

Advance Topics in Software Engineering

CSE-6324-004

School Management System

Project Report

Prof: Farnaz Farahanipad

Team Members

Pavan kumar Gannarapu -1002076404

Sai Rohit Kalyan Gandham – 1002070724

11/09/2023

Table of Contents

- 1 Abstract**
- 2 Introduction**
- 3 Project-scope**
- 4 Requirements-analysis**
- 5 Project planning**
- 6 Design-and-architecture**
- 7 Implementation**
- 8 Testing**
- 9 Maintenance**
- 10 Conclusion**
- 11 Recommendations**
- 12 References**

1. Abstract

Our user-friendly School Management System serves as both a powerful administrative tool and a safe communication platform. This tool streamlines difficult administrative tasks and can be installed on any system. It assures data security throughout, allowing administrators to manage activities like attendance and grading more quickly. The integrated email feature allows for simple and safe communication among stakeholders, encouraging a collaborative environment. With a focus on user-friendliness and security, our technology enables educational institutions to easily improve administrative efficiency and communication channels.

2. Introduction

Our School Management System, which is at the forefront of educational innovation, emerges as a critical response to the dynamic issues that modern educational institutions face. We propose a comprehensive system adapted to the individual needs of each school in response to the pressing need for efficient school management. This proposal demonstrates our commitment to empowering educational missions by developing a cutting-edge School Management System. We hope to upgrade administrative procedures by proactively addressing essential objectives and goals, reaffirming the institution's commitment to providing top-tier education. Join us on this transformative journey, as innovation meets utility to reshape the school administration environment..

3. Project Scope

The project's scope includes the following important areas:

1. **Database Development:** Our project begins with the development of a solid database structure. This foundation will securely contain critical information, such as student and faculty information, academic records, and administrative data, while also maintaining data integrity and accessibility.
2. **User Interface Design:** We Created an intuitive and user-friendly interface is part of the development process. This design will meet the various needs of administrators, instructors, students, and parents, resulting in a seamless and engaging user experience.
3. **Feature Development:** We prioritize the deployment of critical features required for effective school administration. These include student enrollment, attendance monitoring, fee management, academic records management, and sophisticated communication tools, all of which improve the system's overall functionality.
4. **Security safeguards:** Because data security is so important, our project includes advanced safeguards such as data encryption. This protects sensitive information while also ensuring the integrity and confidentiality of educational data.
5. **Integration Capabilities:** The system is meant to be versatile and capable of integrating seamlessly with other educational systems and resources. This interoperability creates a unified educational ecosystem, allowing for a more coordinated approach to education management.

The scope of our project is the holistic construction of a School Management System, which includes everything from a secure database and user-friendly interface to important features, tough security measures, and integration possibilities. This holistic approach strives to redefine and raise school administration norms.

4. Requirement Analysis

Client Requirements

- 1) **Automation of Administrative Tasks:** Streamline and automate processes such as student enrollment, attendance tracking, fee management, and payroll for staff members.

- 2) **Academic Records Management:** Manage student academic records such as grades, transcripts, , and class registrations fee.
- 3) **Communication Enhancement:** Using messaging, notifications, and announcements, makes it easier for instructors, students, and parents to interact with each other.
- 4) **Reporting and Analytics:** Provide comprehensive reporting and analytics capabilities to track student performance, attendance trends, and financial data.
- 5) **Security and Compliance:** Ensure sensitive data security and confidentiality while adhering to data protection rules.
- 6) **User-Friendly Interface:** Create an easy-to-use interface for administrators, professors and students, to use

Based on the client requirements we have created functional Requirements ,Non functional Requirements ,use cases and user stores

Functional Requirements:

1. **User Authentication:** Users must be able to securely log in with unique credentials
2. **User Roles:** The system should distinguish between user roles with differing levels of access and permissions, guaranteeing proper data visibility.
3. **Database Management:** Store and manage student and faculty information, academic records, and administrative data in an organized database in an efficient manner.
4. **User Interface:** Create an intuitive and responsive user interface that allows all user types to navigate easily.
5. **Student Enrollment:** Create a module that allows administrators to enroll new students and collect necessary information and also allow to edit the details.
6. **Teacher Enrollment:** Create a module that allows administrators to enroll new teacher and collect necessary information and also allow to edit the details.
7. **Monitoring Attendance:** Allow instructors to record and monitor student attendance, by marking the student day-day present/absent and creating reports for study.
8. **Fee Management:** Provide a module that allows administrators to manage and track fee payments and Email the payment receipts.
9. **Academic records management:** enables the development, storage, and retrieval of academic records such as grades , transcripts and student attendance.
10. **Communication tools:** Integrate secure communication technologies, such as emails or messaging, to allow for smooth collaboration across stakeholders.
11. **Help:** Create a module help in which user can email if there is any issue with the application.

Non-Functional Requirements:

1. **Security:** To protect sensitive information, use strong data encryption and secure authentication procedures.
2. **Scalability:** To support potential future development, design the system to handle a scalable amount of users and data points.
3. **Performance:** Ensure optimal system performance with short response times for user engagements.
4. **Reliability:** The system should be available and dependable, with low maintenance downtime.
5. **Usability:** Prioritize a user-friendly interface with clear navigation and low user training requirements.

Use Cases:

1. **User Login:** As a user, I wish to log into the system securely using my unique credentials.
2. **Enroll a Student:** As an administrator, I want to be able to easily enroll a new student and edit student details and capture their pertinent information.
3. **Attendance:** As an instructor, I want to keep track of and monitor student attendance in each class.
4. **Manage Fees:** As an administrator, I want to be able to manage and track fee payments easily, as well as provide alerts for overdue payments.
5. **Academic detail:** As a teacher I want to add exam marks and attendance tracking for each student.
6. **Access Academic Records:** I wish to access academic records, such as grades and transcripts, as a student or parent.
7. **User assistance:** As a user I need help in send email if there is un responsive element to intimate to developer

User stories:

1. As an instructor, I want a simple interface for entering and tracking student grades.
2. As an instructor, I want a simple interface for entering and tracking student attendance.
3. As an administrator, I want to be able to generate detailed data on student enrollment, attendance, and academic performance.
4. As an administrator I want to know fee details of each student and send email to the of fee payment receipts.
5. As a system administrator, I wish to undertake routine maintenance without interfering with the system's availability.
6. As an administrator I want to send email to know abilities of the application

Updates during project

There were major revisions and updates to requirements throughout the project. Initially focusing on essential functionality, the decision was made throughout development to improve the entire user experience by improving the user interface (GUI). Recognizing the significance of visual appeal, this enhancement intended to improve engagement and accessibility.

Furthermore, the addition of new features, such as files reflecting total teachers and fees collected, demonstrates an adaptive reaction to changing needs. These upgrades bring the system more in line with user expectations and give administrators with useful information.

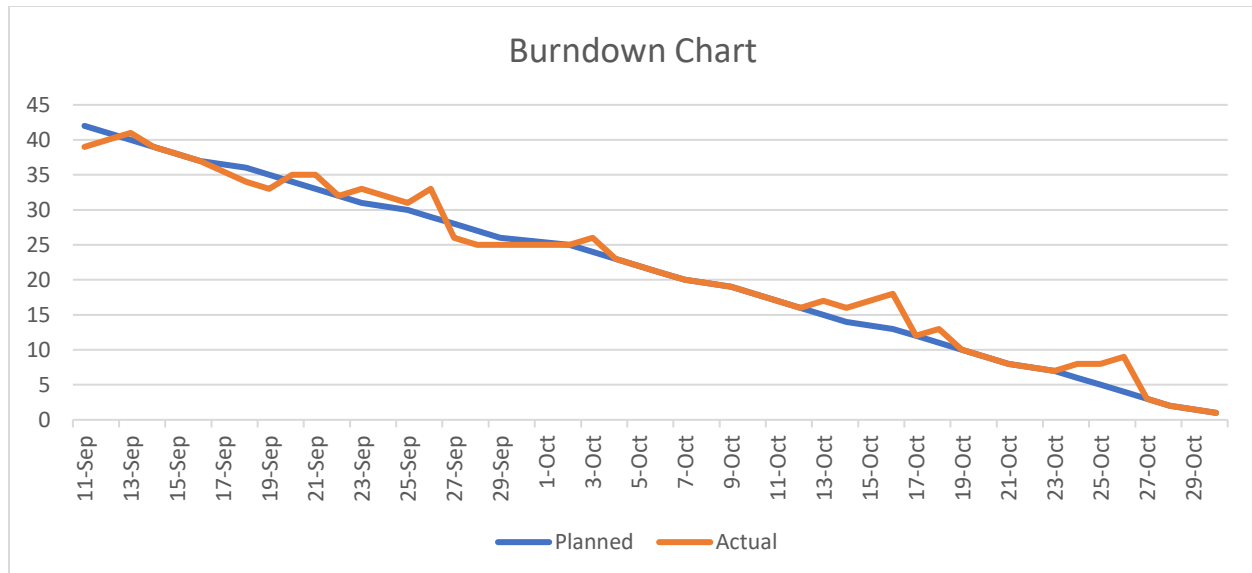
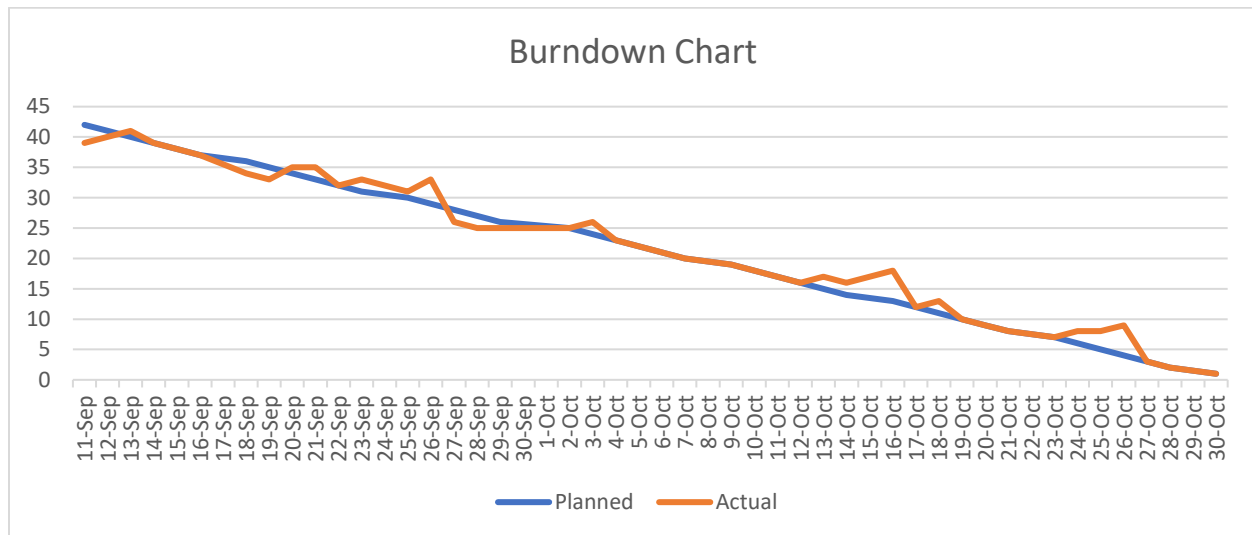
These modifications emphasize the project's adaptability and reactivity, displaying a dedication to continual improvement and user pleasure.

