is considered additional information. By employing custom classes that reflect the structure of the SQL database, the program's operation, processing, and data understanding are significantly enhanced, ensuring a robust and user-friendly school register system.

Address (Line 10)

```
Public Class Address 'Stores address of students, parents, gurdians, teachers, etc (Is also a table in SQL database)
    Public Property ID As Integer
    Public Property HouseNumberName As String '(Note: Houses may be number but often can have name and not a number)
    Public Property StreetName As String
    Public Property LocalityName As String '(optional user input)
    Public Property CityTownName As String
    Public Property Postcode As String
    Public Property CountryName As String '(optional user input, defulat United Kingdom)
End Class
```

Purpose: Stores address information of students, parents, guardians, teachers, etc.
Properties:

- ID: Unique identifier for each address entry.
- HouseNumberName: String representing the house number or name.
- StreetName: String representing the name of the street.
- LocalityName: String representing the locality (optional).
- CityTownName: String representing the city or town.
- Postcode: String representing the postal code.
- CountryName: String representing the country name (optional, defaulting to United Kingdom).

Student (Line 32)

```
Public Class Student 'Student records (Is also a table in
SQL database)
    Public Property ID As Integer
    Public Property FirstName As String
    Public Property MiddleName As String '(optional user
input)
    Public Property LastName As String
    Public Property DateOfBirth As Date
    Public Property Address As Address '(Not in SQL
database, but AddressID is, and is saved as a foregin key
for `Address` table)'
    Public Property EmergencyGurdianContact() As
EmergencyGurdianContact() '(Not in SQL at all.
EmergencyGurdianContact refrers to StudentID, not student
to EmergencyGurdianContact. This use is just for the
program)'
End Class
```

Purpose: Represents student records.
Properties:

- ID: Unique identifier for each student.
- FirstName, MiddleName, LastName: Strings representing the student's name.
- DateOfBirth: Date representing the student's date of birth.
- Address: Object of type Address representing the student's address.
- EmergencyGuardianContact: Array of EmergencyGuardianContact objects representing emergency contacts associated with the student.

EmergencyGuardianContact (Line 20)

```
Public Class EmergencyGurdianContact 'Emergency, parental
and gurdian contacts of students (Is also a table in SQL
database)
    Public Property ID As Integer
    Public Property StudentID As Integer 'Foreign key in
SQL database as StudentID refrencing 'student' table
    Public Property Title As String
    Public Property FirstName As String
    Public Property MiddleName As String '(optional user
input)
    Public Property LastName As String
    Public Property StudentRelationship As String
    Public Property PhoneNumber As String '(Note: Saved as
a string as tearting it as an integer will cut leading
zeros)
    Public Property Address As Address '(Not in SQL
database, but AddressID is, and is saved as a foregin key
for `Address` table)
End Class
```

Purpose: Stores emergency, parental, and guardian contacts of students.
Properties:

- ID: Unique identifier for each emergency contact entry.
- StudentID: Foreign key referencing the student associated with this contact.
- Title, FirstName, MiddleName, LastName: Strings representing the contact's name and title.
- StudentRelationship: String describing the relationship with the student.
- PhoneNumber: String representing the contact's phone number.
- Address: Object of type Address representing the contact's address.

Teacher (Line 42)

```
Public Class Teacher 'Teacher records (Is also a table in
SQL database)
    Public Property ID As Integer
    Public Property Title As String
    Public Property FirstName As String
    Public Property MiddleName As String '(optional user
input)
    Public Property LastName As String
    Public Property DateOfBirth As Date
    Public Property Address As Address '(Not in SQL
database, but AddressID is, and is saved as a foregin key
for `Address` table)
End Class
```

Purpose: Represents teacher records.
Properties:

- ID: Unique identifier for each teacher.
- Title, FirstName, MiddleName, LastName: Strings representing the teacher's name and title.
- DateOfBirth: Date representing the teacher's date of birth.
- Address: Object of type Address representing the teacher's address.

ClassCourse (Line 52)

```
Public Class ClassCourcse 'Class record, each class (Is
also a table in SQL database)
    Public Property ID As Integer
    Public Property ClassName As String
    Public Property Teacher As Teacher '(Not in SQL
database, but TeacherID is, and is saved as a foregin key
for `teacher` table)
    Public Property Subject As String
    Public Property RoomName As String '(optional user
input)
End Class
```

Purpose: Represents class records, each corresponding to a specific course.
Properties:

- ID: Unique identifier for each class/course.
- ClassName: String representing the name of the class.
- Teacher: Object of type Teacher representing the teacher assigned to the class.
- Subject: String representing the subject of the class.
- RoomName: String representing the name of the room where the class takes place (optional).

ClassEnrollments (Line 60)

```
Public Class ClassEnrollments 'Record of which students
are in a class (Is also a table in SQL database)
    Public Property ID() As Integer()
    Public Property Student() As Student() '(Not in SQL
database, but StudentID is, and is saved as a foregin key
for `student` table)
    Public Property ClassCourcse As ClassCourcse '(Note:
Spelling mistake of 'course' has occured. However, program
and database is too far devloped for this error to be
rectofoed) (Not in SQL database, but ClassCourcseID is,
and is saved as a foregin key for `ClassCourcse` table)
End Class
```

Purpose: Records which students are enrolled in each class.
Properties:

- ID: Array of integers representing unique identifiers for each enrollment entry.
- Student: Array of Student objects representing the students enrolled in the class.
- ClassCourse: Object of type ClassCourse representing the class/course in which students are enrolled.

ClassSchedule (Line 66)

```
Public Class ClassSchedule 'Record of when a class is (Is
also a table in SQL database)
    Public Property ID As Integer
    Public Property ClassCourcse As ClassCourcse '(Note:
Spelling mistake of 'course' has occured. However, program
and database is too far devloped for this error to be
rectofoed) (Not in SQL database, but ClassCourcseID is,
and is saved as a foregin key for `ClassCourcse` table)
    Public Property DayOfWeek As String
    Public Property StartTime As TimeSpan
    Public Property EndTime As TimeSpan
End Class
```

Purpose: Records the schedule for each class.
Properties:

- ID: Unique identifier for each class schedule entry.
- ClassCourse: Object of type ClassCourse representing the class/course for which the schedule is defined.
- DayOfWeek: String representing the day of the week for the class.
- StartTime, EndTime: TimeSpan objects representing the start and end times of the class.

AttendanceRecord (Line 74)

```
Public Class AttendanceRecord 'Record of each students
attendence (Is also a table in SQL database)
    Public Property ID As Integer
    Public Property Student As Student
    Public Property ClassCourcse As ClassCourcse '(Note:
Spelling mistake of 'course' has occured. However, program
and database is too far devloped for this error to be
```

Purpose: Records the attendance of students in each class.
Properties:

- ID: Unique identifier for each attendance record.
- Student: Object of type Student representing the student whose attendance is recorded.
- ClassCourse: Object of type ClassCourse

rectofoed) (Not in SQL database, but ClassCourcseID is, and is saved as a foregin key for `ClassCourcse` table)
    Public Property ClassSchedule As ClassSchedule '(Not in SQL database, but ClassScheduleID is, and is saved as a foregin key for `ClassSchedule` table)
    Public Property DateRecordDate As Date
    Public Property StartTime As TimeSpan
    Public Property EndTime As TimeSpan
    Public Property StudentIsPresent As Boolean 'Present = true , absent = false
    Public Property KnownReasonOfAbsence As Boolean '(optional user input)
    Public Property AuthorisedAbsence As Boolean '(optional user input)
    Public Property AbsenceNote As String '(optional user input)
    Public Property NotifyGuardian As Boolean '(optional user input)
    Public Property GuardiansNotified As Boolean '(optional user input)
 	
End Class
	representing the class/course attended.
	• ClassSchedule: Object of type ClassSchedule representing the schedule for the attended class.
	• DateRecordDate: Date representing the date of the attendance record.
	• StartTime, EndTime: TimeSpan objects representing the start and end times of the attended class.
	• StudentIsPresent: Boolean indicating whether the student was present.
	• Additional properties for recording reasons for absence, authorization status, absence notes, and notification status.

Secure password inputting

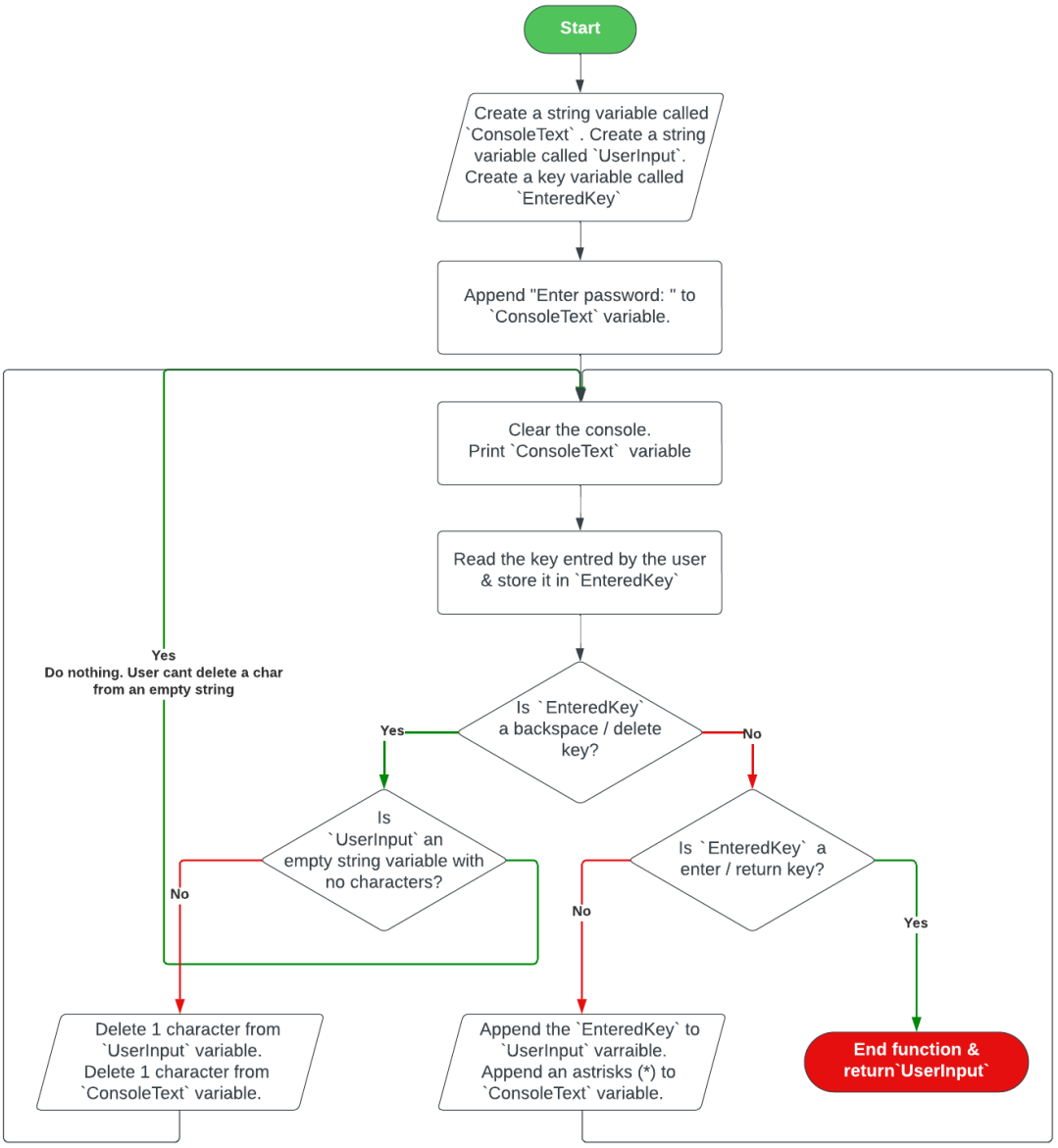
To make my program secure, and keep student / teacher / class data secure, I'll be requiring the user to enter a password to access the system. This will be done by:

- Requiring the user to know the password to proceed with the program
- Hiding the user entered password on the terminal so students and other onlookers cant see the entered password
- Removing any hints of what the password could be by hiding the password length

Issue:
Visual basic console apps do not have a built in method of retrieving passwords from the user securely. If I wanted to get a password input from a user they'd be required to type it in their password which would also show their password on the console and so exposing it, which would compromise the security of the system, and could put students in danger.

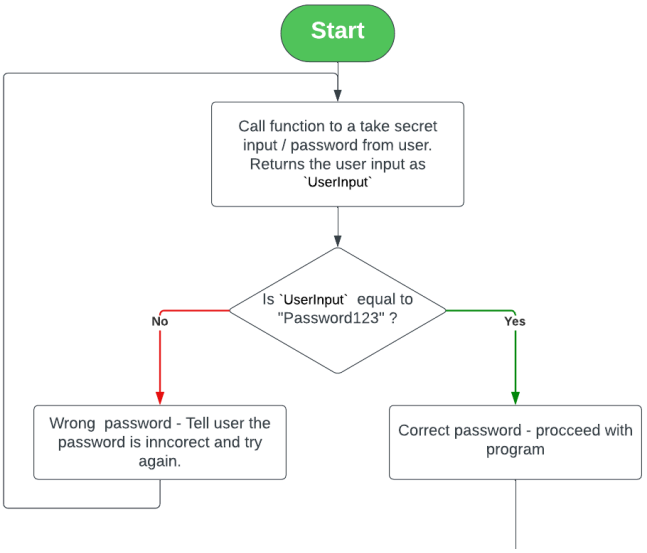
Solution:
I've created a solution to this issue. VB allows the console to be cleared, so removing all the text previously printed/entered onto the console, I can create a function / subroutine that will take the user input and then hide it on the terminal.

Algorithm for solution:
The main function:



This is the `SecretInput` function. This is responsible for taking the user password and hiding the password on the console. As the user enters the keys, if they're alphanumeric and not a delete or enter key, they'll be appended to the `UserInput` variable where then the program will wait and read the user's key again and append it continuously, it'll also print an asterisks on the console to represent the password and to conceal it. If the user enters a delete key, the program will check to see if any chars in the `UserInput`, if not, the program will do nothing, if there are, the program will delete a character from the `UserInput` and delete an asterisks from `ConsoleText`. If the user enters a enter key / return key, this is the end of the password, the function will end and return the `UserInput` variable.

User end / password validator:



This is a very simple algorithm that just checks if the returned password matches the correct password. If they match, the program proceeds, if they don't match, the program will ask the user to enter the password again.

