Islamic University of Technology

Board Bazar, Gazipur, Dhaka

SWE 4304 – Software Project Lab-I

Project Proposal

Hall Verse

Hall Management System

December 14, 2025

SPL Team - 14

Supervisor:

Md. Bakhtiar Hasan
Assistant Professor
Islamic University of Technology

Team Members:

Muntajim Rahman Saimon
ID: 230042103
Bsc. in Software Engineering
Department of Computer Science and Engineering

Ifham Bin Hossain
ID: 230042112
Bsc. in Software Engineering
Department of Computer Science and Engineering

Didarul Shahriar
ID: 230042137
Bsc. in Software Engineering
Department of Computer Science and Engineering

Chapter 1

Project Overview & Motivation

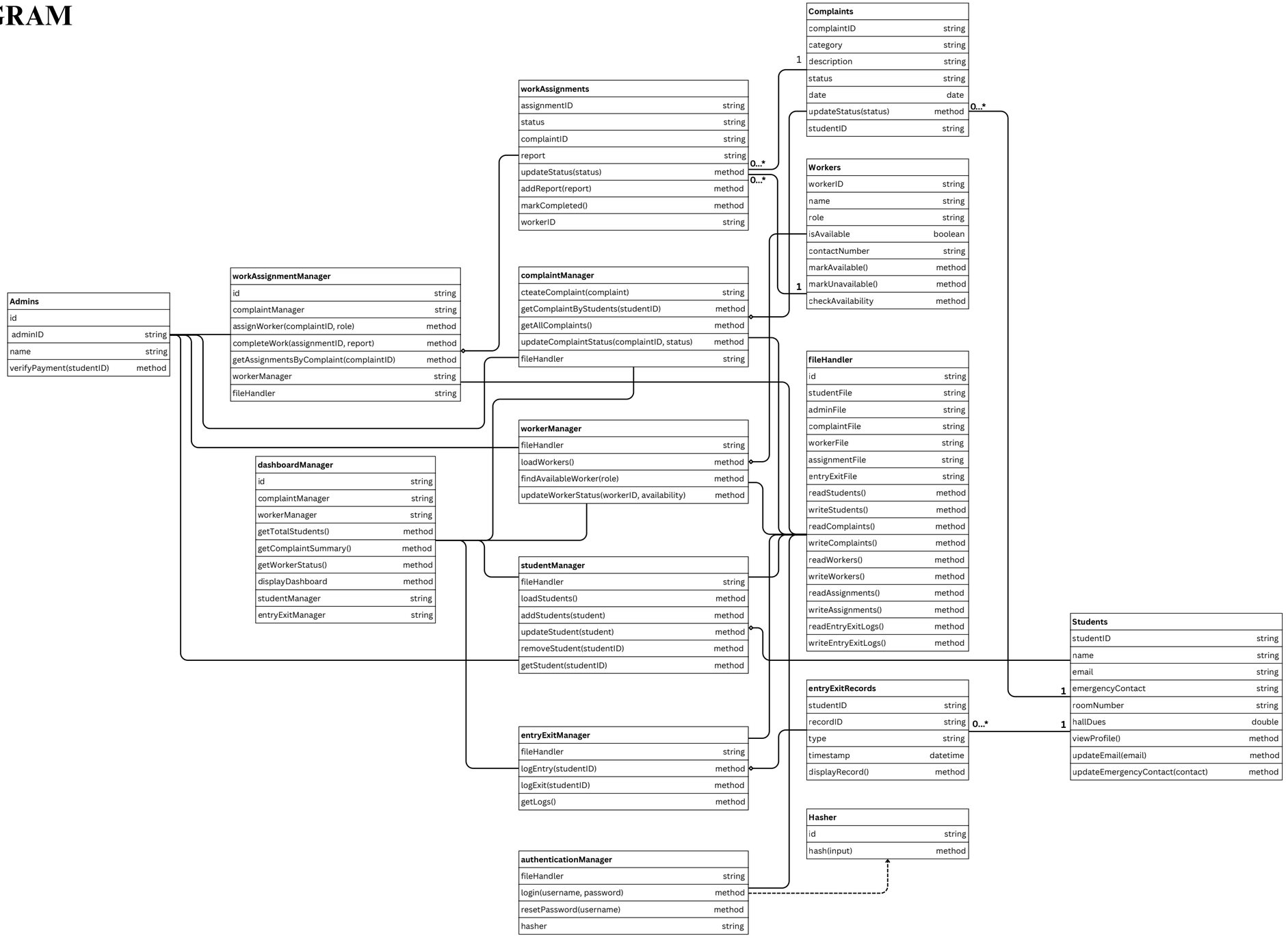
1.1 Overview

Project Name: HallVerse
HallVerse – A hall management system is basically a console-based application designed to digitize and simplify daily hall operations such as student information management, room allocation, payment information, complaints, and other administrative tasks. This project focuses on secure data handling, organized record-keeping, and smooth user interaction through a simple, menu-driven interface. In this project, manual processes are replaced with an organized digital system, and that's how the application improves accuracy, reduces workload, and provides a faster and more efficient experience.

1.2 Motivation

Hall activities in many universities are still managed manually, which often leads to unnecessary delays and confusion. Students are frequently asked for proof upon entry because guards have no quick way to verify hall residents. Complaint processes are also paper-based, making it difficult for students to know whether their issues—such as electrical problems or damaged equipment—are being addressed. These everyday challenges highlighted the need for a centralized digital system where hall information, complaints, room allocations, and payments can be managed more efficiently. Our motivation behind this project is to reduce manual workload, remove uncertainty, and create a smoother and more transparent communication flow between students, guards, and hall authorities.

UML DIAGRAM


Admins
id
adminID string
name string
verifyPayment(studentID) method
workAssignmentManager
id string
complaintManager string
assignWorker(complaintID, role) method
completeWork(assignmentID, report) method
getAssignmentsByComplaint(complaintID) method
workerManager string
fileHandler string
dashboardManager
id string
complaintManager string
workerManager string
