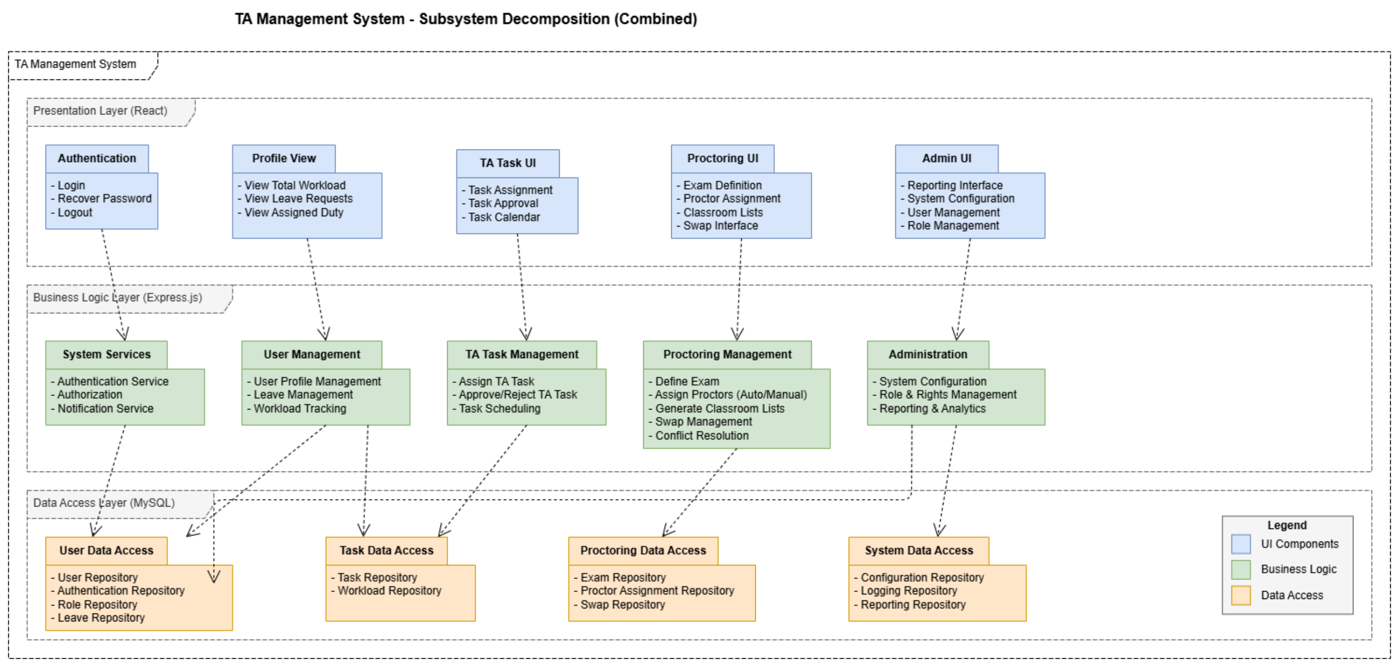
1.0 High-Level Software Architecture

1.1 Subsystem Decomposition



https://viewer.diagrams.net/?tags=%7B%7D&lightbox=1&highlight=0000ff&edit=_blank&layers=1&nav=1&title=subsystem_decomposition.drawio&dark=auto#Uhttps%3A%2F%2Fdrive.google.com%2Fuc%3Fid%3D1JB3DHsHjn22KDxksik8_IT_aSymsnXmR%26export%3Ddownload

We adopted a 3-layer architectural style for the TA Management System to enhance modularity and separation of concerns. The subsystems are organized into Presentation Layer, Business Logic Layer, and Data Access Layer, each with specific components that serve distinct purposes.

1.2 Presentation Layer (React)

The Presentation Layer handles user interaction through a React-based frontend. This layer provides the interface through which users interact with the system.

Authentication

- This package provides login, password recovery, and logout functionality.

- It communicates with the System Services package in the Business Logic Layer.

Profile View

- This package allows users to view their total workload, leave requests, and assigned duties.
- It communicates with the Leave Management package in the Business Logic Layer.

TA Task UI

- This package provides interfaces for task assignment, task approval, and task calendar views.
- It communicates with the TA Task Management package in the Business Logic Layer.

Proctoring UI

- This package contains screens for exam definition, proctor assignment, classroom lists, and swap interface.
- It communicates with both Proctoring Assignment and Proctor Swap packages in the Business Logic Layer.

