

London Metropolitan University, BEng Software Engineering (top-up)

Franchised Delivery at Esoft, Sri Lanka

Coursework Coversheet

Part 2 – Student Feedback

| | |
|---------------------------|-----------------------------|
| Student ID: | Student Name: |
| Module Code: | Module Name: |
| Assignment number: | Esoft Module Leader: |
| Date set: | Date due: |

Strengths (areas with well-developed answers)

Weaknesses (areas with room for improvement)

Additional Comments

Esoft Module Lecturer:

Esoft Module Marker:

Provisional mark as %:

Date marked:

Acknowledgements

This work would not have been possible without the support of all the lecturers and staff at E-Soft Metro Campus, Kandy. I am especially indebted to Mr.Sudesh Bandara, who has been supportive throughout the Application development subject. Most importantly, I wish to thank my parents and sibling for guidance and care. Last but not least, to my friends for unending inspiration.

Table of Contents

Chapter 1

| | |
|--------------------|---|
| Introduction | 1 |
|--------------------|---|

Chapter 2

| | |
|------------------------------|----|
| Design | 2 |
| 02.01 Use Case Diagram..... | 2 |
| 02.02 User Descriptions..... | 3 |
| 02.03 User Stories..... | 11 |
| 02.04 Class Diagram..... | 12 |
| 02.05 ER Diagram..... | 13 |
| 02.06 Data Base..... | 14 |

Chapter 3

| | |
|--|----|
| Development..... | 17 |
| 03.01. User Manual..... | 17 |
| 03.01.01. Splash Screen (Screen 01)..... | 17 |
| 03.01.02. Account Selection Screen (Screen 02)..... | 17 |
| 03.01.03. Login Screen (Screen 03)..... | 18 |
| 03.01.04. Admin Dashboard (Screen 04)..... | 19 |
| 03.01.05. Manage Class Details (Screen 05)..... | 20 |
| 03.01.06. Manage Student Details (Screen 06)..... | 20 |
| 03.01.07. Manage Staff Details (Screen 07)..... | 21 |
| 03.01.08. Manage Course and Subject (Screen 08)..... | 22 |
| 03.01.09. Manage Course and Subject (Screen 09)..... | 22 |
| 03.01.010. Screen Flow for Staff Account Holders (Screen 3 for staff members)..... | 23 |
| 03.01.011. Dashboard for Staff (Screen 4 for staff members)..... | 24 |
| 03.01.012. Map Subject to Class (Screen 5 for staff members)..... | 25 |
| 03.01.013. Map Student to subject (Screen 6 for staff members)..... | 26 |
| 03.01.014. Add Marks (Screen 7 for staff members)..... | 26 |
| 03.01.015. Add Marks (Screen 8 for staff members)..... | 27 |
| 03.02. Source Code..... | 29 |

Chapter 04

| | |
|---------------|----|
| Testing | 30 |
|---------------|----|

Chapter 05

| | |
|-----------------|----|
| Conclusion..... | 43 |
|-----------------|----|

CHAPTER 01

INTRODUCTION

The main economic factors including human labour shall be used in the most efficient manner in order to get the best out of it. Inefficient usage of labour is a waste of human hours and impacts the economic development. The education sector has become a very labour intense because that requires more and more labour in the upcoming years. Due to the high demand for teachers, and school administrators from one hand and the inefficient usage of resource personals there is a scarcity pertaining to labour in the education sector. In the context of an educational institute the most of the tasks are done manually and technology is yet to be adapted. From the point of enrolment of a student to issuance of the degree certificate; everything is done manually and paper based.

As elsewhere mentioned, the existing educational system demands high amount of labour and as a country without low cost labour in abundance the system is very costly. As an example, the student registration is done manually which requires human labour and inclined to errors. In most schools examination marks are calculated and result sheets are created manually, and this process is time consuming. Also, the storage facilities to preserve these result sheets and the maintenance costs are not negligible. In the existing mechanism erroneous calculations, double entries and other human errors are a very common occurrence. The defects can be easily remedied by introducing modern technological methodologies.

Among means of maximizing labour efficiency, the automating the exiting manual process will achieve guaranteed results. In the context of an educational institute an automated system will accrue positivity through different means. If list down a few, the efficiency will increase, constrains on resources will reduce, labour requirement reduces as the same ends realized through a manual process can be realized using less labour, as the productivity of users increase that will ultimately result in the increased quality of life of the users. Let it be as it may, as given in the given scenario, the “e-Pupil” School Management and Marks Tracking system will increase the efficiency and costs will be reduced of all users. As stated in the given scenario a comprehensive student management system that manages tasks from registration to class assignment, marking examination marks and lecture assignment is proposed.

The proposed system is developed using the C# and .net frame work. The C# language is widely accepted and recognized. Also, the windows forms provide the opportunity to design the user interfaces easily and in less effort.

IDENTIFICATION OF MODEL - UML

a. Use Case

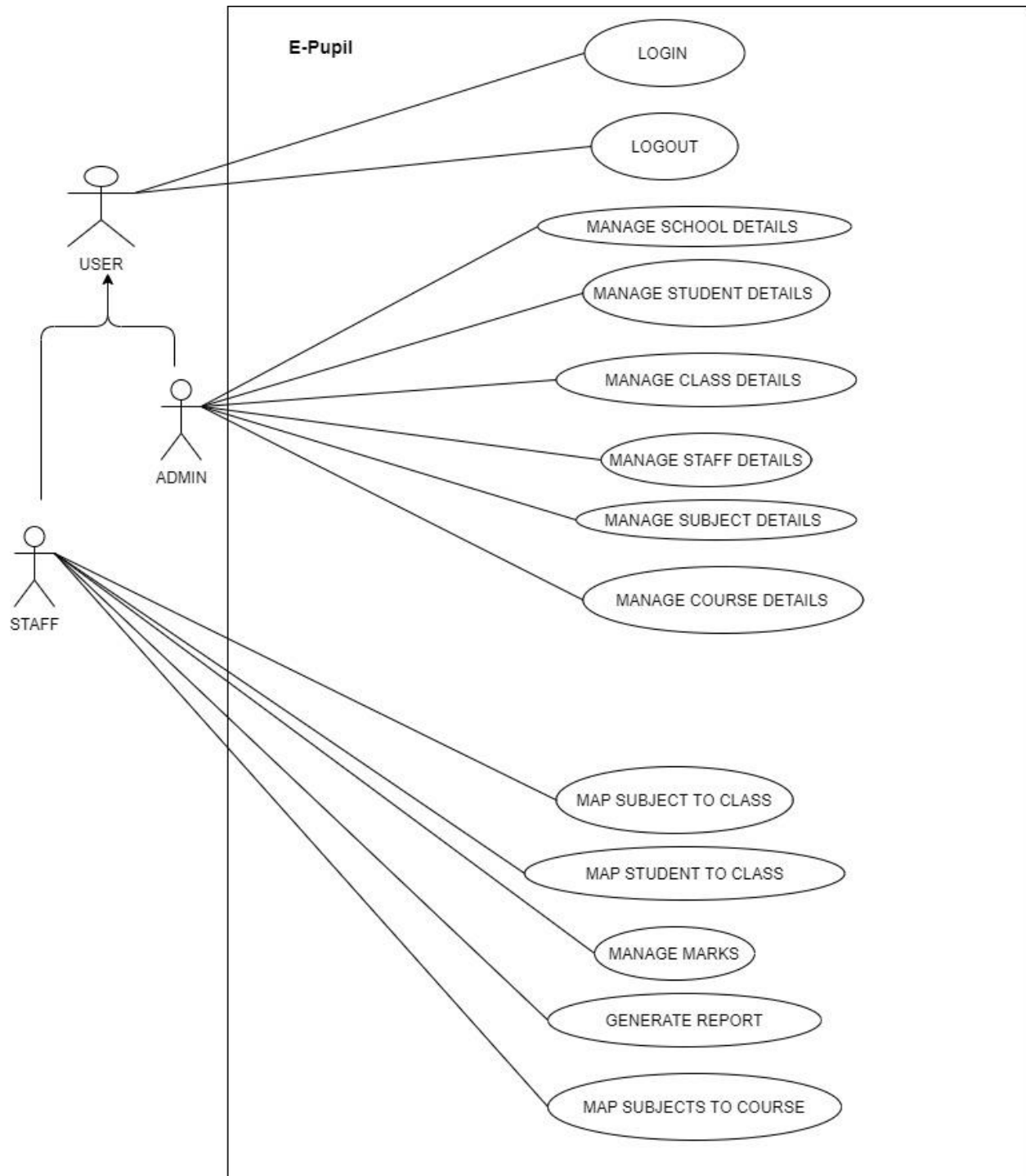


Figure 01-Use Case Diagram

b. Use Description

Use case: Login

| | |
|-----------------------|--|
| Description | The relevant actors must login to the system by proving the given Username and the Password. |
| Actors | Staff or Admin |
| Pre-Condition | The Actor is already registered as a user |
| Post-Condition | The Actor directs to the Dashboard |

| | | |
|------------------------------|--|-----------------------------------|
| Main Success Path | The Actor upon filling username and the password directs to the main Dashboard | |
| Actor Actions | | System Responses |
| 1. Add Username and Password | | 1.1 System verifies details |
| | | 1.2 Directs to the Main Dashboard |

| | | |
|------------------------------|---|-------------------------------|
| Exception Path | The Actor upon filling wrong username and/or the password. Access to the system is denied | |
| Actor Actions | | System Responses |
| 1. Add Username and Password | | 1.1 System verifies details |
| | | 1.2 Display the error message |

Use case: Logout

| | |
|-----------------------|--|
| Description | The Admin or the Staff member could logout from the system. Logout the actor should press the red color bottom on top of the main dashboard. |
| Actors | Staff or Admin |
| Pre-Condition | The Actor is already logged to the system. |
| Post-Condition | Log out from the system. |

| | | |
|-----------------------------|---|-----------------------------|
| Main Success Path | The Actor upon the mouse click logs out from the system | |
| Actor Actions | | System Responses |
| 1. Click the log out button | | 1.1 System verifies details |

Use case: Manage School Detail

| | |
|-----------------------|---|
| Description | Admin could add new details pertaining to the school. |
| Actors | Admin |
| Pre-Condition | The Actor is already logged in to the system. |
| Post-Condition | Save the changes to the data base. |

| | |
|--------------------------|---|
| Main Success Path | Admin could add new school and add details. The system will request set of pre-determined information and in order to save as a school. It will be necessary to add all detail requested. |
|--------------------------|---|

| Actor Actions | System Responses |
|---|---|
| 1. Press the Manage school details button and then select add new school details. | 1.1 Check whether all areas have been filled. |
| 2. Press the Save button | 2.1 Save data to the database |

| Alternative Path | Admin changes details of an existing school. | |
|---|---|--|
| Actor Actions | System Responses | |
| 1. Press the Manage school details button and then select change school details. Then select one school detail from the list to edit. | 1.1 Check whether all areas have been filled. | |
| 2. Press the Save button | 2.1 Save data to the database | |

| Exception Path | The Actor doesn't add all relevant details. The system will display a message stating relevant information to be added. | |
|--|--|--|
| Actor Actions | System Responses | |
| 1. Add only some details of the case. | 1.1 Display the error message. | |

