# TA Management System

Deliverable-3 1st Iteration

24.04.2025

**Course:** CS 319

**Section:** 01

**Group No.:** 7

**Instructor:** Eray Tüzün

**Teaching Assistant:** Utku Boran Torun

**Group Members:** Bilge İdil Öziş [ 22102365 ]
Burak Kağan Gür [ 22203365 ]
Daib Malik [ 22201013 ]
Yunus Günay [ 22203758 ]

## 1. Top Two Design Goals:

a. **Usability:** Usability is the most critical design goal for the TA Management System. The application must serve a diverse user base that includes instructors, teaching assistants, dean, and secretary – each with different responsibilities and expectations. To address this, the system is designed to be intuitive, task-oriented, and accessible for all users, regardless of technical proficiency. Instructors should be able to assign TAs, review workload distributions, and configure preferences in no more than 7 clicks, without needing training. TAs must find it easy to view their assigned courses, submit leave or duty requests, and track feedback without navigating complex menus, and again within 7 clicks once they are authenticated by the system. Some instructors need a clear overview of the entire assignment matrix, including over-/under-loaded TAs, and must-have preferences. Similarly, the dean and department secretary may need to access reports, see charts, and accept/reject requests within 8 clicks. All of the above mentioned actions take no more than 4 seconds to take effect.

To achieve the best user experience, our dashboard and sidebar are tailored to each role and present only relevant options. Icons and color-coded badges help visually distinguish TA categories (e.g., must-have, avoided, preferred), while tabs and searchable tables help users find and act on information in under 2 seconds. All buttons that save information are blue and ones that risk potential deletion are colored red, so the user can make decisions without hesitation. All delete options also ask for confirmation before deleting anything. Dialog boxes and inline warnings guide users during sensitive actions (like overriding must-have TA constraints), reducing error rates. A red notification dot with a number in it for unread notifications and alert system ensures that urgent updates (e.g., a TA's leave approval or a reassignment) are never missed. We also avoid clutter by using collapsible sections and categorized filters, enabling users to focus on their tasks without distraction. The sidebar has proper, intuitive names like Reports, TA Assignment, Proctoring Assignments, and more that won't require a user to guess which functionality lies where. This makes it easy for a new user to settle in quickly and navigate their way smoothly. We made sure to follow a design layout that is common with most applications these days that users are familiar with. For example, the Logout button always appears in the sidebar, so there is no need to go to a specific page (like settings or profile) to exit. Contact button shows immediately on the dashboard, so there is no need to go through a list of names/emails on a different page to contact a particular user. Additionally, paths to all critical operations are just 1 or 2 clicks away from the main dashboard, minimizing user effort. Lastly, real-time feedback, such as success toasts and validation warnings, help guide users to correct outcomes.

**b.** **Reliability:** Reliability is the second key design goal of the TA Management System. The platform must consistently perform critical functions such as assigning TAs, validating workloads, and tracking preferences without failure. For example, when an instructor assigns a TA to a course, the system must accurately calculate the resulting workload and prevent violations of minimum or maximum thresholds. If a TA is removed or unavailable, the system must not crash or return corrupted data. We need to keep track of the correct workload distribution while other actions like swapping. Even a load differing by 1 or 2 units can burden the TA substantially. Reliability is ensured by addressing edge cases like avoiding duplicate TA assignments to the same course or preventing grader over-assignment beyond the instructor-specified count. At the same time, we ensure proper load allotment to activities to make sure an action like proctoring and overseeing a lab is not the same. This ensures the TA's workload is assigned fairly.

We rely on Django's ORM to enforce database integrity and transactional safety, so all assignment operations are atomic – either they succeed completely or not at all. Backend logic includes validation to avoid common pitfalls like assigning non-existent users or missing "must-have" TAs without proper acknowledgment. The system also provides fallback behaviors, if the API fails during development or testing. Role-based access control (RBAC) ensures that only authorized users can make sensitive changes, and all assignment actions are logged for later traceability. Updates to allocations, preferences, or leave requests are reflected in real-time, so users always view the latest state of the system, preventing outdated information from influencing decisions.

