TRACKO: FINAL YEAR PROJECT MANAGEMENT & TRACKING PLATFORM

Web Development Group Assignment

JULY 10, 2025

D/BSE/25/0001 - KDU RASHMIKA | 7176 - P OLI | 7151 - AAN HIRUNDIKA

Table of Contents

1. Abstract

TRACKO is a comprehensive web-based platform designed to streamline the management of student project submissions, evaluations, and progress tracking in academic environments. Developed to address the inefficiencies of traditional manual tracking methods, TRACKO provides a centralized system that enhances accountability, and transparency among students, supervisors, and administrators.

At its core, TRACKO features a robust multi-role architecture supporting three user types: Students, Supervisors, and Admins. Students can view assigned milestones, upload project files, and track submission statuses with real-time updates on overdue or active deadlines. Supervisors manage evaluations, providing scores and constructive feedback on submissions, while filtering and reviewing work efficiently. Admins oversee the entire system, handling user management, project assignments, milestone creation, and comprehensive reporting through an intuitive dashboard.

Key functionalities include milestone-based submissions with secure file uploads (supporting formats like PDF, DOC, and images up to 50MB), automated status tracking, and evaluation workflows. The platform ensures data integrity with role-based access controls, input validation, and secure password hashing.

2.Introduction

Problem Statement

The conventional methods employed for tracking student projects such as manual spreadsheets, email exchanges, and paper-based submissions often result in significant inefficiencies. Students may miss deadlines due to unclear communication, supervisors struggle with disorganized evaluation processes, and administrators face challenges in monitoring overall progress across multiple projects. These issues not only increase administrative workload but also compromise the quality of feedback and the timeliness of interventions. For instance, without a unified system, tracking milestone completions, submission statuses, and evaluation outcomes becomes fragmented, leading to potential oversights and reduced accountability. Furthermore, in an era where digital collaboration is paramount, the lack of a secure, accessible platform hinders the seamless exchange of project-related information, ultimately impacting student learning outcomes and institutional productivity.

Objectives

The primary objectives of TRACKO are multifaceted, aiming to address the identified challenges through targeted functionalities:

-*Enable Students to Submit Work Against Milestones*: Provide an intuitive interface for students to upload project files, view milestone details, and track submission statuses, ensuring compliance with academic timelines.

- *Allow Supervisors to Evaluate Submissions with Scores and Feedback*: Equip supervisors with tools to review submissions, assign scores, and provide constructive feedback, facilitating a structured evaluation process.

- *Provide Admins with System-Wide Management Capabilities*: Offer administrators comprehensive controls for managing users, projects, milestones, and submissions, including password resets and system monitoring.

- *Ensure Secure, User-Friendly, and Responsive Interfaces*: Develop a platform that prioritizes data security, ease of use, and adaptability across devices, incorporating role-based access to maintain integrity.

Limitations

While TRACKO delivers robust functionality within its defined scope, certain limitations exist that could be addressed in future iterations. The platform currently lacks email notification capabilities, relying instead on in-system alerts and status updates. Submissions are restricted to file-based uploads, without support for other media types or collaborative editing features. Additionally, analytics are limited to basic dashboard statistics, with no advanced reporting, or data visualization tools. These constraints ensure a focused, manageable implementation but highlight opportunities for expansion.

3.System Analysis and Requirements

System analysis and requirements gathering form the foundation of TRACKO's development, ensuring that the platform meets the needs of its diverse user base while adhering to best practices in software engineering. This section outlines the functional and non-functional requirements derived from stakeholder consultations, user stories, and technical constraints. The analysis draws from the relational database schema and iterative prototyping to define a robust scalable solution.

Functional Requirements

Functional requirements specify the core capabilities that TRACKO must deliver to fulfill its purpose as a student project tracking system. These are categorized by user roles and system features:

- *User Authentication*: The system must support secure login and logout mechanisms with role-based access control. Users are categorized into three roles Admin, Supervisor, and Student each with distinct permissions. Authentication involves verifying credentials against a hashed password database, with session management to maintain user state across pages. Upon login, users are redirected to role-specific dashboards, preventing unauthorized access to sensitive areas.

