

Phase 4 — Progress Report

Campus Room Schedule and Management System

Team 11 — CSAI 203

Course: CSAI 203 - Introduction to Software
Engineering

Instructor: Dr. Mohamed Rakha

Semester: Fall 2025

Team: 11

Date Prepared: December 6, 2025

Contents

1	Executive Summary	2
1.1	Scope Achieved	2
2	Project Status — Feature Checklist	2
3	Completed Work — Technical Details	2
3.1	Database and Migrations	2
3.2	Authentication	3
3.3	Room Management	3
3.4	Dashboard	3
3.5	Excel Import & Issue Reporting (Prototype)	3
4	Tests and Quality Assurance	3
5	Screenshots & Evidence	4
6	Known Issues & Risks	4
7	Next Steps (Phase 5 Priorities)	4
8	Appendix	4

1 Executive Summary

This progress report documents Phase 4 (Core Functionality Prototype) development status for the Campus Room Schedule and Management System. The team focused on implementing an MVP that covers the highest-priority functional requirements. The objective for Phase 4 was to implement approximately 50% of the SRS functional requirements with working, testable functionality.

1.1 Scope Achieved

In Phase 4 we implemented and verified the following components:

- User authentication (login/logout) with secure password hashing — **Done**
- Room entity management (list, add, toggle status) — **Done**
- Dashboard: today's schedules listing and status computation — **Done**(countdown timers deferred)
- Basic Excel/CSV import endpoint (prototype) — **Partial**
- Issue reporting (basic form + persistence) — **Partial**
- Unit tests: authentication, dashboard — **Done**

2 Project Status — Feature Checklist

FR ID	Feature	Status
FR-1	Authentication (login/logout), user model, role field	Done
FR-2	Excel / CSV import (basic)	Partial
FR-3	Room management (list, add, toggle status)	Done
FR-4	Dashboard (today's schedule, status)	Done
FR-5	Manual status override (toggle)	Done
FR-6	Issue reporting (create/list)	Partial

Table 1: Phase 4 functional status

3 Completed Work — Technical Details

3.1 Database and Migrations

- SQLAlchemy configured with SQLite for development (`src/config.py`).
- Flask-Migrate initialized; initial migration created and applied.
- Core models implemented: `User`, `Room`, `Schedule` (prototype), `Issue` (prototype).

3.2 Authentication

- **User** model: `id`, `name`, `email` (indexed), `password_hash`, `role`.
- Passwords hashed with Werkzeug utilities (production: can switch to `bcrypt`).
- Routes: `/auth/login` (GET/POST), `/auth/logout`.
- Session management via Flask-Login.
- Admin user seeded via `scripts/seed.py`.

3.3 Room Management

- **Room** model: `id`, `building`, `number`, `status`.
- Routes: `/rooms` (list), `/rooms/add` (GET/POST), `/rooms/{id}/toggle` (POST).
- Templates: `rooms_list.html`, `room_form.html`. Simple UI for functionality verification.

3.4 Dashboard

- **Schedule** model prototype: `date`, `open_time`, `close_time`, `room_id`.
- `/dashboard` route queries today's schedules and computes Open/Closed status.
- Countdown timers and auto-refresh deferred to Phase 5 (front-end JS).

3.5 Excel Import & Issue Reporting (Prototype)

- Import endpoint implemented as prototype; uses Pandas when invoked (import done lazily to avoid CLI import errors).
- Issue reporting form persists minimal issue records: `id`, `room_id`, `reporter_id`, `description`, `status`, `created_at`.

4 Tests and Quality Assurance

- Unit tests created and executed:
 - `tests/test_auth.py`: test for valid login (passes)
 - `tests/test_dashboard.py`: dashboard loads (passes)
- Local test command: `pytest -q`
- Integration: basic manual integration performed (`login` → `dashboard` → `rooms`).

5 Screenshots & Evidence

(Insert screenshots here in final PDF)

- Login page — `screenshots/login.png`
- Dashboard — `screenshots/dashboard.png`
- Rooms list — `screenshots/roomslist.png`
- Issue report form | `screenshots/issuereport.png`

6 Known Issues & Risks

- **Pandas / NumPy binary incompatibility:** resolved by pinning compatible versions in `requirements.txt`. Import of heavy libs (pandas) is now lazy within route handlers.
- **Excel import:** column-mapping and validation not implemented (deferred to Phase 5).
- **Role-based decorators:** simple role checks used; formal decorators/ACL to be added in Phase 5.
- **UI polish:** minimal styling only (Phase 5 will include CSS/JS improvements).

7 Next Steps (Phase 5 Priorities)

1. Complete Excel import with column mapping and validation (Yousef).
2. Implement countdown timers and auto-refresh on dashboard (Mohamed).
3. Add full issue lifecycle (status changes, audit logs) and search/filter for rooms (Ahmed).
4. Add role-based decorators and finer-grained RBAC.
5. Add CI (GitHub Actions) and Dockerization for deployment (Mohamed).
6. Polish UI and create final presentation materials.

8 Appendix

- SRS document: included in `/docs` (reference).
- Work Distribution document: this file integrates the Phase 4 workload.
- Commands to reproduce environment:

```
python -m venv venv
venv\Scripts\activate
pip install -r requirements.txt
flask db upgrade
python scripts/seed.py
python run.py
```

Prepared by:

Ahmed Ayman Mostafa — 202401612

Mohamed Ahmed Fouad — 202401032

Yousef Hossameldin Mohamed — 202402592