Campus Room Schedule and Management System

Phase 4 Progress Report

Team 11 – CSAI 203: Introduction to Software Engineering

Date Submitted: December 6, 2025
Team Members:

Ahmed Ayman Mostafa – 202401612

Mohamed Ahmed Fouad – 202401032

Yousef Hossameldin Mohamed – 202402592

---

1. Introduction

This document presents the **official Progress Report for Phase 4 (Core Functionality Prototype)** of the *Campus Room Schedule and Management System*.
The project aims to replace the current manual room scheduling approach, which relies heavily on spreadsheets, with an integrated, database-driven web platform.

Phase 4 focuses on delivering the **minimum viable functional system (50% of the core requirements)** while ensuring correctness, proper architecture, and adherence to Software Engineering best practices, including MVC separation, modular design, and code documentation.

2. Phase 4 Objectives

Phase 4 was structured around four primary goals:

1. **Establish the software architecture**
    - Flask application factory pattern
    - SQLAlchemy ORM models
    - Flask-Migrate infrastructure
2. **Implement 5 major functional requirements (FRs):**
    - FR-1 Authentication
    - FR-3 Room Management

   - FR-4 Dashboard (simple version)
   - FR-5 Manual Room Status Override
   - FR-2 Basic Schedule Import
   - FR-6 Basic Issue Reporting
3. **Develop 3 unit tests**, one per member, verifying the correctness of the implemented core functionality.
4. **Complete integration**, resolve merge conflicts, and validate system behavior as one coherent application.

3. Completed Functional Requirements

The following table summarizes all features successfully implemented during Phase 4.

FR ID	Functional Requirement	Owner	Status
FR-1	User Authentication (Login/Logout + Role Field)	Ahmed	Completed
FR-3	Room Management (List, Add Room)	Ahmed	Completed
FR-4	Dashboard Showing Today's Schedules	Mohamed	Completed
FR-5	Manual Room Status Override	Mohamed	Completed
FR-2	Basic File Import (.xlsx/.csv → Schedule Table)	Yousef	Completed
FR-6	Issue Reporting (Submit + List Issues)	Yousef	Completed

All features were developed following the **MVC architecture** and merged through GitHub using separate feature branches.

4. System Architecture Summary (Phase 4)

4.1 MVC Pattern

The system follows a clean separation of layers:

- **Models:** SQLAlchemy ORM classes (User, Room, Schedule, Issue)

- **Views:** Templates using Jinja2 (login.html, rooms_list.html, dashboard.html...)
- **Controllers:** Blueprints handling routes (auth.py, admin.py, dashboard.py, rooms.py, issues.py, imports.py)

4.2 Database

- SQLite used for development
- Fully initialized with Flask-Migrate
- Initial migration created: tables for Users, Rooms, Schedules, Issues

4.3 Blueprints Implemented

- /auth — Authentication
- /admin — Room Management
- /dashboard — Room schedule display
- /rooms — Status override
- /issues — Issue reporting
- /import — File upload + schedule loading

5. Work Completed by Each Team Member

5.1 Ahmed Ayman Mostafa — Authentication & Room Management

Implemented Features

1. **Database Setup**
    - Configured SQLAlchemy
    - Initialized Flask-Migrate
    - Database created and connected successfully
2. **FR-1: Authentication**
    - User model (id, name, email, password_hash, role)
    - Password hashing using Werkzeug
    - Login route
    - Logout route

    - Session handling via Flask-Login

3. **FR-3: Room Management**
    - Room model
    - /rooms list page
    - /rooms/add page
    - Admin-only behavior (checked manually in Phase 4)

Unit Test

- test_auth.py: Valid login test
- Status: **Passed**

5.2 Mohamed Ahmed Fouad — Dashboard & Status Override

Implemented Features

1. **FR-4: Dashboard**
    - Schedule model
    - Controller: /dashboard
    - Displays today's schedules
    - Calculates real-time status (open/closed)
2. **FR-5: Status Override**
    - /rooms/<id>/toggle
    - Toggles room status between *Available* → *Occupied*
    - Instant update on the dashboard
3. **Base Template**
    - Navigation bar
    - Flash messaging
    - Unified layout for all views

Unit Test

- test_dashboard.py: Dashboard render test
- Status: **Passed**

5.3 Yousef Hossameldin — File Import & Issue Reporting

Implemented Features

1. **FR-2: Basic Schedule Import**
   - File upload (CSV/XLSX)
   - Pandas parsing
   - Direct insertion into Schedule table
   - Assumes fixed column structure (Phase 5 adds validation)
2. **FR-6: Issue Reporting**
   - Issue model
   - /issues/report form
   - /issues listing
3. **File Upload Configuration**
   - Upload directory
   - File validation (.xlsx/.csv)

Unit Test

- test_issues.py: Issue creation test
- Status: **Passed**

6. Integration & Testing

All team members followed the Git workflow:

1. Each member developed on a separate feature branch.
2. Pull requests were created, reviewed, and merged into main.
3. Final integration testing was performed.

4. Conflicts were resolved collaboratively.
5. Application tested end-to-end.

Final Result

Application runs without errors
All features accessible
All migrations functional
Unit tests passing

7. Known Limitations (To Be Fixed in Phase 5)

1. No countdown timers on the dashboard
2. No audit log for status override
3. No advanced Excel validation
4. No role-based access decorator
5. UI not polished
6. No search/filter functionality
7. No exporting reports

All these components are planned for Phase 5.

8. Conclusion

Phase 4 successfully delivered more than **50% of the system functionality**, demonstrating:

- Correct system architecture
- Working authentication
- Room management
- Dashboard integration
- File import
- Issue reporting

- Automated tests

The project is now on track for the **Phase 5 full delivery**, which will focus on feature completeness, UI enhancement, search/filter, exporting, and deployment (Docker + CI/CD).