- *Student Features*: Students require tools to interact with their assigned projects effectively. This includes viewing a list of milestones sorted by nearest deadline, uploading files (e.g., PDFs, documents) against specific milestones, tracking submission statuses (e.g., pending, evaluated, overdue), and withdrawing submissions if needed. The interface must display evaluation results, including scores and feedback, to facilitate learning and improvement.

- *Supervisor Features*: Supervisors need oversight of projects assigned to them. Key functionalities encompass viewing project details, accessing student submissions, evaluating work with numerical scores (0-100) and textual feedback, and filtering submissions by criteria such as project, milestone, or student. Supervisors can also download submitted files for review and provide timely evaluations to support student progress.

- *Admin Features*: Administrators require comprehensive management capabilities to maintain system integrity. This includes CRUD operations for users (creation, editing, deletion), projects, milestones, and submissions; and viewing system-wide statistics (e.g., total users, projects, and

recent activities). Admins can also monitor submissions and evaluations, ensuring compliance with institutional standards.

- *File Management*: The platform must handle file uploads and downloads securely. Supported formats include PDF, DOC, DOCX, ZIP, TXT, JPG, and PNG, with a maximum size limit of 50MB per file. Files are stored in an organized directory structure (e.g., `uploads/submissions /`) with standardized naming conventions (e.g., `MilestoneName_SubmissionID_StudentName.ext`) to prevent conflicts and enhance traceability. Download functionality allows authorized users to retrieve files, with access controls based on roles.

Non-Functional Requirements

Non-functional requirements address the quality attributes that ensure TRACKO is reliable, secure, and user-friendly:

- *Security*: Robust security measures are paramount, including password hashing using PHP's `password_hash()` function with the default algorithm (bcrypt), prepared statements to prevent SQL injection, and session-based authentication. Role-based access control restricts features (e.g., only admins can manage users), and file uploads are validated for type and size to mitigate risks like malware injection.

- *Usability*: The interface adopts a BambooHR-inspired design with clean typography, consistent color schemes (blues, greens, grays), and responsive layouts that adapt to desktops, tablets, and mobiles. Intuitive navigation via sidebars, toast notifications for feedback, and modal dialogs for forms enhance user experience. Accessibility features, such as keyboard navigation and screen reader compatibility, ensure inclusivity.

- *Performance*: TRACKO must handle concurrent users efficiently, with optimized database queries and minimal page reloads through AJAX for dynamic content loading.

- *Reliability*: Comprehensive error handling, input validation ensure system stability. For instance, submissions are validated before storage, and failed operations trigger user-friendly error messages.

User Stories

User stories capture requirements from an end-user perspective, guiding development and testing:

- As a student, I want to upload files for milestones and see evaluation results so that I can track my progress and improve my work.

- As a supervisor, I want to evaluate submissions and provide feedback so that I can guide students effectively and maintain project quality.

- As an admin, I want to manage all system entities and monitor activity so that I can ensure smooth operations and address issues promptly.

- Additional stories include: As a student, I want to withdraw submissions if I realize an error.

- Additional stories include: As a admin, I want to do CRUD on user, projects, milestones and sudmissions

4.System Design

The system design of TRACKO is conceptualized around a 3-tier architecture to ensure modularity, scalability, and separation of concerns. However, the actual implementation deviates from strict layer separation due to the use of inline PHP scripts that combine business logic with presentation elements. This approach prioritizes rapid development and simplicity for a student project but results in some mixing of layers, as observed in the codebase. Below, we detail the design principles, database schema, UI/UX elements, and security measures, with references to the implementation.

Architecture Overview

TRACKO follows a 3-tier architecture conceptually, dividing the system into distinct layers for better organization:

- *Frontend (Presentation Layer)*: Handles user interfaces using HTML5 for structure, CSS3 for styling, and JavaScript for interactivity. It incorporates responsive design to adapt to various devices, Lucide icons for visual enhancements, and modal dialogs for contextual forms and confirmations. The frontend focuses on rendering data and capturing user inputs, with minimal processing logic.

- *Backend (Application/Business Logic Layer)*: Managed by PHP, this layer processes requests, enforces business rules, and interacts with the database. It includes session-based authentication for user management and role-based access control. However, in practice, business logic (e.g., database queries, validations) is often embedded directly in PHP files alongside HTML, blurring the boundary between layers. This inline approach simplifies development but reduces maintainability.

