Phase 1

Agile Software Engineering Project

CSE233

Yousef Amr Mohamed Hassen Gamie
22P0274

PROJECT PRODUCT BACKLOG

EPIC 1 — Authentication & User Roles

US-AUTH-1: Login

As a user, I want to log into the system so that I can access my dashboard.
Story Points: 3

Acceptance Criteria

- System authenticates user using Firebase email/password authentication.
- The system fetches the user's role from Firestore (users/{uid}.role).
- After successful login, the system redirects based on role:
  - Student → /student/dashboard
  - Professor → /professor/dashboard
  - Admin → /admin/dashboard
- Invalid credentials show an error message.
- Login page must display loading state while authenticating.

EPIC 2 — Facilities / Room Scheduling Module

US-STU-1 — View Assigned Rooms

As a student, I want to view only the rooms assigned to my registered courses/exams so that I know where my classes are held.
Story Points: 5

Acceptance Criteria

- System fetches student’s enrolled courses from enrollments/{studentId}.
- For each course, system retrieves the corresponding room from courseRooms.
- Student only sees rooms linked to their own courses.
- Student **cannot see rooms** unrelated to them.
- Rooms must display:
    - Room name
    - Room type
    - Location
    - Capacity
- Rooms appear in a structured list.
- Rooms must update in real time if Firestore data changes.

US-PROF-1 — View Room Schedules

As a professor, I want to view room schedules so that I can check available time slots.
Story Points: 3

Acceptance Criteria

- The system retrieves all bookings for a specific room.
- Bookings must be displayed in a **weekly calendar view**.
- Each schedule entry must show:
  - Course
  - Professor
  - Date
  - Start time
  - End time
- Time slots marked as “Booked” must be visually distinguishable.
- Rooms under maintenance must be visually highlighted.
- Calendar updates in real time if bookings or maintenance status changes.

US-PROF-2 — Request a Room

As a professor, I want to request a room for a lecture/exam/event so that I can hold an academic activity.
Story Points: 3

Acceptance Criteria

- Professor selects:
  - Room
  - Date
  - Start time
  - End time
  - Course
- Request is stored in Firestore under roomRequests with status = “pending”.
- If the room is under maintenance → **request is blocked** with an error message.
- If the selected time slot overlaps with an existing booking → **request is blocked**.
- The system validates time conflicts using Firestore data.
- Successful requests appear instantly in professor’s “My Requests” page.

US-PROF-3 — Cancel My Pending Request

As a professor, I want to cancel a room request I previously submitted so that I have flexibility.
Story Points: 3

Acceptance Criteria

- System loads professor’s requests where status == "pending".
- Professor can only cancel requests that:
    - Belong to them
    - Are still pending
- On cancellation:
    - The request document is deleted from Firestore.
    - The list refreshes automatically.
- Admin view must reflect cancellation immediately.

US-ADMIN-0 — View Pending Requests

As an admin, I want to view all pending requests so that I can accept or reject them.
Story Points: 3

Acceptance Criteria

- System retrieves **all roomRequests where status == "pending"**.
- Each request must display:
  - Request ID
  - Room
  - Course
  - Professor
  - Date
  - Start time
  - End time
  - CreatedAt timestamp
- Requests appear in a **structured list/table**.
- Requests sorted by createdAt, newest first.
- Only admins can access this view.
- List updates in real time on approve/reject/cancel.

US-ADMIN-1 — Add New Room

As an admin, I want to add new rooms so that they become available in the system.
Story Points: 3

Acceptance Criteria

- Admin enters:
    - Room name
    - Room type
    - Capacity
    - Location
- Room is stored in Firestore under rooms.
- Saved room appears immediately in room lists for all roles.
- Form validates missing fields.
- Duplicate room names must be prevented or warned.

US-ADMIN-2 — Edit Room Information

As an admin, I want to edit room details so that the system reflects real changes.
Story Points: 3

Acceptance Criteria

- Admin can update:
    - Room name
    - Room type
    - Capacity
    - Location
- Updates applied immediately to Firestore.
- All users see updated information instantly.
- Editing is restricted to admin role only.