5. Project Planning

Gantt chart

[illegible]

Burndown chart



Reasons for being ahead or behind:

Ahead of schedule:

The project experienced fast development on 11-Sep, 12-Sep, and 18-Sep to 20-Sep, as well as 27-Sep to 28-Sep during the phases of requirement analysis, database building, and documentation. Efficient teamwork and a clear grasp of needs aided in the execution of tasks more quickly. Effective database development and documentation practices aided in the simplification of processes, allowing for faster progress. This encouraging trend demonstrates the value of rigorous planning and execution throughout the early stages of a project.

Behind of schedule:

Dates: 20-Sep, 21-Sep, 23-Sep, 25-Sep, 26-Sep, 2-Oct, 12-Oct to 17-Oct, and 24-Oct to 27-Oct.

Delay in project mainly occurred in development and integration phases, these dates could indicate issues with deployment, interface integration and testing processes. These phases frequently entail sophisticated coordination and debugging, affecting overall progress.

Project timeline:

Task Name	Start Date	End Date	Assigned To	Status
Project Planning and Analysis	09/11/23	09/15/23		Completed
Requirement gathering	09/11/23	09/12/23	Pavankumar G	Completed
Define project scope and objectives	09/13/23	09/15/23	Rohit	Completed
Identify key needs	09/13/23	09/15/23	Pavankumar G	Completed
System Design	09/16/23	09/21/23		Completed
ER- Design	09/16/23	09/16/23	Rohit	Completed
Database design	09/18/23	09/19/23	Pavankumar G	Completed
User interface design	09/20/23	09/21/23	Pavankumar G	Completed
Development	09/21/23	10/24/23		Completed
Database implementation	09/21/23	09/22/23	Rohit	Completed
Database Tables	09/21/23	09/21/23	Rohit	Completed
Connection	09/22/23	09/22/23	Rohit	Completed
Front-end development	09/21/23	10/24/23	Pavankumar G	Completed
Login page	09/21/23	09/21/23	Pavankumar G	Completed
Student page	09/22/23	09/22/23	Pavankumar G	Completed
Teacher page	09/24/23	09/26/23	Pavankumar G	Completed
Admin page	09/27/23	09/28/23	Pavankumar G	Completed
report card	09/29/23	09/29/23	Pavankumar G	Completed
welcome page	10/02/23	10/02/23	Pavankumar G	Completed
Menu	10/02/23	10/02/23	Pavankumar G	Completed
Fee payment	10/03/23	10/04/23	Pavankumar G	Completed
Edit teacher & student pages	10/05/23	10/06/23	Pavankumar G	Completed
Add student & teacher pages	10/07/23	10/11/23	Pavankumar G	Completed
Analysis of student	10/13/23	10/18/23	Pavankumar G	Completed
Attendance	10/19/23	10/24/23	Pavankumar G	Completed
Back-end Development	09/22/23	10/17/23	Rohit	Completed
Login page	09/22/23	09/22/23	Rohit	Completed
Student page	09/23/23	09/25/23	Rohit	Completed
Teacher page	09/26/23	09/27/23	Rohit	Completed
Admin page	09/28/23	09/29/23	Rohit	Completed
report card	10/02/23	10/02/23	Rohit	Completed
welcome page	10/02/23	10/03/23	Rohit	Completed
Menu	10/04/23	10/04/23	Rohit	Completed
Fee payment	10/05/23	10/06/23	Rohit	Completed
Edit teacher & student pages	10/06/23	10/09/23	Rohit	Completed
Add student & teacher pages	10/12/23	10/13/23	Rohit	Completed

Analysis of student	10/16/23	10/16/23	Rohit	Completed
Attendance	10/17/23	10/17/23	Rohit	Completed
Testing	10/18/23	10/24/23	Rohit	Completed
Manual Testing	10/18/23	10/20/23	Pavankumar G	Completed
White Box Testing	10/18/23	10/20/23	Rohit	Completed
Automation Testing	10/20/23	10/24/23	Rohit	Completed
Documentation	10/23/23	10/23/23	Pavankumar G	Completed
User documentation	10/23/23	10/23/23	Pavankumar G	Completed
Technical documentation	10/23/23	10/23/23	Rohit	Completed
Deployment and support	10/24/23	11/03/23	Rohit	In Progress
Deployment planning and execution	10/24/23	10/26/23	Rohit	Completed
User training	10/25/23	10/27/23	Pavankumar G	Completed
Post-launch support	10/27/23	11/03/23	Rohit	Completed