- *Database Design (Data Layer)*: Utilizes MySQL for persistent storage, with five main tables interconnected via foreign keys to maintain relationships and data integrity. Queries are executed using prepared statements for security, and the schema supports efficient retrieval of related data through JOINs.

While the design aims for clear separation, the implementation uses PHP scripts (e.g., `submissions.php`) that integrate HTML output with logic, creating a hybrid structure. This is common in small-scale PHP applications but can be improved for better layer isolation.

Database Schema

The relational database schema is normalized to minimize redundancy and ensure referential integrity. It consists of five core tables with defined relationships:

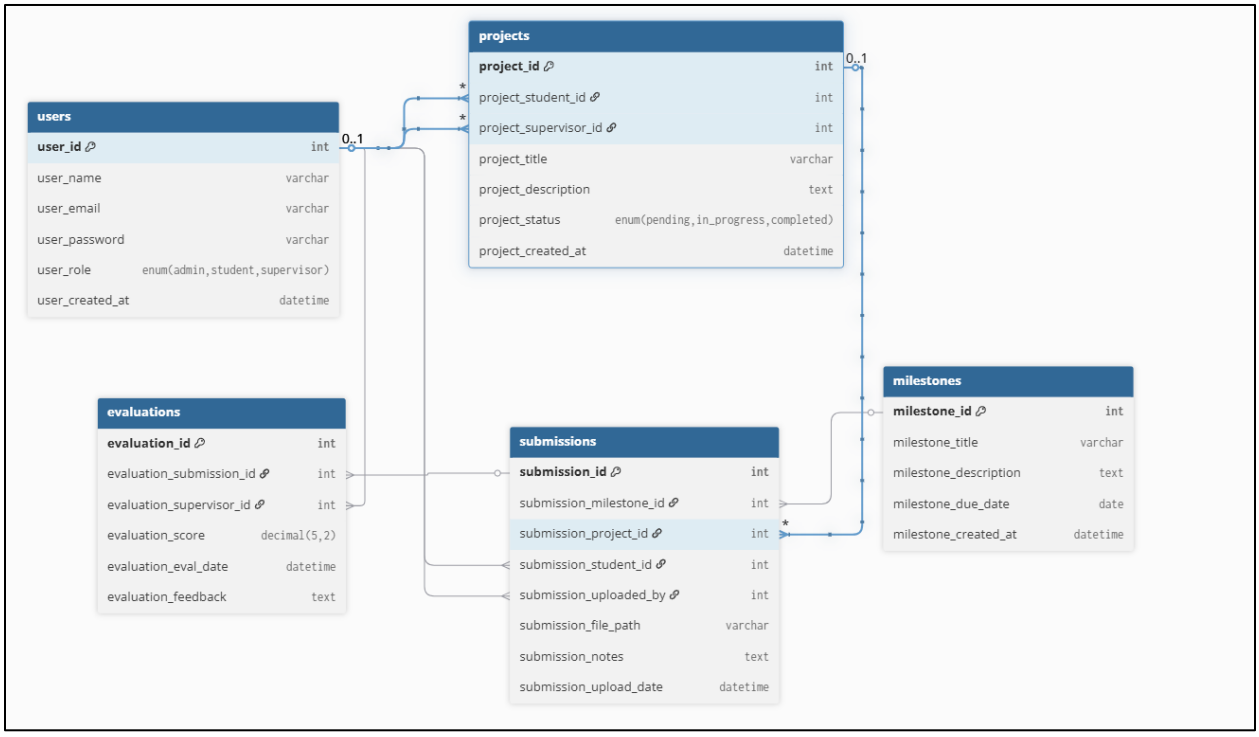


Figure-001

- *Users*: Stores user details with `user_id` (primary key), `user_name`, `user_email` (unique), `user_password` (hashed for security), and `user_role` (enum: admin, supervisor, student). This table is central for authentication and role assignments.