The code to the solution:

The main function:

Visual basic (Line 420)

```
Module SecretInputHandlerModule
    ' Variables explained:
    '   PreviousConsoleText - The prompt/text to print to the terminal
    '   SecretCharacter - The character to use to hide the input
    Public Function Function_SecretInput(ByRef PreviousConsoleText As String, ByRef
SecretCharacter As Char) As String

        Dim ConsoleKey As ConsoleKeyInfo
        Dim input As String = "" 'Holds the user entred secret (passwords, etc) as each
character is inputed
        Dim PCT_WithoutHidenChars As String = PreviousConsoleText 'Holds the previous
console text, that is without the secret chars (*), makes it so number of chars in
password is unknown for security

        While ConsoleKey.Key <> 13 'While char input is not enter key

            Console.Clear()
            HUI.TextFormat.Colour.ColouriseText(PreviousConsoleText)

            ConsoleKey = Console.ReadKey()

            If ConsoleKey.Key = 8 Then 'If the entered key is backspace/delete

                If input.Length > 0 Then 'Prevents the user from deleting the input when
there are 0 chars in it
                    input = input.Substring(0, input.Length - 1)
                    PreviousConsoleText = PreviousConsoleText.Substring(0,
PreviousConsoleText.Length - 1)
```

```
                End If

            Else ' If it's not backspace/delete then put the character in the input
variable
                input = input & ConsoleKey.KeyChar
                PreviousConsoleText = PreviousConsoleText & SecretCharacter

            End If

        End While

        'clears and hides the length of user entered password
        PreviousConsoleText = PCT_WithoutHidenChars
        Dim RepeatedSecretCharacter = SecretCharacter & SecretCharacter &
SecretCharacter
        PreviousConsoleText = PreviousConsoleText & RepeatedSecretCharacter & "[HIDDEN
PASSWORD]" & RepeatedSecretCharacter & Environment.NewLine
        Console.Clear()
        HUI.TextFormat.Colour.ColouriseText(PreviousConsoleText)

        Return input.Substring(0, input.Length - 1)

    End Function

End Module
```

User end / password validator:

Visual basic (Line 465)

```
Module module1
    ' Entrance into program:
    ' - Note: The program will enter through Module1 and to sub Main
    '         This is where the user will have to provide in their login credentials to access the
program
    ' - Page layer: 000 (None)

    'Variable that stores the text visible to the console so far - Used for secret input
    Public PreviousConsoleTextGlobal As String = ""

    Sub Main(args As String()) ' #Sub entrance into program:

        'Declare any variables to use in 'sub main' of program:
        'Password for program
        Dim LogonPassword As String = "Password123"

        HUI.TextFormat.Colour.ColouriseText("<!$white_background$!><!$black_text$!> School register
system <!$reset$!>" & Environment.NewLine & Environment.NewLine)
        PreviousConsoleTextGlobal &= "<!$white_background$!><!$black_text$!> School register system
<!$reset$!>" & Environment.NewLine & Environment.NewLine

        HUI.TextFormat.Colour.ColouriseText("To access the application, please enter the password."
& Environment.NewLine)
        PreviousConsoleTextGlobal &= "To access the application, please enter the password." &
Environment.NewLine

        'Ask for password and loop until the password is correct
        While True

            HUI.TextFormat.Colour.ColouriseText(Environment.NewLine & "Password: ")
```

```
            PreviousConsoleTextGlobal &= Environment.NewLine & "Password: "

            Dim UserInput_LogonPassword As String = Function_SecretInput(PreviousConsoleTextGlobal,
"*") 'Ask user to enter password securely

            If UserInput_LogonPassword = LogonPassword Then

                HUI.TextFormat.Colour.ColouriseText(Environment.NewLine & "<!$green_text$!>Password
was correct! Logging in now.<!$reset$!>")
                PreviousConsoleTextGlobal &= (Environment.NewLine & "<!$green_text$!>Password was
correct! Logging in now.<!$reset$!>")

                For time As Integer = 0 To 6 ' 5 seconds ' set to 6 in irl
                    Thread.Sleep(200) ' 250 milliseconds = 0.25 seconds
                    HUI.TextFormat.Colour.ColouriseText("<!$green_text$!>.<!$reset$!>")
                    PreviousConsoleTextGlobal &= ("<!$green_text$!>.<!$reset$!>")
                Next

                Exit While ' Exit the loop if the password was correct

            Else
                HUI.TextFormat.Colour.ColouriseText(Environment.NewLine & "<!$red_text$!>Password
was incorrect! Please try again...<!$reset$!>")
                PreviousConsoleTextGlobal &= Environment.NewLine & "<!$red_text$!>Password was
incorrect! Please try again...<!$reset$!>"
            End If
        End While

        'Go to main menue through user logon
        MainMenue.MainMenue()

    End Sub

End Module
```

Human user interface / human-computer interface: Program page layers with use of stacks, custom class data types and objects oriented programming

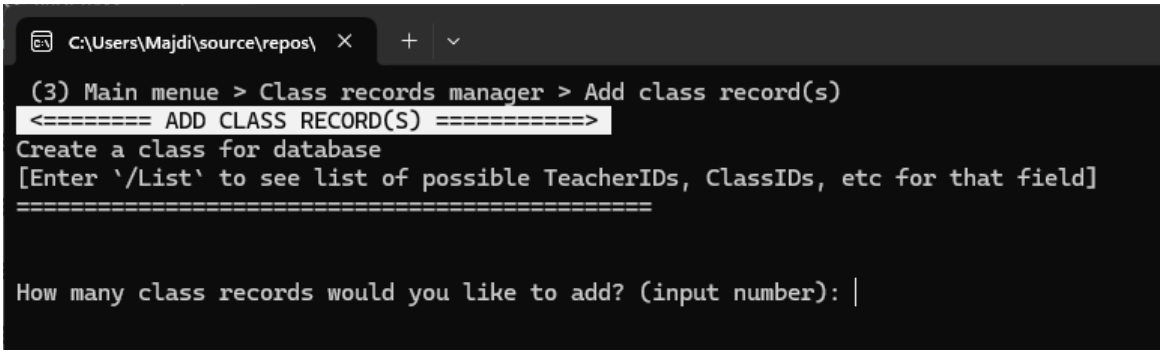
A human user interface (HUI), also commonly referred to as a human-computer interface (HCI) or simply a user interface (UI), is the point of interaction between a human user and a computer system. It encompasses all the elements that enable users to interact with a system, software, or device, including input mechanisms (like keyboards, mice, touchscreens, and voice commands), output devices (such as monitors, speakers, and printers), and the graphical user interface (GUI) or command-line interface (CLI) that users interact with to perform tasks.

The goal of a HUI is to facilitate efficient and intuitive interaction between humans and computers, allowing users to input commands or data and receive feedback or output in a way that is understandable and manageable for them. A well-designed user interface can enhance user experience, productivity, and accessibility, while a poorly designed one can lead to frustration, errors, and inefficiency.

For this project, I'll be using HUI through a CLI (command-line interface) using visual basics.
I'll be using the command line interface as it requires less processing and using VB console app, it can be run on nearly all devices, such as windows, mac, chromeOS, linux, etc.

Issue / solution:
However, I want to create a good user experience through UI, and CLI is very limited in its abilities. Due to this I'll make my own *'graphical'* user interface. I'll use text on the console app to decorate and show visual elements of the program. Such as creating headers and input sections.

Here's an example of the console header/section divider:



The lines of `===` are used to separate the header of the page from the main content.

Using CLI limits me from being able to create multiple windows and new pages when a user interacts with the program. However, I can emulate the behaviour of accessing new pages in VB by clearing the console using the `console.clear` command and rewriting the page. This will be done by the use of stacks, custom data types and built in function to clear the console.

Pseudocode demo & diagram of solution:

```
PSEUDO CODE:

New class called `PageManger`
      Public shared new stack of string type called `PageLayerStack`
      Declare `CurrentPage` as string

      Function PrintToPage (parameters: `ContentToWrite` as string)
            Print to console `ContentToWrite`
            Add `ContentToWrite` onto string `CurrentPage`
      End Function

      Function `ForwardPage`
            Clear console
            Push `CurrentPage` onto stack `PageLayerStack`
            CurrentPage = ``
      End Function


      Function `BackPage`
            Clear console
            `PreviousPage` = Peek at top layer of `PageLayer`
            Print to console `PreviousPage`
            Remover the current stack layer
      End Function

End class
```

Scene 1: Pseudo code to setup of page creation code

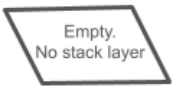
```
PSEUDO CODE:
...
PrintToPage("<p1: Header of page>
Hello world

Enter one of the following:
Option 1
Option 2
Option 3

User input:")
...
Rest of code handling user input and
where to take the user
```

```
<p1: Header of page>
Hello world

Enter one of the following:
 1.   Option 1
 2.   Option 2
 3.   Option 3

User input: 1
```

Live value of `CurrentPage` variable:

```
<p1: Header of page>
Hello world

Enter one of the
following:
  1.    Option 1
  2.    Option 2
  3.    Option 3

User input:
```

Live value of `PageLayerStack` variable:



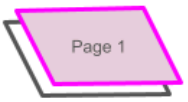
Program prints to console, that text is saved in `CurrentPage`. User is prompted to select a page. They enter "1" - program will take them to "Option 1" - the 2nd page

Scene 2: Page 1: Writing to console and user inputs page number

```
PSEUDO CODE:
...
ForwardPage()
PrintToPage("<p2: New page>
Hello world, again.

Enter one of the following:
Option 1
Option 2
Option 3
Go back page

User input: ")
... Rest of code handling user input
and where to take the user
```

```
<p2: New page>
Hello world, again.

Enter one of the following:
 1.   Option 1
 2.   Option 2
 3.   Option 3
 4.   Go back page

User input: 2
```

```
<p2: New page>
Hello world, again.

Enter one of the
following:
Option 1
Option 2
Option 3
Go back page

User input: 2
```



As user enters the new page, `ForwardPage()` pushes the `CurrentPage` text onto the stack. Page 1 from before is now saved on the stack, and clears `CurrentPage`. The next line `PrintToPage(...` creates the page 2's content and saves it on the now cleared `CurrentPage` variable. User then selects the next page, "option 2" - the 3rd page

Scene 3: Page 2: Create new page

```
PSEUDO CODE:
...
ForwardPage()
PrintToPage("<p3 Another...
...
If input 4:
      BackPage()
...
```

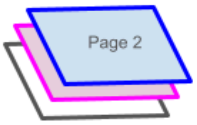
```
<P3 Another new page>
3rd page created

Enter one of the following:
 1.   Option 1
 2.   Option 2
 3.   Option 3
 4.   Go back page

User input: 4
```

Scene 4: Page 3: Create a new page, and user goes back page

Live value of `CurrentPage` variable:

```
<P3 Another new page>
3rd page created

Enter one of the
following:
Option 1
Option 2
Option 3
Go back page

User input:
```

Live value of `PageLayerStack` variable:



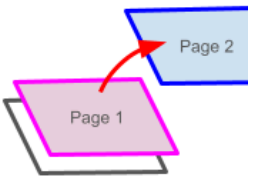
Similar to as before, user enters the new page, `ForwardPage()` pushes the `CurrentPage` text onto the stack. Page 2 from before is now saved on the stack, and clears `CurrentPage`. The next line `PrintToPage(...` creates the page 3's content and saves it on the now cleared `CurrentPage` variable. User then selects to go back to previous page, "Go back page" - the 2nd page. `BackPage()` clears the console and shows the previous page layer. Then pops that layer off.

```
PSEUDO CODE:
...
ForwardPage()
PrintToPage("<P2 New page...")
...
If input 4:
      BackPage()
...
```

```
<P2 New page>
Hello world, again.

Enter one of the following:
 1.   Option 1
 2.   Option 2
 3.   Option 3
 4.   Go back page

User input: 4
```

Scene 5: Go back to page 2

```
<p2: New page>
Hello world, again.

Enter one of the
following:
Option 1
Option 2
Option 3
Go back page

User input: 2
```



Again, the user wishes to go back a page, as the same from before, the program will use `BackPage()` to clear the console and print the previous page, then pop that layer off.

```
PSEUDO CODE:
...
PrintToPage("<Header of page>
Hello world

Enter one of the following:
Page 1
Page 2
Page 3

User input:")
...
```

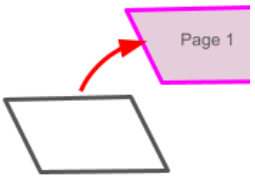
```
<Header of page>
Hello world

Enter one of the following:
 1.   Page 1
 2.   Page 2
 3.   Page 3

User input:
```

Scene 6: Go back to Page 1

Live value of `CurrentPage` variable:

```
<p1: Header of page>
Hello world

Enter one of the
following:
  1.    Page 1
  2.    Page 2
  3.    Page 3

User input:
```

Live value of `PageLayerStack` variable:



Code for solution:

The code to this solution is in the code PDF file attached with this document and along with the VB file.

I created this solution by using object oriented programming in VB by making custom classes. One big benefit of object-oriented programming is that it helps with code organisation and reusability. By breaking down the program into objects, I can easily manage and update different parts of the code without affecting the whole program. Plus, I can reuse objects in other parts of my program or in different projects, saving time and reducing errors. It also promotes modular and scalable code, making it easier to maintain and collaborate on larger projects.

All of these objects, classes and methods are in the `HUI` class of my code [CODE LINE: 94 - 413]. HUI (human user interface) deals with creating pages, writing to pages, exiting pages, storing user input to pages, etc.

Code section / description	Page	Line number
HUI class	Page 3 - 12	94 - 413

Public Class HUI ' Human user interface Class containing all the Human user interface code.		
Page class (as part of HUI class) **Public Class Page** This sub class in the HUI class contains the custom data types for page creation, retrieval and stacking.	Page 3 - 8	97 - 265
Data class (as part of Page class, nested with in HUI class) **Public Class Data** This nested class is the custom data type for the pages metadata in the program. In the demo/diagram shown before, for the UI page solution, the stack holding the page layers were of the string type. However, in the implementation of this solution, the stack is of a custom data type called `data` (accessible by `HUI.Page.Data`) composed of other data types and a custom `new` method. The custom data class contains data such as the PageTitle, PageDescription, PageContentString, etc. `PageContentString` is continuously appended to as content is written to the console. The subroutine `New` in this class dictates the default values of this class and how it will behave.	Page 3 - 4	100 - 126

Creation of a new stack holding page layers (as part of Page class, nested with in HUI class)
This code creates a stack instance of the custom data type `data` (`HUI.Page.Data`) that will store the series of pages throughout the run of this program.
Code:
`' Make shared stack instance of pages type`
`Private Shared PagesLayer As New Stack(Of Data)()`
	Page 4	129

Create a new page / forward / next page (as part of Page class, nested with in HUI class) **Public Shared Sub EnterView(ByVal NewPageData As Data)** Takes in parameters for the page title, description, layer etc. Creates a new page and displays it to the user. Pushes the new page onto the stack of pages. Displays the navigation path at the top. Prints the header, description, possible navigation, and content of the page. This is the main code responsible for making these new pages.	Page 4 - 5	132 - 170
Write and WriteLine to console and page (as part of Page class, nested with in HUI class) **Public Shared Sub Write(ByVal AddToPageString As String)** **Public Shared Sub WriteLine(ByVal AddToPageString As String)** These two classes are custom methods that allow the function to write or writeline to the console and write to the pages. Rather than writing to the console then saving what has been written to the console constantly for every time I want to print somthing, I created a method which will write content to the console and also save it to the current open page with one single command. Using a class to make the method `HUI.Page.WriteLIne("Hello world!")` & `HUI.Page.Write("Hello world!")`	Page 6	172 - 180 & 182 - 190
Read input from user, appended it to current page (as part of Page class, nested with in HUI class)	Page 7 - 8	227 - 238 &

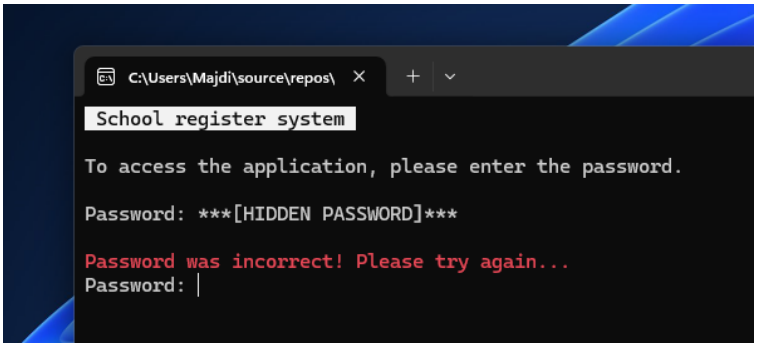
Public Shared Function ReadLine() As String **Public Shared Function Read() As String** These are similar to the write/writeline methods. It reads the input from the user and appends it to the current open page stack. Without this, if a user entered data on 1 page then went to another page and came back to the original page, their data would have been lost.		240 - 250