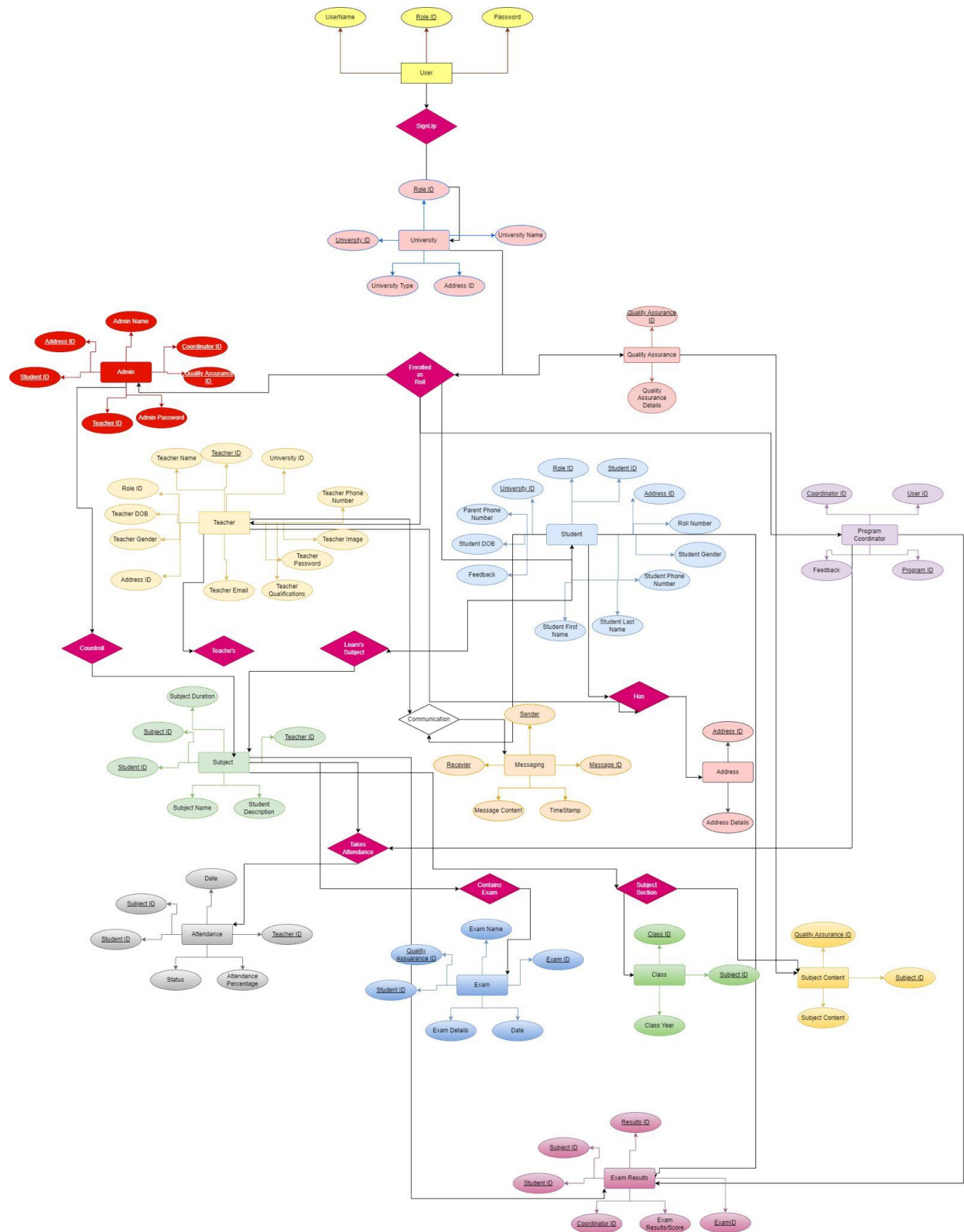
Responsibilities

Task Name	Assigned To
Project Planning and Analysis	
Requirement gathering	Pavankumar G
Define project scope and objectives	Rohit
Identify key needs	Pavankumar G
System Design	
ER- Design	Rohit
Database design	Pavankumar G
User interface design	Pavankumar G
Development	
Database implementation	Rohit
Database Tables	Rohit
Connection	Rohit
Front-end development	Pavankumar G
Login page	Pavankumar G
Student page	Pavankumar G
Teacher page	Pavankumar G
Admin page	Pavankumar G
report card	Pavankumar G
welcome page	Pavankumar G
Menu	Pavankumar G
Fee payment	Pavankumar G
Edit teacher & student pages	Pavankumar G
Add student & teacher pages	Pavankumar G
Analysis of student	Pavankumar G
Attendance	Pavankumar G

Back-end Development	Rohit
Login page	Rohit
Student page	Rohit
Teacher page	Rohit
Admin page	Rohit
report card	Rohit
welcome page	Rohit
Menu	Rohit
Fee payment	Rohit
Edit teacher & student pages	Rohit
Add student & teacher pages	Rohit
Analysis of student	Rohit
Attendance	Rohit
Testing	Rohit
Manual Testing	Pavankumar G
Automation Testing	Rohit
Documentation	Pavankumar G
User documentation	Pavankumar G
Technical documentation	Rohit
Deployment and support	Rohit
Deployment planning and execution	Rohit
User training	Pavankumar G
Post-launch support	Rohit

6. Design and Architecture

Er-Diagram



- Relationship between students and enrollments:
- A student may be enrolled in multiple classes.
- StudentID in Enrollments references StudentID in Students.
- Relationship between teachers and classes:
- A single teacher can instruct many classes.
- TeacherID in Classes referencing TeacherID in Teachers is a foreign key.
- Relationship between classes and enrollments:

Multiple students may enroll in the same class.

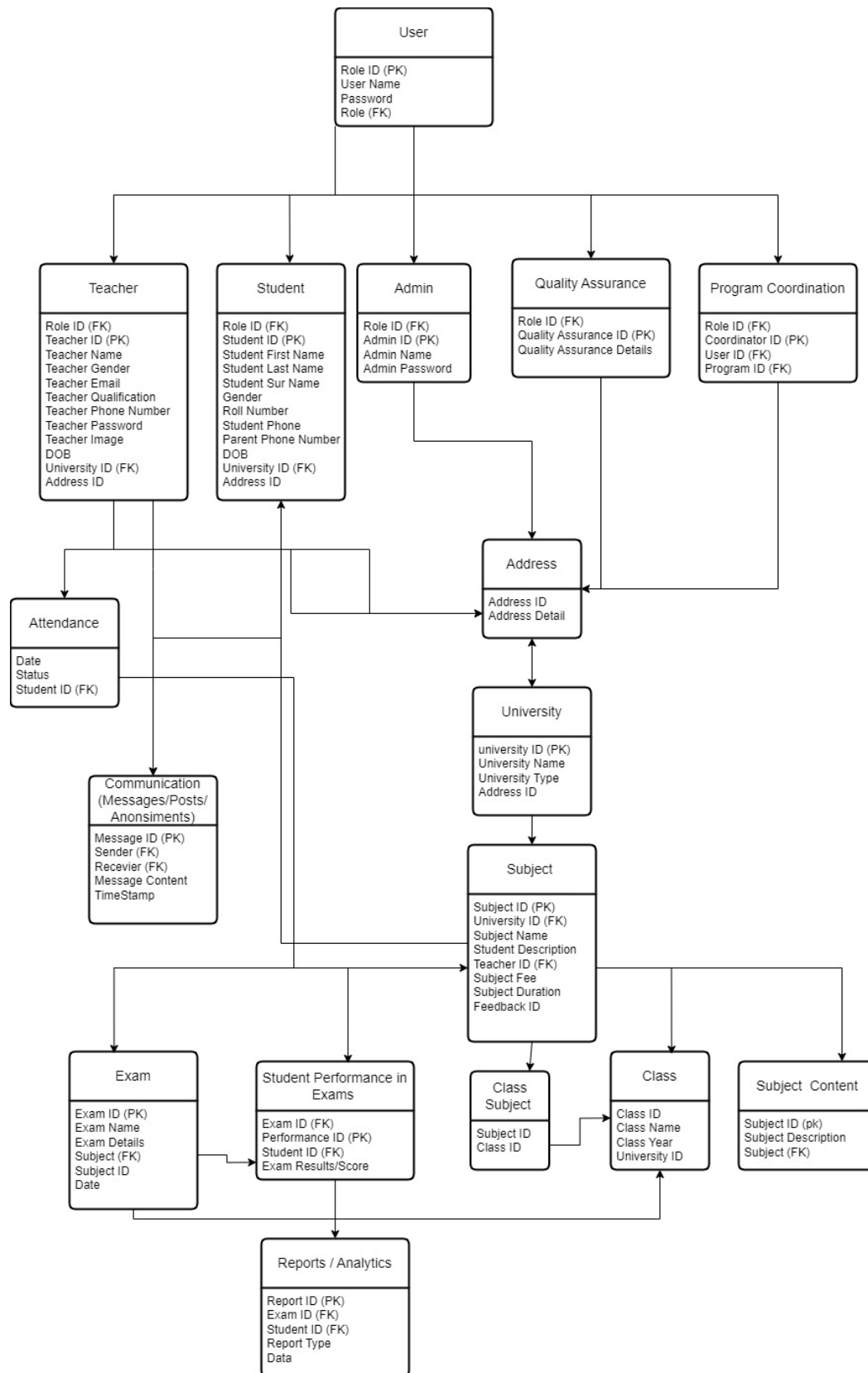
Foreign Key: Enrollment ClassID referencing ClassID in Classes.

Relationship between Students and Attendance:

DB-Schema

- Each student can be registered in numerous classes (Students to Enrollments subjects has a one-to-many relationship).
- Each instructor can teach several classes (teachers to classes have a one-to-many relationship).
- Each class can have many enrollments (Classes to Enrollments has a one-to-many relationship).
- Attendance is related to both students and classes (a many-to-many relationship exists between students and classes).
- Fees are linked to specific students (a one-to-many relationship: Students to Fees).

This schema provides the basic functions of a School Management System, such as monitoring students, classes, teachers, enrollments, attendance, and fees.



7. Implementation

Environment:

OS -- Windows
GUI -- QT Designer
Back-End -- Python
Database -- SqlLite

Tools:

PyQt
Pycharm
DBeaver -- For Database Manager

Major Components

1. Student Administration Module: Manages student records, enrollment details, attendance, performance, and parent/guardian communication.
2. Teacher Management Module: This module is in charge of managing teacher data, subject allocation, timetables, and communication with administration.
3. Course Management Module: Curriculum information, course structures, subject distribution, and academic calendars are all handled by the Course Management Module.
4. Attendance Management Module: This module monitors student attendance, creates reports, and manages absenteeism records.
5. Exam and Grading Module: This module manages exam schedules, grades, mark entry, and result production.
6. Financial Module: This module is in charge of fee collection, financial records, salary management, and budgeting.

Challenges

1. Automation via GUI: Due to differences in screen resolutions, UI modifications, and platform behaviors, automating tasks in an application using PyAutoGUI or Pywinauto is difficult. It can be difficult to ensure stability across several environments.
2. Database Operations: - When integrating database operations inside a broader system, establishing connections, guaranteeing accurate data entry, and resolving exceptions or failures can be difficult.
3. Email Configuration: Sending emails via SMTP requires proper authentication, and utilizing Gmail as the SMTP server may encounter challenges owing to security settings such as two-factor authentication or app-specific passwords.
4. Python Application Packaging: Creating executable files (.exe) from Python scripts can be tricky, especially when the application includes complicated dependencies, third-party libraries, or several modules.
5. Data Entry Automation: - When automating data entry into forms or apps, it may be difficult to simulate actual user behavior and handle different input scenarios such as text fields, combo boxes, or date pickers.
6. Error Handling: It is critical to handle exceptions when interacting with databases, file operations, or external services (such as sending emails). To maintain system robustness, proper error handling and graceful fallback procedures are required.