Use case: Manage Student Detail

| | |
|-----------------------|--|
| Description | Admin could add new details pertaining to the Student. |
| Actors | Admin |
| Pre-Condition | The Actor is already logged in to the system. |
| Post-Condition | Save the changes to the data base. |

| Main Success Path | Admin could add new Student and add details. The system will request set of pre-determined information and in order to save as a Student, it will be necessary to add all details requested. | |
|---|--|--|
| Actor Actions | System Responses | |
| 1. Press the Manage Student details button and then select add new Student details. | 1.1 Check whether all areas have been filled. | |
| 3. Press the Save button | 2.1 Save data to the database | |

| Alternative Path 01 | Admin changes details of an existing Student. | |
|--|---|--|
| Actor Actions | System Responses | |
| 1. Press the Manage Student details button then select change student detail from the list. Then select the Student from the list. | 1.1 Check whether all areas have been filled. | |
| 2. Press the Save button | 2.1 Save data to the database | |

| | | |
|----------------------------|------------------------------------|--|
| Alternative Path 02 | Admin delete details of a student. | |
|----------------------------|------------------------------------|--|

| Actor Actions | System Responses |
|---------------------------------|--|
| 1. Select the relevant student. | 1.1 fills the relevant data to text boxes. |
| 2. Press the delete button | 2.1 delete data from the database |

| Exception Path | The Actor doesn't add all relevant details or add a detail that is already in the system. The system will display a message stating relevant information to be added. |
|--|--|
| Actor Actions | System Responses |
| 3. Add only some details of the case. | 1.1 Display the error message. |

Use case: Manage Class Detail

| | |
|-----------------------|--|
| Description | Admin could add new details pertaining to the Class. |
| Actors | Admin |
| Pre-Condition | The Actor is already logged in to the system. |
| Post-Condition | Save the changes to the data base. |

| Main Success Path | Admin could add new Class and add details. The system will request set of pre-determined information and in order to save a Class, it will be necessary to add all details requested. |
|---|---|
| Actor Actions | System Responses |
| 4. Press the Manage Class details button and then select add new Class details. | 1.1 Check whether all areas have been filled. |
| 5. Press the Save button | 2.1 Save data to the database |

| Alternative Path 01 | Admin changes details of an existing class. |
|---|---|
| Actor Actions | System Responses |
| 1. Press the Manage class details button then select change class detail. Then select the relevant class to edit from the list. | 1.1 Check whether all areas have been filled. |
| 2. Press the Save button | 2.1 Save data to the database |

| Alternative Path 02 | Admin delete details of an existing class. |
|-------------------------------|--|
| Actor Actions | System Responses |
| 3. Select the relevant class. | 1.1 fills the relevant data to text boxes. |
| 4. Press the delete button | 2.1 delete data from the database |

| Exception Path | The Actor doesn't add all relevant details or add a detail that is already in the system. The system will display a message stating relevant information to be added. |
|--|--|
| Actor Actions | System Responses |
| 1. Add only some details of the case. | 1.1 Display the error message. |

Use case: Manage Staff Detail

| | |
|--------------------|--|
| Description | Admin could add new details pertaining to the Staff. |
| Actors | Admin |

| | |
|-----------------------|---|
| Pre-Condition | The Actor is already logged in to the system. |
| Post-Condition | Save the changes to the data base. |

| | | |
|---|--|---|
| Main Success Path | Admin could add new Staff and add details. The system will request set of pre-determined information and in order to save a new Staff member, it will be necessary to add all details requested. | |
| Actor Actions | | System Responses |
| 1. Press the Manage Staff details button and then select add new Staff details. | | 1.1 Check whether all areas have been filled. |
| 2. Press the Save button | | 2.1 Save data to the database |

| | | |
|--|---|---|
| Alternative Path 01 | Admin changes details of an existing Staff. | |
| Actor Actions | | System Responses |
| 1. Press the Manage Staff details button then select change Staff detail. Then select the relevant Staff member to edit from the list. | | 1.1 Check whether all areas have been filled. |
| 2. Press the Save button | | 2.1 Save data to the database |

| | | |
|--------------------------------------|---|--|
| Alternative Path 02 | Admin delete details of an existing Staff member. | |
| Actor Actions | | System Responses |
| 5. Select the relevant staff member. | | 1.1 fills the relevant data to text boxes. |
| 6. Press the delete button | | 2.1 delete data from the database |

| | | |
|--|--|--------------------------------|
| Exception Path | The Actor doesn't add all relevant details or add a detail that is already in the system. The system will display a message stating relevant information to be added. | |
| Actor Actions | | System Responses |
| 1. Add only some details of the case. | | 1.1 Display the error message. |

Use case: Manage Subject Detail

| | |
|-----------------------|--|
| Description | Admin could add new details pertaining to a new Subject. |
| Actors | Admin |
| Pre-Condition | The Actor is already logged in to the system. |
| Post-Condition | Save the changes to the data base. |

| | | |
|---|--|---|
| Main Success Path | Admin could add new Subject and add details. The system will request set of pre-determined information and in order to save a new Subject, it is necessary to add all details requested. | |
| Actor Actions | | System Responses |
| 1. Press the Manage Subject details button and then select add new Subject details. | | 1.2 Check whether all areas have been filled. |
| 2. Press the Save button | | 2.1 Save data to the database |

| | | |
|--|---|---|
| Alternative Path | Admin changes details of an existing Subject. | |
| Actor Actions | | System Responses |
| 1. Press the Manage Subject details button then select change Subject detail. Then select the relevant Subject member to edit from the list. | | 1.1 Check whether all areas have been filled. |
| 2. Press the Save button | | 2.1 Save data to the database |

| | | |
|--|--|--------------------------------|
| Exception Path | The Actor doesn't add all relevant details or add a detail that is already in the system. The system will display a message stating relevant information to be added. | |
| Actor Actions | | System Responses |
| 1. Add only some details of the case. | | 1.1 Display the error message. |

Use case: Manage Course Detail

| | |
|-----------------------|--|
| Description | Admin could add new details pertaining to a new Corse. |
| Actors | Admin |
| Pre-Condition | The Actor is already logged in to the system. |
| Post-Condition | Save the changes to the data base. |

| | | |
|---|--|---|
| Main Success Path | Admin could add new Course and add details. The system will request set of pre-determined information and in order to save a new course, it is necessary to add all details requested. | |
| Actor Actions | | System Responses |
| 1. Press the Manage Course details button and then select add new Course details. | | 1.1 Check whether all areas have been filled. |
| 2. Press the Save button | | 2.1 Save data to the database |

| | | |
|--|---|--|
| Alternative Path | Admin changes details of an existing Subject. | |
| Actor Actions | | System Responses |
| 1. Press the Manage Course details button then select change Course detail. Then select the relevant Course to edit from the list. | | 1.1Check whether all areas have been filled. |
| 2. Press the Save button | | 2.1 Save data to the database |

| | | |
|--|--|--------------------------------|
| Exception Path | The Actor doesn't add all relevant details or add a detail that is already in the system. The system will display a message stating relevant information to be added. | |
| Actor Actions | | System Responses |
| 1. Add only some details of the case. | | 1.1 Display the error message. |

Use case: Map Subject to a Class

| | |
|--------------------|---|
| Description | Staff could match a subject to a certain class in the list. |
|--------------------|---|

| | |
|-----------------------|---|
| Actors | Staff |
| Pre-Condition | The Actor is already logged in to the system. |
| Post-Condition | Save the changes to the data base. |

| | | |
|---|--|---|
| Main Success Path | Staff can add the subject to the relevant class with the time. It is necessary to add all details requested. | |
| Actor Actions | | System Responses |
| 1. Press map subject to class from the staff dashboard. | | 1.1 Check whether all areas have been filled. |
| 2. Press the Save button. | | 2.1 Save data to the database. |

| | | |
|--|---|-----------------------------------|
| Alternative Path 01 | Staff changes details of an existing class and subject. | |
| Actor Actions | | System Responses |
| 1. Then select the relevant class and subject to edit from the list. | | 1.1 Fills the data to text boxes. |
| 2. Press the Save button. | | 2.1 Save data to the database. |

| | | |
|-------------------------------|---|--|
| Alternative Path 02 | Staff deletes details of an existing class and subject. | |
| Actor Actions | | System Responses |
| 1. Select the relevant class. | | 1.1 Fills the relevant data to text boxes. |
| 2. Press the delete button. | | 2.1 delete data from the database. |

| | | |
|--|--|--------------------------------|
| Exception Path | The Actor doesn't add all relevant details or add a detail that is already in the system. The system will display a message stating relevant information to be added. | |
| Actor Actions | | System Responses |
| 2. Add only some details of the case. | | 1.1 Display the error message. |

Use case: Map Student to a Subject

| | |
|-----------------------|---|
| Description | Staff could match a student to relevant subjects in the list. |
| Actors | Staff |
| Pre-Condition | The Actor is already logged in to the system. |
| Post-Condition | Save the changes to the data base. |

| | | |
|---|--|---|
| Main Success Path | Staff can add the student to the relevant subject with the time. It is necessary to add all details requested. | |
| Actor Actions | | System Responses |
| 1. Press map subject to student from the staff dashboard. | | 1.2 Check whether all areas have been filled. |
| 2. Press the Save button. | | 2.1 Save data to the database. |

| | | |
|----------------------------|---|--|
| Alternative Path 01 | Staff changes details of an existing schedule of a student. | |
|----------------------------|---|--|

| Actor Actions | System Responses |
|--|-----------------------------------|
| 1. Then select the relevant student and subject to edit from the list. | 1.1 Fills the data to text boxes. |
| 2. Press the Save button. | 2.1 Save data to the database. |

| Alternative Path 02 | Staff deletes details of an existing student schedule. |
|---------------------------------|--|
| Actor Actions | System Responses |
| 1. Select the relevant student. | 1.1 Fills the relevant data to text boxes. |
| 2. Press the delete button. | 2.1 delete data from the database. |

| Exception Path | The Actor doesn't add all relevant details or add a detail that is already in the system. The system will display a message stating relevant information to be added. |
|--|--|
| Actor Actions | System Responses |
| 1. Add only some details of the case. | 1.1 Display the error message. |

Use case: Add Marks

| | |
|-----------------------|---|
| Description | Staff could add marks pertaining to an Examination. |
| Actors | Staff |
| Pre-Condition | The Actor is already logged in to the system. |
| Post-Condition | Save the changes to the data base. |

| Main Success Path | Staff could add new Marks for an examination. |
|--|---|
| Actor Actions | System Responses |
| 1. Press the Manage Marks button from the dashboard and then select add new marks. | 1.2 Check whether all areas have been filled. |
| 2. Press the Save button | 2.1 Save data to the database |

| Alternative Path | Admin changes details of an existing student marks. |
|--|---|
| Actor Actions | System Responses |
| 1. Press the Manage marks button then select change added marks. Then select the relevant examination and the entry to change from the list. | 1.1 Check whether all areas have been filled. |
| 2. Press the Save button | 2.1 Save data to the database |

| Exception Path | The Actor doesn't add all relevant details or add a detail that is already in the system. The system will display a message stating relevant information to be added. |
|--|--|
| Actor Actions | System Responses |
| 1. Add only some details of the case. | 1.1 Display the error message. |

Use case: Generate Report

| | |
|-----------------------|---|
| Description | Staff could print a report as provided in the system. |
| Actors | Staff |
| Pre-Condition | The Actor is already logged in to the system. |
| Post-Condition | Save the changes to the data base. |

| | | |
|--|--|--|
| Main Success Path | Staff could get new reports from the system. | |
| Actor Actions | | System Responses |
| 1. Press the Generate Button from the dashboard and then select the relevant report. | | 1.3 Check whether all areas have been filled to generate a report. |
| | | 1.2 Print the relevant report. |

| | | |
|--|---|--------------------------------|
| Exception Path | If the relevant information have not been added to get the relevant report. | |
| Actor Actions | | System Responses |
| 1. Select the relevant report from the system. | | 1.1 Display the error message. |

c. User Stories

Actor: Staff

01. As a first time user/user I want to click on the desktop icon and open the app, so I can use the app.
02. As a user I need to login to the application and use the services provided.
03. As a user I need to add student to a particular course, so student can link with a course.
04. As a user I need to add subject to a course, so courses and relevant subjects can be linked.
05. As a user I need to generate reports, so I can use those to take administrative decisions.
06. As a user I need to add marks, so I can use those to take administrative decisions.
07. As a user I need to map subject to a class, so class and relevant subjects can be linked.
08. As a user I need to map student to a subject, so student and relevant subjects can be linked.