getTotalStudents() method
getComplaintSummary() method
getWorkerStatus() method
displayDashboard method
studentManager string
entryExitManager string
workAssignments
assignmentID string
status string
complaintID string
report string
updateStatus(status) method
addReport(report) method
markCompleted() method
workerID string
complaintManager
cteateComplaint(complaint) string
getComplaintByStudents(studentID) method
getAllComplaints() method
updateComplaintStatus(complaintID, status) method
fileHandler string
workerManager
fileHandler string
loadWorkers() method
findAvailableWorker(role) method
updateWorkerStatus(workerID, availability) method
studentManager
fileHandler string
loadStudents() method
addStudents(student) method
updateStudent(student) method
removeStudent(studentID) method
getStudent(studentID) method
entryExitManager
fileHandler string
logEntry(studentID) method
logExit(studentID) method
getLogs() method
authenticationManager
fileHandler string
login(username, password) method
resetPassword(username) method
hasher string
Complaints
complaintID string
category string
description string
status string
date date
updateStatus(status) method
studentID string
Workers
workerID string
name string
role string
isAvailable boolean
contactNumber string
markAvailable() method
markUnavailable() method
checkAvailability method
fileHandler
id string
studentFile string
adminFile string
complaintFile string
workerFile string
assignmentFile string
entryExitFile string
readStudents() method
writeStudents() method
readComplaints() method
writeComplaints() method
readWorkers() method
writeWorkers() method
readAssignments() method
writeAssignments() method
readEntryExitLogs() method
writeEntryExitLogs() method
entryExitRecords
studentID string
recordID string
type string
timestamp datetime
displayRecord() method
Hasher
id string
hash(input) method
Students
studentID string
name string
email string
emergencyContact string
roomNumber string
hallDues double
viewProfile() method
updateEmail(email) method
updateEmergencyContact(contact) method
1
0...*
1
0...*
0...*
1
1
0...*
1

Chapter 2

Key Features

The Hall Management System integrates secure authentication, structured information handling, and administrative monitoring to support efficient hall operations. The features are designed to ensure data accuracy, role-based access, and smooth student–authority interaction. The following sections provide an overview of the major functionalities implemented within the system.