US-ADMIN-3 — Mark Room as Under Maintenance

As an admin, I want to mark a room as unavailable so that users know it cannot be booked.
Story Points: 3

Acceptance Criteria

- Admin toggles a “maintenance” flag on the room document.
- Room details must display maintenance status to all users.
- Professors cannot request a room while maintenance is active.
- Admin cannot approve bookings for rooms under maintenance.
- Future bookings in maintenance range must appear as **blocked**.

US-ADMIN-4 — Approve/Reject Room Requests

As an admin, I want to approve or reject professor room requests.
Story Points: 3

Acceptance Criteria

- Admin sees list of pending requests.
- Approving a request:
  - Creates a booking in bookings
  - Changes request status to "approved"
  - Prevents time conflicts with:
    - Existing bookings
    - Maintenance status
- Rejecting a request:
  - Sets request status to “rejected”
  - Stores rejection reason
- UI must update automatically after any action.
- Double booking must not be possible.

EPIC 3 — Administrative Office Automation

US-13: Manage Student Records

As an admin, I can create/update student profiles.
Story Points: 5

Acceptance Criteria

- Admin can:
  - Create student record
  - Update existing student record
- Student profile fields include:
  - Name
  - Department
  - Academic year
  - Courses enrolled
- Data stored under students/{studentId}.
- Changes must reflect instantly for all dependent modules (e.g., room assignment).

US-14: Generate Transcripts

As an admin, I want to auto-generate student transcripts.
Story Points: 8

Acceptance Criteria

- System retrieves:
    - Student data
    - Course grades
    - GPA
- Transcript generated in PDF or HTML format.
- Transcript must include:
    - Student name
    - Student ID
    - Course list
    - Credits
    - Grades
    - GPA
- Admin can download/print transcript.
- Transcript generation must handle missing data gracefully.

US-15: Handle Admission Applications

As an admissions officer, I want to manage incoming applications.
Story Points: 8

Acceptance Criteria

- System displays list of submitted applications.
- Each application shows:
  - Applicant name
  - Program applied for
  - Status
- Admin can:
  - Approve
  - Reject
  - Request more documents
- Application status updates in real time.
- Approved applicants stored in students collection.

EPIC 4 — Resource Allocation

US-16: Add Equipment / Software Resource

As an admin, I want to add a resource (equipment or software).
Story Points: 5

Acceptance Criteria

- Admin enters:
  - Resource name
  - Type
  - Quantity
  - Department
- Resource saved in Firestore under resources.
- Resources appear immediately to admin dashboard.

US-17: Allocate Resource to a Department

As admin, I want to allocate resources to departments.
Story Points: 5

Acceptance Criteria

- Admin selects:
  - Resource
  - Receiving department/faculty
  - Quantity allocated
- Allocation recorded under resourceAllocations.
- System ensures:
  - Allocation quantity ≤ available quantity

- Allocation updates instantly.

US-18: Track Resource Usage

As admin, I want to track which department/faculty/student is using each resource.
Story Points: 8

Acceptance Criteria

- System displays:
    - Resource
    - Who is using it
    - Department
    - Allocation date
    - Quantity used
- Usage displayed in a structured table.
- Admin can filter by:
    - Resource type
    - Department
    - User
- Data updates automatically when allocations change.

EPIC 5 — System Dashboard & Reporting

US-19: Admin Dashboard

As an admin, I want an overview of room usage, requests, resources, etc.
Story Points: 8

Acceptance Criteria

- Dashboard must show:
    - Number of rooms
    - Pending room requests
    - Rooms under maintenance
    - Recent bookings
    - Resource allocation summary
- Data updates in real time.
- Only admins can access.

US-20: Professor Dashboard

As a professor, I want to view my requests, booking history, and upcoming schedules.
Story Points: 5

Acceptance Criteria

- Dashboard displays:
    - Upcoming bookings
    - Pending requests
    - Past bookings
- Data updates automatically.