2. **Trade-offs:**

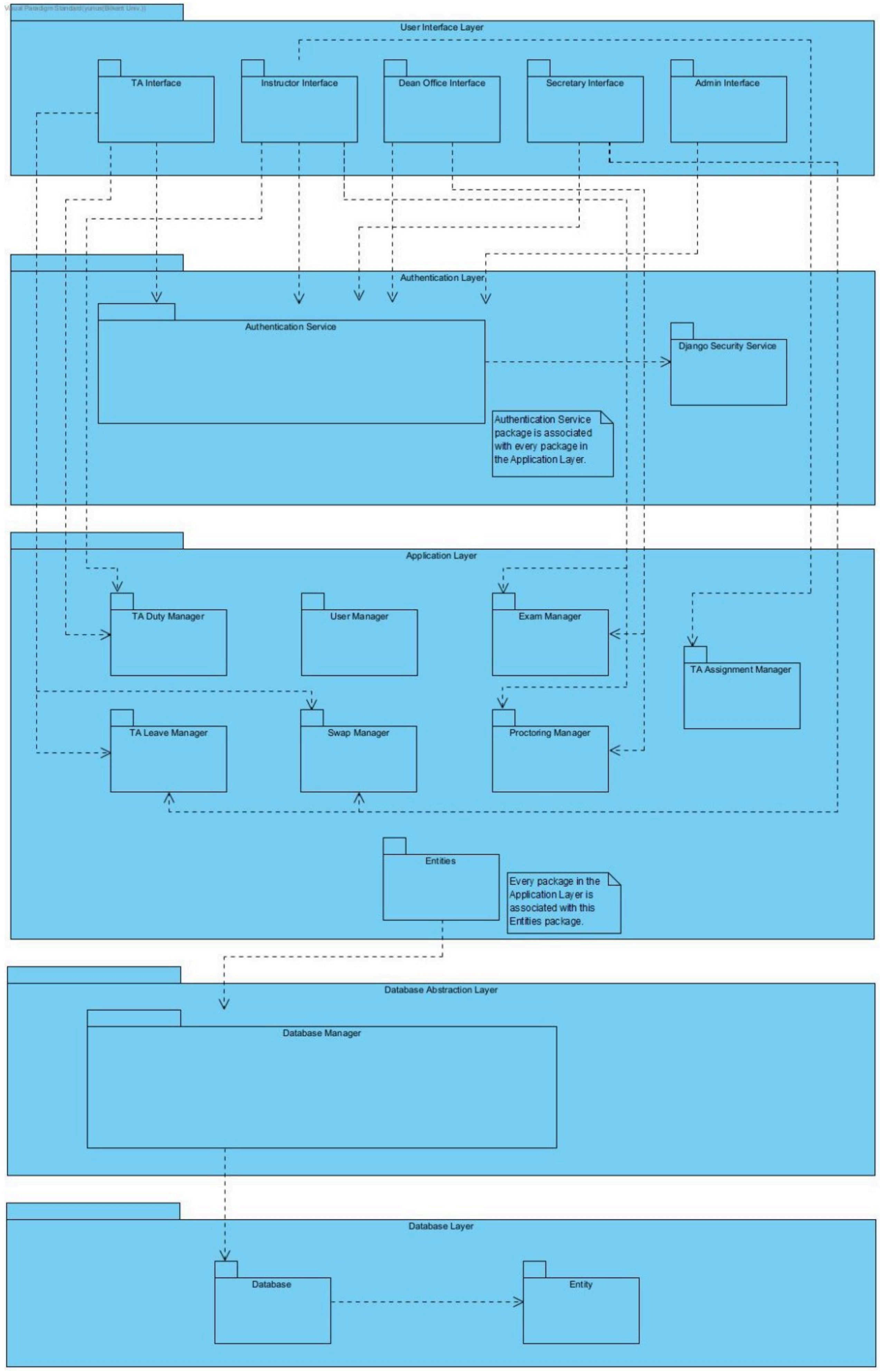
   a. **Usability vs. Reliability :** A classic trade-off exists between reliability and usability. For example, the backend enforces strict checks to ensure must-have TAs are included in an assignment. While this prevents violations of instructor preferences and ensures reliable outcomes, it can frustrate users who wish to proceed with incomplete assignments due to real-world constraints (e.g., illness or resignation). To resolve this, we implemented a “force” override mechanism with warnings. This slightly reduces system rigidity, but it allows instructors to act pragmatically. The compromise maintains system reliability (through logs and prompts) while improving usability.

   b. **Reliability vs. Performance:** To ensure accurate and up-to-date allocation data, we re-fetch and recalculate TA loads and preferences upon every page refresh or assignment change. So even when reports are being generated, they fetch data in real-time on every click. While this improves reliability by preventing stale views, it can introduce slight delays or increase server load during peak times. Instructors accessing many courses simultaneously may experience slower response times due to the backend’s thorough validation routines.

Caching or partial fetches could improve performance but at the cost of up-to-date accuracy, hence we prioritize reliability.

c. **Usability vs. Security:** Usability and security often exist in a state of tension, and this is particularly true in multi-role educational systems like our TA Management platform. To maximize usability, we aim to minimize friction – avoiding unnecessary login prompts, reducing the number of confirmation dialogs, and allowing features like auto-saving or pre-filled forms. For instance, instructors are able to assign TAs within 5 clicks (from dashboard), and TAs can submit requests without navigating deep forms. However, prioritizing usability can open up potential vulnerabilities. A highly usable system might allow TAs to remain logged in for extended sessions or expose user-related data (like workload stats or assignments) to roles that don't strictly require it. To address this, we implement role-based access control (RBAC), session timeout limits, and conditional visibility at the UI level – but these protections sometimes result in extra clicks, access denial popups, or context switches, which slightly hurt the seamless experience.

## 3. Subsystem Decomposition Diagram


Visual Paradigm Standard(yunus(Bilkent Univ.))
User Interface Layer
TA Interface
Instructor Interface
Dean Office Interface
Secretary Interface
Admin Interface
Authentication Layer
Authentication Service
Django Security Service
Authentication Service package is associated with every package in the Application Layer.
Application Layer
TA Duty Manager
User Manager
Exam Manager
TA Assignment Manager
TA Leave Manager
Swap Manager
Proctoring Manager
Entities
Every package in the Application Layer is associated with this Entities package.
Database Abstraction Layer
Database Manager
Database Layer
Database
Entity