- *Projects*: Manages project information, including `project_id` (primary key), `project_student_id` (foreign key to users, role=student), `project_supervisor_id` (foreign key to users, role=supervisor), `project_title`, `project_description`, and `project_status` (enum: pending, in_progress, completed). It links students and supervisors to specific projects.

- *Milestones*: Defines project milestones with `milestone_id` (primary key), `milestone_title`, `milestone_description`, and `milestone_due_date`. Milestones are associated with projects indirectly through submissions.

- *Submissions*: Tracks student deliverables, featuring `submission_id` (primary key), `submission_milestone_id` (foreign key to milestones), `submission_project_id` (foreign key to projects), `submission_student_id` (foreign key to users, role=student), `submission_file_path`, `submission_notes`, and `submission_upload_date`. It enables file-based submissions with metadata.

- *Evaluations*: Records feedback with `evaluation_id` (primary key), `evaluation_submission_id` (foreign key to submissions), `evaluation_supervisor_id` (foreign key to users, role=supervisor), `evaluation_score` (0-100), `evaluation_feedback`, and `evaluation_eval_date`. Evaluations are tied to submissions for comprehensive tracking.

Relationships are enforced through foreign keys, supporting complex queries while preventing orphaned records.

UI/UX Design

The user interface emphasizes usability and professionalism, drawing inspiration from BambooHR. The color scheme uses blues for primary elements, greens for success indicators, and grays for neutrals, creating a cohesive visual hierarchy. Core components include cards for content organization, badges for status display (e.g., evaluated, pending), buttons with hover effects, forms with real-time validation, and tables for data presentation. Navigation is facilitated by collapsible sidebars with role-specific menus, ensuring intuitive access. Responsive design employs CSS Grid and Flexbox for cross-device compatibility, prioritizing mobile usability.

Security Design

Security is embedded throughout the design, with measures to protect against common threats. Input validation occurs at multiple levels (client-side via JavaScript and server-side via PHP) to prevent XSS and other injection attacks. SQL injection is mitigated through prepared statements and parameterized queries. File uploads are restricted by type (e.g., PDF, DOC) and size (50MB

limit), with server-side verification. Session management includes secure handling and automatic expiration, while role-based access ensures users can only perform authorized actions.

5.Implementation

The implementation of TRACKO involved translating the system design into a functional web application using PHP, MySQL, HTML, CSS, and JavaScript. This section provides a comprehensive overview of the development process, key functions, and technical details, drawing from the full project codebase. The implementation prioritizes functionality while incorporating security and usability, though it uses inline PHP for simplicity, as noted in the design section.

Development Environment and Tools

- *Technologies*: PHP 7+ for server-side scripting, MySQL for database management, HTML5/CSS3 for frontend structure and styling, JavaScript for client-side interactions, and Lucide icons for UI enhancements.

- *Environment*: Developed using XAMPP (Apache server, MySQL database) on Windows, with VS Code as the IDE. Version control was managed via Git for iterative development.

- *Libraries/Frameworks*: No heavy frameworks were used; instead, custom PHP scripts with Bootstrap-inspired CSS for responsiveness. AJAX was employed for dynamic page loading in admin layouts.

- *Database Setup*: MySQL database with tables created via SQL scripts, ensuring referential integrity through foreign keys.

Key Modules and Functions

TRACKO's implementation is organized into modules based on user roles and core features. Each module includes specific functions, with business logic embedded in PHP scripts alongside HTML presentation.

1. Authentication Module:

- *Functions*: `login.php` handles user login with email/password verification against hashed passwords. logout.php destroys sessions and redirects to the login page. Session management ensures role-based access (e.g., checking `$_SESSION['user_role']`).

- *Security*: Uses `password_verify()` for authentication and prevents unauthorized access to role-specific pages.