Actor: Admin

01. As a first time user/user I want to click on the desktop icon and open the app, so I can use the app.
02. As a user I need to login to the application and use the services provided.
03. As a user I need to add school details, so school details will be saved in the database.
04. As a user I need to edit school details, so school details can be changed in the database.
05. As a user I need to delete school, so school details will be deleted from the database.
06. As a user I need to add course details, so course details will be saved in the database.
07. As a user I need to edit course details, so course details can be changed in the database.
08. As a user I need to delete course, so course details will be deleted from the database.
09. As a user I need to add student details, so student details will be saved in the database.
10. As a user I need to edit student details, so student details can be changed in the database.
11. As a user I need to delete student, so student details will be deleted from the database.
12. As a user I need to add subject details, so subject details will be saved in the database.
13. As a user I need to edit subject details, so subject details can be changed in the database.
14. As a user I need to delete subject, so subject details will be deleted from the database.
15. As a user I need to add student details, so student details will be saved in the database.
16. As a user I need to edit student details, so student details can be changed in the database.
17. As a user I need to delete student, so student details will be deleted from the database.
18. As a user I need to add staff details, so staff details will be saved in the database.
19. As a user I need to edit staff details, so staff details can be changed in the database.
20. As a user I need to delete staff, so staff details will be deleted from the database.

d. Class Diagram

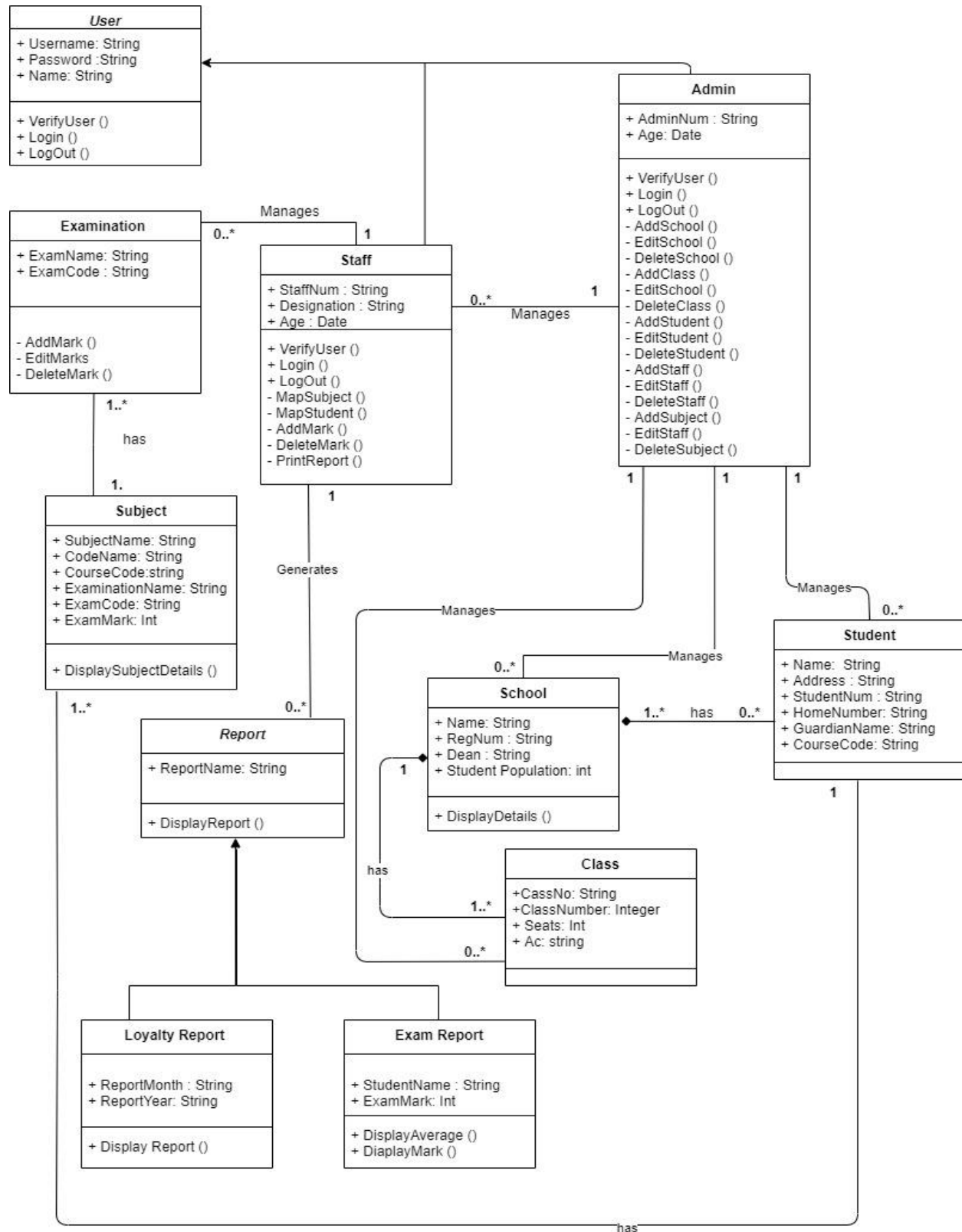


Figure 02 Class Diagram

e. ER Diagram

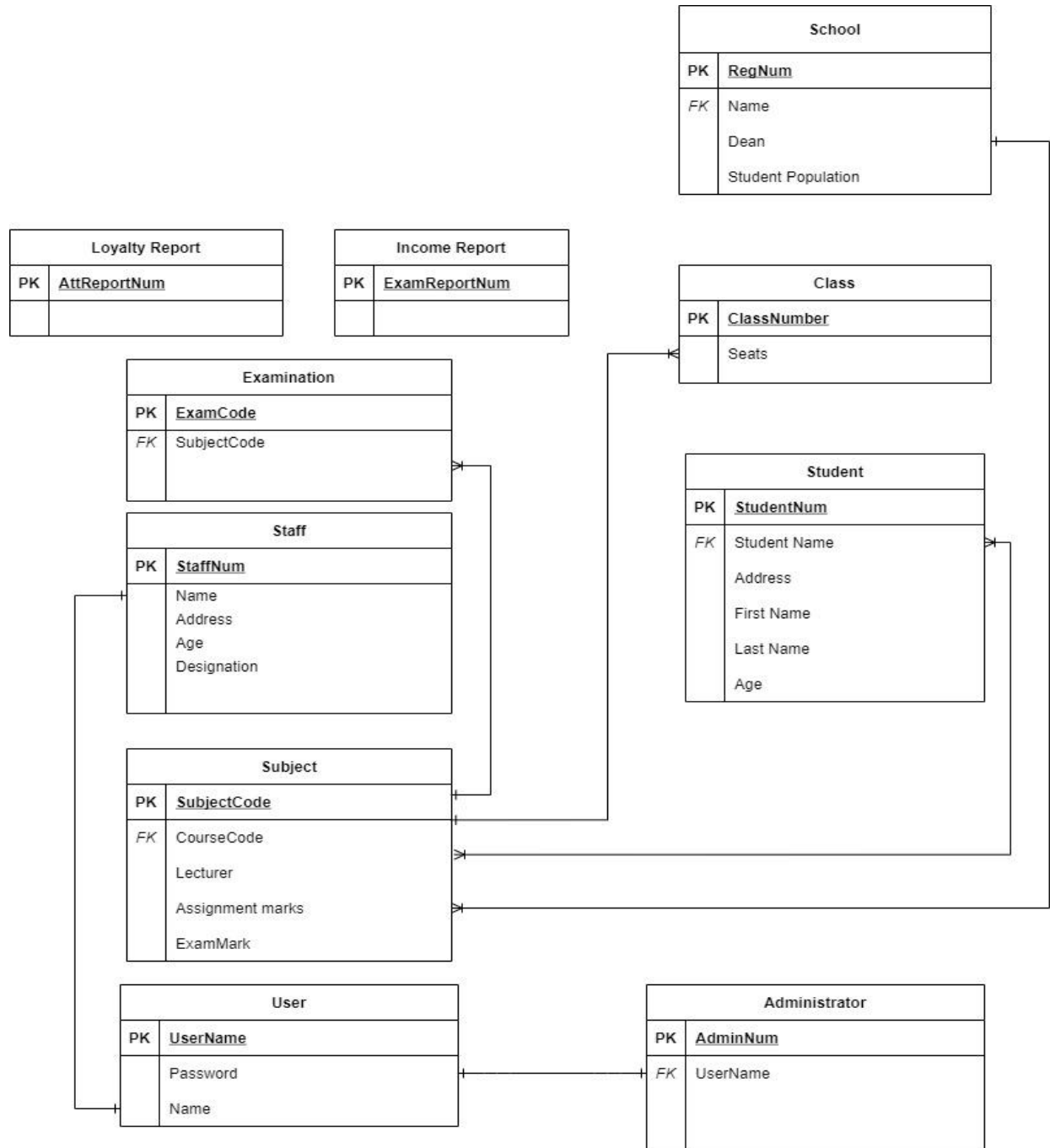
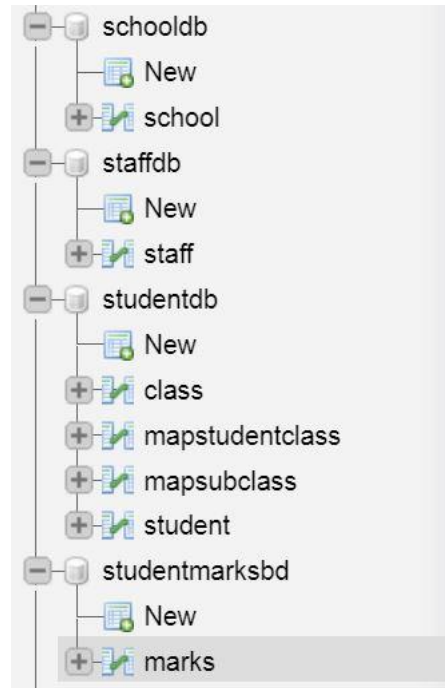