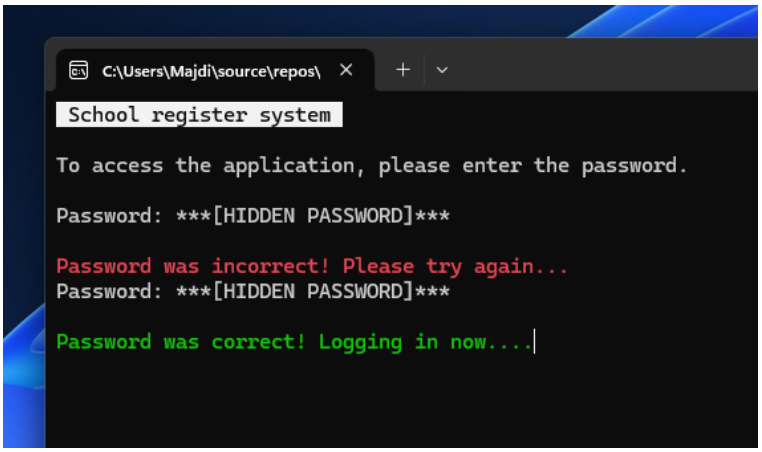
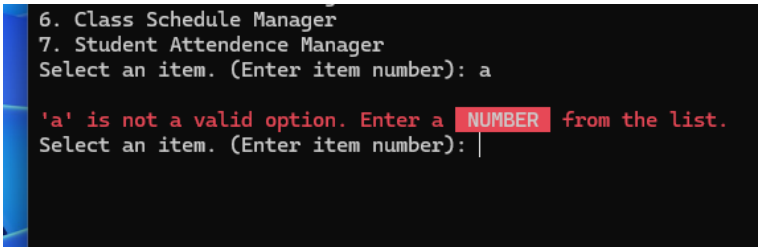
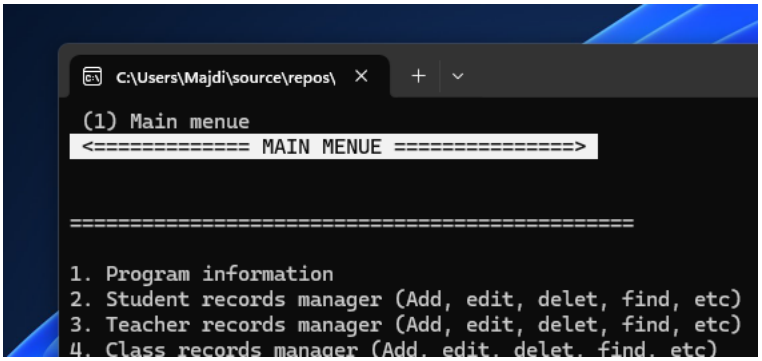
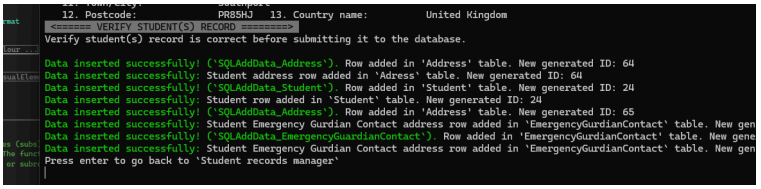
Human user interface / human-computer interface: Console colour and visual elements text formatting

Code section / description	Page	Line number
HUI class **Public Class HUI ' Human user interface** Class containing all the Human user interface code.	Page 3 - 12	94 - 413
Text Format class (as part of HUI class) **Public Class TextFormat** This sub class in the HUI class contains the methods to print coloured text to the console along with text elements such as a header/divider.	Page 8 - 12	267 - 411
Data class (as part of Page class, nested with in TextFormat class) **Public Shared Sub ColouriseText(input As String)** This nested method accessible by `HUI.TextFormat.Colour.ColouriseText()` takes in a string parameter. This string parameter is the text to output to the console, within it it also contains tags such as <!$yellow_text$!>, <!$red_background$!>, etc and <!$reset$!>. These determine the colour of the text to output. For example this parameter: “Hello <!$green_background$!>world!<!$reset$!> This is an <!red_text!>example of how <!$cyan_background$!>text would<!$reset$!> be colour formatted!” Would be processed and outputted as such: Hello world! This is an example of how text would be colour formatted! This is done by taking the input string and splitting it into segments based on the specified tags. The default foreground and background colours are initially set to empty strings. The code then iterates through each segment, searching for the tags. If a tag is found, it extracts the tag content and the remaining content. If the tag content contains "_text", it means it specifies the colour for the text. The function "PrintColourConsole" is called to print the coloured text using the extracted foreground colour and background colour. If the tag content contains "_background", it means it specifies the colour for the background. Again, the function "PrintColourConsole" is called to print the coloured text with the specified background colour. If the tag content is "reset", it means it is specifying to reset the foreground and	Page 9 - 11	290 - 373

background colours. The function "PrintColourConsole" is called to print the text with the colours reset. If the tag is not recognized, the segment is written as is. The function "GetConsoleColor" is used to convert the tag content to the ConsoleColor enum. The function "PrintColourConsole" sets the foreground and background colour of the console if they are specified. It prints the text and then resets the colour to default. To use the nested method "ColouriseText" within the TextFormat class, it is called using the full name "HUI.TextFormat.Colour.ColouriseText()". This is a long method name to use throughout the text, however the `HUI.page.Write` (and writeline) will also output colourised text as it passes it's parameters to it.		
Visual elements creator, header/divider creator. (as part of Page class, nested with in TextFormat class) **Public Shared Function GenerateHeader(title As…** Using only a console based interface makes my program more portable and easier to access on more devices, however it limits the ability to create advanced visual elements and ideas. This can make it harder to separate different sections and other elements apart. One way to do this though is to make a line/header to spit sections apart clearly. This method does this. The method "GenerateHeader" in the VisualElements class is used to generate a formatted header with a specified title and optional parameters such as border character, maximum length, and inclusion of brackets. This method returns the formatted header string.	Page 11	377 - 406

Screenshots of evidence for console text colouring:

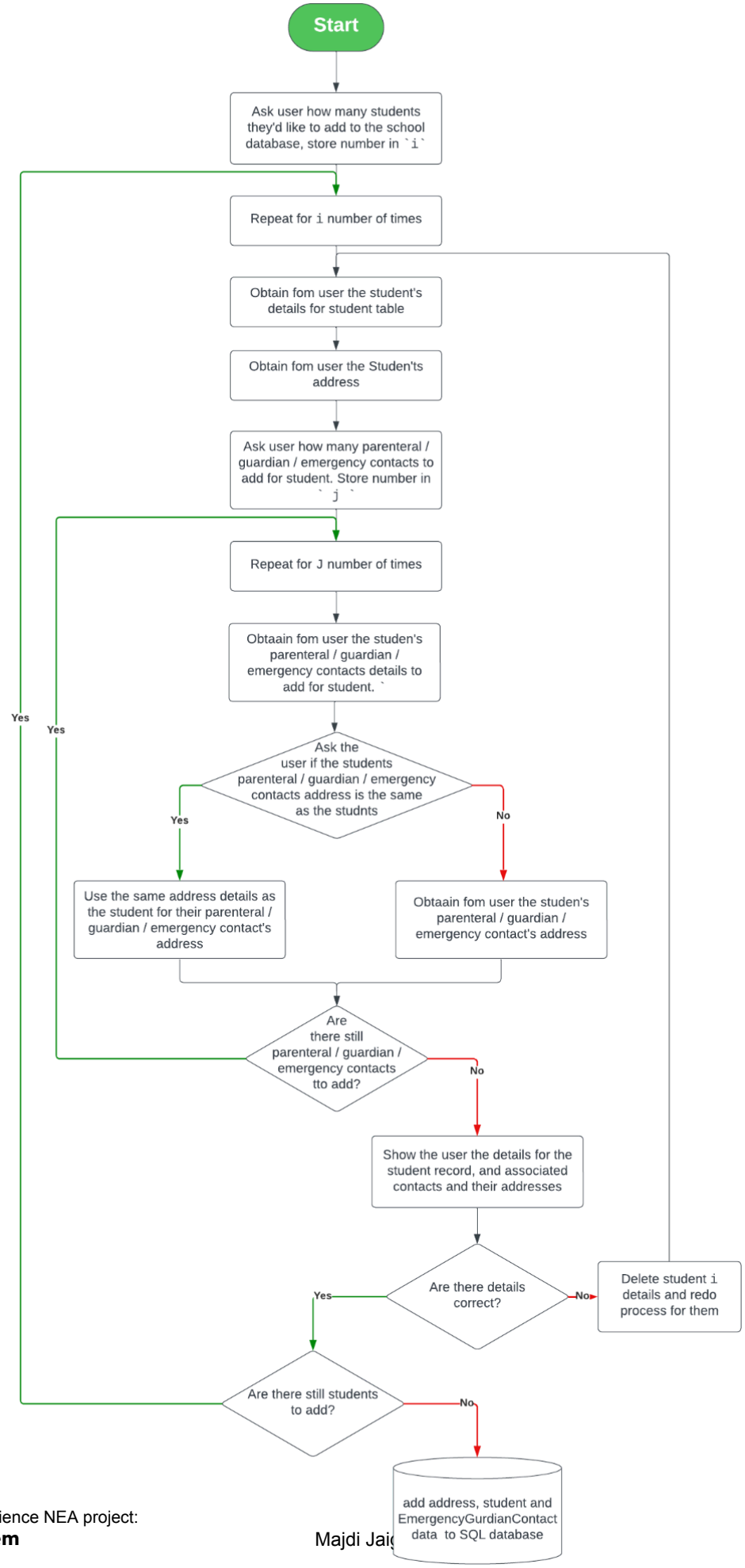
#	Screenshot(s)	Caption/description
1		Showcases how text colour can be used to indicate an error or invalid input - in this case an incorrect password - this draws the user's attention and showcases somthing is wrong It also showcases how background colour can be used to focus on an element, the "school register system" is highlighted in white, bringing focus to it as well.

2		Opposite to an error or an issue, this green text indicates to the user that the password they imputed is correct
3		Showcases how text colour and background colour indicate an error.
4		The use of the header and divider separate components and make the focus of the screen clearer
5		

SQL database: Adding data to the database

Adding student record and other associated records

The system will store the students' information who are in this school in a microsoft SQL express server database. The students' details will span across 3 different tables, these will be `student` - the main table holding the student's personal information, `EmergencyGurdianContact` - storing the student's parenteral / guardian / emergency contacts, and `address` - the table holding the student's address and their emergency contact's address. When adding a student, these 3 tables will be the initial table that holds student content. The data input process from the user looks like this:


Start
Ask user how many students they'd like to add to the school database, store number in `i`
Repeat for i number of times
Obtain fom user the student's details for student table
Obtain fom user the Studen'ts address
Ask user how many parenteral / guardian / emergency contacts to add for student. Store number in ` j `
Repeat for J number of times
Obtaain fom user the studen's parenteral / guardian / emergency contacts details to add for student. `
Ask the user if the students parenteral / guardian / emergency contacts address is the same as the studnts
Yes
No
Use the same address details as the student for their parenteral / guardian / emergency contact's address
Obtaain fom user the studen's parenteral / guardian / emergency contact's address
Are there still parenteral / guardian / emergency contacts tto add?
Yes
No
Show the user the details for the student record, and associated contacts and their addresses
Are there details correct?
Yes
No
Delete student i details and redo process for them
Are there still students to add?
Yes
No
add address, student and EmergencyGurdianContact data to SQL database

The details added to the SQL database happen in this order:

1. The student Address is first added and the new ID is obtained for that row
2. The Student details are added and the ID obtained from adding the address is used as the Address ID. The ID is obtained for this new student row.
3. The Student's emergency/guardians(') Address(es) are added to the Address table and the ID for this new row obtained
4. The EmergencyGurdianContact are added to the table and use the previous StudentID obtained by adding the student, and use the AddressID by adding the contact's address.

Technical code for this solution:

Code section / description	Page	Line number
SQL function code to add address to the database **Function SQLAddData_Address(connectionString As St…** This SQL function within the SQLAddData module is responsible for adding for students, Emergency/Guardian Contacts, teachers addresses. By creating this into a function, and creating it so the function takes in parameters `ByRef` and returns a boolean, I'm able to reuse the code and do less reception. The function returns a boolean to indicate a successful data addition or unsuccessful one along with potential error messages. This function is passed in 4 parameters; `connectionString As String, ByRef AddressDetails As Address, Optional ByRef ErrorString As String = "", Optional ByVal ShowSuccessErrorMessages As Boolean = True` connectionString is simply the connection details to the server, rather than making this a consent or embedding it in each SQL function I've made this a variable, in case in the future I'd like to edit the program to connect to different multiple servers or change user login method. `AddressDetails ` Is as a custom data type of Address, this is `ByRef`, this means that the function can change the values of this parameter passed in by the calling block of the code. This is done as assigning the ID to a new address row is done by the SQL server, and so this function obtains this ID for the new row and assigns the value to `AddressDetails.id` which can be accessed by the calling code section. `ErrorString` & `ShowSuccessErrorMessages` are both optional parameters that handle the errors/success that occur in the attempt to add data to the table. `ShowSuccessErrorMessages` is an optional boolean that by default is true, it prompts the function if it should write error / success messages to the console or not. This is done as it allows me to reuse the same function again throughout the code without having to make modifications for every instance of using/calling this function. For example, I may want to handle an error differently in one very specific part of the code while not in the rest. `ErrorString` is the error message the SQL server returns. This can be used by the calling block of code or by the function itself to show the error. If the `ShowSuccessErrorMessages` is true, the `SQLAddData_INTERNAL_PROCESS_ErrorSuccessMessage` function will	Page 19 - 20	706 - 756

process and write the error message. This function is discussed more in “Data validation, formatting, errors and success messages” of this document SQL select query: `Dim sqlQuery As String = "INSERT INTO Address (HouseNumberName, StreetName, LocalityName, CityTownName, Postcode, CountryName) OUTPUT INSERTED.ID, INSERTED.HouseNumberName, INSERTED.StreetName, INSERTED.LocalityName, INSERTED.CityTownName, INSERTED.Postcode, INSERTED.CountryName VALUES (@HouseNumberName, @StreetName, @LocalityName, @CityTownName, @Postcode, @CountryName)"`		
SQL function code to add student Emergency / Guardian Contacts to the database **Function SQLAddData_EmergencyGuardianContact(connect…** This SQL function within the SQLAddData module, similarly to `SQLAddData_Address` adds the student’s emergency/guardian contacts details to the SQL database table `EmergencyGurdianContact`. This is also a function to make it reusable, which means less reception of code throughout the program. It uses the same parameters and error handling method as `SQLAddData_Address`. SQL query: `Dim sqlQuery As String = "INSERT INTO EmergencyGurdianContact (StudentID, Title, FirstName, MiddleName, LastName, StudentRelationship, PhoneNumber, AddressID) OUTPUT INSERTED.ID, INSERTED.StudentID, INSERTED.Title, INSERTED.FirstName, INSERTED.MiddleName, INSERTED.LastName, INSERTED.StudentRelationship, INSERTED.PhoneNumber, INSERTED.AddressID VALUES (@StudentID, @Title, @FirstName, @MiddleName, @LastName, @StudentRelationship, @PhoneNumber, @AddressID)"`	Page 20 - 22	758 - 812
SQL function code to add student data row to the databaseFunction SQLAddData_Student(connectionString As String, ByRef st… This SQL function within the SQLAddData module, similarly to `SQLAddData_Address` adds the student’s details to the SQL database table `Student`. This is also a function to make it reusable, which means less reception of code throughout the program. It uses the same parameters and error handling method as `SQLAddData_Address`. SQL query: `Dim sqlQuery As String = "INSERT INTO Student (FirstName, MiddleName, LastName, DateOfBirth, AddressID) OUTPUT INSERTED.ID, INSERTED.FirstName, INSERTED.MiddleName, INSERTED.LastName, INSERTED.DateOfBirth, INSERTED.AddressID VALUES (@FirstName, @MiddleName, @LastName, @DateOfBirth, @AddressID)"`	Page 22 - 23	814 - 862