Third party libraries

1. SonarQube:
 - a. SonarQube is an open-source tool for continuous code quality inspection. It analyzes static code to detect code smells, defects, and security issues. SonarQube gives detailed statistics and metrics to developers, assisting them in maintaining high-quality code throughout the development process.
 - b. Key characteristics include:
 - Metrics for Code Quality
 - Detection of Security Vulnerabilities
 - Inspection on a continuous basis
 - Connection to CI/CD Pipelines
2. Designer of QT:
 - a. QT Designer is a graphical user interface (GUI) design tool used in the Qt application framework for developing and designing interfaces. It streamlines the user interface design process by allowing developers to visually design and layout UI components. QT Designer provides code that can be embedded in Qt applications, making it easier to create visually beautiful and interactive apps.
 - b. Key characteristics include:
 - Interface Design Using Drag-and-Drop
 - Integration of Visual Component Layout with Qt Framework
 - Rapid UI Prototyping

Some of the other libraries

Package	Version	Latest version
EasyProcess	1.1	1.1
MainApp	0.1	0.1
MouseInfo	0.1.3	0.1.3
Pillow	10.1.0	10.1.0
PyAutoGUI	0.9.54	0.9.54
PyGetWindow	0.0.9	0.0.9
PyMsgBox	1.0.9	1.0.9
PyMySQL	1.1.0	1.1.0
PyQt5	5.15.10	5.15.10
PyQt5-Qt5	5.15.2	▲ 5.15.11
PyQt5-sip	12.13.0	12.13.0
PyQt5-stubs	5.15.6.0	5.15.6.0
PyRect	0.2.0	0.2.0
PyScreeze	0.1.29	▲ 0.1.30
altgraph	0.17.4	0.17.4
comtypes	1.2.0	1.2.0
coverage	7.3.2	7.3.2
entrypoint2	1.1	1.1
mss	9.0.1	9.0.1
mysql	0.0.3	0.0.3
mysql-connector-python	8.1.0	▲ 8.2.0
mysqlclient	2.2.0	2.2.0

UI Implementation screenshots



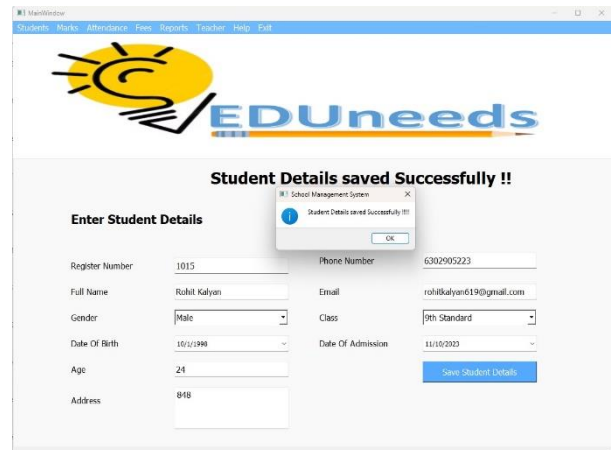
Administrator Login

Enter the User Name and Password

User Name:

Password:

FIGURE 1 ADMINISTRATOR LOGIN WITH USERNAME



Student Details saved Successfully !!

Enter Student Details

Register Number: Phone Number:

Full Name: Email:

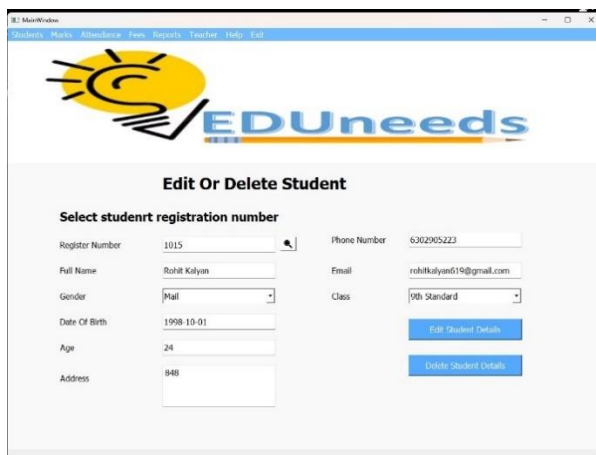
Gender: Class:

Date Of Birth: Date Of Admission:

Age:

Address:

FIGURE 2 REGISTER THE NEW STUDENT



Edit Or Delete Student

Select student registration number

Register Number: Phone Number:

Full Name: Email:

Gender: Class:

Date Of Birth:

Age:

Address:

Figure 3 Edit or Delete Student details.



Add Mark Details

Register Number:

Exam Name:

Date of Exam:

English:

Maths:

Science:

Social:

Edit / Delete Mark

Select Registration Number

Register Number:

Exam Name:

Date of Exam:

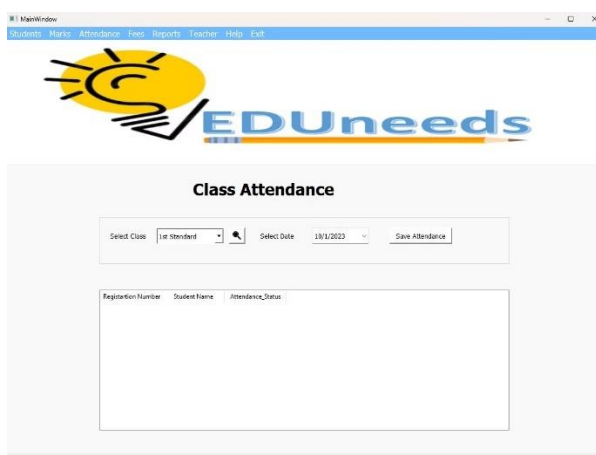
English:

Maths:

Science:

Social:

Figure 4 Add or edit or delete marks of each student

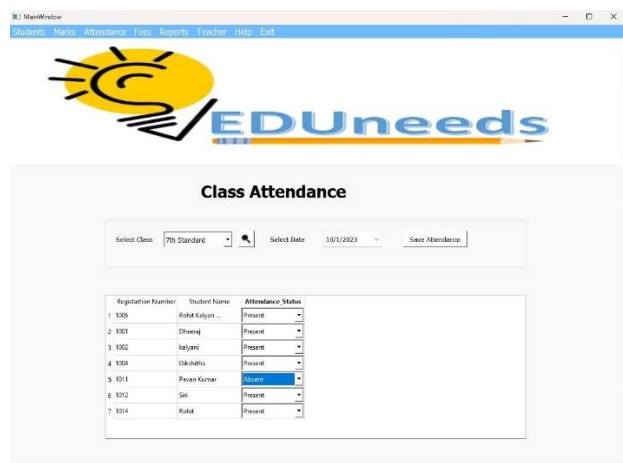


Class Attendance

Select Class: Select Date:

Registration Number	Student Name	Attendance Status

FIGURE 5 ADD ATTENDANCE FOR EACH STUDENT



Class Attendance

Select Class: Select Date:

Registration Number	Student Name	Attendance Status
1 1008	Rohit Kalyan	Present
2 1009	Omraj	Present
3 1002	Kalyani	Present
4 1004	Dikshita	Present
5 1011	Pravin Kumar	Absent
6 1012	Sim	Present
7 1014	Rishi	Present

FIGURE 6 SELECT DATE AND ADD ATTENDANCE

Edit / Delete Attendance

Select Reg Number And Date

Register Number: 1006

Attendance Date: 11/10/2023

Student Name: Rohit Kalyan Gandham

Attendance Status: Present

[Edit Attendance](#) [Delete Attendance](#)

FIGURE 7 EDIT OR DELETE ATTENDANCE

Fees Payment Details

Enter Fess Details

Register Number: 1015

Student Name: Rohit Kalyan

Class: 9th Standard

Total Amount Need to Pay: 9000

[Save Fees Details](#) [Delete Fees Details](#)

FIGURE 8 FEE PAYMENT DETAILS PAGE

Fees Payment Details

Enter Fess Details

Register Number: 1015

Student Name: Rohit Kalyan

Class: 9th Standard

Total Amount Need to Pay: 8500

[Save Fees Details](#) [Delete Fees Details](#)

FIGURE 9 ADD FEE DETAILS AND SAVE

Fees Payment Details

Enter Fess Details

Register Number: 1015

Student Name: Rohit Kalyan

Class: 9th Standard

Total Amount Need to Pay: 9000

[Save Fees Details](#) [Delete Fees Details](#)

1st Term is born paid Successfully and Balance amount Need to pay is 8500

FIGURE 10 SUCCESSFULLY PAID FEE

Administrator Reports

Student Reports

Total Students in School: 13

Select Class: 1st Standard

Teacher Reports

Total Teacher: 13

Attendance Reports

Select Date: 10/1/2023

Total School Attendance: 13

FIGURE 11 ADMINISTRATOR REPORTS PAGE

Add Teacher Details

Enter Teacher Details

Instructor Name: Rohit

Instructor ID: 2392

Instructor Subject: Maths

[Save Details](#) [Delete Details](#)

Teacher Details Added Successfully !!!