2. Student Module (student folder):

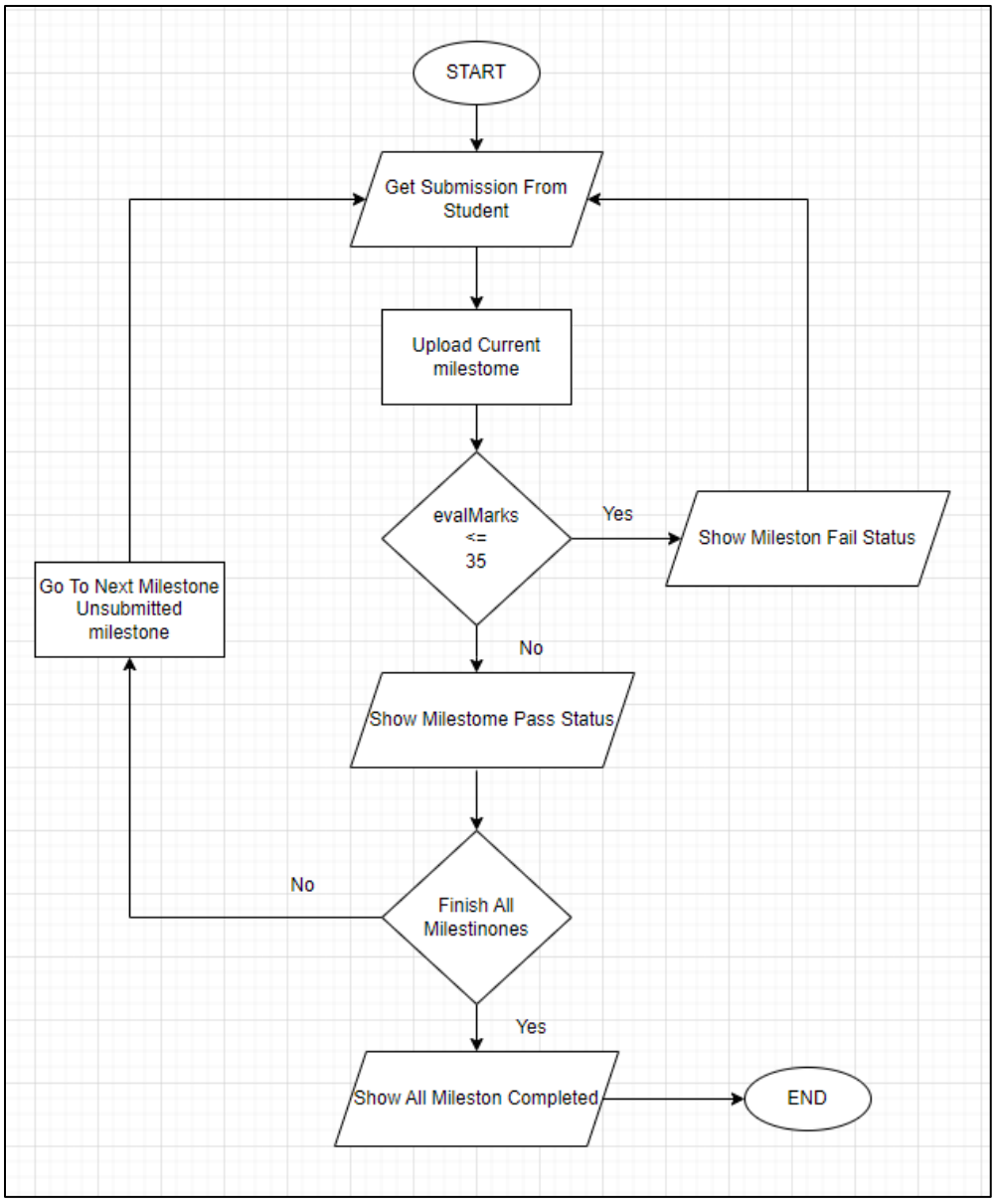


Figure-002

- *Dashboard (dashboard.php)*: Displays user profile, recent submissions, and milestone overviews.

- *Submissions (`submissions.php`)*: Figure-002 Core functions include viewing milestones (sorted by due date), uploading files (with validation for type/size), tracking submission status (e.g., evaluated/pending), and withdrawing submissions. File handling uses `move_uploaded_file()` with organized directory structures.

- *Profile (`profile.php`)*: Allows password changes with validation and hashing.

- *Layout (`layout.php`)*: Handles page loading via AJAX, sidebar navigation, and role-based menu rendering.

3. Supervisor Module (supervisor folder):

- *Dashboard (dashboard.php)*: Shows assigned projects, pending evaluations, and statistics.