Adding teacher record and other associated records

With the similar methods and code as `Adding student record and other associated records`, the teachers record asks the user for the teachers details and asks for confirmation and stores them. The teacher’s record is broken into two tables, `teacher` and `address`, where again the teacher’s address is stored in a different table and is referenced in the teacher’s table by `AddressID`

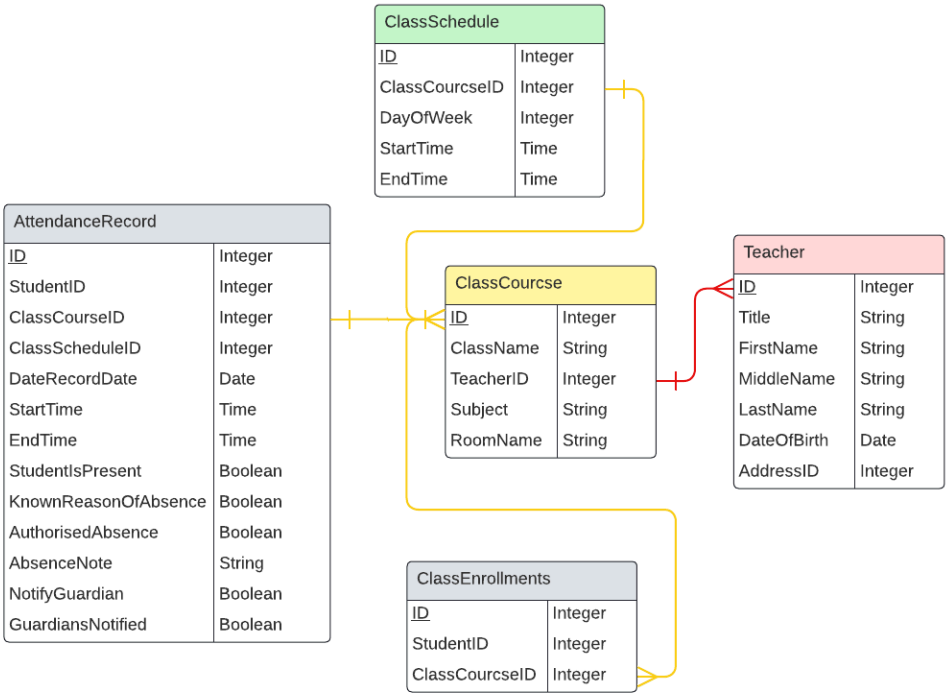
Code section / description	Page	Line number
SQL function code to add teacher to the database **Function SQLAddData_Teacher(connectionString As St…** SQL select query: `Dim sqlQuery As String = "INSERT INTO Teacher (Title, FirstName, MiddleName, LastName, DateOfBirth, AddressID) OUTPUT INSERTED.ID, INSERTED.Title, INSERTED.FirstName, INSERTED.MiddleName, INSERTED.LastName, INSERTED.DateOfBirth, INSERTED.AddressID VALUES (@Title, @FirstName, @MiddleName, @LastName, @DateOfBirth, @AddressID)"`	Page 23 - 25	864 - 914

SQL database: Deleting data from the database

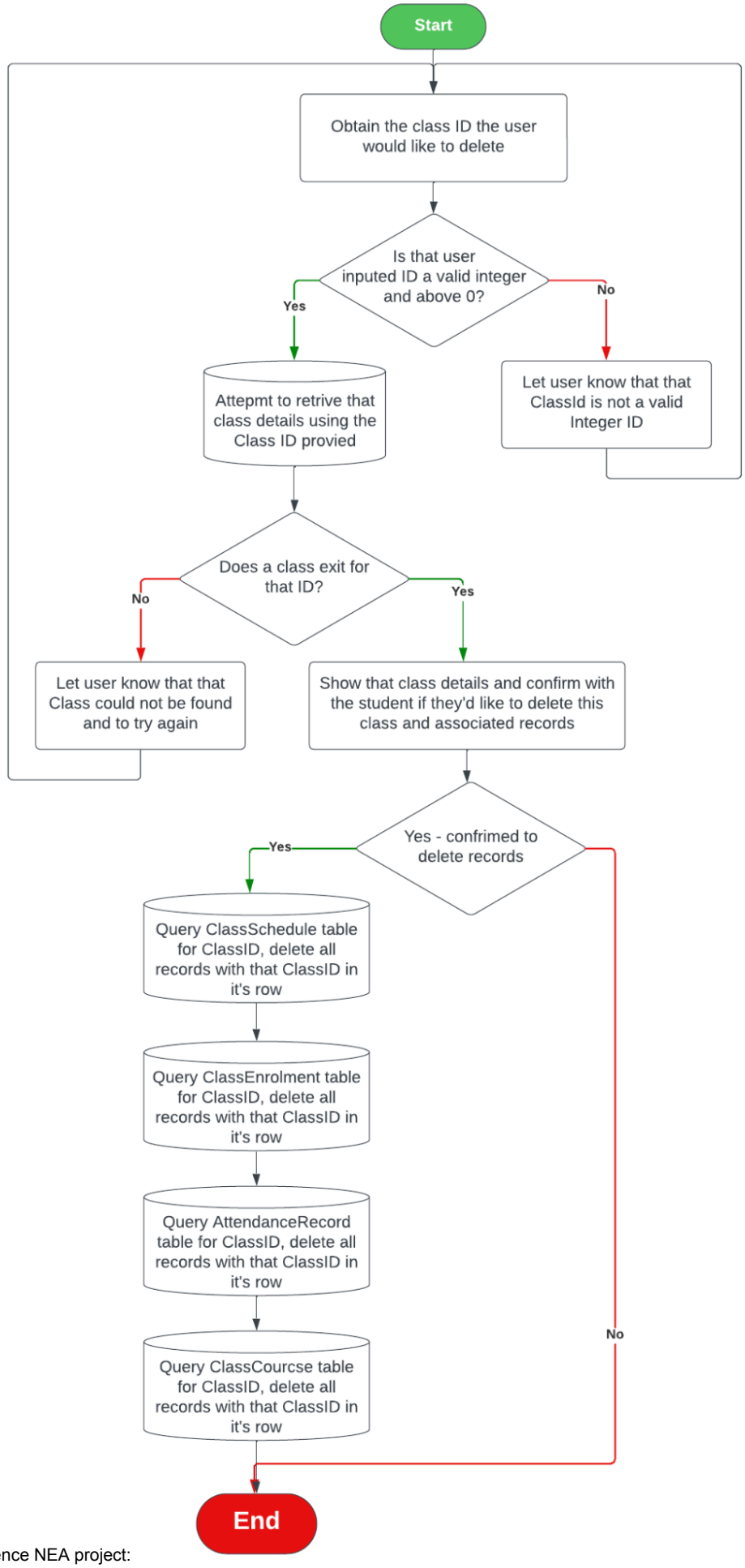
Deleting class records and other associated records

As records such as teachers, students, classes and other tables reference each other and data is spread throughout the database. Deleting data from the database requires that multiple of referenced and reliant records are deleted. For example, the deletion of a class / class course.

Deleting a class from the system would require multiple dependent rows to be deleted. Here's the connection of Class courses on other tables.:



So, if a user wanted to delete a class, the scheduling of that class would need to be deleted, the class enrolments would also need to be deleted, and the attendance record for that class would also need to be deleted. This flowchart showcases the solution:


Start
Obtain the class ID the user would like to delete
Is that user inputed ID a valid integer and above 0?
Yes
No
Attepmt to retrive that class details using the Class ID provied
Let user know that that ClassId is not a valid Integer ID
Does a class exit for that ID?
No
Yes
Let user know that that Class could not be found and to try again
Show that class details and confirm with the student if they'd like to delete this class and associated records
Yes - confrimed to delete records
Yes
No
Query ClassSchedule table for ClassID, delete all records with that ClassID in it's row
Query ClassEnrolment table for ClassID, delete all records with that ClassID in it's row
Query AttendanceRecord table for ClassID, delete all records with that ClassID in it's row
Query ClassCourcse table for ClassID, delete all records with that ClassID in it's row
End

Code to the solution:

Code section / description	Page	Line number

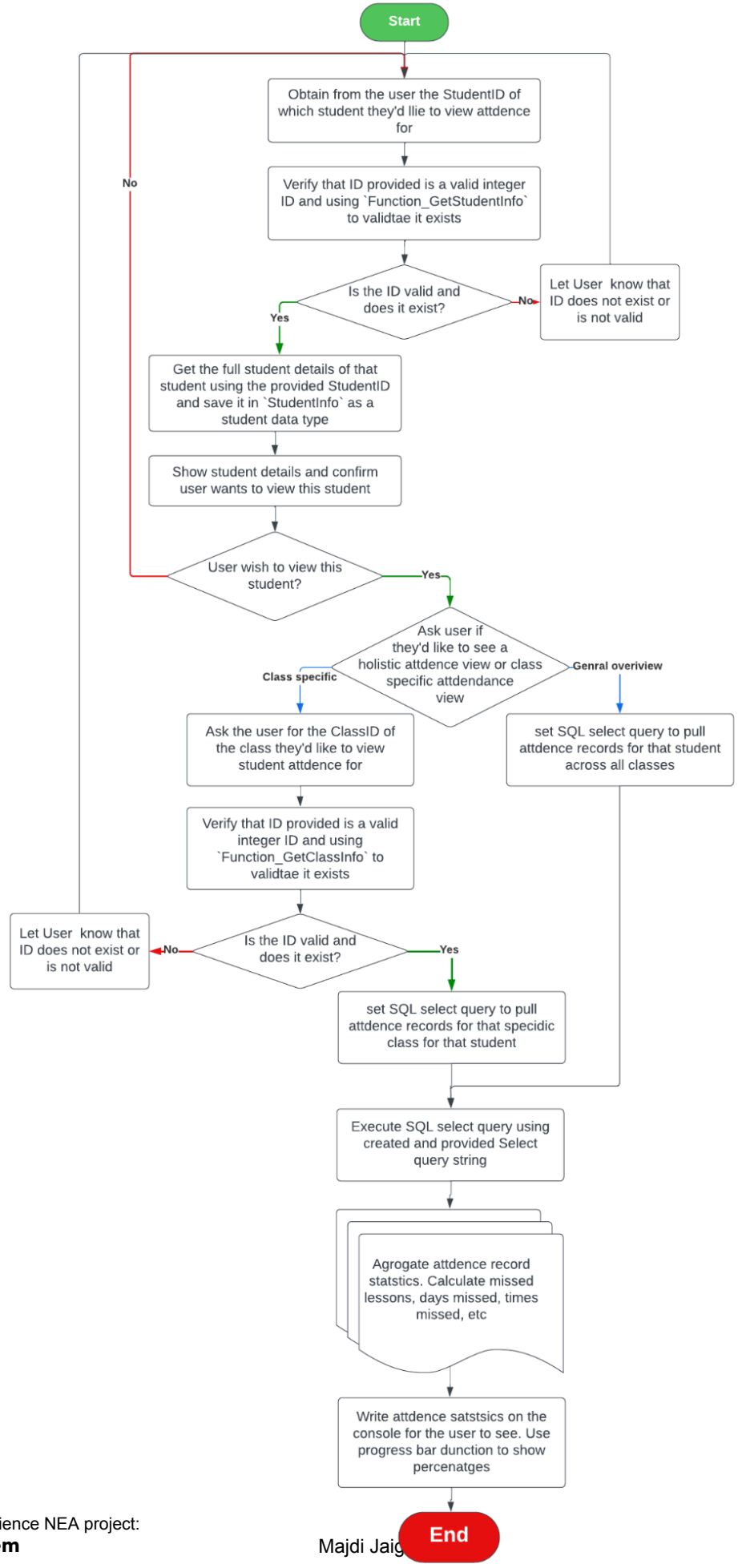
SQL function code to add teacher to the database
Sub Function_DeleteClassCourcse(ByVal ClassCourcseID As Inte…
 	
Records need to be deleted in a particular order as not to break SQL table restraints.
 	
SQL select query to delete class enrolments:
`Dim deleteClassEnrollmentsQuery As String = "DELETE FROM ClassEnrollments WHERE ClassCourcseID = @ClassCourcseID"`
 	
SQL select query to delete attendance records:
`Dim deleteAttendanceQuery As String = "DELETE FROM AttendanceRecord WHERE ClassID = @classID"`
 	
SQL select query to delta class schedules:
`Dim deleteScheduleQuery As String = "DELETE FROM ClassSchedule WHERE ClassID = @classID"`
 	
SQL select query delete the class course:
`Dim deleteClassQuery As String = "DELETE FROM ClassCourcse WHERE ID = @classID"`
	Page 40-41	1494 - 1533

SQL database: Retrieving and finding data from database

Retrieving student attendance data from the database

A core part of a school register system is the ability to retrieve and view the attendance records of students. To get this information, multiple tables will need to be queried using SQL select statements.

Algorithm / flowchart:


Start
Obtain from the user the StudentID of which student they'd llie to view attdence for
Verify that ID provided is a valid integer ID and using `Function_GetStudentInfo` to validtae it exists
Is the ID valid and does it exist?
No
Let User know that ID does not exist or is not valid
Yes
Get the full student details of that student using the provided StudentID and save it in `StudentInfo` as a student data type
Show student details and confirm user wants to view this student
User wish to view this student?
No
Yes
Ask user if they'd like to see a holistic attdence view or class specific attdendance view
Class specific
Genral overiview
Ask the user for the ClassID of the class they'd like to view student attdence for
set SQL select query to pull attdence records for that student across all classes
Verify that ID provided is a valid integer ID and using `Function_GetClassInfo` to validtae it exists
Let User know that ID does not exist or is not valid
No
Is the ID valid and does it exist?
Yes
set SQL select query to pull attdence records for that specidic class for that student
Execute SQL select query using created and provided Select query string
Agrogate attdence record statstics. Calculate missed lessons, days missed, times missed, etc
Write attdence satstsics on the console for the user to see. Use progress bar dunction to show percenatges
End

Code section / description	Page	Line number
Sub and SQL commands to retrieve student attendance record and to show it to the user **Sub MMO_SAM_ViewAttendanceForStudent()** Subroutine as part of `MMO_StudentAttendanceManager` module that prompts the user for a StudentID to view the student's attendance record, and for the classID if the user would like to see a specific class statistics. The subroutine works by querying the database for attendance records, then calculates the statistics for that student's attendance. This is the maths calculation code in the sub:	Page 141 - 154	5251 - 5687