- Professor cannot view or edit other professors' data.

US-21: Student Dashboard

As a student, I want a simple dashboard showing my courses, rooms, and schedules.
Story Points: 3

Acceptance Criteria

- Dashboard displays:
    - Student’s enrolled courses
    - Assigned rooms
    - Schedule for the week
- Data updates in real time.
- Only the logged-in student can view their data.

PHASE 1 OVERVIEW

Objective of Phase 1 it is a 2 sprints phase that is intended to Deliver a fully functional core of the **Facilities / Room Scheduling Module** with role-based access control.

Phase 1 key Features

- Role-based authentication (Student / Professor / Admin) using Firebase Auth + Firestore roles
- Students can only view rooms assigned to their enrolled courses/exams
- Professors can view any room's weekly schedule and availability
- Professors can submit and cancel room booking requests
- Real-time conflict detection (no double booking)
- Real-time maintenance blocking
- Admin can add, edit, delete, and mark rooms as "under maintenance"
- Admin can view, approve, or reject professor requests (with automatic booking creation on approval)
- All changes reflected instantly across all roles thanks to Firestore real-time listeners
- UI built with Next.js 14 (App Router)
- Firestore security rules implemented

User Roles

Student

- Can ONLY view rooms related to their registered courses/exams

Professor

- Can view room schedules
- Can request rooms for lectures/exams/events
- Can cancel their pending requests

Admin

- Can add/edit rooms
- Can mark rooms under maintenance
- Can approve/reject professor room requests

Role	**Permissions for phase**
Student	Login → View only assigned rooms → See room details
Professor	Login → View any room schedule → Request room → Cancel own pending requests → View own requests
Admin	Login → Full CRUD on rooms → Toggle maintenance → View/Approve/Reject all requests

Technology Stack

Frontend: Next.js, chakra ui, zustand, js

Backend: Firebase Authentication, Firestore , storage, and realtime database

Phase backlog (User stories for sprint)

US-AUTH-1— Login

As a user, I want to log into the system so that I can access my dashboard.
Story Points: 3

Acceptance Criteria

- System authenticates user using Firebase email/password authentication.
- The system fetches the user's role from Firestore (users/{uid}.role).
- After successful login, the system redirects based on role:
  - Student → /student/dashboard
  - Professor → /professor/dashboard
  - Admin → /admin/dashboard
- Invalid credentials show an error message.
- Login page must display loading state while authenticating.

Student Stories

US-STU-1 — View Assigned Rooms

As a student, I want to view only the rooms assigned to my registered courses/exams so that I know where my classes are held.
Story Points: 5

Acceptance Criteria

- System fetches student's enrolled courses from enrollments/{studentId}.
- For each course, system retrieves the corresponding room from courseRooms.
- Student only sees rooms linked to their own courses.
- Student **cannot see rooms** unrelated to them.
- Rooms must display:
  - Room name
  - Room type
  - Location
  - Capacity
- Rooms appear in a structured list.
- Rooms must update in real time if Firestore data changes.

Professor Stories

US-PROF-1 — View Room Schedules

As a professor, I want to view room schedules so that I can check available time slots.
Story Points: 3

Acceptance Criteria

- The system retrieves all bookings for a specific room.
- Bookings must be displayed in a **weekly calendar view**.
- Each schedule entry must show:
    - Course
    - Professor
    - Date
    - Start time
    - End time
- Time slots marked as “Booked” must be visually distinguishable.
- Rooms under maintenance must be visually highlighted.
- Calendar updates in real time if bookings or maintenance status changes.

US-PROF-2 — Request a Room

As a professor, I want to request a room for a lecture/exam/event so that I can hold an academic activity.
Story Points: 3

Acceptance Criteria

- Professor selects:
  - Room
  - Date
  - Start time
  - End time
  - Course
- Request is stored in Firestore under roomRequests with status = "pending".
- If the room is under maintenance → **request is blocked** with an error message.
- If the selected time slot overlaps with an existing booking → **request is blocked**.
- The system validates time conflicts using Firestore data.
- Successful requests appear instantly in professor's "My Requests" page.