FIGURE 12ADD NEW INSTRUCTOR

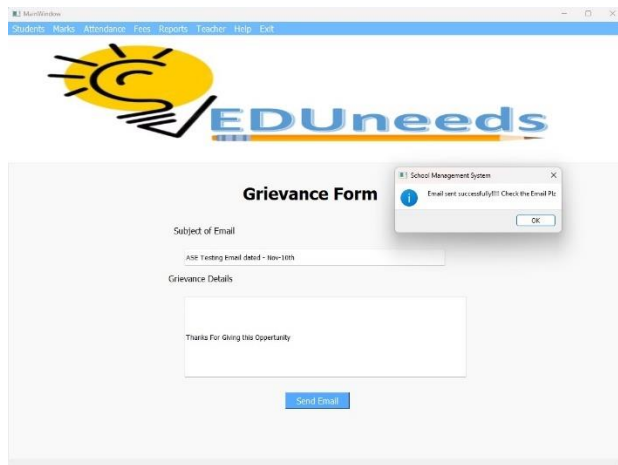


FIGURE 13 GRIEVANCE FORM ABOUT APPLICATION

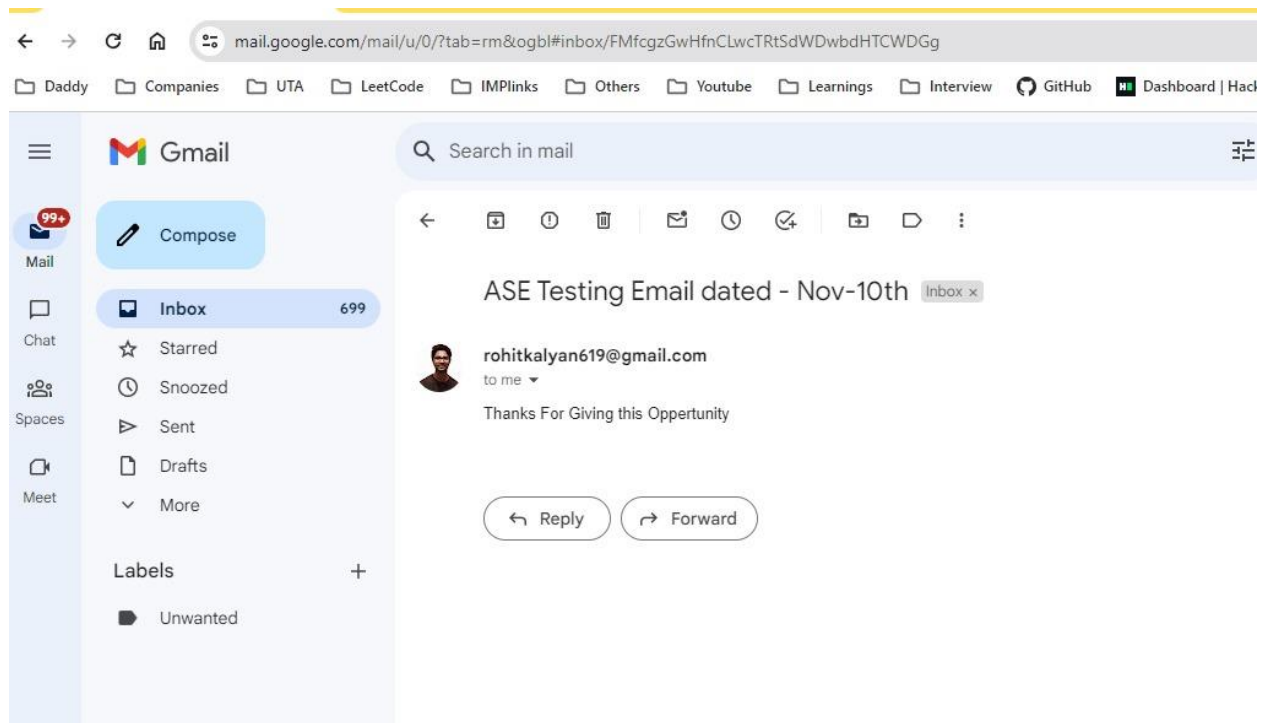


FIGURE 14 EMAIL AFTER GRIEVANCE SUCCESSFULLY RAISED

8. Testing

Functional Testing

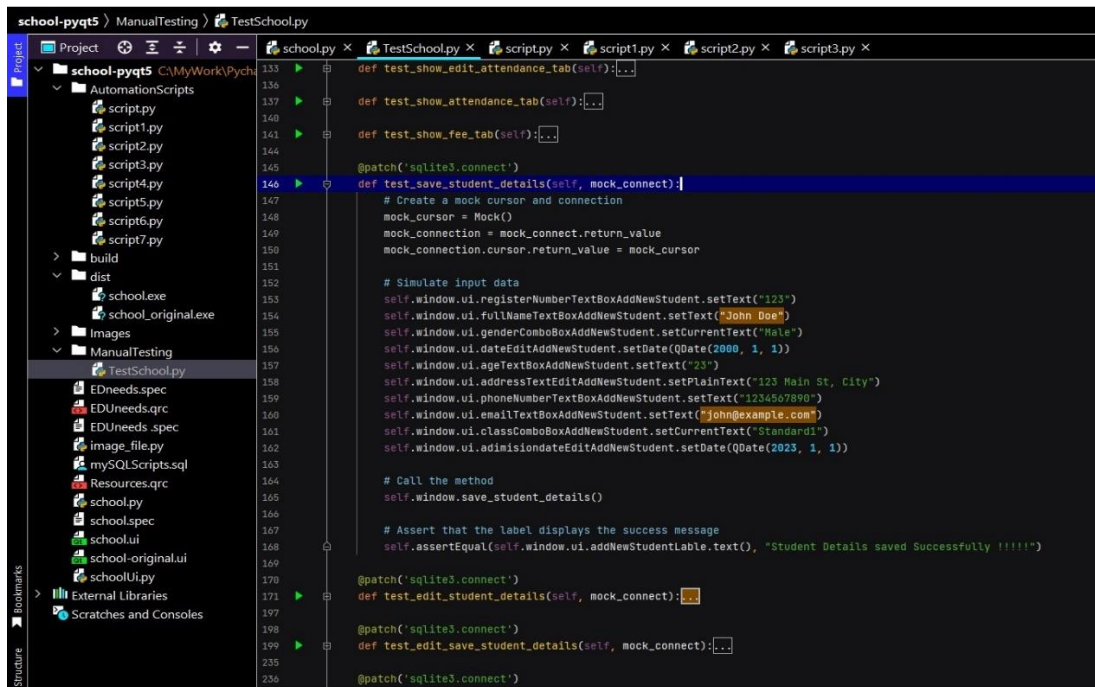
- Unit Test -- unittest, pytest, QTest
- White-Box-Testing -- SonarQube light
- Automation Testing -- pyautogui

Non functional Testing for Performance:

- We performed Reliability Testing The primary purpose is to discover potential failures and weaknesses in the code in order to ensure that the software runs consistently without interruptions or unexpected behavior.
- We performed Compatibility Testing the major goal is to offer uniform user experience and seamless functionality across all platforms.

Unit Testing

This is the sample image for unit testing



The screenshot displays the PyCharm IDE interface. On the left, the 'Project' view shows a directory structure for 'school-pyqt5'. The 'ManualTesting' folder is expanded, revealing 'TestSchool.py' and several test specification files. The main editor window shows the code for 'TestSchool.py'. The code includes several test functions: 'test_show_edit_attendance_tab', 'test_show_attendance_tab', 'test_show_fee_tab', and 'test_save_student_details'. The 'test_save_student_details' function is currently selected and highlighted. It uses the 'unittest.mock' module to patch 'sqlite3.connect' and create a mock cursor and connection. The function simulates input data for a new student, calls the 'save_student_details' method, and asserts that the success message is displayed. Other test functions like 'test_edit_student_details' and 'test_edit_save_student_details' are also visible, each with a corresponding '@patch' decorator.

```
133 def test_show_edit_attendance_tab(self):...
```

```
136
```

```
137 def test_show_attendance_tab(self):...
```

```
140
```

```
141 def test_show_fee_tab(self):...
```

```
144
```

```
145 @patch('sqlite3.connect')
```

```
146 def test_save_student_details(self, mock_connect):
```

```
147     # Create a mock cursor and connection
```

```
148     mock_cursor = Mock()
```

```
149     mock_connection = mock_connect.return_value
```

```
150     mock_connection.cursor.return_value = mock_cursor
```

```
151
```

```
152     # Simulate input data
```

```
153     self.window.ui.registerNumberTextBoxAddNewStudent.setText("123")
```

```
154     self.window.ui.fullNameTextBoxAddNewStudent.setText("John Doe")
```

```
155     self.window.ui.genderComboBoxAddNewStudent.setCurrentText("Male")
```

```
156     self.window.ui.dateEditAddNewStudent.setDate(QDate(2000, 1, 1))
```

```
157     self.window.ui.ageTextBoxAddNewStudent.setText("23")
```

```
158     self.window.ui.addressTextEditAddNewStudent.setPlainText("123 Main St, City")
```

```
159     self.window.ui.phoneNumberTextBoxAddNewStudent.setText("1234567890")
```

```
160     self.window.ui.emailTextBoxAddNewStudent.setText("john@example.com")
```

```
161     self.window.ui.classComboBoxAddNewStudent.setCurrentText("Standard1")
```

```
162     self.window.ui.admissiondateEditAddNewStudent.setDate(QDate(2023, 1, 1))
```

```
163
```

```
164     # Call the method
```

```
165     self.window.save_student_details()
```

```
166
```

```
167     # Assert that the label displays the success message
```

```
168     self.assertEqual(self.window.ui.addNewStudentLabel.text(), "Student Details saved Successfully !!!!!!")
```

```
169
```

```
170 @patch('sqlite3.connect')
```

```
171 def test_edit_student_details(self, mock_connect):...
```

```
172
```

```
173 @patch('sqlite3.connect')
```

