FINAL INTERNATIONAL UNIVERSITY

SFWE343/SOFT343/COMP342 -

Software Analysis & Design

Course Project Introduction

As part of this course, you will be working on practical, real-world system development projects. The goal is to help you apply theoretical knowledge to problem-solving, system design, and implementation while working collaboratively in teams.

- There are +40 students enrolled in this class.
- You will be divided into 8 groups
- Each group must select one project from the provided list.
- Once a project has been selected by a group, it will no longer be available to other groups.
- A Team & Project Selection Form will be opened later for registration.
- Group formation and project selection will be on a first-come, first-served basis.

This setup ensures:

1. **Collaboration:** You will learn how to work in teams with shared responsibilities.
2. **Diversity of Output:** Each group will work on a different system, reducing duplication and encouraging innovation.
3. **Ownership & Accountability:** Since no two groups will work on the same project, you will have full responsibility for the design, development, and presentation of your chosen system.

By the end of the course, each group will present:

- A working prototype of their system.
- Documentation covering requirements, system design, and implementation.
- A reflection on teamwork and problem-solving challenges faced during development.

1. Course Outlines Management System

Objective:

Centralize course outline management, ensuring consistency across faculties.

Features & Workflow:

- Generate outlines based on the instructor inputs
- Detect any potential wrong information.
- Ensure a flexible workflow of approval
- Ensure a synchronous edit in all outlines
- Check the outline content.

System Architecture:

- **Frontend:** Web portal (React/Angular).
- **Backend:** Django REST / Laravel API.
- **Database:** PostgreSQL for metadata, AWS S3 for file storage.
- **Auth:** University Single Sign-On (SSO).

Implementation Notes:

- Should include a **course code + semester tagging system**.
- Versioning: Allow multiple versions of outlines but keep “final” marked.

2. Ethic Committee Form Submission

Objective:

Digitize and streamline ethics approval workflows for research projects.

Features & Workflow:

1. Researcher submits ethics application form online (upload attachments like research proposal, consent forms).
2. System routes to the relevant ethics committee.
3. Committee reviews → approve/reject/revision request.
4. Researcher notified automatically.
5. Archive of all ethics requests (linked to researcher’s profile).

System Architecture:

- **Frontend:** Dynamic forms engine (Form.io or custom React forms).
- **Backend:** Workflow engine (Camunda, Flowable, or Django).
- **Database:** MySQL/Postgres with audit logs.

- **Storage:** Cloud storage for large files.

Implementation Notes:

- Add **document checklist validation** (no incomplete submissions).
- Multi-level approval required (Faculty → Ethics Committee → Admin).

3. Reminder System for Deans & Directors

Objective:

Ensure quarterly meetings are scheduled and not forgotten.

Features & Workflow:

- Admin sets meeting schedule (4 per year, every 3 months).
- System sends reminders to dean & secretary **1 month, 1 week, 1 day** before meeting.
- Agenda/Minutes can be attached.
- Dashboard for past/upcoming meetings.

System Architecture:

- **Backend:** Cron jobs with Django Celery / Node.js scheduler.
- **Frontend:** Minimal dashboard for scheduling.
- **Database:** PostgreSQL.
- **Integration:** Google Calendar / Google API.

Implementation Notes:

- Must allow **custom reminders** beyond quarterly defaults.
- Generate **automatic attendance reports**.

4. Office Placement Application

Objective:

Digitally track and manage faculty office assignments.

Features & Workflow:

- Database of office numbers, capacity, building/floor.
- Record of **occupiers** (faculty/staff).
- Search & filter by department, occupant, or office number.
- Admin panel to update changes.
- Exportable reports for planning.

System Architecture:

- **Frontend:** React with advanced search.
- **Backend:** Django REST.
- **Database:** PostgreSQL (with relational mapping).
- **Optional:** Campus map integration with office visualization.

Implementation Notes:

- Should track **office history** (who occupied it previously).
- Flag conflicts (e.g., office overcapacity).

---

5. Thesis Check Platform

Objective:

Support academic integrity by allowing students to self-check their thesis drafts before submission.