2.1 User Authentication System

The user authentication module serves as the primary security system of the application. It verifies user identity and restricts system access based on predefined roles such as Student or Admin. Authentication is performed through a username–password system, with passwords hashed before storage to ensure confidentiality and security. A recovery process is also implemented to assist users who forget their login credentials.

- **Username and Password Authentication**
  – Users log in using valid credentials (Student or Admin).
  – Passwords are securely hashed before being stored in the csv files.

- **Role-Based Access Control**
  – Students and admins are granted separate levels of access.
  – Students can only interact with student-specific features.
  – Administrators receive privileges to manage system-wide operations.

- **Account Recovery**
  – A password-reset mechanism is available for users.
  – Users must correctly answer their security questions to reset their password.

2.2 Student Profile Management

This module provides structured management of student information, enabling both students and administrators to handle profile-related data. Students can view and update selected personal details, while administrators see student records.

- **Student Features**
  – View personal profile information.
  – Update editable fields such as email or emergency contact.
  – Check hall dues.
  – View room number and associated details.

- **Admin Features**
  – Add new student records when required.
  – Remove student records when a student graduates or leaves the hall.
  – Verify student payments and update dues status.

2.3 Complaint & Issue Reporting System

This feature ensures the communication channel between students and hall authorities regarding issues and concerns. Students can submit complaints and track their complaint status, while admins are equipped with tools to process and manage these complaints efficiently.

- **Student Features**
  – File complaints related to electricity, water, room conditions, or other issues.
  – Track complaint status (Pending, In-Progress, Resolved).
- **Admin Features**
  – View all submitted complaints along with student details and room numbers.
  – Update complaint status as issues are addressed.
  – View student room information when required for context.

2.4 Admin Dashboard

The admin dashboard views all statistics and operational information in a single interface. This feature supports decision-making and enables effective supervision of hall activities.

- **Dashboard Capabilities**
  – Manage all users, including adding and removing accounts.
  – Display the total number of registered students.
  – Display total complaints.
  – Track complaints solved, pending, or in progress.
  – Show the number of vacant seats in the hall.

2.5 Additional Features

Additional operational features provide enhanced monitoring and activity tracking within the hall. The system maintains entry and exit logs to ensure accountability and improve security.

- **Entry/Exit Management**
  – Users must authenticate (username and password) to mark hall entry or exit.
  – Every entry and exit is recorded with timestamps.

- **Activity Records**
  – View daily entry counts.
  – View daily exit counts.
  – Maintain a searchable history of all entry/exit events.

Chapter 3

Tools & Technologies

In the development of our project, we have selected a set of tools and technologies that ensure smooth collaboration, efficient coding, and reliable version management.

3.1 Version Control

We are using **Git** for version control to track changes in our codebase and manage contributions from all team members. Additionally, **Github** serves as our remote repository which allows us to collaborate, review code, and maintain an organized development workflow.

3.2 Development Environment

For writing and managing our code, we are using **Visual Studio Code**. It provides a lightweight yet powerful environment with extensions that enhance productivity, code navigation, debugging, and C++ development support.

3.3 Programming Language

We will be using **C++** as our primary programming language in this project because **C++** offers strong performance and structured programming capabilities, making it suitable for building a menu-driven console-application like our Hall Management System.

Chapter 4

Proposed Project Timeline

Initial Planning and System Design (Week 6)

During this phase, all required classes are identified, some may be added later if necessary, class skeletons are created, and data that we will assign for various classes, such as students, admins, complaints, and entry/exit logs, are finalized.

Building Core Functionalities (Week 7 – Week 8)

These two weeks involve developing the essential features of the system. This includes user authentication, password hashing, and role-based access (Student/Admin). Basic student profile features, such as viewing and updating personal information, are implemented. The complaint reporting system has also been started, allowing students to submit complaints and store them in files.