Visual basic (Line 5545)

```
        ' Total number of lessons attended and absent
        Dim totalLessons As Integer =
ArrayOfStudentClassAttendanceRecord.Length
        Dim totalPresent As Integer =
ArrayOfStudentClassAttendanceRecord.Where(Function(x) x.StudentIsPresent =
True).Count()
        Dim totalAbsent As Integer = totalLessons - totalPresent

        ' Ensure totalAbsent is not zero to prevent division by zero error
        If totalAbsent = 0 Then
            totalAbsent = 1
        End If

        ' Percentage of attendance
        Dim attendancePercentage As Double = (CDbl(totalPresent) /
CDbl(totalLessons)) * 100

        ' Total number of unauthorized absences
        Dim totalUnauthorizedAbsences As Integer =
ArrayOfStudentClassAttendanceRecord.Where(Function(x) x.AuthorisedAbsence =
False).Count()

        ' Ensure totalAbsent is not zero to prevent division by zero error
        If totalAbsent = 0 Then
            totalAbsent = 1
        End If

        ' Percentage of absences being unauthorized
        Dim unauthorizedAbsencePercentage As Double =
(CDbl(totalUnauthorizedAbsences) / CDbl(totalAbsent)) * 100

        ' Lessons missed on each day of the week
        Dim lessonsMissedPerDayOfWeek As New Dictionary(Of DayOfWeek,
Integer)()

        ' Total lessons scheduled on each day of the week
        Dim totalLessonsPerDayOfWeek As New Dictionary(Of DayOfWeek,
Integer)()
```

```
        ' Initialize dictionaries with 0
        For Each dayOfWeek As DayOfWeek In
[Enum].GetValues(GetType(DayOfWeek))
            lessonsMissedPerDayOfWeek(dayOfWeek) = 0
            totalLessonsPerDayOfWeek(dayOfWeek) = 0
        Next

        ' Iterate over attendance records to calculate lessons missed per
day of week
        For Each attendanceRecord As AttendanceRecord In
ArrayOfStudentClassAttendanceRecord
            Dim dayOfWeek As DayOfWeek =
attendanceRecord.DateRecordDate.DayOfWeek
            totalLessonsPerDayOfWeek(dayOfWeek) += 1
            If attendanceRecord.StudentIsPresent = False Then
                lessonsMissedPerDayOfWeek(dayOfWeek) += 1
            End If
        Next

        ' Calculate percentage of lessons missed on each day of the week
        Dim missedLessonsPercentagePerDayOfWeek As New Dictionary(Of
DayOfWeek, Double)()

        For Each kvp As KeyValuePair(Of DayOfWeek, Integer) In
totalLessonsPerDayOfWeek
            Dim dayOfWeek As DayOfWeek = kvp.Key
            Dim totalLessonsOnDay As Integer = kvp.Value
            Dim missedLessonsOnDay As Integer =
lessonsMissedPerDayOfWeek(dayOfWeek)
            If totalLessonsOnDay = 0 Then
                missedLessonsPercentagePerDayOfWeek(dayOfWeek) = 0
            Else
                missedLessonsPercentagePerDayOfWeek(dayOfWeek) =
(CDbl(missedLessonsOnDay) / CDbl(totalLessonsOnDay)) * 100
            End If
        Next


        ' Absence hours:
        ' Dictionary to store absences per hour
        Dim absencesPerHour As New Dictionary(Of Integer, Integer)()

        ' Initialize the dictionary with 0 absences for each hour
        For hour As Integer = 9 To 17
            absencesPerHour(hour) = 0
        Next

        ' Iterate over attendance records to tally absences per hour
        For Each attendanceRecord As AttendanceRecord In
ArrayOfStudentClassAttendanceRecord
            ' Get the hour of the lesson start time
            Dim startHour As Integer = attendanceRecord.StartTime.Hours

            ' Increment the absence count for the corresponding hour
            absencesPerHour(startHour) +=
If(attendanceRecord.StudentIsPresent = False, 1, 0)
```

```
        Next

        ' Calculate total lessons per hour
        Dim totalLessonsPerHour As New Dictionary(Of Integer, Integer)()

        For hour As Integer = 9 To 17
            Dim currentHour = hour
            totalLessonsPerHour(hour) =
ArrayOfStudentClassAttendanceRecord.Where(Function(x) x.StartTime.Hours =
currentHour).Count()
        Next
```

Using the maths calculated it then shows these stats to the user on the console. For representing percentages, I've created a function that makes a visual progress / percentage bar. This function is called by `GeneratePercentageBar(missedLessonsPercentagePerDayOfWeek(DayOfWeek.Wednesday))`

Function to create a visual element of a progress / percentage bar to show on console
Function GeneratePercentageBar(ByVal percentage As Integer) As Stri…
This is a function in the `TextFormat` module that creates a visual text element of a progress bar using the unicode charters of solid blocks (█), and use of semi solid solid blocks (▒). Used together, these can represent a percentage number in a visual manner.

This is the maths calculation and code for the bar:

Visual basic (Line 530)

```
    Function GeneratePercentageBar(ByVal percentage As Integer) As String
        'Make progress bar

        ' Ensure percentage is within range 0 to 100
        If percentage < 0 Then
            percentage = 0
        ElseIf percentage > 100 Then
            percentage = 100
        End If

        ' Calculate number of full blocks and remaining empty blocks
        Dim fullBlocks As Integer = percentage \ 5
        Dim emptyBlocks As Integer = (100 - percentage) \ 5

        ' Generate the percentage bar string
        Dim percentageBar As String = New String("", fullBlocks) & New
String("", emptyBlocks)

        Return percentageBar
    End Function
```

The function returns a string of the created progress / percentage bar.

Page 14 - 15

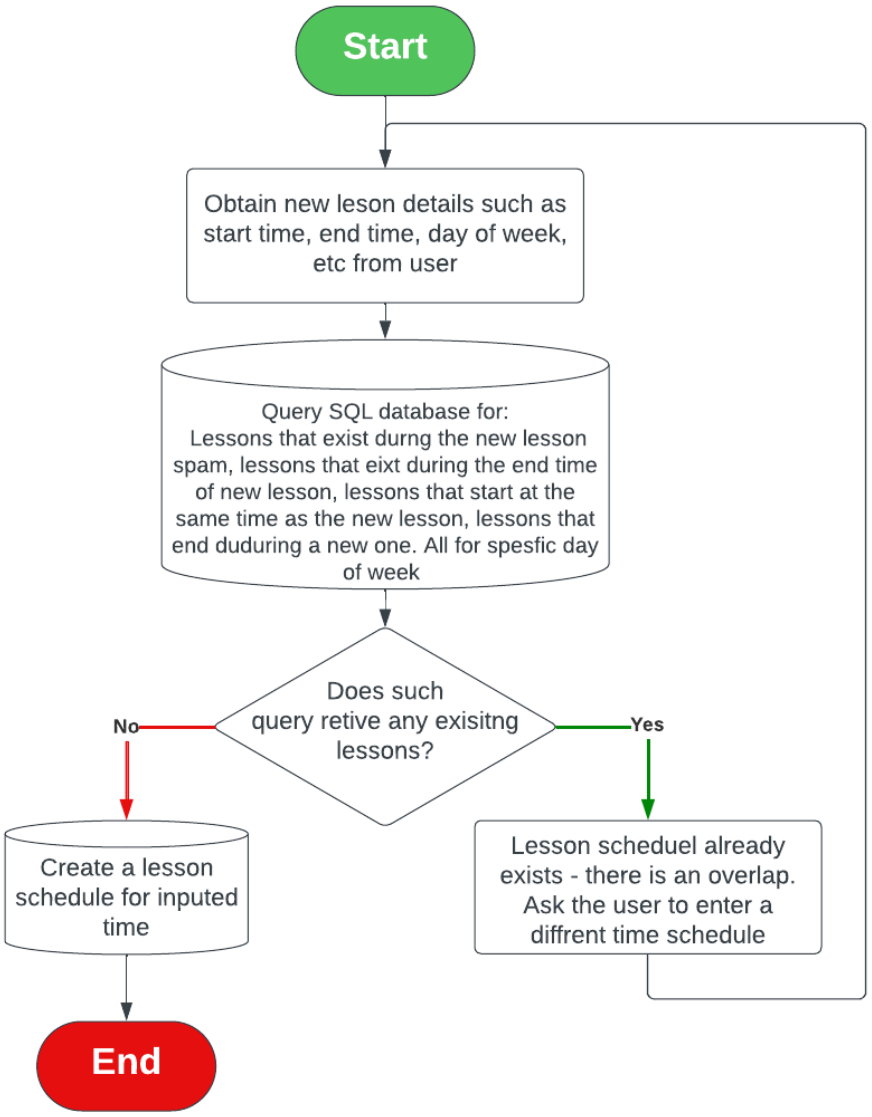
530 - 548

SQL database & lesson time logic: Creating lesson schedules and preventing lesson overlap

In this deployment of my school register system, in this demo, the school can only have one lesson at a time. In a real life school environment multiple lessons could be happening concurrently for different year/grade groups, classes, schedules, etc. But for simplification of demonstrating this system, there can only be one lesson at a time, to implement the ability to have multiple lessons at once, the use of schedule groups/year groups would be used. This would be used to match the potential new class lesson time to an already existing one of that class by simply seeing if a lesson already exists for a less scheduled group.

The creation of a lesson schedule and checking if a lesson time does not overlap works by taking the user input start and end time for a new lesson and the day of the week for that lesson and comparing it against the SQL database table `ClassSchedule`.

Flowchart:



Code for solution:

Code section / description	Page	Line number
SQL function code to check if a new lesson would overlap an already existing lesson **Public Function Function_CheckForOverlaps(ClassID As Integer, Da…** Function to check to see if a new proposed lesson would overlap an existing lesson. It takes in the parameters `ClassID As Integer, DayOfWeek As Integer, startTime As TimeSpan, endTime As TimeSpan, ByVal connectionString As String`, using these it checks to see if a lesson exists during these times. It outputs a boolean true or false to indicate if a lesson does overlap, by default when leasing, it assumes a lesson overlap is false. SQL select query: `Dim query As String = "SELECT ID, ClassID, DayOfWeek, StartTime, EndTime FROM ClassSchedule WHERE DayOfWeek = @DayOfWeek AND ((@startTime < EndTime AND @startTime >= StartTime) OR (@endTime > StartTime AND @endTime <= EndTime) OR (@startTime = StartTime) OR (@endTime = EndTime))"`	Page 25 - 26	918 - 951
SQL function/sub code to check if a new lesson would overlap an already existing lesson **Public Sub Function_AddLesson(ClassID As Integer, DayOfWeek A…** Function/sub to create a new lesson schedule. This function takes in the parameters: `ClassID As Integer, DayOfWeek As Integer, startTime As TimeSpan, endTime As TimeSpan, ByVal connectionString As String` it then calls the `Function_CheckForOverlaps` function and passes the parameters in to see if an already existing lesson would overlap it. If so, the function lets the user know and prevents the lesson creation. If there are no overlaps, the function will create the lesson using the provided parameters, SQL select query: `Dim query As String = "INSERT INTO ClassSchedule (ClassID, DayOfWeek, StartTime, EndTime) VALUES (@ClassID, @DayOfWeek, @startTime, @endTime)"`	Page 26	954 - 974

Data validation of user input, data formatting, verification of existence of data, formatting, and error / success messages

This section covers the parts of the code and SQL database that validate that a data value is valid. Valid in the format it is in, valid as a possible option, valid that it exists and that a process has been successful or not. It ensures that users enter correct data in a correct format, such as dates, it ensures that user selects an item from a list that exists, ensures data added to the SQL database is correct, and a process has been done successfully.

SQL query: error / success messages

Briefly mentioned previously in this document, the program indicates when errors or successful data inserting into SQL tables have occurred. For example, when adding a student or teacher record. This function:

Visual basic (Line 685)

```
    Sub SQLAddData_INTERNAL_PROCESS_ErrorSuccessMessage(ShowSuccessErrorMessages As Boolean,
Succeeded As Boolean, SQLTableName As String, NameOfCallingMethod As String, SuccsessIDReturn As
String, ErrorString As String)
        'A fubction to be used to show error/sucess messages. This shoulf only be used by functions
in this module to give response to user

        'Message templates: To show during a sucess or failure of an SQL add attempt
        Dim SQLAddData_SuccessMessageTemplate As String = "<!$green_text$!>Data inserted
successfully! (`{0}`).<!$reset$!> Row added in '{1}' table. New generated ID: {2}"
        Dim SQLAddData_ErrorMessageTemplate As String = "<!$red_text$!>Error with data inserted!
(`{0}`).<!$reset$!> Failed to add row in '{1}' table. Error: {2}"

        If ShowSuccessErrorMessages = True Then 'Function is responsible/expected to show error
messages, as aposed to the calling block

            If Succeeded = True Then
                HUI.Page.WriteLine(String.Format(SQLAddData_SuccessMessageTemplate,
NameOfCallingMethod, SQLTableName, SuccsessIDReturn))
            Else
                HUI.Page.WriteLine(String.Format(SQLAddData_ErrorMessageTemplate,
NameOfCallingMethod, SQLTableName, ErrorString))
            End If
        End If

    End Sub
```

It's called for example by when adding a teacher to the database:

Visual basic (Line 910)

```
SQLAddData_INTERNAL_PROCESS_ErrorSuccessMessage(ShowSuccessErrorMessages, Succeeded, SQLTableName,
MethodBase.GetCurrentMethod().Name, FunctionReturnID, ErrorString)
```

The parameters passed through are: `ShowSuccessErrorMessages` is a boolean that determines if the function should print messages to the console, or if the original calling function/sub will handle error/success messages. `Succeeded` is a boolean which is true when succeeded and false when an attempt fails. `SQLTableName` is the table in which data was attempted to be added to. `MethodBase.GetCurrentMethod().Name` is a custom class library method that outputs the

current function it is in, for example, when the `SQLAddData_INTERNAL_PROCESS_ErrorSuccessMessage` function/sub is called by `SQLAddData_Teacher` the code `MethodBase.GetCurrentMethod().Name` will output the string of the current sub/function, in this case “SQLAddData_Teacher”. `FunctionReturnID` is the ID of the newly created row, if the query was successful. `ErrorString` shows any error messages, if no errors have occurred when trying to add a record, it will be an empty string.

Simply by using a subroutine to manage showing messages to the console, it means less code repetition needs to occur and if I want to make an edit on how a message is shown I can do so once and have it apply for all SQL queries.

Date input, formatted in correct manner

When taking date data from the user, it’s important that the provided date is in the correct format, this ensures that the program reads the date correctly, the user hasn't made a mistake in formatting, and that the value is saved correctly in the SQL database. This is used to validate the DOB’s entered by the user for teachers, students, and for class attendance dates.

Visual basic (Line 665)

```
    Sub SQLAddData_INTERNAL_PROCESS_ErrorSuccessMessage(ShowSuccessErrorMessages As Boolean,
Succeeded As Boolean, SQLTableName As String, NameOfCallingMethod As String, SuccsessIDReturn As
String, ErrorString As String)
        'A fubction to be used to show error/sucess messages. This shoulf only be used by functions
in this module to give response to user

        'Message templates: To show during a sucess or failure of an SQL add attempt
        Dim SQLAddData_SuccessMessageTemplate As String = "<!$green_text$!>Data inserted
successfully! (`{0}`).<!$reset$!> Row added in '{1}' table. New generated ID: {2}"
        Dim SQLAddData_ErrorMessageTemplate As String = "<!$red_text$!>Error with data inserted!
(`{0}`).<!$reset$!> Failed to add row in '{1}' table. Error: {2}"

        If ShowSuccessErrorMessages = True Then 'Function is responsible/expected to show error
messages, as aposed to the calling block

            If Succeeded = True Then
                HUI.Page.WriteLine(String.Format(SQLAddData_SuccessMessageTemplate,
NameOfCallingMethod, SQLTableName, SuccsessIDReturn))
            Else
                HUI.Page.WriteLine(String.Format(SQLAddData_ErrorMessageTemplate,
NameOfCallingMethod, SQLTableName, ErrorString))
            End If
        End If

    End Sub
```

This code defines a function called `Function_ValidateDate_ddmmyyyy` that validates whether a given date string is in the format "dd/mm/yyyy" and returns a boolean value indicating its validity. Here's how it works:

1. **Parameters:** The function takes a single parameter `dateString`, which is a string representing a date in the "dd/mm/yyyy" format.

2. **Validation Process:**

a. It initialises a `CultureInfo` object with the culture set to "en-GB" (English - United Kingdom). This culture is chosen because it uses the "dd/mm/yyyy" date format.
b. It defines an array `formats()` containing a single format string "dd/MM/yyyy", which specifies the expected format of the date string.
c. It declares a `DateTime` variable `dateValue` which will hold the parsed date if the parsing is successful.
d. It then uses the `DateTime.TryParseExact` method to attempt to parse the `dateString` using the specified format and culture. This method returns `True` if the parsing succeeds and stores the parsed date in the `dateValue` variable; otherwise, it returns `False`.

3. **Return Value:** The function returns the result of the `DateTime.TryParseExact` method, which indicates whether the parsing of the date string was successful (`True` for valid format, `False` for invalid format).

The function checks if a given date string is in the format "dd/mm/yyyy" by attempting to parse it using the specified format and culture. If the parsing succeeds, it returns `True`, indicating that the date string is in the valid format; otherwise, it returns `False`.