US-PROF-3 — Cancel My Pending Request

As a professor, I want to cancel a room request I previously submitted so that I have flexibility.
Story Points: 3

Acceptance Criteria

- System loads professor's requests where status == "pending".
- Professor can only cancel requests that:
    - Belong to them
    - Are still pending
- On cancellation:
    - The request document is deleted from Firestore.
    - The list refreshes automatically.
- Admin view must reflect cancellation immediately.

Admin Stories

US-ADMIN-0 — View Pending Requests

As an admin, I want to view all pending requests so that I can accept or reject them.
Story Points: 3

Acceptance Criteria

- System retrieves **all roomRequests where status == "pending"**.
- Each request must display:
  - Request ID
  - Room
  - Course
  - Professor
  - Date
  - Start time
  - End time
  - CreatedAt timestamp
- Requests appear in a **structured list/table**.
- Requests sorted by createdAt, newest first.
- Only admins can access this view.
- List updates in real time on approve/reject/cancel.

US-ADMIN-1 — Add New Room

As an admin, I want to add new rooms so that they become available in the system.
Story Points: 3

Acceptance Criteria

- Admin enters:
    - Room name
    - Room type
    - Capacity
    - Location
- Room is stored in Firestore under rooms.
- Saved room appears immediately in room lists for all roles.
- Form validates missing fields.
- Duplicate room names must be prevented or warned.

US-ADMIN-2 — Edit Room Information

As an admin, I want to edit room details so that the system reflects real changes.
Story Points: 3

Acceptance Criteria

- Admin can update:
    - Room name
    - Room type
    - Capacity
    - Location
- Updates applied immediately to Firestore.
- All users see updated information instantly.
- Editing is restricted to admin role only.

US-ADMIN-3 — Mark Room as Under Maintenance

As an admin, I want to mark a room as unavailable so that users know it cannot be booked.
Story Points: 3

Acceptance Criteria

- Admin toggles a “maintenance” flag on the room document.
- Room details must display maintenance status to all users.
- Professors cannot request a room while maintenance is active.
- Admin cannot approve bookings for rooms under maintenance.
- Future bookings in maintenance range must appear as **blocked**.

US-ADMIN-4 — Approve/Reject Room Requests

As an admin, I want to approve or reject professor room requests.
Story Points: 3

Acceptance Criteria

- Admin sees list of pending requests.
- Approving a request:
  - Creates a booking in bookings
  - Changes request status to "approved"
  - Prevents time conflicts with:
    - Existing bookings
    - Maintenance status
- Rejecting a request:
  - Sets request status to “rejected”
  - Stores rejection reason
- UI must update automatically after any action.
- Double booking must not be possible.

SPRINT 1 GOAL

Deliver the core foundation of the Facilities Module:

- Student limited room viewing
- Room schedule viewing
- Professors can submit room requests
- Professors can cancel their own pending requests
- Firestore database (EAV-compatible) initialized
- Authentication + role-based access implemented

SPRINT 1 — BACKLOG (SELECTED USER STORIES)

US-AUTH-1— Login

US-STU-1 — View Assigned Rooms

US-PROF-1 — View Room Schedules

US-PROF-2 — Request a Room

US-PROF-3 — Cancel My Pending Request

SPRINT 1 — development tasks

Backend

B1 — Initialize Firebase project

- Setup Firestore
- Setup Firebase Auth
- Setup Firestore security rules

B2 — Create Firestore collections

- rooms/
- bookings/
- roomRequests/
- enrollments/
- courseRooms/

B3 — Implement Student Room Filtering

- Fetch student courses based on enrollment
- Map courses → assigned rooms
- Return list of student rooms only

B4 — Implement Room Schedule Query

- Query bookings by room
- Query maintenance flag
- Construct weekly calendar

B5 — Implement Room Request Creation

- Validate maintenance
- Validate conflict checking
- Insert pending request

B6 — Implement Cancel Request Function

- Security rule → only the request owner
- Delete request document

Frontend

F1 — Project Setup

- Create Next.js app
- Install Firebase SDK
- Setup /lib/firebase.js
- Setup global role-based route protection