Figure 03 ER Diagram

f. Database



Data base is a very important for any software development work because it contains. All the data based have been developed using the MySQL. The main table's structures have been given bellow.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|-----------------------------|------------|-------------|--------------------|------------|------|---------|----------|----------------|
| <input type="checkbox"/> 1 | StdRNum | int(3) | | | No | None | | AUTO_INCREMENT |
| <input type="checkbox"/> 2 | StdFN | varchar(10) | utf8mb4_general_ci | | No | None | | |
| <input type="checkbox"/> 3 | StdLN | varchar(20) | utf8mb4_general_ci | | No | None | | |
| <input type="checkbox"/> 4 | StdCur | varchar(30) | utf8mb4_general_ci | | No | None | | |
| <input type="checkbox"/> 5 | StdYer | varchar(10) | utf8mb4_general_ci | | No | None | | |
| <input type="checkbox"/> 6 | StdTP | varchar(15) | utf8mb4_general_ci | | No | None | | |
| <input type="checkbox"/> 7 | StdAdd | varchar(40) | utf8mb4_general_ci | | No | None | | |
| <input type="checkbox"/> 8 | DoB | date | | | No | None | | |
| <input type="checkbox"/> 9 | StdPayment | varchar(10) | utf8mb4_general_ci | | No | None | | |
| <input type="checkbox"/> 10 | StdPaid | varchar(10) | utf8mb4_general_ci | | No | None | | |
| <input type="checkbox"/> 11 | StdDue | varchar(10) | utf8mb4_general_ci | | No | None | | |

Figure 04 Student table structure

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|----------------------------|---------------------|-------------|--------------------|------------|------|---------|----------|----------------|
| <input type="checkbox"/> 1 | RNum | int(11) | | | No | None | | AUTO_INCREMENT |
| <input type="checkbox"/> 2 | User ID | varchar(10) | utf8mb4_general_ci | | No | None | | |
| <input type="checkbox"/> 3 | Account Type | varchar(6) | utf8mb4_general_ci | | No | None | | |
| <input type="checkbox"/> 4 | First Name | varchar(11) | utf8mb4_general_ci | | No | None | | |
| <input type="checkbox"/> 5 | Last Name | varchar(11) | utf8mb4_general_ci | | No | None | | |
| <input type="checkbox"/> 6 | DoB | date | | | No | None | | |
| <input type="checkbox"/> 7 | Phone | int(11) | | | No | None | | |
| <input type="checkbox"/> 8 | Address | varchar(30) | utf8mb4_general_ci | | No | None | | |
| <input type="checkbox"/> 9 | Designation | varchar(10) | utf8mb4_general_ci | | No | None | | |

Figure 05 Staff table structure

| # | Name | Type | Collation | Attributes | Null | Default |
|-----------------------------|---------------------|-------------|--------------------|------------|------|---------|
| <input type="checkbox"/> 1 | FGrade | varchar(4) | utf8mb4_general_ci | | No | None |
| <input type="checkbox"/> 2 | RName | varchar(10) | utf8mb4_general_ci | | No | None |
| <input type="checkbox"/> 3 | CourseC | varchar(10) | utf8mb4_general_ci | | No | None |
| <input type="checkbox"/> 4 | Fn | varchar(10) | utf8mb4_general_ci | | No | None |
| <input type="checkbox"/> 5 | Ln | varchar(10) | utf8mb4_general_ci | | No | None |
| <input type="checkbox"/> 6 | Subject01 | varchar(10) | utf8mb4_general_ci | | No | None |
| <input type="checkbox"/> 7 | Exam01 | varchar(10) | utf8mb4_general_ci | | No | None |
| <input type="checkbox"/> 8 | Assignment01 | varchar(10) | utf8mb4_general_ci | | No | None |
| <input type="checkbox"/> 9 | Subject02 | varchar(10) | utf8mb4_general_ci | | No | None |
| <input type="checkbox"/> 10 | Exam02 | varchar(10) | utf8mb4_general_ci | | No | None |
| <input type="checkbox"/> 11 | Assignment02 | varchar(10) | utf8mb4_general_ci | | No | None |

Figure 06 Marks table

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|----------------|-------------|--------------------|------------|------|---------|----------|-------|
| 1 | Date | date | | | No | None | | |
| 2 | Subject | varchar(10) | utf8mb4_general_ci | | No | None | | |
| 3 | Time | varchar(10) | utf8mb4_general_ci | | No | None | | |
| 4 | Class | varchar(10) | utf8mb4_general_ci | | No | None | | |

Figure 07 Date Subject Table

| # | Name | Type | Collation | Attributes | Null | Default | Comments |
|---|-----------------|-------------|--------------------|------------|------|---------|----------|
| 1 | SchoolIN | varchar(30) | utf8mb4_general_ci | | No | None | |
| 2 | SRN | varchar(10) | utf8mb4_general_ci | | No | None | |
| 3 | Dean | varchar(20) | utf8mb4_general_ci | | No | None | |

Figure 08 School table

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra |
|---|----------------|-------------|--------------------|------------|------|---------|----------|-------|
| 1 | FName | varchar(10) | utf8mb4_general_ci | | No | None | | |
| 2 | LName | varchar(10) | utf8mb4_general_ci | | No | None | | |
| 3 | Date | date | | | No | None | | |
| 4 | Subject | varchar(10) | utf8mb4_general_ci | | No | None | | |
| 5 | Time | varchar(10) | utf8mb4_general_ci | | No | None | | |
| 6 | Class | varchar(10) | utf8mb4_general_ci | | No | None | | |

Figure 09 Class time Table


| # | Name | Type | Collation | Attributes | Null | Default | Comments |
|---|---|-------------|--------------------|------------|------|---------|----------|
| 1 | ClassN  | varchar(10) | utf8mb4_general_ci | | No | None | |
| 2 | ClassName | varchar(10) | utf8mb4_general_ci | | No | None | |
| 3 | Seat | varchar(10) | utf8mb4_general_ci | | No | None | |
| 4 | A/C | varchar(10) | utf8mb4_general_ci | | No | None | |

Figure 10 Class Table

CHAPTER 03

03.01 Development

User Manual

Loading Screen (Screen 01)

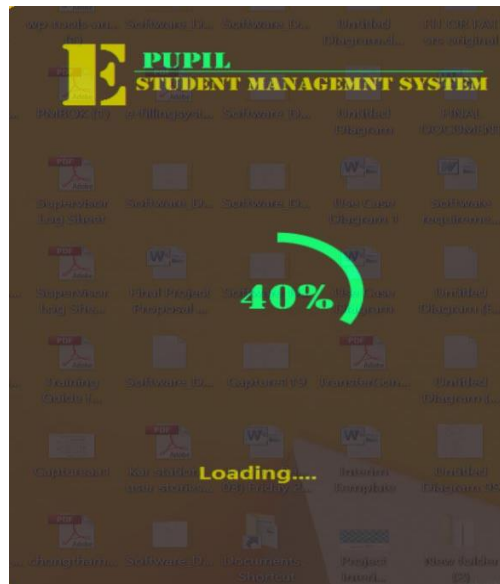


Figure 11

This is the first screen that will load first. As soon as the circle completes one round the program will be launched.

Account Selection Screen (Screen 02)

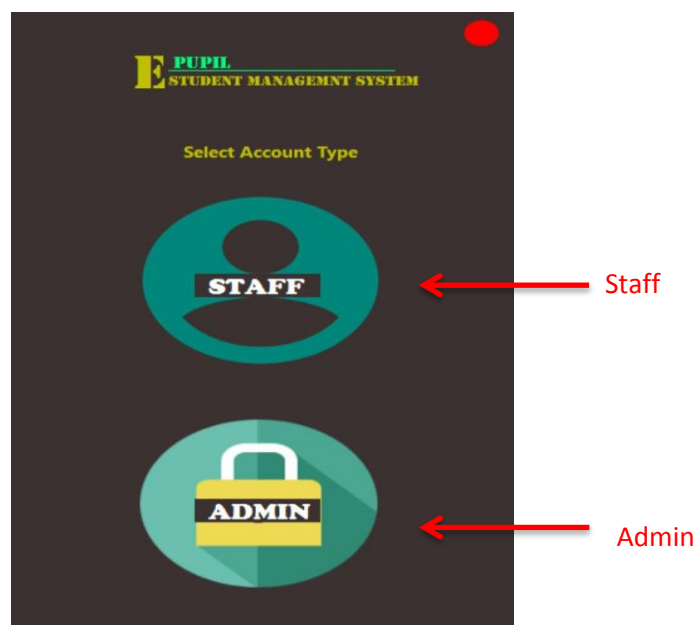


Figure 12

From the second screen the user shall select the type of the account want to access. From the red circle the user can close the program. The first part contains information on if the user selected the admin account type.

Screen Flow for Admin Account Holders

Login Screen (Screen 03)

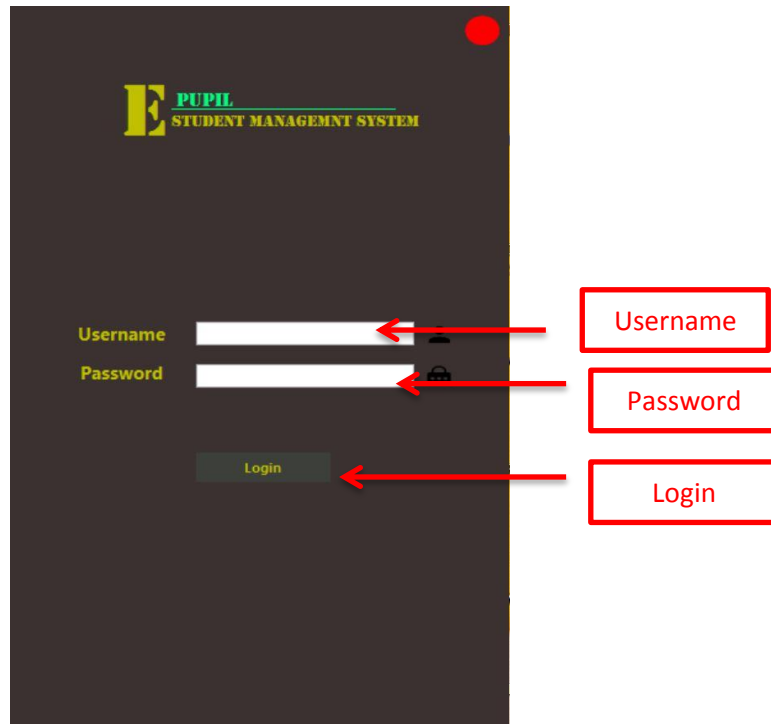


Figure 13

User can use this screen to add username and password. If the Password is incorrect password incorrect message will display.