Day of the week validation

The day of the week is stored as an integer in the SQL table. It's used for the ClassSchedule table to hold the value of when a scheduled class is.

Visual basic (Line 676)

```
Function IsValidDayOfWeek(dayOfWeekInput As String, ByRef dayOfWeek As DayOfWeek) As Boolean
    ' Function to validate the day of the week input

    Dim result As Boolean = [Enum].TryParse(Of DayOfWeek)(dayOfWeekInput, True, dayOfWeek)
    Return result
End Function
```

1. **Parameters:**
    a. dayOfWeekInput: A string representing a day of the week, like "Monday", "Tuesday", etc.
    b. dayOfWeek: A variable that will hold the corresponding DayOfWeek value if the input is valid.

2. **Validation Process:**
    a. It attempts to parse the dayOfWeekInput string into a DayOfWeek enumeration value.
    b. If parsing succeeds, it returns True, and the parsed DayOfWeek value is stored in the dayOfWeek variable.
    c. If parsing fails (i.e., the input string does not represent a valid day of the week), it returns False.

3. **Return Value:** True if the input is a valid day of the week and False otherwise.

The function validates whether a given string input represents a valid day of the week by attempting to parse it into a DayOfWeek enumeration value. If parsing succeeds, it returns True, and the parsed DayOfWeek value is stored in the dayOfWeek parameter; otherwise, it returns False.

Verification of a record existing for an ID

The day of the week is stored as an integer in the SQL table. It's used for the ClassSchedule table to hold the value of when a scheduled class is.

Visual basic (Line 4743 page: 127)

```
                While True
                    While True

                        HUI.Page.Write($"Enter in the class ID: ")
                        Dim UserInput_ClassID As String = Console.ReadLine

                        If Integer.TryParse(UserInput_ClassID,
AttendanceRecordDetails.ClassCourcse.ID) Then
                            Exit While
                        Else
                            HUI.Page.WriteLine("<!$red_text$!>'" & UserInput_ClassID & "' is not a
valid ID. Enter a <!$reset$!><!$red_background$!> NUMBER ID <!$reset$!><!$red_text$!>.")
                        End If

                    End While

                    AttendanceRecordDetails.ClassCourcse = Function_GetClassInfo(connectionString,
AttendanceRecordDetails.ClassCourcse.ID)
                    If AttendanceRecordDetails.ClassCourcse Is Nothing Or
AttendanceRecordDetails.ClassCourcse.ID = 0 Or AttendanceRecordDetails.ClassCourcse.Teacher Is
Nothing Or AttendanceRecordDetails.ClassCourcse.Teacher.ID = 0 Then
                        HUI.Page.WriteLine("<!$red_text$!>'" &
AttendanceRecordDetails.ClassCourcse.ID & "' is not a valid ID, or could not be found. Enter a
<!$reset$!><!$red_background$!> NUMBER ID <!$reset$!><!$red_text$!>.")
                    Else
                        Exit While
                    End If

                End While
```

Ths code snippet focuses on validating a class ID entered by the user and ensuring that it corresponds to an existing record in the SQL database.

1. **User Input Validation:**
    a. The code uses nested While True loops to repeatedly prompt the user to enter a class ID until a valid one is provided.
    b. Within the inner nested loop, it displays a prompt asking the user to enter the class ID (UserInput_ClassID).
2. **Validation Logic:**
    a. It attempts to parse the user input (UserInput_ClassID) into an integer using Integer.TryParse.
    b. If parsing is successful (TryParse returns True), it assigns the parsed value to AttendanceRecordDetails.ClassCourcse.ID (as an integer field representing the class ID) and exits the inner loop.

c. If parsing fails (TryParse returns False), it displays an error message indicating that the entered value is not a valid ID, prompting the user to enter a valid numeric ID.

3. **Database Lookup:**
    a. After obtaining a valid class ID from the user, it calls Function_GetClassInfo (function that retrieves class information from the database) to fetch details about the class corresponding to the provided ID.
    b. If the returned AttendanceRecordDetails.ClassCourcse object is Nothing or has an ID of 0, or if the associated teacher information is missing, it indicates that the provided class ID does not exist in the database or is invalid. It displays an appropriate error message.
    c. If the class information is successfully retrieved and validated, the outer loop is exited, and the code proceeds with the validated class information.

This ensures that the user-entered class ID is valid and corresponds to an existing record in the SQL database by repeatedly prompting the user until a valid ID is provided and performing a database lookup to verify the existence of the provided ID. If the ID is not valid or does not exist in the database, appropriate error messages are displayed to the user.

SMS message sender, notifying student absences by use of Automate, HTTP, APIs, JSON and web requests

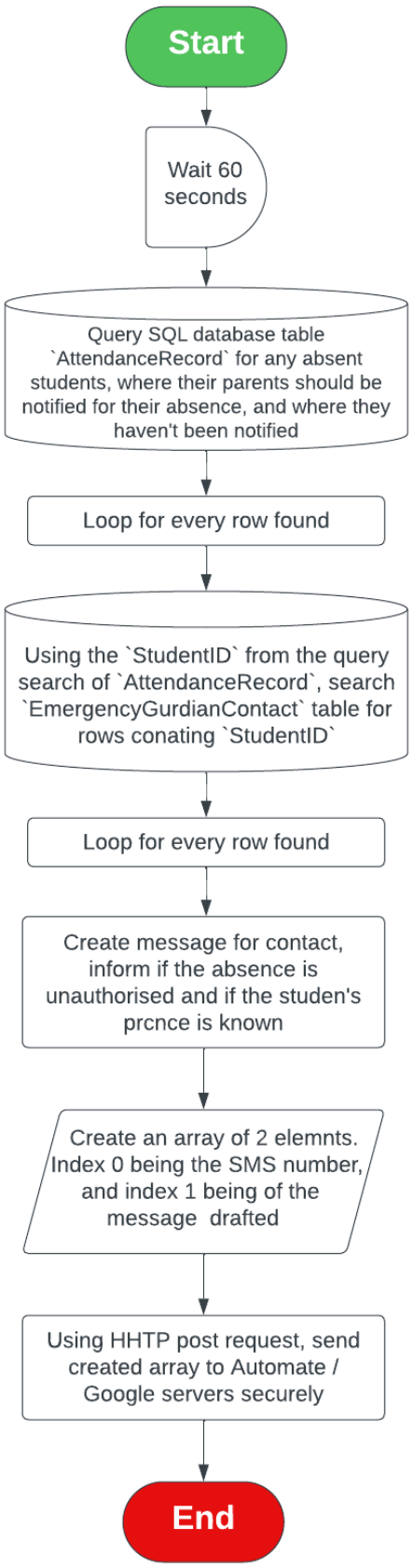
The client, Jeni, said that she'd like to have the ability to notify the student's parents if their child has been marked absent for them. To create this solution, I've created a separate program called `SchoolRegisterSystemBackEnd`. This program stays open in the background and runs on the main server/computer hosting the SQL database.

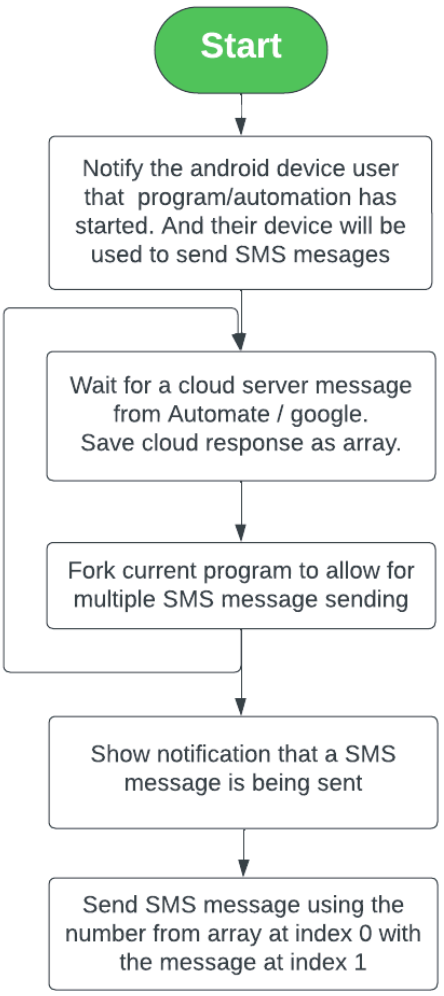
The program will repeatedly check the `AttendanceRecord` table in the SQL table every 60 seconds for new records of student attendances where the student is marked absent, the students' contact should be notified, and they haven't been notified yet. It will then find that student's emergency / guardian contacts and send them an SMS message to their number, telling them: their child missed a lesson, if the teacher knows why the student is absent, and if the absence has been authorised.

This SMS message sending capability is made possible by the android app "Automate". It allows me to send a HTTP JSON payload post request to a server using google account authentication, this data message is then sent to my Android device running the Automate app, which then reads the data as an array, where index 0 is the number and index 1 is the message to send. Using the `SMS message` block, I can send these messages. The function/automation I created also shows a notification to that current Android device that it is being used to send SMS messages.

Using this method of sending an SMS to parents means that school won't need to pay for an online SMS message sending service, which can become expensive.

Flowchart:





Left: School server **Right: Android device**

The code solution for the school server:

Visual basic (Line 433, Different document, page: 11)

```
    'Send Data SMS
    Function SendData(ByVal VARIABLE1 As String, ByVal VARIABLE2 As String) As String
        Dim url As String = "https://llamalab.com/automate/cloud/message"
        Dim secret As String = "--REMOVED FOR PRIVACY AND SECURITY--"
        Dim email As String = "--REMOVED FOR PRIVACY AND SECURITY--"
        Dim device As String = "GOOGLE Pixel 6"
        Dim priority As String = "normal"

        Dim jsonPayload As String = "{""secret"":""" & secret & """,""to"":""" & email &
""",""device"":""" & device & """,""priority"":""" & priority & """,""payload"":[""" & VARIABLE1 &
""","""" & VARIABLE2 & """]}"""

        Dim request As WebRequest = WebRequest.Create(url)
        request.Method = "POST"
        request.ContentType = "application/json"

        Dim data As Byte() = Encoding.UTF8.GetBytes(jsonPayload)
        request.ContentLength = data.Length

        Dim stream As Stream = request.GetRequestStream()
        stream.Write(data, 0, data.Length)
        stream.Close()

        Dim response As WebResponse = request.GetResponse()
        Dim responseStream As Stream = response.GetResponseStream()
        Dim reader As New StreamReader(responseStream)
        Dim responseString As String = reader.ReadToEnd()

        reader.Close()
        responseStream.Close()
        response.Close()

        Return responseString
    End Function
```

This code defines a function named SendData that sends data via SMS using an HTTP request to an Automate server.

1. **Parameters:**
    a. VARIABLE1 and VARIABLE2: These are strings containing the data to be sent via SMS.
2. **URL and Authentication:**
    a. It defines the URL where the HTTP request will be sent (url variable).
    b. It also defines authentication details such as a google account secret key (secret variable) and a google email address (email variable).
3. **Payload Creation:**
    a. It constructs a JSON payload containing the secret key, google account email, device name, message priority, and the data to be sent (VARIABLE1 and VARIABLE2).
    b. The JSON payload is constructed as a string (jsonPayload variable).
4. **HTTP Request Setup:**
    a. It creates a WebRequest object with the specified URL.
    b. It sets the request method to POST and content type to "application/json".

c. It converts the JSON payload string into bytes and sets the request's content length.

5. **Sending the Request:**
    a. It obtains a stream from the request (stream variable) and writes the JSON payload data to it.
    b. It closes the stream after writing.
6. **Handling the Response:**
    a. It sends the HTTP request using request.GetResponse().
    b. It reads the response stream (responseStream) and converts it to a string (responseString) using a StreamReader.
    c. It closes the reader and response stream after reading.
    d. It returns the response string.
7. **Return Value:**
    a. The function returns the response received from the server, which may contain information about the success or failure of the SMS sending process.

The function sends data via SMS by constructing a JSON payload containing the necessary information, sending an HTTP POST request to the specified URL with the payload, and returning the response received from the server.

Visual basic (Line 477, Different document, page: 12)

```
    Sub Main()
        SendData("--REMOVED FOR PRIVACY AND SECURITY--", ("School Register system messaging
service:" & NewLine & "You may recive contious messages from this sytem.").ToString) ' Send SMS
message to android device, mesging itself informing the user that it is being used

        Dim WaitTime As Integer = 60000 ' Loop every 60 seconds
        Dim SecondsCount As Integer = 0
        While True
            Thread.Sleep(WaitTime)
            Console.WriteLine("")
            Console.WriteLine(SecondsCount & ": Starting & searching: ")
            ReadAndUpdateAttendance(connectionString)
            SecondsCount += WaitTime
        End While
    End Sub
```

This is the entrance into the program, this sends an SMS message to the android device from itself, notifying the user that their device will be used by the system to send messages. It also creates a loop, with a 60 second delay to continuously look for absent students.
It then calls the `ReadAndUpdateAttendance` subroutine.

Visual basic (Line 491, Different document, page: 13)

```
    Sub ReadAndUpdateAttendance(connectionString As String)
        Dim query As String = "SELECT ID, StudentID, ClassCourseID, ClassScheduleID, DateRecordDate,
StartTime, EndTime, KnownReasonOfAbsence, AuthorisedAbsence FROM AttendanceRecord WHERE
StudentIsPresent = 0 AND NotifyGuardian = 1 AND GuardiansNotified = 0"

        Using connection As New SqlConnection(connectionString)
            connection.Open()
```

```
            Using command As New SqlCommand(query, connection)
                Using reader As SqlDataReader = command.ExecuteReader()
                    While reader.Read()
                        Dim recordId As Integer = reader.GetInt32("ID")
                        Dim studentId As Integer = reader.GetInt32("StudentID")
                        Dim ClassCourseID As Integer = reader.GetInt32("ClassCourseID")
                        Dim ClassScheduleID As Integer = reader.GetInt32("ClassScheduleID")
                        Dim DateRecordDate As Date = reader("DateRecordDate")
                        Dim StartTime As TimeSpan = reader("StartTime")
                        Dim EndTime As TimeSpan = reader("EndTime")
                        Dim KnownReasonOfAbsence As Boolean =
reader.GetBoolean("KnownReasonOfAbsence")
                        Dim AuthorisedAbsence As Boolean = reader.GetBoolean("AuthorisedAbsence")

                        ProcessStudentAbsence(recordId, studentId, ClassCourseID, ClassScheduleID,
DateRecordDate, StartTime, EndTime, KnownReasonOfAbsence, AuthorisedAbsence)
                        UpdateGuardiansNotified(connectionString, recordId)
                    End While
                End Using
            End Using
        End Using
    End Sub
```

This sub reads the attendance record of the students who are marked absent, parents/emergency/guardian contacts need to be notified and haven't been notified yet. It retires these rows and loops for each row, it then calls `ProcessStudentAbsence()` and ` UpdateGuardiansNotified()` passing in their parameters

Visual basic (Line 517, Different document, page: 13)

```
    Sub ProcessStudentAbsence(recordId As Integer, studentId As Integer, ClassCourseID As Integer,
ClassScheduleID As Integer, DateRecordDate As Date, StartTime As TimeSpan, EndTime As TimeSpan,
KnownReasonOfAbsence As Boolean, AuthorisedAbsence As Boolean)

        ' Get students emergency list contact numbers
        Dim ArrayOfEmergencyGurdianContactInfo() As EmergencyGurdianContact =
Function_GetEmergencyGuardianInfo(connectionString, studentId)

        Dim StudentsInfo As Student = Function_GetStudentInfo(connectionString, studentId)

        Dim ClassCourseInfo As ClassCourcse = Function_GetClassInfo(connectionString, ClassCourseID)

        For i As Integer = 0 To ArrayOfEmergencyGurdianContactInfo.Length - 1
            Dim SMSMessageString As String = "Dear " & ArrayOfEmergencyGurdianContactInfo(i).Title &
" " & ArrayOfEmergencyGurdianContactInfo(i).FirstName & " " &
ArrayOfEmergencyGurdianContactInfo(i).MiddleName & " " &
ArrayOfEmergencyGurdianContactInfo(i).LastName
            SMSMessageString += NewLine & StudentsInfo.FirstName & " " & StudentsInfo.LastName & "
has been marked absent for their lesson "
            SMSMessageString += ClassCourseInfo.ClassName & " (" & ClassCourseInfo.Subject & ") at "
& StartTime.ToString & " - " & EndTime.ToString & ", " & DateRecordDate.ToLongDateString & "."

            If AuthorisedAbsence = True Then
                SMSMessageString += NewLine & "This absence has been authorised by us"
            Else
                SMSMessageString += NewLine & "This absence has not been authorised by us"
            End If

            If KnownReasonOfAbsence = True Then
                SMSMessageString += NewLine & "Reason of absence was given for " &
```

```
StudentsInfo.FirstName & "s absence to lesson"
            Else
                SMSMessageString += NewLine & "No reason was give for " & StudentsInfo.FirstName &
"'s absence to lesson"
            End If

            'Who to message:
            Dim PhoneNumber As String = ArrayOfEmergencyGurdianContactInfo(i).PhoneNumber

            Console.WriteLine("")
            Console.WriteLine("SMS Message to number: " & PhoneNumber)
            Console.WriteLine("Message: " & SMSMessageString)
            SendData(PhoneNumber, SMSMessageString.ToString) 'HARD CODED NUMBER FOR TESTING. replace
number wirh `PhoneNumber`
            Console.WriteLine("Message sent.")


        Next

    End Sub
```

In this sub, it fist calls `Function_GetEmergencyGuardianInfo` to get an array of the student's contacts, then it calls `Function_GetStudentInfo` to retrieve the students information such as their name, and then it calls `Function_GetClassInfo` to get information on the class they've missed.

