Department of Computer Engineering, Bilkent University

CS319 Object Oriented Software Engineering

Group 11

Design Goals & Subsystem Decomposition Diagram - D3

Section 1 - Eray Tüzün

03/05/2025

Furkan Mert Aksakal 22003191
Furkan Komaç 22102165
Alper Biçer 22003097
Erkan Can Arslan 22103948
Deniz Yazıcı 21902557

Contents

Top 2 Design Goals

The design goals stem from the analysis of the **non‑functional requirements** for the TA Management System. While building the platform, the two goals below will have highest priority—even when trade‑offs arise, as explained at the end of each section.

Usability

The system must be effortless for every stakeholder—TAs, instructors, and department staff—to learn and operate.

- **Intuitive dashboards**: Each role lands on a tailored home‑screen that shows only the actions relevant to it (e.g., "Log Duty," "Approve Tasks," "Assign Proctors"). Common actions are reachable in **≤ 2 clicks**.
- **Guided forms & smart defaults:** When a TA records a duty, the form pre‑selects the current course/section and auto‑computes effort hours from a drop‑down template, reducing typing and errors.
- **Real‑time feedback:** Toast notifications confirm a submission or flag validation errors **< 1 s** after the user clicks "Save," so the user never wonders whether an action succeeded.
- **Accessible visual cues:** Colour‑blind‑safe status badges (green = approved, yellow = pending, red = rejected) make state obvious at a glance.
- **Search & filter everywhere:** All tables (TA list, proctor roster, duty log) support type‑ahead search and multi‑facet filtering, enabling staff to locate information in **< 5s**.

Reliability

The platform must consistently perform critical operations—duty logging, proctor assignment, leave blocking—without error or data loss.

- **ACID transactions:** Every create/update/delete on duties, proctor slots, or leave requests runs inside a single database transaction; if any sub‑step fails, nothing commits.

- **Exactly-once scheduling:** A message system sends tasks to the part of the system that assigns proctors. It's designed to make sure a TA never gets accidentally assigned to the same exam twice.
- **Health-checked micro-services:** If any part of the system crashes, it restarts automatically.
- **Versioned audit log:** Immutable, append-only records (timestamp, actor, before/after JSON diff) allow rapid forensic analysis and restore trust when disputes arise.

Possible Trade-offs

1. ***Reliability vs Cost-effectiveness*** – Running redundant database replicas and a message bus increases infrastructure spend. We limit replicas to exam periods and scale down during off-peak months.
2. ***User-friendliness vs Efficiency*** – Rich UI components and live search add client-side weight, which may slightly increase initial page-load time. Lazy-loading and code-splitting will mitigate the impact.
3. ***User-friendliness vs Security*** – Keeping workflows "password-free" for instructors (SSO token only) improves UX but heightens risk if tokens leak. We counterbalance with 15-minute idle timeouts and device-bound refresh tokens.

By prioritising **User-friendliness** and **Reliability**, the TA Management System delivers a **smooth daily workflow** for its users while **protecting exam integrity** and historical workload records, in line with established system-design best practices.

Subsystem Decomposition Diagram

link: https://drive.google.com/file/d/1qpHeOokRLjLRHgvIW3bL7bTl1J7oLOyb/view?usp=sharing