Admin Dashboard (Screen 04)

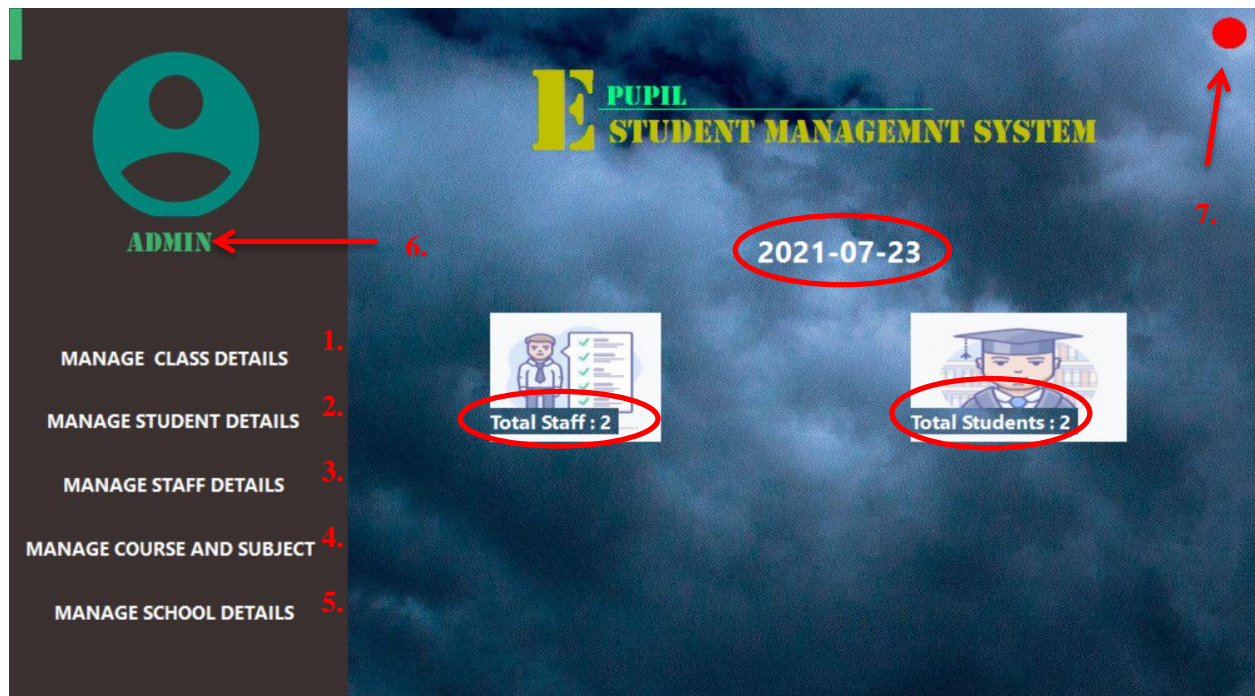


Figure 14

This is the main panel for administrators. On the Screen Date will be displayed and total number of students and total staff members will be displayed. Account holder can select the tasks he/she wants to complete from the panel on the left. As numbered from 1 to 5 the user can select the relevant buttons to manage different aspects of the system. As shown by number 06 the ADMIN denotes the type of the account. The red colour button on the top can be used to close the application and return to the login screen.

Manage Class Details (Screen 05)

1. Class Number

2. Class Name

3. Seats

4. A/C - Non A/C ☐ A/C ☐ NON A/C

SAVE **EDIT** **DELETE** **CLEAR**

| | | | |
|-----------|------|----|-------|
| 1 | C1L1 | 20 | AC |
| 2 | C1L2 | 25 | NONAC |
| New Entry | | | |

Figure 15

The Admin can select the 'Manage Class Details' buttons to add relevant details pertaining to the school. As highlighted above when selected the relevant button the green colour panel will appear on the right corner (circled in red colour). The admin can add new data, edit new data, and delete data and clear values in the text boxes at once. In order to edit details it is required to select the relevant filed from the below list. The system will furnish the details in to the textboxes so that the admin can edit. Admin should add all the details requested to add data to the database (1. Class Number, 2. Class Name, 3. Seat and 4.A/C).

Manage Student Details (Screen 06)

Search Student Number

First Name Last Name

Course ☐ Software Engineering ☐ Management

Year

Telephone Number

Address

Date of Birth 2021-07-21

Payment (Rs.) Paid (Rs.) Due (Rs.)

SAVE **EDIT** **CLEAR** **DELETE**

| | | | | | | | | | | |
|----------|-------|--------|-----------|------|----------|-----------|----------|---------|---------|---------|
| 20210... | Amal | Perera | softwa... | 2020 | 12365... | Kandy | 2000-... | 120,000 | 110,000 | 10,000 |
| 20210... | Kusal | Perera | Manag... | 2019 | 07667... | 22 Sad... | 2019-... | 200,000 | 100,000 | 100,000 |

Figure 16

The Admin can use this screen to add details about the students, edit, delete or clear all textboxes. Also, admin can use the search button to search for students in the list below. It is required to add all the details as shown above to save data to the database. In order to edit or delete an item it is required to select the relevant item from the list.

Manage Staff Details (Screen 07)

Search by Number

Account Type ☐ Staff ☐ Admin

First Name **Last Name**

ID Number

Date of Birth

Mobile Number

Address

Designation

| | | | | | | | | |
|-------|------------|-------|--------|--------|------------|------------|--------------|---------|
| 12335 | 8988898... | staff | Nayana | Perera | 2021-07... | 9877765... | 1233, Sa... | Manager |
| 12336 | 1222898... | staff | Yo | Mohan | 2002-03... | 0 | 1, Perade... | Teacher |
| | | | | | | | | |
| | | | | | | | | |

Figure 17

The admin can use this form to manage details about the staff members. Also, admin can use the search button to search for employee in the list below. It is required to add all the details as shown above to save data to the database. In order to edit or delete an item it is required to select the relevant item from the list.

Manage Course and Subject (Screen 08)

Course

Course Code

Subject 1 Exam ☐ Assignment ☐ Lecturer Lec. Hours

Subject 2 Exam ☐ Assignment ☐ Lecturer Lec. Hours

| | | | | | | |
|---------|---------|-----------|---|---|----------|----|
| Law | law01 | Land | 1 | 2 | Assam | 1 |
| Mana... | Mana... | Basic ... | 1 | 2 | Abdul... | 30 |

| | | | | |
|----------|---|---|---------|----|
| Advan... | 1 | 2 | Sama... | 30 |
|----------|---|---|---------|----|

Figure 18

Admin can use this form to add course and subject details. Also, in order to save a detail it is required to fill all the textboxes.

Manage Course and Subject (Screen 09)

School Name

Registration Number

Dean

Student Population

| | | |
|-------------------------|------------|--------------------|
| KANDY SCHOOL OF STUDIES | #123123123 | Dr. Saliya Peiries |
|-------------------------|------------|--------------------|

Figure 19

The Admin can use this for to furnish all the details pertaining to the system. It is noteworthy as shown above the student population is automatically calculated by the system based on the number of students registered. Please refer the screen 2 instruction on how to add, delete, and edit students.

Screen Flow for Staff Account Holders (Screen 3 for staff members)

This is a continuation form the screen 3 above.

The screenshot shows the login interface for the E-Pupil Student Management System. It features a dark blue background with the system's logo at the top left. The login form consists of two white input fields labeled 'Username' and 'Password' in yellow text. To the right of these fields, red arrows point to red boxes containing the labels 'Username' and 'Password'. Below the input fields is a green 'Login' button, with a red arrow pointing to a red box containing the label 'Login'.

Figure 20

User can use this screen to add username and password. If the Password is incorrect password incorrect message will display.

Dashboard for Staff (Screen 4 for staff members)

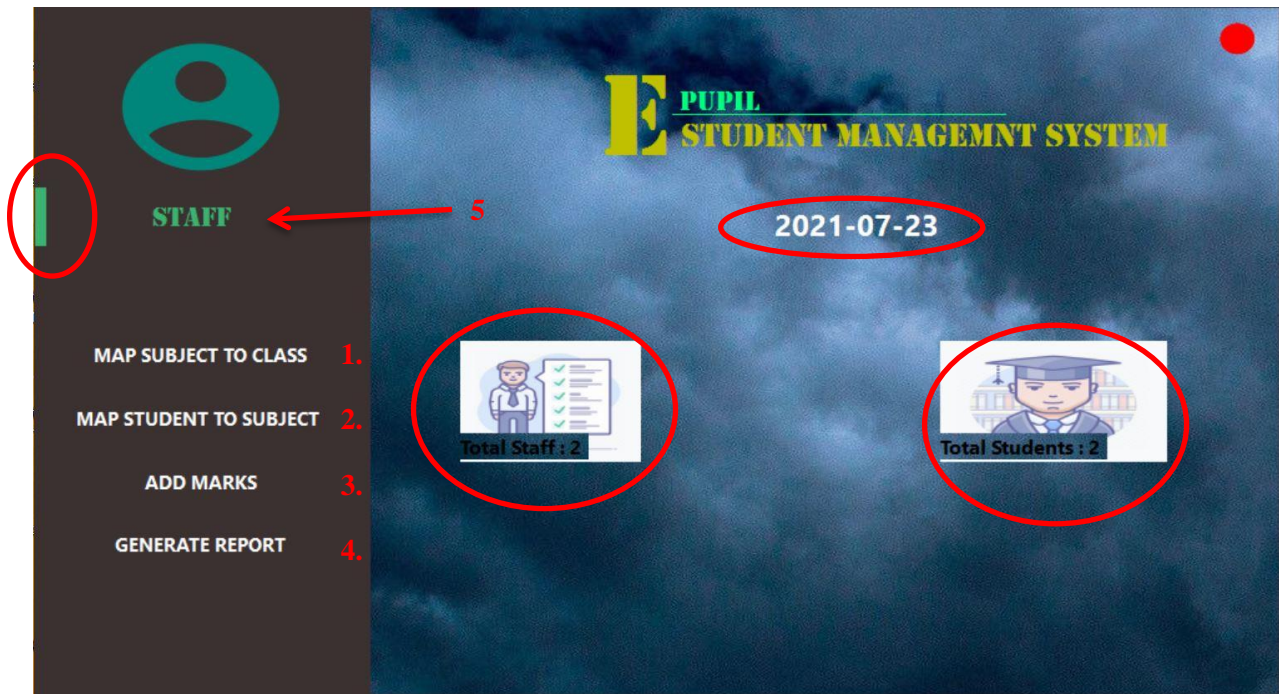


Figure 21

This is the main panel for staff members. On the Screen Date will be displayed and total number of students and total staff members will be displayed. Account holder can select the tasks he/she wants to do from the panel on the left (1. Map subject to class, 2.Map student to subject, 3.Add Marks, 4.General Report). As numbered from 1 to 4 the user can select the relevant buttons to manage different aspects of the system. As shown by number 05 the ADMIN denotes the type of the account.

Map Subject to Class (Screen 5 for staff members)

The interface is titled "STAFF" and includes a sidebar with the following options:

- MAP SUBJECT TO CLASS
- MAP STUDENT TO SUBJECT
- ADD MARKS
- GENERATE REPORT

The main form contains the following fields and buttons:

- Search** (highlighted with a red circle)
- Date** (2021-07-22)
- Subject** (dropdown menu)
- Time** (dropdown menu)
- Class** (dropdown menu)
- SAVE**, **EDITE**, **CLEAR**, and **DELETE** buttons
- Close Button** (highlighted with a red box and a red arrow pointing to a red dot in the top right corner)

The table below shows the mapped data:

| Date | Subject | Time | Class |
|------------|---------|-------|-------|
| 2021-07-22 | Law | 09-10 | L1C1 |
| 2000-02-08 | BCS | 10-11 | L1C2 |
| 2000-02-08 | MSC | 01-02 | L1C5 |
| 2021-07-23 | SE | 10-11 | L1C2 |
| 2021-07-23 | SE | 10-11 | L1C2 |
| 2021-07-22 | Law | 10-11 | L1C1 |
| 2000-02-08 | MSC | 01-02 | L1C5 |
| 2000-02-08 | BCAS | 10-11 | L1C2 |
| 2021-07-24 | ASD | 10-11 | L1C2 |

Figure 22

Staff members can use this form to match subject and class with the relevant time. Also, the date the relevant date can be selected. As shown in the above diagram the staff member can add classes, edit, clear from boxes and delete the relevant details form the database.

Map Student to subject (Screen 6 for staff members)

Search Name

First Name

Last Name

Date

Subject

Time

Class

SAVE **EDITE** **CLEAR** **DELETE**

| | | | | | |
|------|-------|------------|----------|-------|------|
| Amal | Silva | 2021-07-25 | Land Law | 10-11 | C1L1 |
|------|-------|------------|----------|-------|------|

Figure 23

This form can be used to map student to a class. All other features will be same as in other windows.