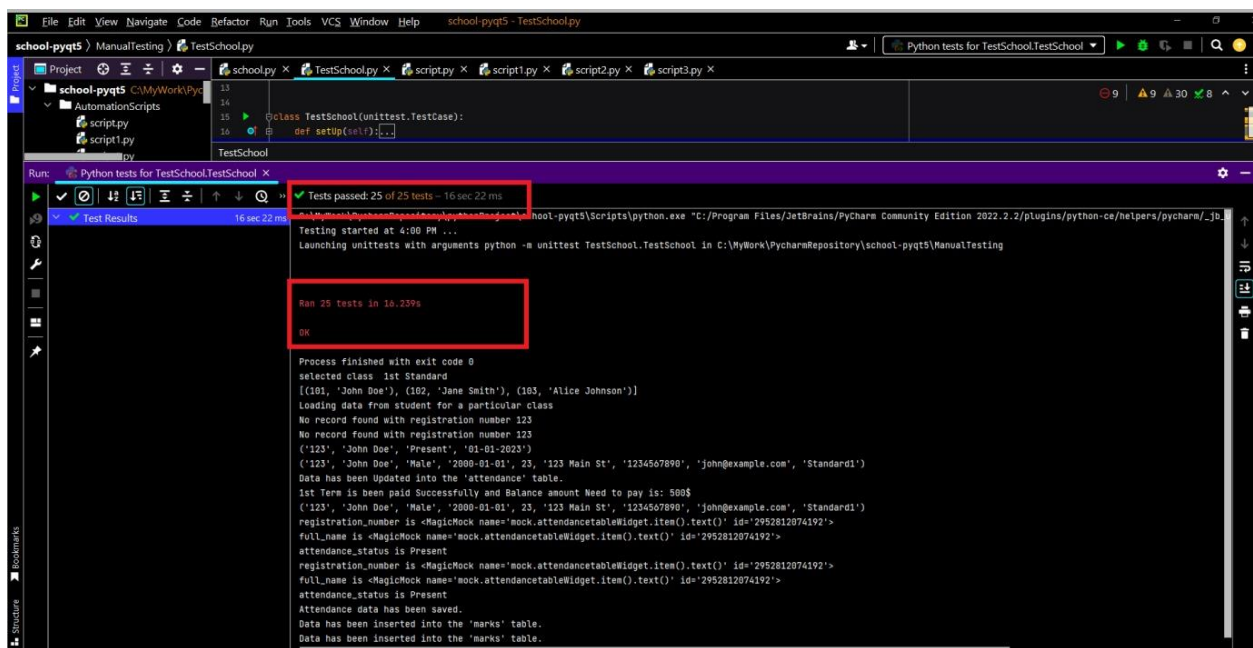
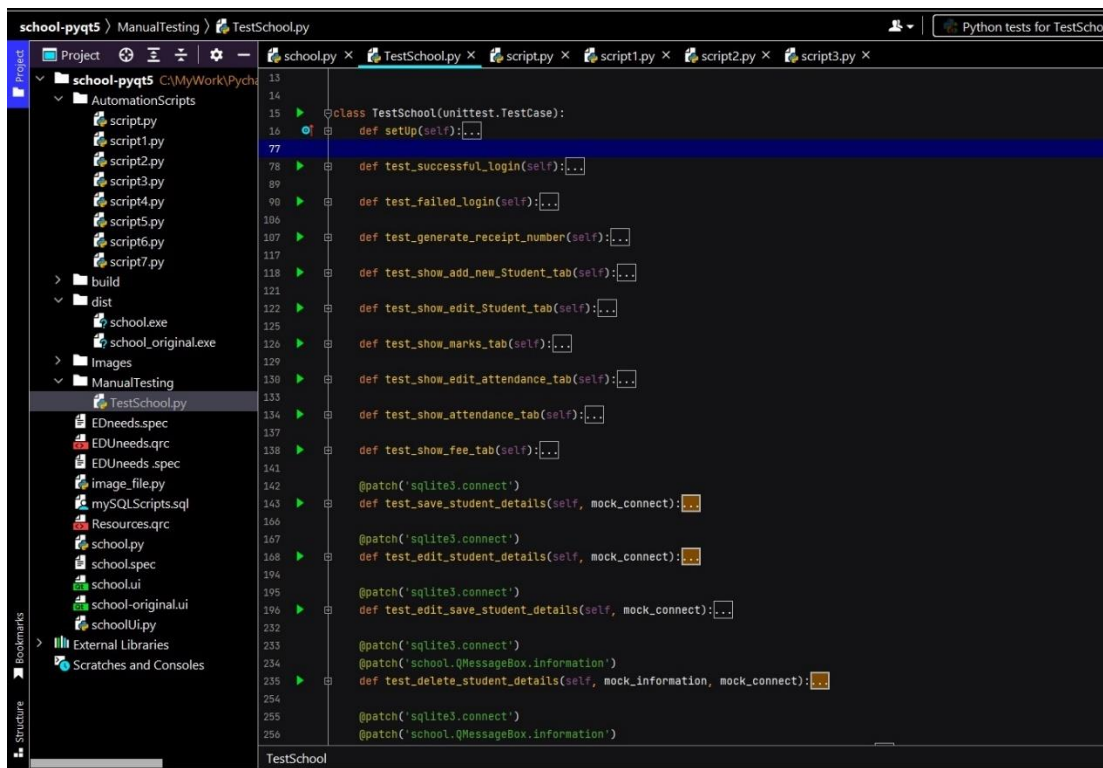
```
174 def test_edit_save_student_details(self, mock_connect):...
```

```
175
```

```
235
```

