CSE-299, Section – 01
1) Md. Tushar Khan, ID: 2232344642 Group - 6
2) Md. Minhaz Ahamed Rohan, ID: 2232065642
3) Md. Ashike Rabby, ID: 2021900642

Final Report

Student Portal Management System

1. Introduction

In recent years, educational institutions have increasingly turned to digital solutions to streamline operations and improve student experiences. One critical area that demands efficient management is student information. Traditional manual systems are time-consuming, error-prone, and lack proper data security. To address these challenges, this project introduces a **Student Portal Management System (SPMS)** developed using **C++ and the Qt framework**.

This desktop application is designed to allow students to create accounts, securely log in, and manage their academic and personal data. It features a responsive graphical user interface (GUI), robust backend logic, and persistent storage using CSV files. The platform not only enhances student data management but also serves as a practical implementation of modern software engineering principles.

The development of this system demonstrates proficiency in object-oriented programming, GUI development, file handling, and user authentication, making it a valuable academic and practical contribution.

2. Project Objectives

The core goal of this project is to create a simple, secure, and extensible system for students to manage their information. Specific objectives include:

- **User Authentication**: To design a secure signup and login mechanism using password hashing and masking.

- **Information Submission**: To allow students to enter and update their data including personal details, documents, and photos.
- **Persistent Storage**: To ensure all information is saved in CSV files and remains accessible across sessions.
- **User Interface**: To develop an intuitive and visually appealing GUI using Qt Designer.
- **Security and Privacy**: To protect sensitive user data from unauthorized access.
- **Future-Proof Design**: To enable scalability and future integration with modules like course management or payments.

3. System Features

The Student Portal Management System offers several key features aimed at enhancing usability, performance, and security.

3.1 Signup & Login System

- **Password Hashing**: SHA256 (or similar) hashing is implemented so that user passwords are never stored in plain text.
- **Validation & Masking**: Passwords are hidden during entry. Input fields are validated to ensure no fields are left empty or contain invalid data.
- **CSV-based Credentials**: User credentials are stored securely in a separate CSV file, isolating authentication from other data.

3.2 Student Dashboard

Upon successful login, users are directed to a personalized dashboard. Features include:

- Display of submitted information.
- Option to upload or change profile photo.
- Button to edit details with validation prompts.

3.3 Student Information Management

Students can submit the following:

- Full Name
- Parent's Name
- Date of Birth (with calendar input)
- Nationality and Blood Group (dropdown menus)
- Current CGPA
- ID Number (auto-validated)
- Upload of documents (via file picker)
- Upload of profile photo

3.4 File Handling

All information is saved in structured CSV files:

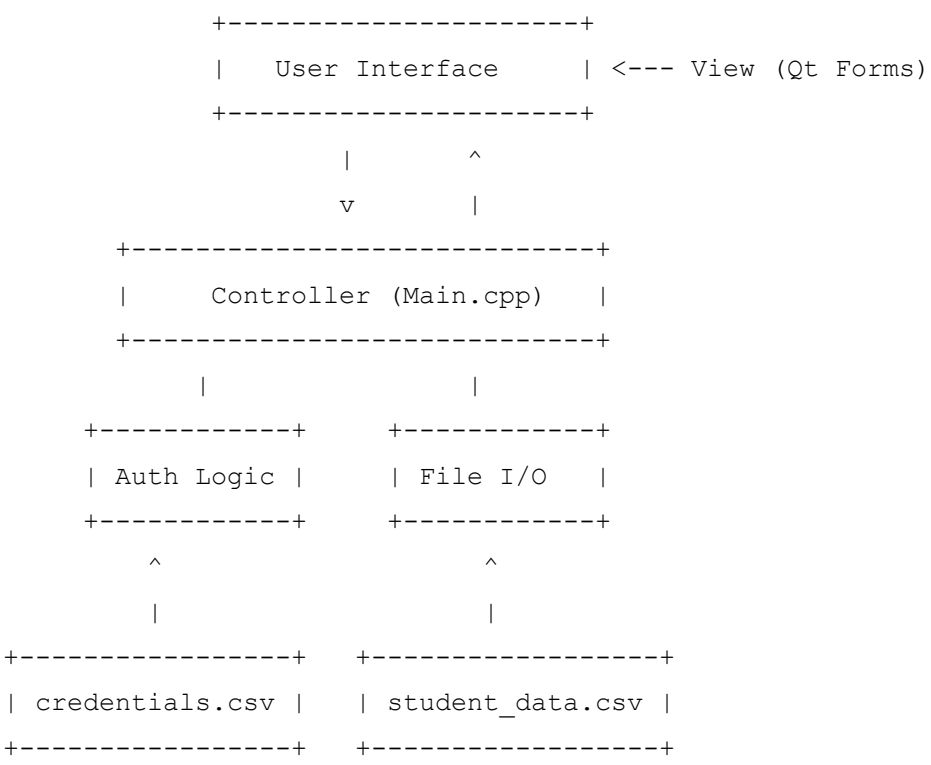
- `credentials.csv` – stores usernames and hashed passwords
- `student_data.csv` – stores detailed student profiles
- `images/` – stores uploaded profile pictures