Add Marks (Screen 7 for staff members)

Search Student Number

Final Grade

Registration Number

Course Code

First Name

Last Name

Subject

Exam Marks

Assignment

SAVE **EDITE** **CLEAR**

| | | | | | | | |
|-------|-------|--------|--------|----------|------|-----|-----------------|
| 1 | 00012 | law101 | Kamal | Harris | Land | 25 | 60 |
| 23 | #law | Kavi | Na | Land | 88 | 70 | Comm... 80 80 |
| 23341 | #law | Ravi | Naadun | Land law | 78 | 60 | Comm... 60 8060 |
| a | a | q | q | 1qa | qa | 11q | |

Figure 23

Staff members can use this window to add details about the student marks.

Add Marks (Screen 8 for staff members)

STAFF

MAP SUBJECT TO CLASS

MAP STUDENT TO SUBJECT

ADD MARKS

GENERATE REPORT

MONTHLY INCOME

STUDENT MARKS

LOYALTY REPORT

Result Sheet

Registration Number

First Name Last Name

Course Year

Print Student Mark

Figure 23

If the user click on the generate report button from the left panel. A drop down list will be shown and the user can select the type of the report wanted to generate.

03.02 Source Code

This application has been developed using Microsoft Visual Studio Community 2019 Version 16.9.4. C# is the language used.

Class DB

Class DBconnection establishes the connection between database and the app.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using MySql.Data.MySqlClient;
7
8  namespace EpupilSplashScreen
9  {
10     8 references
11     class DBconnect
12     {
13         20 references
14         MySqlConnection connect = new MySqlConnection("datasource=localhost;port=3306;username=root;password=root;database=studentdb");
15         public MySqlConnection getconnection
16         {
17             get
18             {
19                 return connect;
20             }
21         }
22         13 references
23         public void openConnect()
24         {
25             if (connect.State == System.Data.ConnectionState.Closed)
26                 connect.Open();
27         }
28         25 references
29         public void closeconnect()
30         {
31             if (connect.State == System.Data.ConnectionState.Open)
32                 connect.Close();
33         }
34     }
35 }
```

Figure 24

Account Class

User authentication is handled by this class.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Data;
7  using MySql.Data.MySqlClient;
8  using EpupilSplashScreen;
9
10 namespace EpupilSplashScreen
11 {
12     2 references
13     class Accounts : UserLoginClass
14     {
15         2 references
16         public string username { get; set; }
17         2 references
18         public string password { get; set; }
19         1 reference
20         public string fullname { get; set; }
21
22         1 reference
23         public bool validate_user()
24         {
25             bool check = false;
26
27             connectdb.Open();
28
29             MySqlDataReader rd;
30
31             using (var cmd = new MySqlCommand())
32             {
33                 cmd.CommandText = "SELECT* FROM users where User_Name=@user AND User_Password=@password";
34                 cmd.CommandType = CommandType.Text;
35                 cmd.Connection = connectdb;
36                 cmd.Parameters.Add("@user", MySqlDbType.VarChar).Value = username;
37                 cmd.Parameters.Add("password", MySqlDbType.VarChar).Value = password;
38
39                 rd = cmd.ExecuteReader();
40                 while (rd.Read())
41                 {
42                     check = true;
43                     fullname = rd.GetString("Name");
44                 }
45                 connectdb.Close();
46             }
47
48             return check;
49         }
50     }
51 }
```

Figure 25

Student Class

Student class is one of the main classes in this development. This class is utilized when adding, editing and editing student details.

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using MySql.Data.MySqlClient;
7  using System.Data;
8
9  namespace EpullipLashScreen
10 {
11     [Reference]
12     class RegisteredStudentClass
13     {
14         (Disconnect connect = new MySqlConnection());
15
16         public bool insertStudent(string StdFirstName, string StdLastName, string Course, DateTime StdId, string Year,
17             string Address, string StdPhone, string Payment, string Paid, string Due)
18         {
19             MySqlCommand command = new MySqlCommand("INSERT INTO `student` (`StdFN`, `StdLN`, `StdCur`, `DoB`, `StdYr`, `StdAdR`, `StdTP`, `StdPaym`, `StdPaid`, `StdDue`) VALUES (@fn, @ln, @cur, @dob, @yr, @add, @p, @payment, @paid, @due)", connect.getConnection());
20
21             //run, fname, lname, curs, bdate, year, address, tele, payment, paid, due)
22
23             command.Parameters.Add("@fn", MySqlDbType.VarChar).Value = StdFirstName;
24             command.Parameters.Add("@ln", MySqlDbType.VarChar).Value = StdLastName;
25             command.Parameters.Add("@cur", MySqlDbType.VarChar).Value = Course;
26             command.Parameters.Add("@dob", MySqlDbType.Date).Value = StdId;
27             command.Parameters.Add("@yr", MySqlDbType.VarChar).Value = Year;
28             command.Parameters.Add("@tp", MySqlDbType.VarChar).Value = StdPhone;
29             command.Parameters.Add("@add", MySqlDbType.VarChar).Value = Address;
30             command.Parameters.Add("@payment", MySqlDbType.VarChar).Value = Payment;
31             command.Parameters.Add("@paid", MySqlDbType.VarChar).Value = Paid;
32             command.Parameters.Add("@due", MySqlDbType.VarChar).Value = Due;
33
34             connect.openConnection();
35             if (command.ExecuteNonQuery() == 1)
36             {
37                 connect.closeconnect();
38                 return true;
39             }
40             else
41             {
32                 connect.closeconnect();
43                 return false;
44             }
45         }
46     }
47 }

```

Figure 26

```

74
75 //create a function edit for student
76
77 public bool updateStudent(string StdFirstName, string StdLastName, string Course, DateTime StdId, string Year,
78     string Address, string StdPhone, string Payment, string Paid, string Due)
79 {
80     MySqlCommand command = new MySqlCommand("UPDATE INTO `student` SET `StdLN`=@ln, `StdCur`=@cur, `DoB`=@dob, `StdYr`=@yr, `StdAdR`=@add, `StdTP`=@tp, `StdPaym`=@payment, `StdPaid`=@paid, `StdDue`=@due WHERE `StdFN`=@fn", connect.getConnection());
81
82     //student.insertStudent(run, fname, lname, curs, bdate, year, address, tele, payment, paid, due)
83
84     command.Parameters.Add("@fn", MySqlDbType.VarChar).Value = StdFirstName;
85     command.Parameters.Add("@ln", MySqlDbType.VarChar).Value = StdLastName;
86     command.Parameters.Add("@cur", MySqlDbType.VarChar).Value = Course;
87     command.Parameters.Add("@dob", MySqlDbType.Date).Value = StdId;
88     command.Parameters.Add("@yr", MySqlDbType.VarChar).Value = Year;
89     command.Parameters.Add("@tp", MySqlDbType.VarChar).Value = StdPhone;
90     command.Parameters.Add("@add", MySqlDbType.VarChar).Value = Address;
91     command.Parameters.Add("@payment", MySqlDbType.VarChar).Value = Payment;
92     command.Parameters.Add("@paid", MySqlDbType.VarChar).Value = Paid;
93     command.Parameters.Add("@due", MySqlDbType.VarChar).Value = Due;
94
95     connect.openConnection();
96     if (command.ExecuteNonQuery() == 1)
97     {
98         connect.closeconnect();
99         return true;
100     }
101     else
102     {
103         connect.closeconnect();
104         return false;
105     }
106 }
107

```

Figure 27

```

110 //Create a function to delete data
111 [reference]
112 public bool deleteStudent(string StdFirstName)
113 {
114     MySqlCommand command = new MySqlCommand("DELETE FROM `student` WHERE `StdFN`=@fn", connect.getConnection());
115
116     //@fn
117     command.Parameters.Add("@fn", MySqlDbType.VarChar).Value = StdFirstName;
118
119     connect.openConnection();
120     if (command.ExecuteNonQuery() == 1)
121     {
122         connect.closeconnect();
123         return true;
124     }
125     else
126     {
127         connect.closeconnect();
128         return false;
129     }
130 }
131
132 //table
133 [reference]
134 public DataTable getStudentlist()
135 {
136     MySqlCommand command = new MySqlCommand("SELECT * FROM `student`", connect.getConnection());
137     MySqlDataAdapter adapter = new MySqlDataAdapter(command);
138     DataTable table = new DataTable();
139     adapter.Fill(table);
140     return table;
141 }
142
143 }
144
145

```

Figure 28

```

34 connect.openConnection();
35 if (command.ExecuteNonQuery()==1)
36 {
37     connect.closeconnect();
38     return true;
39 }
40 else
41 {
42     connect.closeconnect();
43     return false;
44 }
45
46
47
48 // Create a function to execute the count query(total)
49 // reference
50 public string Count(string query)
51 {
52     MySqlCommand command = new MySqlCommand(query, connect.getConnection());
53     connect.openConnection();
54     string count = command.ExecuteScalar().ToString();
55     connect.closeconnect();
56     return count;
57 }
58
59 // to get the total total students
60 // reference
61 public string totalStudent()
62 {
63     return Count("SELECT COUNT(*) FROM student");
64 }
65
66 //to search student (first name, last name)
67 // reference
68 public DataTable searchStudent(string searchdata)
69 {
70     MySqlCommand command = new MySqlCommand("SELECT * FROM 'student' WHERE CONCAT('StdFN','StdLN') LIKE 'X' + searchdata + 'X'", connect.getConnection());
71     MySqlDataAdapter adapter = new MySqlDataAdapter(command);
72     DataTable table = new DataTable();
73     adapter.Fill(table);
74     return table;
75 }

```

Figure 29

Staff Class

Staff class is one of the main classes in this development. This class is utilized when adding, editing and editing student details

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using MySql.Data.MySqlClient;
7 using System.Data;
8
9
10 namespace EpupilSplashScreen
11 {
12     // reference
13     class StaffClass
14     {
15         StaffDb connect = new StaffDb ();
16
17         // reference
18         public bool insertstaff(string ID, string at, string fn, string ln, DateTime dob, string phone, string address, string design)
19         {
20             MySqlCommand command = new MySqlCommand("INSERT INTO 'staff' ('User ID', 'Account Type', 'First Name', 'Last Name', 'DoB', 'Phone', 'Address', 'Designation') VALUES (@v, @at, @fn, @ln, @dob, @phn, @address, @jb)", connect.getConnection());
21
22             // User ID, Account Type, First Name, Last Name, DoB, Phone, Address, Designation
23
24             command.Parameters.Add("@v", MySqlDbType.VarChar).Value = ID;
25             command.Parameters.Add("@at", MySqlDbType.VarChar).Value = at;
26             command.Parameters.Add("@fn", MySqlDbType.VarChar).Value = fn;
27             command.Parameters.Add("@ln", MySqlDbType.VarChar).Value = ln;
28             command.Parameters.Add("@dob", MySqlDbType.DateTime).Value = dob;
29             command.Parameters.Add("@phn", MySqlDbType.VarChar).Value = phone;
30             command.Parameters.Add("@address", MySqlDbType.VarChar).Value = address;
31             command.Parameters.Add("@jb", MySqlDbType.VarChar).Value = design;
32
33             connect.openConnection();
34             if (command.ExecuteNonQuery() == 1)
35             {
36                 connect.closeconnect();
37                 return true;
38             }
39             else
40             {
41                 connect.closeconnect();
42                 return false;
43             }
44         }
45
46         // Create a function to execute the count query(total)
47         // reference
48         public string Count(string query)
49         {
50             MySqlCommand command = new MySqlCommand(query, connect.getConnection());
51             connect.openConnection();
52             string count1 = command.ExecuteScalar().ToString();
53             connect.closeconnect();
54             return count1;
55         }
56     }
57 }