Development for Mid-Presentation (Week 9-10)

By weeks 9 & 10, we will build to a point that includes login, password reset, viewing/updating student information, submitting complaints, and basic entry/exit logging. A clean menu interface and sample data are prepared so the progress can be demonstrated clearly.

Advanced Module Implementation (Week 10 – Week 12)

By weeks 10 and 11, we will focus on completing the remaining major features. This includes developing the admin dashboard (total students, complaints, vacant seats), improving the complaint management system, and completing the full entry/exit tracking module. All modules are connected, so the system works as a whole.

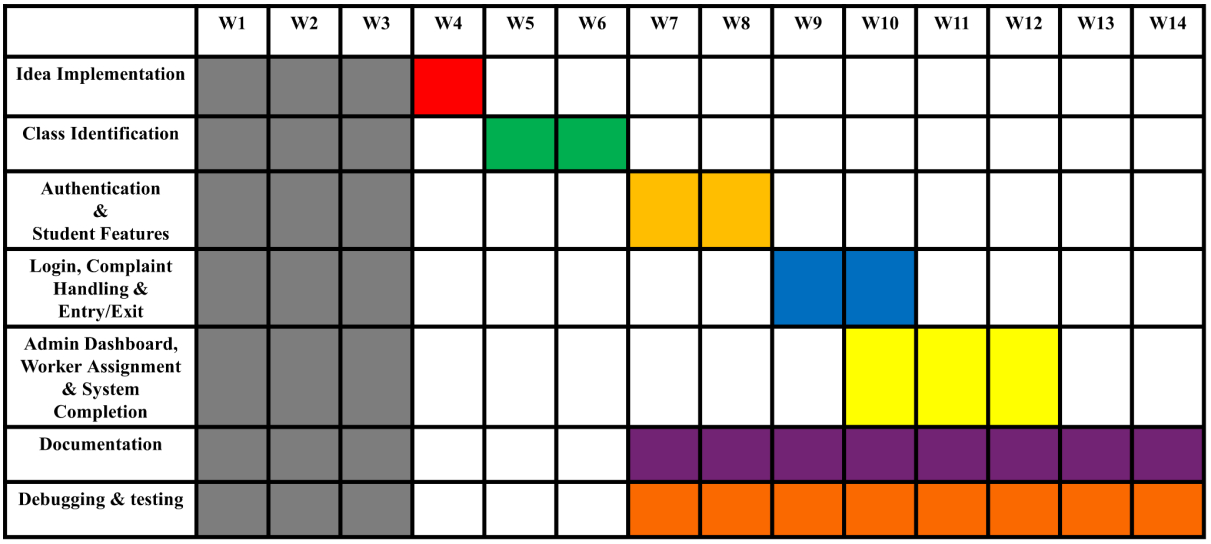
Testing and Refinement

Testing & refinement will be done throughout the whole time period of the project. After building each module & implementations, we will perform functional testing to verify correctness, identify issues, and ensure smooth integration with existing components. If necessary, modifications or improvements will be applied immediately to maintain the system, aligning with the project features.

Final Documentation and Submission Preparation

Documentation will also be maintained continuously throughout the entire development process. As each module is designed and implemented, we will record the implementation details, our debugging and testing outcomes. It will help us maintain a clear workflow for our project.

Timeline



Action taken based on the feedback

1) **Timeline Incorrect:**

- Initially, there was a wrong debugging, testing, and documentation phase. Even before the initiation of the implementation, we kept a timeline for debugging, testing, and documentation.

Now, it has been fixed with proper week distribution with the implementation of features.

2) **Features should be planned more granularly:**

- Initially, there were just two implementations, One is implementation of the students feature, & another was the implementation of the authorities feature. Because of this, there was a hindrance to getting a clear idea about the implementation

Now, it has been fixed with a more granular structure, dividing the implementations into small parts, which will give a clear idea about our feature development.