4. System Architecture

The application follows a **Model-View-Controller (MVC)** architecture:

- **Model**: Handles data operations, reading from and writing to CSV files.
- **View**: Constructed with Qt Designer, it defines how each page (login, signup, dashboard) looks and interacts.
- **Controller**: Implements logic for validation, navigation, and action handling.

Diagram:



5. Implementation Details

5.1 Technology Stack

- **Language**: C++
- **Framework**: Qt 5/6 (cross-platform GUI development)
- **Data Format**: CSV
- **IDE**: Qt Creator

5.2 Code Organization

Files are organized as:

- `main.cpp` – entry point, manages navigation
- `login.cpp` / `signup.cpp` – logic for login/signup
- `student.cpp` – logic for profile display and update
- `filemanager.cpp` – abstraction for CSV operations

5.3 Password Hashing Implementation

```
cpp
CopyEdit
QString hashPassword(QString password) {
    QByteArray byteArray = QCryptographicHash::hash(password.toUtf8(),
QCryptographicHash::Sha256);
    return QString(byteArray.toHex());
}
```

5.4 CSV Parsing Logic

```
cpp
CopyEdit
QFile file("student_data.csv");
if (file.open(QIODevice::ReadOnly | QIODevice::Text)) {
    QTextStream in(&file);
    while (!in.atEnd()) {
        QString line = in.readLine();
        QStringList fields = line.split(",");
        // Process each field
    }
}
```

6. Testing and Evaluation

Testing involved the following phases:

Unit Testing

Each function was tested in isolation:

- `validateInput()`
- `hashPassword()`
- `saveToCSV()`
- `uploadPhoto()`

Integration Testing

- Login to dashboard transition
- Save/edit info reflected correctly in CSV
- Image upload shown in dashboard preview

User Feedback

Testers were asked to use the system and report on usability. Feedback indicated:

- Intuitive design
- Smooth navigation
- Suggestions for better error messages (implemented)

7. Challenges and Limitations

7.1 Technical Challenges

- **Password security**: Hashing was essential but required external library considerations for stronger encryption.
- **CSV race conditions**: Writing to files simultaneously led to data overwrites; mitigated with file locks.
- **Photo scaling**: Ensuring images displayed properly required manual resizing via Qt's `QPixmap`.

7.2 Limitations

- **CSV storage**: Not suitable for large datasets or concurrent access.
- **No database**: Lacks relational data features; not ideal for integration with institutional systems.
- **No admin role**: Current version is student-facing only.

8. Conclusion and Future Work

The Student Portal Management System successfully achieves its aim of providing a secure, efficient, and user-friendly platform for managing student data. It showcases the use of modern C++ principles, GUI design with Qt, and file handling techniques. It is ideal for small institutions or academic demos.

Future Work:

1. **Database Integration**: Use of SQLite or MySQL for scalability.
2. **Role Management**: Add admin/teacher dashboards.
3. **Course & Grade Modules**: Let students enroll in and track courses.
4. **Payment Module**: Integrate fee tracking and payment records.
5. **Web & Mobile Version**: Deploy as a web app using Qt for WebAssembly or convert to React/Flutter frontend.