```

Figure 30

```

55 // to get the total staff members
56 2 references
57 public string totalstaff()
58 {
59     return Count1("SELECT COUNT(*) FROM staff");
60 }
61
62
63 //to search student (first name, last name)
64 1 reference
65 public DataTable searchstaff (string searchdata)
66 {
67     MySqlCommand command = new MySqlCommand("SELECT * FROM 'staff' WHERE CONCAT('First Name','Last Name') LIKE 'X' + searchdata + 'X'", connect.getConnection());
68     MySqlDataAdapter adapter = new MySqlDataAdapter(command);
69     DataTable table = new DataTable();
70     adapter.Fill(table);
71     return table;
72 }
73
74 //edit student
75 0 references
76 public bool editstaff(string ID, string at, string fn, string ln, DateTime dob, string phone, string address, string design)
77 {
78     MySqlCommand command = new MySqlCommand("UPDATE INTO 'staff' SET 'Account Type'=@at, 'First Name'=@fn, 'Last Name'=@ln, 'DoB'=@dob, 'Phone'=@phn, 'Address'=@address, 'Designation'=@jd WHERE 'User ID'=@in", connect.getConnection());
79     //User ID, 'Account Type', 'First Name', 'Last Name', 'DoB', 'Phone', 'Address', 'Designation'
80
81     command.Parameters.Add("@in", MySqlDbType.VarChar).Value = ID;
82     command.Parameters.Add("@at", MySqlDbType.VarChar).Value = at;
83     command.Parameters.Add("@fn", MySqlDbType.VarChar).Value = fn;
84     command.Parameters.Add("@ln", MySqlDbType.VarChar).Value = ln;
85     command.Parameters.Add("@dob", MySqlDbType.Date).Value = dob;
86     command.Parameters.Add("@phn", MySqlDbType.VarChar).Value = phone;
87     command.Parameters.Add("@address", MySqlDbType.VarChar).Value = address;
88     command.Parameters.Add("@jd", MySqlDbType.VarChar).Value = design;
89
90     connect.openConnection();
91     if (command.ExecuteNonQuery() == 1)
92     {
93         connect.closeconnect();
94         return true;
95     }
96     else
97     {
98         connect.closeconnect();
99         return false;
100     }
101 }

```

Figure 31

```

102 //delete staff member
103 1 reference
104 public bool deletestaff(string fn)
105 {
106     MySqlCommand command = new MySqlCommand("DELETE FROM 'staff' WHERE 'User ID'=@in", connect.getConnection());
107
108     //User ID, 'Account Type', 'First Name', 'Last Name', 'DoB', 'Phone', 'Address', 'Designation'
109
110     command.Parameters.Add("@in", MySqlDbType.VarChar).Value = fn;
111
112     connect.openConnection();
113     if (command.ExecuteNonQuery() == 1)
114     {
115         connect.closeconnect();
116         return true;
117     }
118     else
119     {
120         connect.closeconnect();
121         return false;
122     }
123 }
124 //table
125 1 reference
126 public DataTable getstafflist()
127 {
128     MySqlCommand command = new MySqlCommand("SELECT * FROM 'staff'", connect.getConnection());
129     MySqlDataAdapter adapter = new MySqlDataAdapter(command);
130     DataTable table = new DataTable();
131     adapter.Fill(table);
132     return table;
133 }
134
135
136
137
138
139
140

```

Figure 32

Class Class

This class handles adding, deleting and editing data pertaining to the class.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using MySql.Data.MySqlClient;
7  using System.Data;
8
9  namespace EpupilsSplashScreen
10 {
11     2 references
12     class ClassClass
13     {
14
15         DBconnect connect = new DBconnect();
16
17         2 references
18         public bool insertclass(string ClassN, string ClassName, string Seat, string AC)
19         {
20             MySqlCommand command = new MySqlCommand("INSERT INTO `class`(`ClassN`, `ClassName`, `Seat`, `A/C`) VALUES (@cn, @cname, @seat, @ac)", connect.getconnection());
21
22             command.Parameters.Add("@cn", MySqlDbType.VarChar).Value = ClassN;
23             command.Parameters.Add("@cname", MySqlDbType.VarChar).Value = ClassName;
24             command.Parameters.Add("@seat", MySqlDbType.VarChar).Value = Seat;
25             command.Parameters.Add("@ac", MySqlDbType.VarChar).Value = AC;
26
27             connect.openConnect();
28             if (command.ExecuteNonQuery() == 1)
29             {
30                 connect.closeconnect();
31                 return true;
32             }
33             else
34             {
35                 connect.closeconnect();
36                 return false;
37             }
38         }
39     }
```

Figure 33

```
63
64
65     1 reference
66     public bool deleteclass(string ClassN)
67     {
68         MySqlCommand command = new MySqlCommand("DELETE FROM `class` WHERE `ClassN`=@cn", connect.getconnection());
69
70         command.Parameters.Add("@cn", MySqlDbType.VarChar).Value = ClassN;
71
72         connect.openConnect();
73         if (command.ExecuteNonQuery() == 1)
74         {
75             connect.closeconnect();
76             return true;
77         }
78         else
79         {
80             connect.closeconnect();
81             return false;
82         }
83     }
84     //table
85     1 reference
86     public DataTable getClasslist()
87     {
88         MySqlCommand command = new MySqlCommand("SELECT * FROM `class`", connect.getconnection());
89         MySqlDataAdapter adapter = new MySqlDataAdapter(command);
90         DataTable table = new DataTable();
91         adapter.Fill(table);
92         return table;
93     }
94 }
95
96
```

Figure 34

```

40 0 references
41 public bool updateclass(string ClassN, string ClassName, string Seat, string AC)
42 {
43     MySqlCommand command = new MySqlCommand("UPDATE INTO `class` SET `ClassName`=@cname, `Seat`=@seat, `A/C`=@ac WHERE `ClassN`=@cn", connect.getconnection());
44     command.Parameters.Add("@cn", MySqlDbType.VarChar).Value = ClassN;
45     command.Parameters.Add("@cname", MySqlDbType.VarChar).Value = ClassName;
46     command.Parameters.Add("@seat", MySqlDbType.VarChar).Value = Seat;
47     command.Parameters.Add("@ac", MySqlDbType.VarChar).Value = AC;
48
49     connect.openConnect();
50     if (command.ExecuteNonQuery() == 1)
51     {
52         connect.closeconnect();
53         return true;
54     }
55     else
56     {
57         connect.closeconnect();
58         return false;
59     }
60 }
61
62 3 references
63 public bool deleteclass(string ClassN)
64 {
65     MySqlCommand command = new MySqlCommand("DELETE FROM `class` WHERE `ClassN`=@cn", connect.getconnection());
66
67     command.Parameters.Add("@cn", MySqlDbType.VarChar).Value = ClassN;
68
69     connect.openConnect();
70     if (command.ExecuteNonQuery() == 1)
71     {
72         connect.closeconnect();
73         return true;
74     }
75     else
76     {
77         connect.closeconnect();
78         return false;
79     }
80 }
81
82

```

Figure 35

MapClass class

This class is used to add, edit ad delete information about the school.

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using MySql.Data.MySqlClient;
7  using System.Data;
8
9
10 namespace EpupilSplashScreen
11 {
12     2 references
13     class mapsubClass
14     {
15         DBconnect connect = new DBconnect();
16
17         2 references
18         public bool insertclass(DateTime dato, string Stsub, string Sttime, string Class) //to insert data to database
19         {
20             MySqlCommand command = new MySqlCommand("INSERT INTO `mapsubclass` ( `Date`, `Subject`, `Time`, `Class`) VALUES (@da, @sj, @ti, @cl)", connect.getconnection());
21             command.Parameters.Add("@da", MySqlDbType.VarChar).Value = dato;
22             command.Parameters.Add("@sj", MySqlDbType.VarChar).Value = Stsub;
23             command.Parameters.Add("@ti", MySqlDbType.VarChar).Value = Sttime;
24             command.Parameters.Add("@cl", MySqlDbType.VarChar).Value = Class;
25
26
27             connect.openConnect();
28             if (command.ExecuteNonQuery() == 1)
29             {
30                 connect.closeconnect();
31                 return true;
32             }
33             else
34             {
35                 connect.closeconnect();
36                 return false;
37             }
38
39         }
40
41     }

```

Figure 35


```

42 0 reference
43 public bool editclass(DateTime dato, string Stsub, string Sstime, string Class) //to insert data to database
44 {
45     MySqlCommand command = new MySqlCommand("UPDATE INTO `mapsubclass` SET `Date`=@da, `Time`=@ti, `Class`=@cl WHERE `Subject`=@sj ", connect.getconnection());
46
47     command.Parameters.Add("@da", SqlDbType.VarChar).Value = dato;
48     command.Parameters.Add("@sj", SqlDbType.VarChar).Value = Stsub;
49     command.Parameters.Add("@ti", SqlDbType.VarChar).Value = Sstime;
50     command.Parameters.Add("@cl", SqlDbType.VarChar).Value = Class;
51
52     connect.openConnect();
53     if (command.ExecuteNonQuery() == 1)
54     {
55         connect.closeconnect();
56         return true;
57     }
58     else
59     {
60         connect.closeconnect();
61         return false;
62     }
63 }
64
65 1 reference
66 public DataTable searchsul(string searchdata)
67 {
68     MySqlCommand command = new MySqlCommand("SELECT * FROM `mapsubclass` WHERE CONCAT(`Date`, `Subject`) LIKE '%" + searchdata + "%'", connect.getconnection());
69     MySqlDataAdapter adapter = new MySqlDataAdapter(command);
70     DataTable table = new DataTable();
71     adapter.Fill(table);
72     return table;
73 }
74
75 1 reference
76 public bool deleteclass(string Sstime)
77 {
78     MySqlCommand command = new MySqlCommand("DELETE FROM `mapsubclass` WHERE `Time`=@ti", connect.getconnection());
79
80     // @ti
81     command.Parameters.Add("@ti", SqlDbType.VarChar).Value = Sstime;
82
83     connect.openConnect();
84     if (command.ExecuteNonQuery() == 1)
85     {
86         connect.closeconnect();
87         return true;
88     }
89     else
90     {
91         connect.closeconnect();
92         return false;
93     }
94 }

```

Figure 36

```

96 //table
97 1 reference
98 public DataTable getclasslist()
99 {
100     MySqlCommand command = new MySqlCommand("SELECT * FROM `mapsubclass` ", connect.getconnection());
101     MySqlDataAdapter adapter = new MySqlDataAdapter(command);
102     DataTable table = new DataTable();
103     adapter.Fill(table);
104     return table;
105 }
106
107 }
108
109