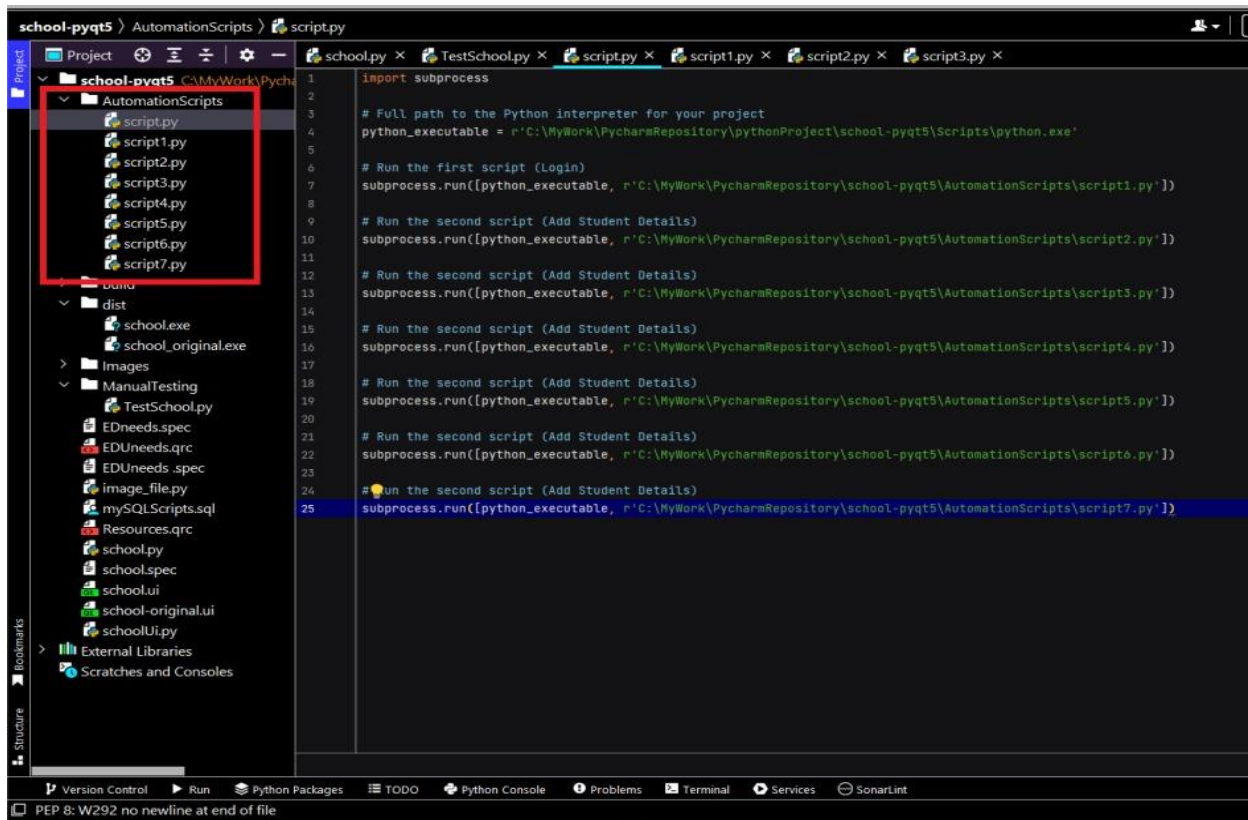
```
236 @patch('sqlite3.connect')
```

Image for unit testing of all methods show in the below image

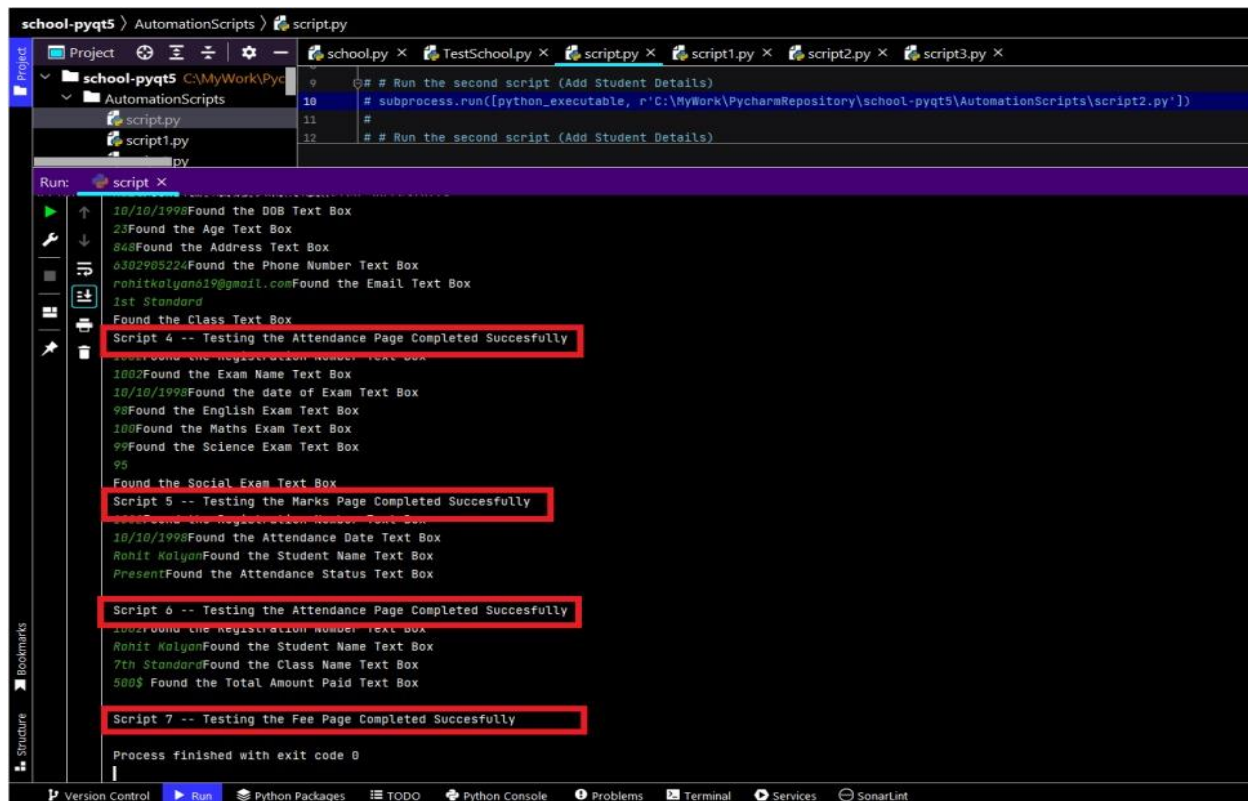


Unit testing is executed and you can find the result in the above image the total test case passed is 25

Automation Testing



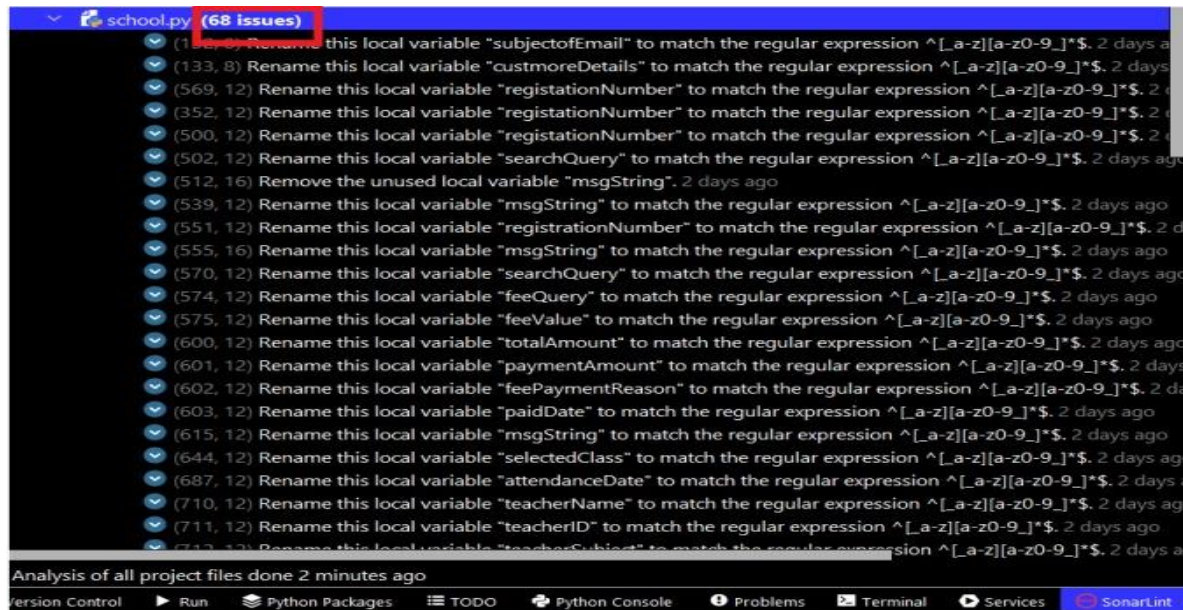
These the 7-test scripts we used in the automation testing



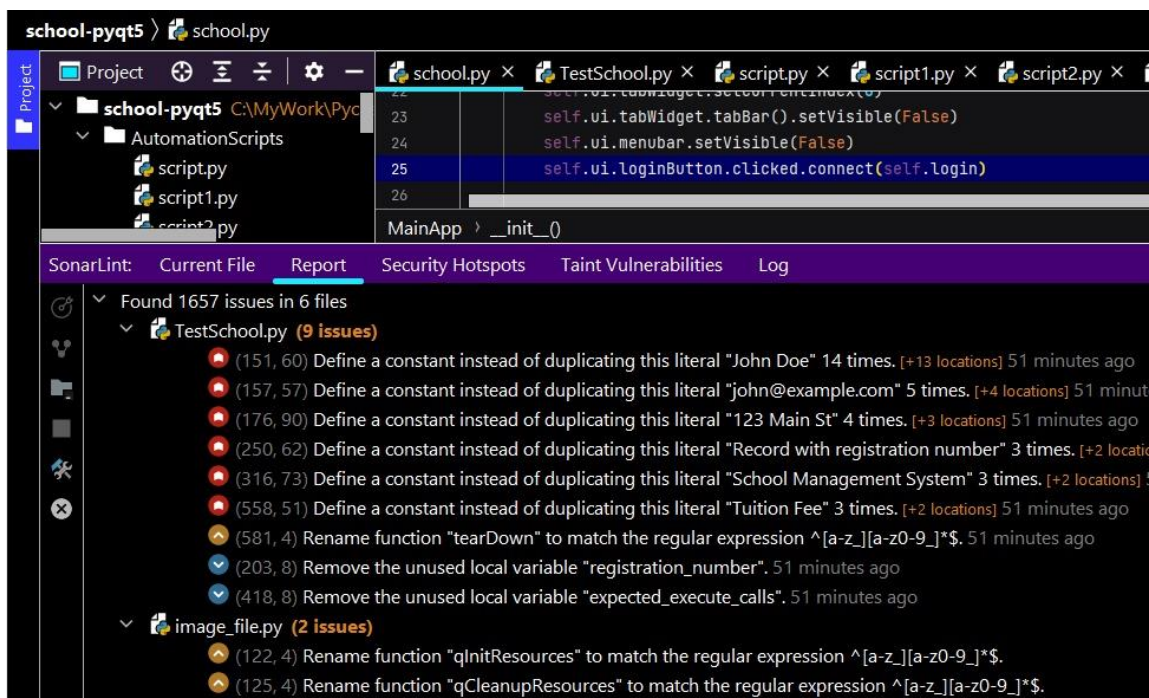
The results of the Automation test script results are shown in the above image

Issues and Bugs

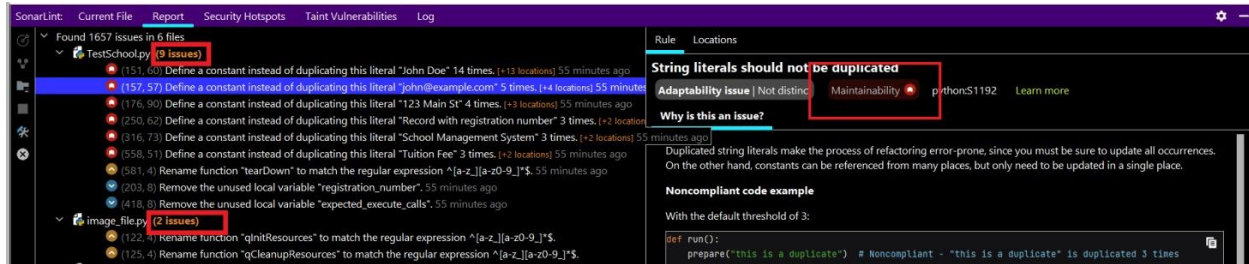
Total 68 bugs are found – resolved bugs 57, Managed -- 11



These are 11 issue found which are maintainable



These 11 issues below are Maintainable below is the screenshot



Code Coverage and Quality:

Command to know the Code Coverage is:

cmd: coverage run -m ManualTesting.TestSchool

Code Coverage Report Command is:

cmd: coverage report -m

- 1) Code Coverage:
 - Main Application: 99%
 - User Interface: 70%
- 2) Code Quality:
 - Main Application 99%

1.Code coverage:

- a. Main Application (99%): A high code coverage percentage shows that a large chunk of the codebase for the main application is performed during testing. This implies extensive testing and a lower chance of undiscovered faults in important functionalities.
- b. While 70% code coverage for the UI is good, it may merit additional attention. Improving UI code coverage can result in more thorough testing of the user interface, lowering the risk of UI-related errors.