Description

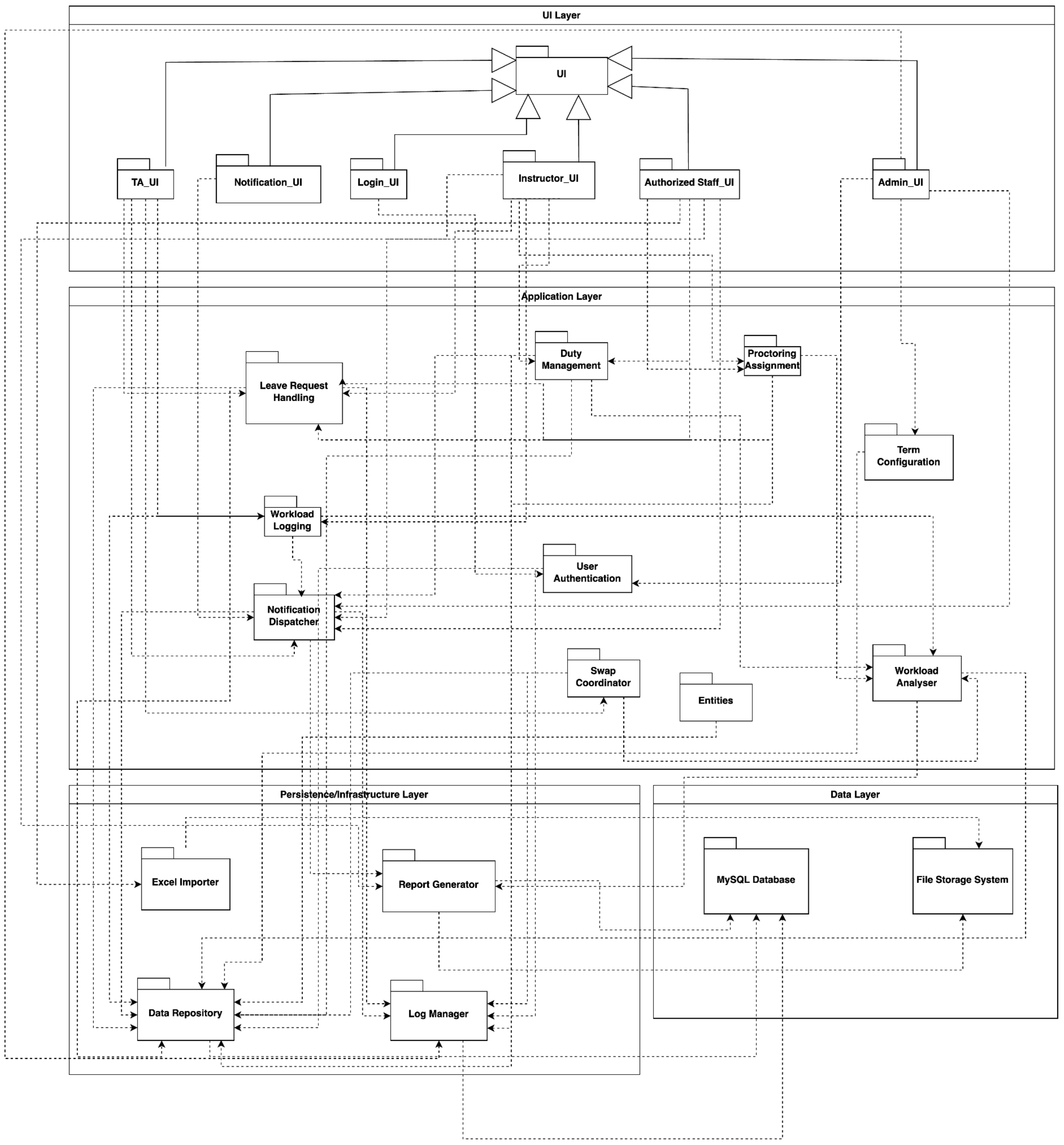
The system architecture is organized into four main layers: UI Layer, Application Layer, Persistence/Infrastructure Layer, and Data Layer.

The UI Layer contains the interfaces between users and the software. It includes corresponding interfaces for all user roles. It has a UI package for the UI template and the common elements. But there are dedicated interfaces for each user. For the secretary, dean and other authorized staff, the same UI is used since they have the same privileges in the system. Admin, instructor and TA have their own interfaces. There are also different interfaces for specific purposes such as Login UI and Notification UI.

The Application Layer implements the business logic of the system. There are Leave Request Handling (TAs can request for leaving), Duty Management (all duty processes), Proctoring Assignment (handles proctoring duties), Term Configuration (gathering and editing term information), Workload Logging (saves the workload), Notification Dispatcher (handles which notification goes to whom), User Authentication (handles authentication), Swap Coordinator (handles swap requests and responses), Entities (connects Persistence/Infrastructure Layer), and Workload Analyzer (checks if the workload exceeds for a specific TA) packages. User UIs are connected if that user is authorized to do that process.

The Persistence/Infrastructure Layer acts as a middleware between the business logic and data management. It provides reusable services and tools for data handling and system operations. There are Excel Importer (generates an excel report), Report Generator (displays information), Data Repository (connects Data Layer), and Log Manager (conducts logs).

The Data Layer is responsible for actual data storage and management. MySQL Database (database itself), and File Storage System (stores files).


UI Layer
UI
TA_UI
Notification_UI
Login_UI
Instructor_UI
Authorized Staff_UI
Admin_UI
Application Layer
Duty Management
Proctoring Assignment
Leave Request Handling
Term Configuration
Workload Logging
User Authentication
Notification Dispatcher
Swap Coordinator
Entities
Workload Analyser
Persistence/Infrastructure Layer
Excel Importer
Report Generator
Data Repository
Log Manager
Data Layer
MySQL Database
File Storage System