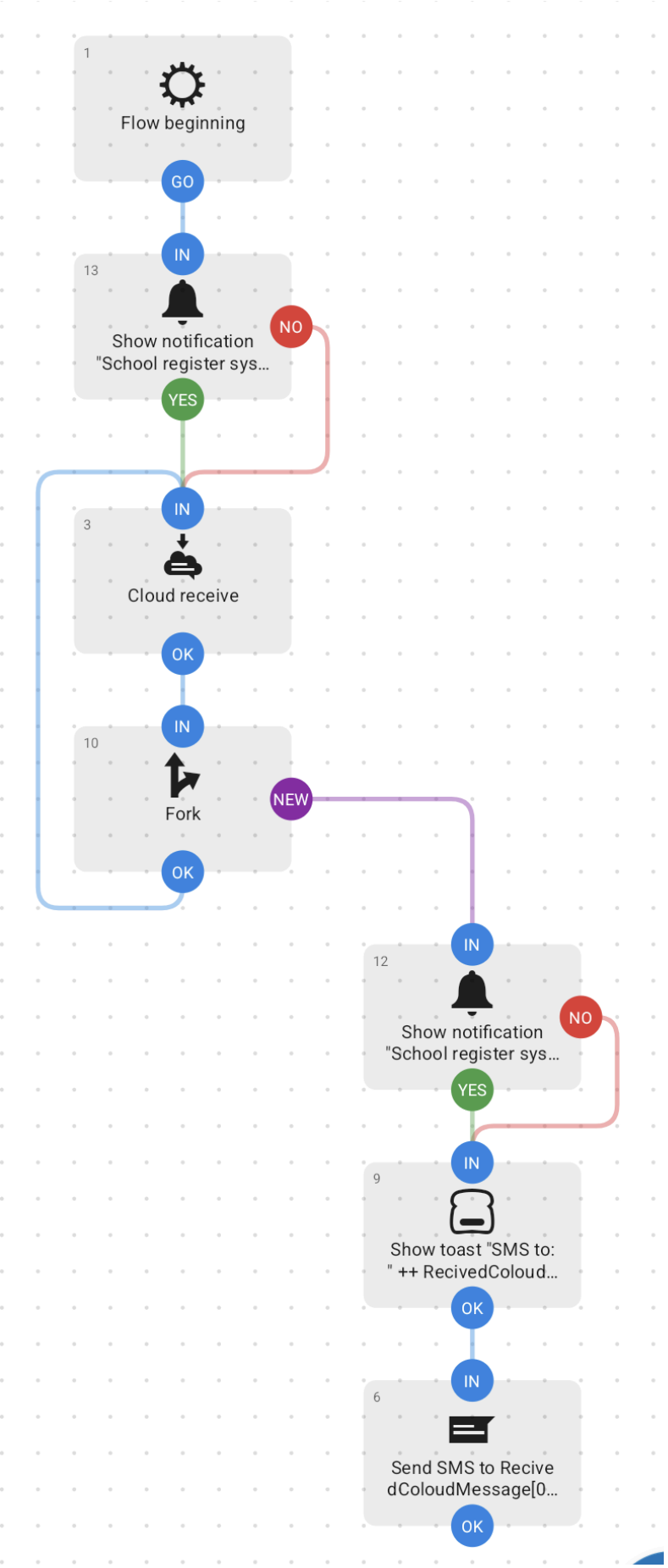
It then loops for each of the emergency guardian contacts the student has, creates a SMS message content using the information queried, and provides information about the absence. It then calls `SendData`, the function to send the SMS message passing in the number and message.

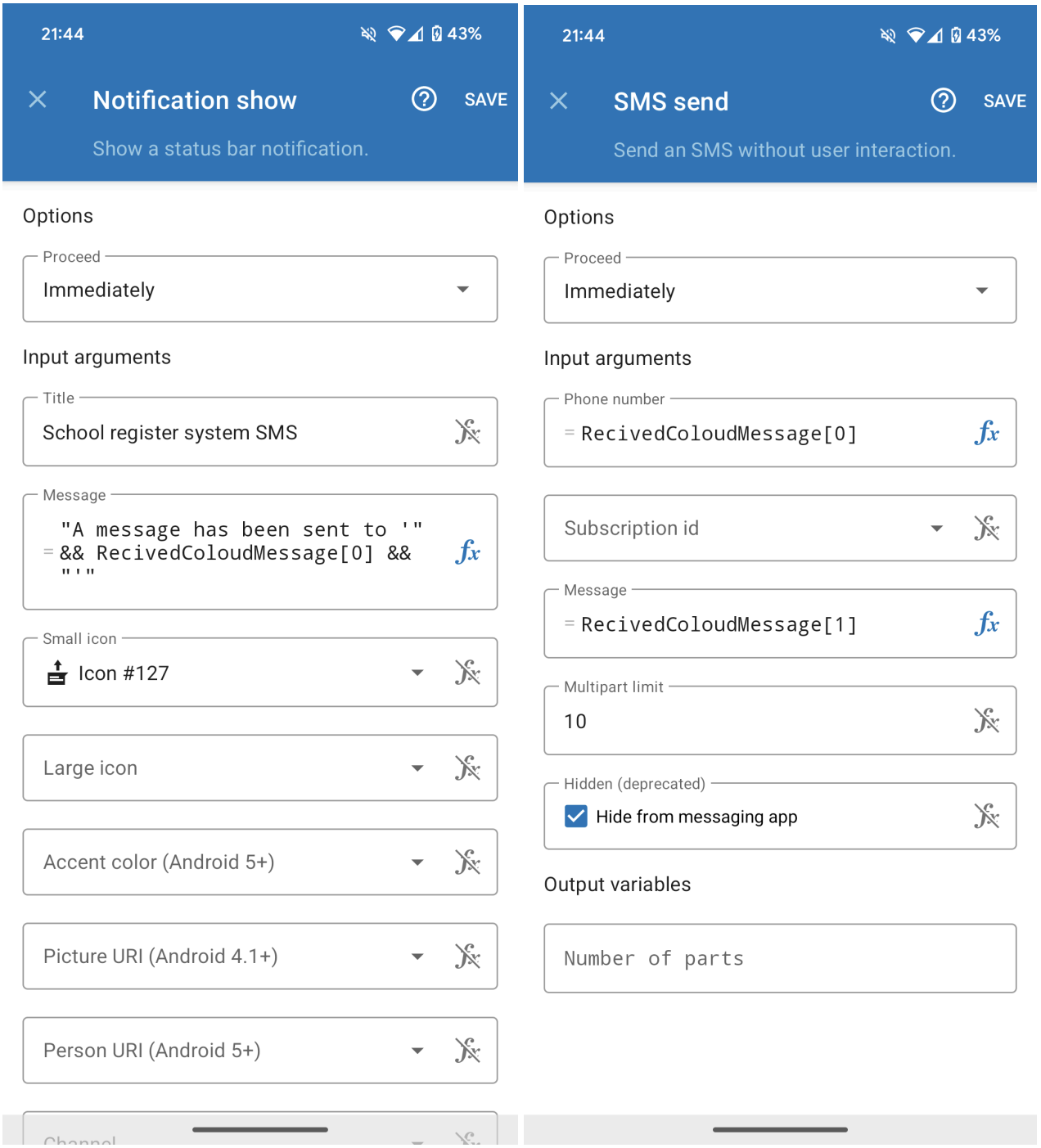
Visual basic (Line 557, Different document, page: 15)

```
    Sub UpdateGuardiansNotified(connectionString As String, recordId As Integer)
        Dim updateQuery As String = "UPDATE AttendanceRecord SET GuardiansNotified = 1 WHERE ID =
@RecordID"

        Using connection As New SqlConnection(connectionString)
            connection.Open()
            Using command As New SqlCommand(updateQuery, connection)
                command.Parameters.AddWithValue("@RecordID", recordId)
                command.ExecuteNonQuery()
            End Using
        End Using
    End Sub
```

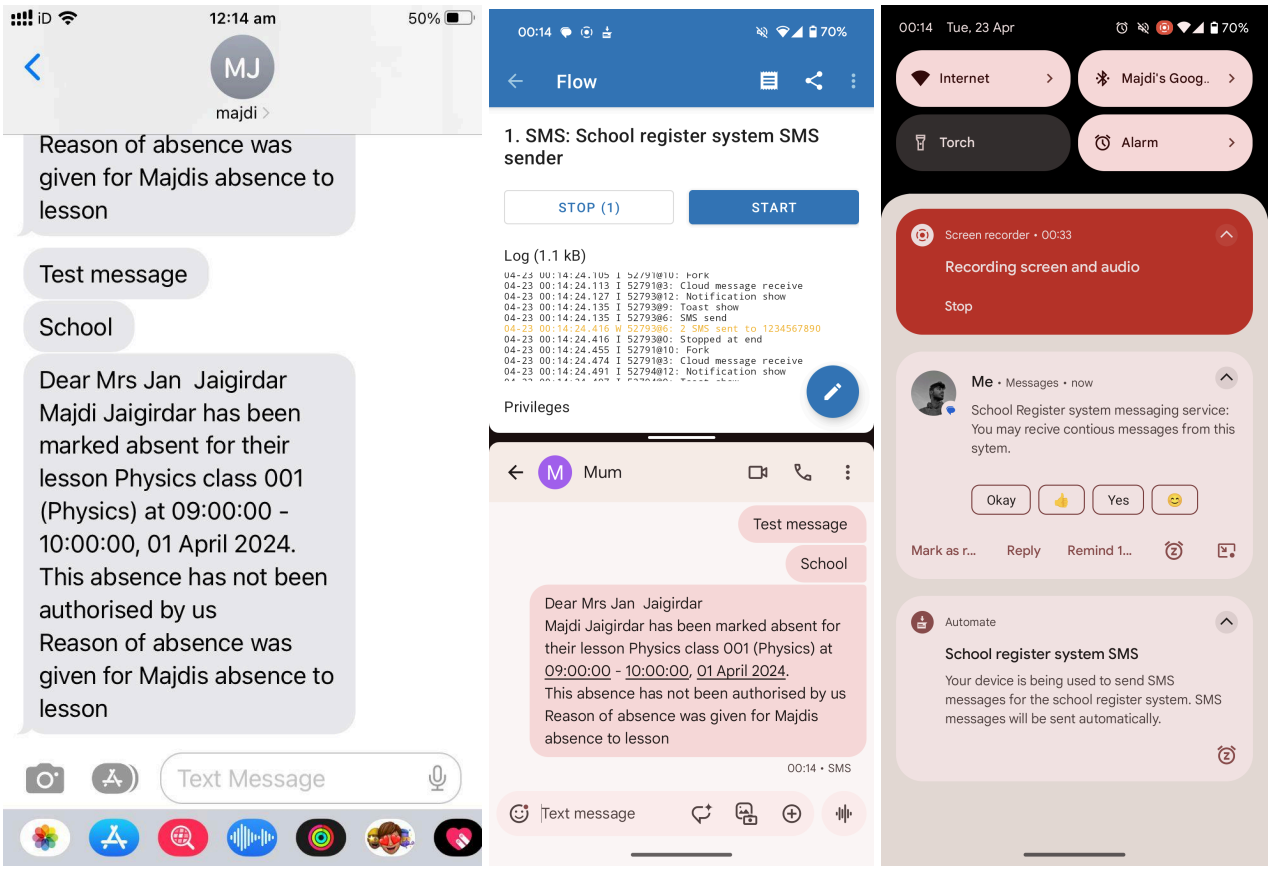
Finally, the program edits the attendance row for that student who was absent, recording that the student's parents/emergency/guardian contacts have been notified.

The Automate solution for the android device screenshot:


1
Flow beginning
GO
IN
13
Show notification "School register sys...
NO
YES
IN
3
Cloud receive
OK
IN
10
Fork
NEW
OK
IN
12
Show notification "School register sys...
NO
YES
IN
9
Show toast "SMS to: " ++ RecivedColoud...
OK
IN
6
Send SMS to Recive dColoudMessage[0...
OK



Evidence of code function on receiver's device and android sender device:


iD
12:14 am
50%
MJ
majdi
Reason of absence was given for Majdis absence to lesson
Test message
School
Dear Mrs Jan Jaigirdar
Majdi Jaigirdar has been marked absent for their lesson Physics class 001 (Physics) at 09:00:00 - 10:00:00, 01 April 2024.
This absence has not been authorised by us
Reason of absence was given for Majdis absence to lesson
Text Message
00:14
70%
Flow
1. SMS: School register system SMS sender
STOP (1)
START
Log (1.1 kB)
Privileges
Mum
Test message
School
Dear Mrs Jan Jaigirdar
Majdi Jaigirdar has been marked absent for their lesson Physics class 001 (Physics) at 09:00:00 - 10:00:00, 01 April 2024.
This absence has not been authorised by us
Reason of absence was given for Majdis absence to lesson
00:14 • SMS
Text message
00:14 Tue, 23 Apr
70%
Internet
Majdi's Goog..
Torch
Alarm
Screen recorder • 00:33
Recording screen and audio
Stop
Me • Messages • now
School Register system messaging service: You may recive contious messages from this sytem.
Okay
Yes
Mark as r...
Reply
Remind 1...
Automate
School register system SMS
Your device is being used to send SMS messages for the school register system. SMS messages will be sent automatically.