2.Code Quality

- a. Main Application (99%): A code quality score of 99% indicates that the code is of high quality in terms of maintainability, readability, and adherence to best practices. This signifies that the core application codebase is well-structured and robust.

Below is the screenshot for code coverage and quality

The screenshot shows an IDE with a Python project named 'school-pyqt5'. The code editor displays a class with methods for saving, editing, and deleting attendance details. The terminal window shows the execution of 25 tests, which passed successfully. Below the test results, a coverage report is displayed, showing the status of various files in the project.

Name	Stats	Miss	Cover	Missing
ManualTesting\TestSchool.py	357	2	99%	583-584
school.py	513	166	68%	67-68, 92, 128-156, 177, 180, 189, 221-222, 247, 249-250, 271, 297, 299-300, 316-317, 342-343, 348-371, 374-408, 416-417, 451-452, 469-470, 492-493, 515, 517-518, 542-543, 558, 5
01-562, 585, 588-589, 618-619, 622-618, 641	661	66	90%	489, 483-703, 706-728, 732-735, 740
schoolUI.py	1419	7	99%	1120-1126
TOTAL	2205	175	92%	

9. Maintenance

1. **Perfective maintenance**: This project presents an exciting opportunity to incorporate a chat functionality and elevate the management of notifications for an enhanced user experience.
2. **Corrective maintenance**: When the system crashes, the unsaved work should be automatically saved, allowing users to resume their work from the last saved point upon restarting.
3. **Preventive maintenance**: Incorporating cutting-edge third-party graphical libraries to enhance report generation and create visually stunning animated Graphical User Interfaces (GUI) for a truly engaging user experience.

Perfective Maintenance (Notification Management and Chat Functionality):

Identification: we Solicit feedback from users (other class members) on a regular basis to discover areas for development, such as the addition of chat capabilities and improved notification handling.

Development: we are Incorporating these improvements into planned updates, making sure they are consistent with the overall project goals.

Testing: Thoroughly test new features to ensure that they integrate seamlessly with current functionalities and to detect and correct any potential faults.

Release: In order to provide a better user experience, we will incorporate perfective modifications into scheduled releases.

Corrective Maintenance (Automatic Saving of Unsaved Work):

Monitoring and detection: we will Use monitoring tools to detect system crashes and identify instances where unsaved work could be jeopardized.

Development: we will Create a remedial solution that preserves unsaved work when the system crashes.

Testing: Thoroughly test the automated saving feature to ensure that it works well and does not bring new problems.

Release: when release we Incorporate the remedial remedy into patches and updates, addressing system stability concerns as soon as possible.

Preventive Maintenance (Third-Party Graphical Libraries for Improved GUI and Reporting):

Evaluation: We will assess the performance and capabilities of third-party graphical libraries on a regular basis to determine their suitability for improving the system's GUI and reporting capabilities.

Integration: During planned updates, we will integrate selected libraries into the system architecture to avoid potential issues and assure compatibility.

Extensive testing should be performed to validate the integration of third-party libraries, ensuring that they improve performance without jeopardizing system stability.

During Preventive maintenance updates will be included in planned releases to improve the system's visual appeal and reporting capabilities.

Management of Bug Fixes:

Reporting: we Created a method for users to report defects, including a dedicated bug tracking system.

Prioritization: we will Sort reported problems according to how they affect system functionality and user experience.

Resolution: we will immediate Assign resources to address and resolve detected defects as soon as possible, ensuring comprehensive testing of bug remedies prior to deployment.

Communication: we always Keep stakeholders up to date on problem fixes via release notes, stressing system stability and reliability improvements.

Scalability and future proofing

1) Cloud Computing Infrastructure:

Scalability: we are planning to use cloud infrastructure to handle variable workloads and user counts. Cloud services may dynamically scale resources, ensuring that system performance stays optimal during peak times.

Future-Proofing: Cloud-based solutions offer flexibility and are frequently updated with new technology, which contributes to the system's future-proofing.

2) Scalability of the database:

Scalability: we need select a database system that allows horizontal scaling, which enables the system to accommodate growing data volume without sacrificing performance.

Future-Proofing: we will select databases that are flexible to new technologies and can interact effortlessly with future data management systems.

3)Security Procedures:

Scalability: Scalable security measures are planned to be implemented to manage rising user numbers and data volumes, protecting the system from potential threats.

Future-Proofing: Updating security methods on a regular basis to handle emerging cybersecurity challenges and to adapt to developing industry standards.

4)Design of the User Interface:

Scalability: Creating a responsive and scalable user interface that can adapt to various devices and screen sizes, delivering a consistent user experience as the user base expands.

Future-Proofing: current focus on user interface design trends and update the interface on a regular basis to suit changing user expectations and usability requirements.

10. Conclusion

The School Management System project transformed school administration by streamlining processes and improving user experiences. A simpler interface, improved notification management, and enhanced security measures are among the accomplishments. Lessons learnt highlight the significance of user participation, Agile techniques, thorough testing, and open communication. Enhancing integration capabilities, offering continuous training, addressing accessibility concerns, and evaluating system performance are all areas for improvement. In the evolving landscape of educational technology, the project displays a commitment to innovation, efficiency, and a user-centric approach. Among the notable accomplishments are:

1. Administration has been streamlined as a result of the development of a strong database structure and user-friendly interface, which has streamlined administrative chores ranging from student enrollment to fee administration.
2. Enhanced User Experience: The addition of chat capability, better notification management, and a visually attractive GUI has enhanced the entire user experience for administrators, teachers, students, and parents.
3. Security Enhancement: To preserve sensitive information and ensure the integrity and confidentiality of educational data, advanced security methods such as data encryption and frequent security updates have been adopted.
4. Scalability and adaptability to changing educational needs are ensured by the system's modular architecture, cloud infrastructure, and scalable database solutions. Continuous upgrades and API design help to secure the system's future.

Lessons Discovered:

1. Continuous user involvement and input are crucial for finding areas for improvement and ensuring that the system closely fits with user expectations.
2. Adopting an Agile development technique has been shown to be effective, providing for flexibility in responding to changing requirements and incorporating iterative improvements.

3. Thorough Testing is Required: Thorough testing is essential for discovering possible issues early in the development process and guaranteeing a stable and reliable system, especially for new features and bug fixes.
4. Clear communication is essential: Maintaining open communication with stakeholders, especially when it comes to updates, patches, and issue fixes, builds confidence and keeps all parties informed.

Areas for Improvement:

1. Enhance the system's integration capabilities to connect with a greater range of educational resources and platforms, encouraging a more interconnected educational ecosystem.
2. Continuous Training and Support: Implement a continuous training and support program for users to ensure they are fully conversant with the system's functionality.
3. Considerations for Accessibility: Evaluate and improve the system's accessibility features to ensure that it accommodates to users with varying demands while complying to accessibility standards.
4. Performance monitoring tools should be used to proactively identify and address any performance bottlenecks, ensuring optimal system performance during peak times.

11. Recommendations

Implement data analytics for informed decision-making to improve student management. Analyze historical student data using descriptive analytics to detect patterns and trends in attendance, academic performance, and behavior. Predictive analytics can foresee future trends, allowing for preventive interventions for children at risk. Use prescriptive analytics to offer individualized academic improvement techniques based on individual student profiles. Implement dashboard visualizations to provide administrators with a comprehensive view of student stats. Continuously increase the accuracy of analytics models using machine learning techniques. This data-driven approach enables educators to make more informed decisions, promoting student success and improving the overall efficiency of the School Management System.

12. References

we have used these external libraries in our project

Package	Version	Latest version
EasyProcess	1.1	1.1
MainApp	0.1	0.1
MouseInfo	0.1.3	0.1.3
Pillow	10.1.0	10.1.0
PyAutoGUI	0.9.54	0.9.54
PyGetWindow	0.0.9	0.0.9
PyMsgBox	1.0.9	1.0.9
PyMySQL	1.1.0	1.1.0
PyQt5	5.15.10	5.15.10
PyQt5-Qt5	5.15.2	▲ 5.15.11
PyQt5-sip	12.13.0	12.13.0
PyQt5-stubs	5.15.6.0	5.15.6.0
PyRect	0.2.0	0.2.0
PyScreze	0.1.29	▲ 0.1.30
altgraph	0.17.4	0.17.4
comtypes	1.2.0	1.2.0
coverage	7.3.2	7.3.2
entrypoint2	1.1	1.1
mss	9.0.1	9.0.1
mysql	0.0.3	0.0.3
mysql-connector-python	8.1.0	▲ 8.2.0
mysqlclient	2.2.0	2.2.0