Admin UI

- This package provides reporting interfaces, system configuration, user management, and role management.
- It communicates with Reporting and System Configuration packages in the Business Logic Layer.

1.3 Business Logic Layer (Express.js)

The Business Logic Layer implements the core functionality of the system using Express.js. It processes data from the Presentation Layer and communicates with the Data Access Layer.

TA Task Management

- This package handles assigning TA tasks, approving/rejecting TA tasks, and task scheduling.

- It communicates with the Task Data Access package in the Data Access Layer.

Leave Management

- This package manages requesting leave, approving/rejecting leave, and leave tracking.
- It communicates with the Task Data Access package in the Data Access Layer.

Proctoring Assignment

- This package handles defining exams, assigning proctors (automatically and manually), and generating classroom lists.
- It communicates with the Proctoring Data Access package in the Data Access Layer.

Proctor Swap

- This package enables initiating swaps by TAs or instructors, approving/rejecting swaps, and resolving conflicts.
- It communicates with the Proctoring Data Access package in the Data Access Layer.

System Services

- This package provides authentication services, authorization, and notification services.
- It communicates with the User Data Access package in the Data Access Layer.

Reporting

- This package generates workload reports, proctoring reports, and analytics.
- It communicates with the System Data Access package in the Data Access Layer.

System Configuration

- This package manages global parameters, role and rights management, and system maintenance.
- It communicates with the System Data Access package in the Data Access Layer.

1.4 Data Access Layer (MySQL)

The Data Access Layer handles database operations using MySQL. It provides data access and storage services to the Business Logic Layer.

User Data Access

- This package manages user repository, authentication repository, and role repository.
- It communicates with the System Services package in the Business Logic Layer.

Task Data Access

- This package manages task repository, workload repository, and leave repository.
- It communicates with TA Task Management and Leave Management packages in the Business Logic Layer.

Proctoring Data Access

- This package handles exam repository, proctor assignment repository, and swap repository.
- It communicates with Proctoring Assignment and Proctor Swap packages in the Business Logic Layer.

System Data Access

- This package manages configuration repository, logging repository, and reporting repository.
- It communicates with Reporting and System Configuration packages in the Business Logic Layer.

2.0 Design Goals

2.1 Modularity

When the variety of users of TA Management System and the various functions that each user will use are examined, each function (user login according to roles, adding completed tasks, permission requests, assigning tasks, reporting, etc.) is designed in separate interfaces in mock-ups and interface designs. These components ensure that the system has the principle of modularity. Each module, that is, each function, can perform its own function with a single page, can be tested independently with various scenarios, and can be updated quickly in case of a change. In this way, readability and analysis are facilitated at the frontend level in the code base, parallel development as backend and frontend within the group is facilitated, and when a new task type, a new function, that is, a module, or a new role is wanted to be added, the impact on other modules and the need for changes are minimized.

2.2 Usability

Since many different roles and numbers of users will use this application during the implementation phase of the application, the usability of the system is important for users and is among our design goals. Thanks to the prepared interfaces, there are consistent and clear guidance menus, form and description fields, instant error/success feedback for any task or leave request provided by the system and authorized user, and labels that allow the desired action to be taken in a practical way; these elements provide the Usability goal. In line with this design goal, even when users (TAs, instructor and authorized staff) use the system for the first time, thanks to these clear guidance and pages that you can easily perform each action, operation. The clarity of these pages comes from the statistics that we collect from users. 80% of people who tested our program said that they could easily access each transaction and perform it without getting confused. It means that they are able to quickly adapt to the use of the application, perform transactions without causing errors and confusion, and easily access the menus and frequently used operations such as “Add Task”, “Request Leave”, “View Assignments”. The simple but understandable design in the interfaces increases efficiency without confusing the user.

3.0 Design Trade-offs

3.1 Modularity vs Rapid Development

In order to provide a modular architecture that considers each function and user, it is necessary to design the interface and service systems of each module separately and integrate them with each other; this prolongs the process of creating the first prototype and the general ready-made application, and since it is a seasonal project, it may limit the possibility of testing and correction to some extent.

3.2 Usability vs Functionality

Designing a simple interface that aims for usability-oriented simplicity and makes user interface components more usable may limit the functionality provided in a modular manner. Considering that various functionality will create complexity for the sake of simplicity and choosing which way to focus was an important decision for our project. This situation and dilemma require balancing the separation of fully functional components with a user-friendly, usable experience.