Features & Workflow:

- Upload thesis draft (PDF/Word).
- Plagiarism check against a local repository or external API.
- Grammar and citation validation.
- Feedback report for students to revise.

System Architecture:

- **Backend**: Python + NLP libraries (spaCy, NLTK).
- **Database**: PostgreSQL (student submissions, thesis metadata).
- **File Storage**: Cloud (AWS S3, Azure).
- **Optional:** Integration with Turnitin or open-source plagiarism APIs.

Implementation Notes:

- Ensure reports are confidential.
- Build plagiarism detection against local thesis repositories for uniqueness.

6. Smart Studying Timetable Generator

Objective

The **Smart Studying Timetable Generator** is a web-based tool that helps students automatically create and manage personalized study timetables. It considers **exam dates,**

class schedules, personal availability, and study goals to generate an optimized, conflict-free plan.

Key Features

- **User Preferences:** Set study times, daily limits, and session lengths.
- **Timetable Generator:** Auto-schedules study sessions around classes, exams, and deadlines.
- **Adaptive Scheduling:** Adjusts based on student feedback (completed/skipped sessions).
- **Calendar Integration:** Sync with Google Calendar or Outlook.
- **Reminders & Notifications:** Alerts before study sessions; weekly study summaries.
- **Progress Dashboard:** Track time spent, completed tasks, and readiness.
- **Drag & Drop Editing:** Students can move sessions, system re-optimizes automatically.

Technical Notes

- **Frontend:** React (Web), Tailwind CSS.
- **Backend:** Django REST / Node.js (API + scheduling engine).
- **Database:** PostgreSQL (study sessions, users, preferences).
- **Scheduler:** Celery + Redis (Python) or BullMQ (Node.js).
- **Integrations:** Google Calendar API, SendGrid/Firebase for notifications.

7. Project Collaboration Platform (HUB)

Objective

The **Project Collaboration Hub** is a digital workspace for students to **plan, manage, and collaborate** on academic or extracurricular projects. It centralizes tasks, files, discussions, and progress tracking, making teamwork more structured and efficient.

Key Features

- **Group Creation:** Students can form teams, invite members, and assign roles.
- **Task Management:** Shared task lists, Kanban board, and milestones.
- **File Sharing:** Upload project documents, resources, and presentations.
- **Communication Tools:** Built-in messaging or discussion boards for coordination.
- **Progress Tracking:** Timeline view with deadlines and completion percentage.
- **Integrations:** Option to connect with Google Drive, GitHub, or MS Teams.

User Roles

- **Student:** Create/join project groups, manage tasks, upload files, chat.
- **Faculty Advisor (Optional):** Monitor project progress, leave feedback.
- **Admin:** Manage user accounts and moderate platform.

Technical Notes

- **Frontend:** React + Tailwind (web app).
- **Backend:** Node.js / Django REST API with WebSockets for real-time updates.
- **Database:** PostgreSQL (users, projects, tasks, messages).
- **File Storage:** AWS S3 or Google Drive API.
- **Notifications:** Email or push alerts for deadlines and updates.

8. Thesis/Capstone Repository

Objective

The **Thesis/Capstone Repository** is a centralized digital archive of past student theses and capstone projects. It allows students and faculty to **search, browse, and reference previous work**, encouraging knowledge sharing and preventing duplicate topics.

Key Features

- **Centralized Storage:** Upload and manage thesis/capstone documents in PDF or Word.
- **Search & Filter:** Search by keyword, student name, supervisor, department, or year.
- **Metadata Records:** Store title, abstract, keywords, and approval details.
- **Access Control:** Define who can view/download (students, faculty, public).
- **Upload & Approval Workflow:** Students upload → faculty/advisor approve → published to repository.
- **Export & Citation:** Auto-generate references in common citation formats.

Technical Notes

- **Frontend:** React (web app), responsive design.
- **Backend:** Django REST / Node.js (API + workflow engine).
- **Database:** PostgreSQL (thesis metadata, user info).
- **File Storage:** AWS S3, Google Drive API, or university server.
- **Search Engine:** ElasticSearch or PostgreSQL full-text search.