```

Figure 37

School Class

This class is used to add, edit ad delete information about the school.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using MySql.Data.MySqlClient;
7  using System.Data;
8
9  namespace EpupilsSplashScreen
10 {
11     2 references
12     class SchoolClass
13     {
14         SchoolDb connect = new SchoolDb();
15
16         2 references
17         public bool insertschool(string Sname, string Snumber, string Dean)
18         {
19             MySqlCommand command = new MySqlCommand("INSERT INTO `school` (`SchoolIN`, `SRN`, `Dean`) VALUES (@sch, @rn, @de)", connect.getconnection);
20
21             command.Parameters.Add("@sch", MySqlDbType.VarChar).Value = Sname;
22             command.Parameters.Add("@rn", MySqlDbType.VarChar).Value = Snumber;
23             command.Parameters.Add("@de", MySqlDbType.VarChar).Value = Dean;
24
25             connect.openConnect();
26             if (command.ExecuteNonQuery() == 1)
27             {
28                 connect.closeconnect();
29                 return true;
30             }
31             else
32             {
33                 connect.closeconnect();
34                 return false;
35             }
36         }
37     }
38 }
```

Figure 37

```
39
40 0 references
41 public bool updatetschool(string Sname, string Snumber, string Dean)
42 {
43     MySqlCommand command = new MySqlCommand("UPDATE INTO `school` SET `SRN` = @rn, `Dean`=@de WHERE `SchoolIN`=@sch", connect.getconnection);
44
45     command.Parameters.Add("@sch", MySqlDbType.VarChar).Value = Sname;
46     command.Parameters.Add("@rn", MySqlDbType.VarChar).Value = Snumber;
47     command.Parameters.Add("@de", MySqlDbType.VarChar).Value = Dean;
48
49     connect.openConnect();
50     if (command.ExecuteNonQuery() == 1)
51     {
52         connect.closeconnect();
53         return true;
54     }
55     else
56     {
57         connect.closeconnect();
58         return false;
59     }
60 }
61
62 //table
63 1 reference
64 public DataTable getschoollist()
65 {
66     MySqlCommand command = new MySqlCommand("SELECT * FROM `school`", connect.getconnection);
67     MySqlDataAdapter adapter = new MySqlDataAdapter(command);
68     DataTable table = new DataTable();
69     adapter.Fill(table);
70     return table;
71 }
72
73
74
75
76
77
78
79
80 }
```

Figure 38

Subject Class

This class is used to add, edit and delete information about the subjects and courses.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.Data;
7  using MySql.Data.MySqlClient;
8
9
10 namespace EpupilsSplashScreen
11 {
12     2 references
13     class SubjectClass
14     {
15         subjectdb connect = new subjectdb();
16
17         1 reference
18         public bool insertcourse(string curs, string curids, string ssl, string eel, string aal, string ll1,
19             string llh1, string ss2, string ee2, string aa2, string ll2, string llh2)
20         {
21             MySqlCommand command = new MySqlCommand("INSERT INTO `course`(`Course`, `CourseID`, `Subject 01`, `Exam 01`,`" +
22                 " `Assig. 01`, `Lecturer 01`, `Lec.hours 01`, `Subject 02`, `Exam 02`, `Assig. 02`, `Lecturer 02`, `Lec.hours 02`) VALUES (@cu,@cid, @s1,@e1,@l1,@lh1, @s2,@e2,@l2,@lh2)", connect.getconnection());
23
24             command.Parameters.Add("@cu", MySqlDbType.VarChar).Value = curs;
25             command.Parameters.Add("@cid", MySqlDbType.VarChar).Value = curids;
26             command.Parameters.Add("@s1", MySqlDbType.VarChar).Value = ssl;
27             command.Parameters.Add("@e1", MySqlDbType.VarChar).Value = eel;
28             command.Parameters.Add("@l1", MySqlDbType.VarChar).Value = aal;
29             command.Parameters.Add("@lh1", MySqlDbType.VarChar).Value = ll1;
30             command.Parameters.Add("@s2", MySqlDbType.VarChar).Value = ss2;
31             command.Parameters.Add("@e2", MySqlDbType.VarChar).Value = ee2;
32             command.Parameters.Add("@a2", MySqlDbType.VarChar).Value = aa2;
33             command.Parameters.Add("@l2", MySqlDbType.VarChar).Value = ll2;
34             command.Parameters.Add("@lh2", MySqlDbType.VarChar).Value = llh2;
35
36             connect.openConnection();
37             if (command.ExecuteNonQuery() == 1)
38             {
39                 connect.closeconnect();
40                 return true;
41             }
42             else
43             {
44                 connect.closeconnect();
45                 return false;
46             }
47         }
48     }
49 }
50
51 //table
```

Figure 39

```
52
53
54
55     1 reference
56     public DataTable getcourselist()
57     {
58         MySqlCommand command = new MySqlCommand("SELECT * FROM `course`", connect.getconnection());
59         MySqlDataAdapter adapter = new MySqlDataAdapter(command);
60         DataTable table = new DataTable();
61         adapter.Fill(table);
62         return table;
63     }
64 }
65
```

Figure 40

CHAPTER 04

TESTING

The test cases used by the system developer to correct the accuracy or the correctness of the system are given below.

| Test No. | Test Name | Testing Process | Expected Result | Actual Result |
|----------|--|---|---|----------------------------|
| 1. | Splash Screen Loading. | Run the application. | Loading the splash screen at the centre of the screen. | As expected. |
| 2. | Selecting the account type. | Click on the admin or staff account button. | Load the login page | As expected. |
| 3. | Entering the correct user name and password. | If admin account is selected then Adding 'admin1' as the username and 'admin1' password. If staff account is selected then adding 'staff' as the username and 'staff' as the password. | Loading the main dash board for staff or admin. | As expected. |
| 4. | Entering the correct user name and password. | If admin account is selected then Adding 'admin' as the username and 'admin' password. If staff account is selected then adding 'staff1' as the username and 'staff1' as the password. | Error message. | As expected. |
| 5. | Checking the buttons on the dashboard. | Click on the 5 buttons on the left navigation panel, if login as an admin. Click on the 4 buttons on the left navigation panel, if log in as a staff. | Relevant screen will appear on the main panel. Also, Number of employees and students must appear. | As expected. |
| 6. | Navigation side bar | When click on the buttons in the navigation panel. | The side bar must appear on top of the | The side bar didn't appear |

| | | | | |
|----|---|---|---|---|
| | | | clicked button. | properly. When selected generate report button. |
| 7. | Staff ability to map subject and class | <p>Click on Save, Edit, Clear and delete bottoms to check whether all the relevant functions work properly.</p> <p>Click on an item on table to select an item.</p> | The save button shall add new entries to the table shown below. | As expected. |
| | | | Clicking on the item from the below table will furnish the data to relevant text boxes. | As expected. |
| | | | Clicking on the clear button will remove all the existing data in the text boxes. | As expected. |
| | | | Clicking on the delete button will remove all the existing data of the selected item from the database. | As expected. |
| 8. | Staff ability to map subject and student. | <p>Click on Save, Edit, Clear and delete bottoms to check whether all the relevant functions work properly.</p> <p>Click on an item on table to select an item.</p> | The save button shall add new entries to the table shown below. | As expected. |
| | | | Clicking on the item from the below table will furnish the data to relevant text boxes. | As expected. |

| | | | | |
|-----|--|--|---|--------------|
| | | | Clicking on the clear button will remove all the existing data in the text boxes. | As expected. |
| | | | Clicking on the delete button will remove all the existing data of the selected item from the database. | As expected. |
| 9. | Staff Adding students marks | Click on Save, Edit, delete and Clear bottoms to check whether all the relevant functions work properly. Click on an item on table to select an item. | The save button shall add new entries to the table shown below. | As expected. |
| | | | Clicking on the item from the below table will furnish the data to relevant text boxes. | As expected. |
| | | | Clicking on the clear button will remove all the existing data in the text boxes. | As expected. |
| 10. | Clicking on the report and Taking printouts. | Click on the print button to print the report. | Printing the relevant reports. | As expected. |
| 11. | Admins ability to manage class details. | Click on Save, Edit, delete and Clear bottoms to check whether all the relevant functions work properly. Click on an item on table to select an item. | The save button shall add new entries to the table shown below. | As expected. |
| | | | Clicking on the item from the below table will furnish the data to relevant text boxes. | As expected. |
| | | | Clicking on the clear button will remove all the existing data in the | As expected. |

| | | | | |
|-----|---|--|---|--------------|
| | | | text boxes. | |
| | | | Clicking on the delete button will remove all the existing data of the selected item from the database. | As expected. |
| 12. | Admins ability to manage student details. | Click on Save, Edit, delete and Clear bottoms to check whether all the relevant functions work properly. Click on an item on table to select an item. | The save button shall add new entries to the table shown below. | As expected. |
| | | | Clicking on the item from the below table will furnish the data to relevant text boxes. | As expected. |
| | | | Clicking on the clear button will remove all the existing data in the text boxes. | As expected. |
| | | | Clicking on the clear button will remove all the existing data in the text boxes. | As expected. |
| 13. | Admins ability to manage staff details. | Click on Save, Edit, delete and Clear bottoms to check whether all the relevant functions work properly. Click on an item on table to select an item. | The save button shall add new entries to the table shown below. | As expected. |
| | | | Clicking on the item from the below table will furnish the data to relevant text boxes. | As expected. |
| | | | Clicking on the clear button will remove all the existing data in the | As expected. |

| | | | | |
|-----|--|--|---|--------------|
| | | | text boxes. | |
| | | | Clicking on the clear button will remove all the existing data in the text boxes. | As expected. |
| 14. | Admins ability to manage subject and course details. | Click on Save bottoms to check whether all the relevant functions work properly. | The save button shall add new entries to the table shown below. | As expected. |
| | | | Clicking on the item from the below table will furnish the data to relevant text boxes. | As expected. |
| | | | Clicking on the clear button will remove all the existing data in the text boxes. | As expected. |
| | | | Clicking on the clear button will remove all the existing data in the text boxes. | As expected. |
| 15. | Admins ability to manage school details | Click on Save, Edit and Clear bottoms to check whether all the relevant functions work properly. | The save button shall add new entries to the table shown below. | As expected. |
| | | | Clicking on the item from the below table will furnish the data to relevant text boxes. | As expected. |
| | | | The edit button shall add edited entries to the table shown below. | As expected. |

CONCLUSION

- **Assumptions**

When developing the following assumptions were assumed. It was assumed that;

- Only two types of employee (staff and admin) will access this application.
- Also, the password and username will be given from the administrator.
- Only limited courses are offered in the university.
- Only limited number of subjects are offered in a course(2 subjects)
- One subject consists of only one exam and assignment.
- The Final grade was developed based on the exam marks and the assignment marks.

- **Developers Review**

This assignment provided a great opportunity to develop my understanding on the .Net framework and also the C# language. The experience I got by designing windows forms was a new experience and I am confident that experience will be an added advantage in the future.

As the first step of development the user interfaces were developed. Though, it was taken a relatively short time to develop the user interfaces. Coding with a new language was time consuming. On the other hand, the designing process also took a considerable amount of time. Also, the vague terms in the course instructions were hard to define, and in that regarding the guidance received from the lecturer was commendable. Also, during the development process the certain decisions taken during the designed process was questioned and had to be reversed. As an example the user case descriptions were inaccurate and were vague.

The main reason for not being able to complete the development on time was the inability to commit fix number off hours during the day. Ultimately that resulted in a massive backlog of work. However, the final delivery was satisfactory and I the experience I received doing this assignment is immense.

- **Future Development**

This system can be further developed in many areas. Including,

- Can allow students to access and view their upcoming classes and exam marks.
- Addition of a picture of every employee and student.
- Addition of new developments
- Also, can add other features like student online attendance registry.