F2 — Student Rooms Page

Path: /rooms

- Fetch student rooms
- Display list
- Show basic information

F3 — Room Schedule Page

Path: /rooms/[roomId]

- Show weekly schedule
- Display maintenance status
- Time-block view

F4 — Request Room Page

Path: /request-room

- Form: room, date, startTime, endTime
- Submit to Firestore
- Input validation built-in

F5 — Pending Request Management

Path: /my-requests

- List professor's pending requests
- Add cancel button
- Connect to cancel request function

UI Components

- C1 — RoomCard
- C2 — WeeklyScheduleGrid
- C3 — RequestRoomForm
- C4 — PendingRequestCard

Documentation Tasks

- D1 — Sprint Planning Document
- D2 — Daily Scrum Notes
- D3 — Sprint Review
- D4 — Sprint Retrospective

SPRINT 1 — ACCEPTANCE TESTS

AT1 — Student Sees Only Assigned Rooms

Given: Student is enrolled in CSE233 & CSE355
When: Student opens /rooms
Then: Only rooms linked to these courses appear

AT2 — Room Schedule Loads Properly

Given: Room 914A has bookings
When: Professor opens /rooms/914A
Then: Weekly schedule displays all booked slots

AT3 — Professor Room Request Works

Given: Room is not in maintenance and time is free
When: Professor submits a request
Then: A “pending” request appears in the database

AT4 — Request Blocked if Room is Maintained

Given: Room 921A is under maintenance
When: Professor tries to request it
Then: System shows error “Room unavailable”

AT5 — Request Cancellation Works

Given: Professor has a pending request
When: They click “Cancel”
Then: Request is deleted

SPRINT 1 — DEFINITION OF DONE

- All selected user stories implemented
- All acceptance criteria met
- Firestore and Auth working
- Code builds without warnings
- Basic UI fully functional
- Connected to backend
- Sprint Review completed
- Retrospective completed
- Code committed to Git

SPRINT 1 — MEETING MINUTES

Sprint Planning

Date: 3/11/2025, 18/11 to 21/11/2025
Attendees: (Scrum Master, PO, Devs)
Stories Selected: US-AUTH-1, US-STU-1, US-PROF-1, US-PROF-2, US-PROF-3

Daily Scrum

Daily Scrum — 18/11/2025

Some time ago:

- Set up Next.js project structure and initialized Firebase config.
- Created Firestore collections for rooms, bookings, enrollments, roomRequests.

Today:

- Implement student → course → room filtering logic.
- Begin building the /rooms student view page.

Blockers:

- Need to confirm Firestore security rules for role-based access.

Daily Scrum — 19/11/2025

Yesterday:

- Finished student room filtering.
- Displayed student-specific rooms on /rooms.

Today:

- Build /rooms/[roomId] schedule page.
- Fetch bookings for selected room and show weekly grid.

Blockers:

- Minor difficulty organizing the weekly schedule layout.

Daily Scrum — 20/11/2025

Yesterday:

- Completed weekly schedule display with booked/free slots.
- Integrated maintenance status display.

Today:

- Build request room form for professors on /request-room.
- Implement request validation (conflicts + maintenance).

Blockers:

- Need to finalize conflict check function for overlapping bookings.

Daily Scrum — 21/11/2025

Yesterday:

- Completed room request submission and validation.
- Requests save as “pending” in Firestore.

Today:

- Implement cancel request page /my-requests.
- Test all sprint features and prepare for Sprint Review demo.

Blockers:

- None.

Sprint Review

Delivered:

- Student room filtering
- Schedule view
- Room request form
- Cancel request function

Demo Summary: to be added after discussion

Sprint Retrospective

Went Well: most of it goes well
Didn’t Go Well: nothing
Improvements for Sprint 2: professor View My Approved/Rejected.