Testing and evidence of function for the program

#	Function, title and description	Proof
	Secure password logon & secret input function Requires users to securely logon using a password to access the program and database. Entering a wrong password will deny you access and make you reenter it. Entering the correct password will load the program and give access to the user to the system. In this video I show: 1. Attempting to logon with the wrong passwords 2. How the program deals with delete keys 3. How the program deals with enter key 4. How the program hides the password after entering a password 5. Logging on with the correct password (in this demo `Password123`)	https://www.loom.com/share/f8080c7e7d5748aeaf7216edf241e007?sid=98629b0c-5529-4204-904d-3bae64814870
	HUI: Pages for program, next and back Created my own form of UI using object oriented programing, custom classes, custom data types and using stacks. Allows users to have a better UI experience by focusing on a function of a program and clearing cluttered console text. Allowing users to go forward and back on pages. It shows the page navigation paths `Main menu > Student records manager > ...`. In this video I show: 1. User navigating the different pages 2. Page being created with my custom created c`HUI` classes 3. The pages being pushed and popped off the stack 4. The header of each page containing the current page title, layer, the page's path, and possible navigations	https://www.loom.com/share/2a908918dc254a7e9f54b5702f70698b?sid=9649a569-c3e4-4b7f-b8fb-5d4cba402eb2
	HUI: Output colourised text and formatting to console output Using object oriented programing, iteration, arrays, string tags outputting text with background and console colour is made possible and is made much more flexible and understandable compared to VB methods of colouring text. This again improves users experience using the program, makes it more easier to use and understand the program. It can be used to highlight and bring focus to more important elements of the program. In these photos I show: 1. How the colourise text looks on the console 2. The different use cases of colourised text on the console	On page 47 - 48 of this document
	SQL: Creation of student record In this video I show: 1. The secure logon before accessing the program/database 2. The student records before adding example student 'Max' 3. The data collecting process for a student 4. The data collecting process for their guardians 5. The data collecting process for both their addresses 6. The data confirmation 7. The sql database featuring the newly added student record, emergency contact and their address	https://www.loom.com/share/77e904ad27664ba39a7b5c5b1c45eedf?sid=b9a71bc2-fcc1-4483-933d-78865cf522a3

	SQL:Creation of teacher record In this video I show: 1. The teacher records before adding example teacher 'David' 2. The data collecting process for a teacher 3. The data collecting process for their addresses 4. The data confirmation 5. The sql database featuring the newly added teacher record and their address	https://www.loom.com/share/4c47735ceb7644918c2f3912fbcf7d20?sid=73823b7c-1b97-49c0-9780-923668f727bd
	SQL: View all and deletion of a teacher record In this video I show: 1. SQL database 2. Logon 3. View all teacher records 4. List possible teachers to delete 5. Delete a teacher record	https://www.loom.com/share/91ed7ca63d2840218f83115983d2496f
	SQL:Creation of Class record In this video I show: 1. The Class records before adding example class English Eng01 2. The data collecting process for a new class 3. The ability to see the current list of teachers in the database available to assign to this class (featuring the use case of pages, where a list of teachers page can be closed and opened) 4. The assignment of a teacher to a class 5. The data confirmation 6. The sql database featuring the newly added class record	https://www.loom.com/share/56fafb5b239b446b90a9e42fb1b253c5?sid=e44379b5-e561-4bdd-9c3b-161ec7eddf79
	SQL:Enrollment of students to a class In this video I show: 1. The Class enrolments table before adding students to a class 2. THe selection of a class to enrol students to 3. The Enrollment of students to a class 4. Conformation of the class and students to enrol	https://www.loom.com/share/ca08cc49b48541efb34b7ec131ae749b?sid=228ae2c1-0a19-4196-b839-3ebda80b52c6
	SQL: Scheduling lesson times for a class In this video I show: 1. Lesson schedule table before addition of new schedule 2. Creating a lesson schedule 3. The database table showing that new schedule	https://www.loom.com/share/c0a077debcc245c6a284161f256b9122?sid=c370d819-87e1-4b3a-8a29-7237bfc557d5
	SQL: Preventing lesson overlap scheduling In this video I show: 1. Attempting to create a lesson over an existing lesson - and being denied as it would cause an overlap 2. Being denied from creating that overlap lesson	https://www.loom.com/share/5395d71fcc434d63996215a89a6d25a3?sid=e8a05659-1dd7-4afe-8bcf-e1fc7ac46820
	SQL: Taking and adding a students attendance In this video I show:	https://www.loom.com/share/b723dd60d5c6406

	1. Take an attendance of class 2. Viewing a list of possible class lessons to take addance for 3. Listening the students to mark 4. Adding details of a student absence	0b92ab951b8f3b138
	SQL: View student attendance for all classes In this video I show: 1. Viewing a student's attendance for all classes	https://www.loom.com/share/f8da035398ab4e4ea39ec96740518e16?sid=b457c90e-c180-496b-a5b6-c5203947c393
	SMS messaging In this first I show the starting of the SMS backend program on the server side. The program reads the student records, and then sends the data to my device. In the second video I show the start of the automation, the program waiting for a cloud message data to be sent to it. Once received,it sends a text message to contact.	https://www.loom.com/share/3d96149b49e44412921c20ef94d54fa2?sid=8028a978-e998-4712-b791-44cbcab9788e https://drive.google.com/file/d/1uCidMCNIGaDWBJ0m2fBXJSwnDwzTQUYp/view?usp=drivesdk Screenshots on page 75 - 77 of this document
	Data validation: Password	https://www.loom.com/share/6efc4f4fe5b64df09ab0f9d28a34099d
	Data validation: Menu options	https://www.loom.com/share/2496452ceb7d463492516c5f4b8bf3c2
	Data validation: date	https://www.loom.com/share/1784ecd675774c6ab4e1ac49a955d26c

Evaluation

Client feedback

Jeni

Hi Majdi,

I have had the opportunity to explore the completed school register system that you have developed, and I must say that overall, I am truly impressed with the solution you have provided. The system offers a range of features that cater to the specific needs of our school and have the potential to greatly benefit our educational processes. Here is a feedback based on my experience using the system:

1. Database System: I appreciate the choice of Microsoft SQL Express as the database system for its lightweight nature, ease of setup, and cost-effectiveness. This choice ensures that the system runs efficiently on our lower-end laptops, enhancing performance.

2. User Authentication: The logon system for accessing the school register is a critical security feature that safeguards our students' and staff data. However, having a separate login for each teacher or user would further enhance user management and data security.

3. SMS Notification Feature: The SMS notification functionality is a standout feature of the system. It effectively notifies parents of student absences without requiring additional costs for an external service, making communication with parents more accessible and efficient.

4. Input Validation and User Guidance: The system's validation of user inputs, verification of existing records, and provision of hints for possible inputs are valuable elements that enhance user experience and data accuracy.

While the system offers numerous advantages, there are a few areas where potential improvements could be considered:

1. Different User Logons: Implementing individual logins for each teacher or user would enhance user-specific access control and privacy, facilitating better user management within the system.

2. Record Editing Functionality: Providing the ability to edit existing records rather than having to delete and recreate them would streamline data management processes and ensure data accuracy.

3. Advanced Student Records Statistics: Enhancing the system to provide more in-depth student advance records statistics would allow for better analysis and tracking of student performance and attendance patterns.

In conclusion, I am grateful for the effort and dedication you put into developing this school register system, which has the potential to transform our school's administrative processes. The system's user-friendly interface, security features, and communication capabilities make it a valuable asset for improving efficiency and communication within our educational environment.

Best regards,
Jeni

Review on objectives

1 - Must Have 2 - Should have 3 - Could have 4 - Won't have

Fully achieved	Partly achieved	Barely achieved	Not achieved

Priority	Objectives
1	1. Use a database system appropriate for my client and software.
1	a. Integrate a system that can easily be accessed by my school register system client software
1	b. Use a database system that can be accessed by other programs and built up on by schools independently to make maintenance and data processing easier
1	c. A system that is lightweight on lower end hardware and can be setup on most various systems with ease
1	d. Use a system that can be externally managed outside my own program so schools have more control and flexibility with their data
1	e. Secure system that will protect sensitive student, teachers and other school related records
2	f. A solution that will allow for multiple concurrent users to access the database.
2	g. Create a backup system of that database to secure backup data from the database locally and on the cloud.
1	2. Create a lightweight client program for the school register system for staff
1	a. Make the client be able to run on lower end, older devices on various platforms and various environments
2	b. A client side computer that doesn't relay on a powerful competitive central computer to do must of the processing
1	c. A secure system that requires users to enter the program with a log in
3	d. A secure logon with multiple accounts giving access / privileges to different users
3	3. Create a lightweight client portal for students / guardians use
3	a. Create a secure login system for students/guardians to access the system
3	b. Allow students/guardians to view their own student details
3	c. Show students/guardians their classes
3	d. Show students/guardians their schedules for their classes
3	e. Allow students/guardians to create requests for authorised absence with authorisation for school and guardians
3	f. Allow students / guardians edit their respective records
1	4. General ability of a functioning school register system
1	a. Be able to mark a student present / absent for a specific class
1	b. Be able to see the statistics of a student's presence / absences for a class
1	c. Be able to see the statistics of a specific student across multiple classes
1	d. Be able to see attendance statistics of students for a singular class
3	e. Remind teachers to take attendance with notifications

1	5. Student records
1	a. Create records of students: name, address, DOB, etc
2	b. Creation of multiple parental / emergency / guardian contacts for students storing address, contact, names etc
1	c. View all student records in the system
1	d. Edit student records
1	e. Delete parental / emergency / guardian contacts for student
1	f. Delete student records and associated data
1	6. Teacher records
1	a. Create records of teachers: Name: address, DOB, etc
1	b. View all teacher records
1	c. Edit a teachers record
1	d. Delete a teacher record and associated data
1	7. Class records
1	a. Create a class in the system
1	b. Assign a teacher to it
1	c. Enrol students into that class
1	d. Schedule times for that class
2	e. Prevent schedule overlaps
1	f. Delete schedules for that class
1	g. Delete class and associated data
2	8. Exam records
2	a. Keep track of students exams for various classes and subjects
2	b. Create exam records
2	c. View exam records
2	d. Show overview of exam results
3	e. Use advanced algorithm to estimate students final grade/results
2	f. Delete exam records
1	9. Create a user interface for system
2	a. Make UI that is intuitive and easy to understand for users
3	b. Don't rely on too many visual effects / transitions as program is intended for lower powered hardware
3	c. The UI should provide instructions, tips and hints on how to use it for the user
3	d. Provide user with possible inputs and suggestions
2	e. Provide user with information on input format
3	f. The UI should not be cultured and should have a primary focus on the main subject
3	g. Use a universal UI language to keep it all consistent
3	h. Show statistical information a clean and understandable format
3	10. Create simple/advanced algorithms and use AI/machine learning to predict student stats
2	a. Use simple/advanced algorithms to find out the day a student misses the most
2	b. Use simple/advanced algorithms to find out the times a student misses the most
2	c. Use simple/advanced algorithms to find out the classes a student misses the most
2	d. Analyse the general attendance of the whole school
2	e. Analyse the general attendance of the whole class

2 3 4	f. Analyse the general attendance of the whole year/grade g. Using advanced algorithm and/or machine learning, and using existing attendance/exam data in the system, estimate a students final grade by comparing their exam results and attendance to other students previous them h. Using advanced algorithm and/or machine learning, suggest potential escalating attendance issues that might occur with a student by comparing their attendance patterns to that of other students who are having poor attendance
1 1 2 1 3 3 2	11. Notify parental / emergency / guardian contacts of their child's absences a. Send SMS messages to contacts when their child misses a lesson b. Make the SMS messaging simple and cheap c. Make the transport of data between devices secure and private d. Inform contact of their students' absence, if it's been authorised and if a reason for absence has been given. e. Make SMS sending process simple f. Keep the functionality lightweight and non-resource intensive
3 3 3 3	12. Create algorithm to intelligently figure out the best school time schedule a. The program will look at the current free time slots and figure out the best lesson schedules b. It will use advanced algorithm to find free slots to put lessons in c. It will provide the user with multiple options of school timetables/schedules
1 1 1 3 1 2	13. Maintain student, teacher and other personal privacy, security & safety a. Require logon access b. Use a secure database method c. Hide user password input d. Prevent SQL injection with parameters e. Validate inputs to ensure security error prevention
3 2 3 3 1 2	14. Use of object oriented programing (OOP) a. When appropriate use OOP b. Create custom OOP using VB classes for functionality c. Use OOP rather than repeating code for following best practice, maintenance and decrease in storage d. Create meaningful and appropriate methods e. Use custom data classes when appropriate
1 2 2 2 2 2 1	15. Code in a uniform manner in which allows other 3rd parties to understand my code, program, solution and data processing a. Use appropriate and understandable variable names b. Use appropriate and understandable module, function & subroutine names c. Use appropriate and understandable class and method names d. Comment throughout the program explaining what code does and the current expected process e. Comments should be in a uniform / standardised manner f. Appropriate uses of custom classes

Personal review & evaluation

My process

Overall on this, I believe I did a fantastic job on this solution and project. With detailed analysis, with a clear background on why I was creating this school register system and interviewing a real client who had a real issue, my Cousin a school teacher in a country where technology availability and capabilities aren't on an equal level with the rest of the world, hindering school technological abilities with many things such as taking attendance. While it may be a small aspect of the education process, it plays a big role in ensuring that everyone, every child, is given a free, fair and optimistic education. Especially where, in bangladesh the government is pushing to give young girls a further education so they can obtain higher earning jobs, improve quality of life and deter from marrying young. Giving them the same equality as their male peers.

I created a clear set of goals and objectives, using the ‘MoSCoW` method to identify the most crucial elements of my program, prioritising them. I also broke objectives down into smaller parts, allowing me to create a path to focus on and to create building blocks/components to the larger goal. This method helped me a lot, as it allowed me to focus on each part separately and not become lost in the grander project.

Designing these objectives component by component, and into smaller systems helped me to understand how I was going to structure my program. Noting that I’d use a central computer to run the database, separate client computers to connect to the database and the use of a 3rd party program on android to plug into my system and use it’s web requests/api abilities.

The designing and breaking down of these components also helped me to break down my code into different sections, functions, modules, methods etc. Going over my design I realised using custom classes, data types, object originated programming would become extremely beneficial for my project, so I implemented these into my code. My program turned out to be a very long one, with 5,000+ lines. This was due to two main reasons: A lot of commenting throughout my program, and also because of using OOP, functions, classes, etc.

I commented on my program thoroughly and continuously, and for the most part using a similar unified way of commenting this is on par with the best standards in computer programming. It proved to be useful when there were errors and when looking back on code I haven’t touched for a while - I could read and understand what was past me, coded.

The use of OOP, functions, classes, etc causing my program to be a lot longer may at first be thought to occur because these methods themselves required a lot of lines of coding. However, the actual reason behind this expansion lies in their facilitation of additional functionality within my code. By employing these programming paradigms, I found it easier and quicker to integrate more features into my code. This streamlined the process, reducing the repetition of complex components and encouraging the use of functions and custom methods, ultimately enhancing the overall efficiency and effectiveness of the program. Meaning I could write more with less errors and hardship.

What I learnt and improved on

Through my computer science NEA project, I learned the importance of detailed analysis and clear goal-setting, particularly in addressing real-world problems like the lack of technological resources in education. By applying methodologies such as the MoSCoW method and breaking down objectives into manageable components, I gained a deeper understanding of project structuring and management. Moreover, the extensive use of OOP, functions, and classes not only contributed to the length of my program but also significantly enhanced its functionality and efficiency. Overall, this project sharpened my skills in problem-solving, project planning, and software development, providing valuable lessons that I will carry forward in my future endeavours.

What I could have done better

Overall I think I did great. However I can identify many areas of improvement:
One of the big ones is that I should have fousedon making more complex algorithms, for example, one to figure out and create a school schedule completely automatically.

Another could be more customised security, while my database and program are very secure, and my program requires a password to access the system. Creating each different logon for each teacher could have possibly been a better implementation of the security method. I could have achieved this by when a new teacher record is created, it creates a one time password, and when that teacher logs in, they must change their password. And these details could have been securely saved on the SQL server, encrypting the password and hashing it.

While I did make many of my repetitive code blocks into functions, I believe I could have functionalized more of my procedures, reducing the number of lines of code. Many parts of my code are very similar and utilise the same process, these processes could be simplified.

An additional thing I believe which could have improved the usability of my system would be the ability to edit records. If an administrator wants to update a student's address, they'd have to delete the whole student off the system and their guardians then re-create a record for them all. Which also means having to re enrol them to classes, and losing their attendance data.