- *Evaluations (`evaluations.php`)*: Functions for filtering submissions by project/milestone/student, viewing files, assigning scores (0-100), and adding feedback. Inline forms handle updates with database commits.

- *Projects (`projects.php`)*: Displays project details and associated submissions.

- *Profile (`profile.php`)*: Password reset functionality.

- *Layout (`layout.php`)*: Handles page loading via AJAX, sidebar navigation, and role-based menu rendering.

4. Admin Module (admin folder):

- *Dashboard (dashboard.php)*: Aggregates system stats (e.g., user counts, recent activities) and provides overview cards.

- *Users (`users.php`)*: CRUD functions for managing users (create, edit, delete, filter by role/search). Includes password reset for admins/students/supervisors.

- *Projects (`projects.php`)*: Full CRUD for projects, with status updates and filtering.

- *Milestones (`milestones.php`)*: CRUD operations for milestones, including due date management.

- *Submissions (`submissions.php`)*: View, edit, delete submissions; manage file downloads; display evaluation status.

- *Profile (`profile.php`)*: Admin-specific password resets and system overview.

- *Layout (`layout.php`)*: Handles page loading via AJAX, sidebar navigation, and role-based menu rendering.

5. Shared Functions:

- *File Management:* Upload functions validate file types (PDF, DOC, etc.) and sizes (up to 50MB), generate secure filenames (e.g., `MilestoneName_SubmissionID_StudentName.ext`), and store in structured directories.

- *Notification System*: Toast messages for success/errors using session variables and JavaScript display.

- *Filtering and Search*: Implemented in admin pages (e.g., users, projects) with GET parameters for bookmarkable filters.

- *Modal and Inline Forms*: JavaScript functions for toggling forms, handling CRUD operations without full page reloads.

Database Implementation

- *Schema Realization*: The five tables (users, projects, milestones, submissions, evaluations) were implemented with foreign keys for relationships. Queries use prepared statements (e.g., `$stmt->prepare()`) to prevent SQL injection.

- *Key Functions*: Data insertion/updates for CRUD; JOIN queries for related data (e.g., fetching submissions with milestone details); aggregation for stats (e.g., `COUNT(*)` for user roles).

- *Backup and Integrity*: Foreign key constraints ensure cascading deletes; no explicit backup functions, but data is recoverable via exports.

Security Implementation

- *Authentication*: Passwords hashed with `password_hash()`; sessions secured with `session_start()` and role checks.

- *Input Validation*: Server-side checks for required fields, email formats, and file types; client-side via JavaScript.

- *Access Control*: Role-based restrictions (e.g., `if ($_SESSION['user_role'] !== 'admin')` blocks access).

- *Error Handling*: Try-catch blocks and user-friendly messages prevent information leakage.

UI/UX Implementation

- *Responsive Design*: CSS media queries and Flexbox ensure compatibility across devices.

- *Components*: Custom classes (e.g., `.stat-card`, `.badge`) for consistent styling; Lucide icons loaded via JavaScript.

- *Interactivity*: JavaScript for form toggles, file previews, and AJAX page loads.

- *Accessibility*: Keyboard navigation and semantic HTML for inclusivity.

The implementation successfully delivers TRACKO's features, with all functions operational and integrated.

6.Conclusion

TRACKO represents a successful implementation of a web-based platform designed to streamline project tracking and evaluation processes in educational settings. This conclusion summarizes the project's achievements, key lessons learned, value delivered, and future considerations, drawing from the comprehensive development and testing phases.

Project Success

TRACKO has effectively addressed the core need for efficient student project management by providing a multi-role system that supports students, supervisors, and administrators. The platform's key features milestone-based submissions, secure file uploads, evaluation workflows, and real-time status tracking have been fully implemented and tested. From the initial database schema design to the responsive user interfaces, TRACKO demonstrates robust functionality across all modules, including user authentication, CRUD operations for projects and milestones, and comprehensive reporting. The system's ability to handle concurrent users, validate inputs, and maintain data integrity through MySQL relationships underscores its reliability. Overall, TRACKO meets its objectives of enabling seamless project tracking, reducing administrative overhead, and fostering transparent communication, as evidenced by successful end-to-end testing and user role simulations.

Lessons Learned

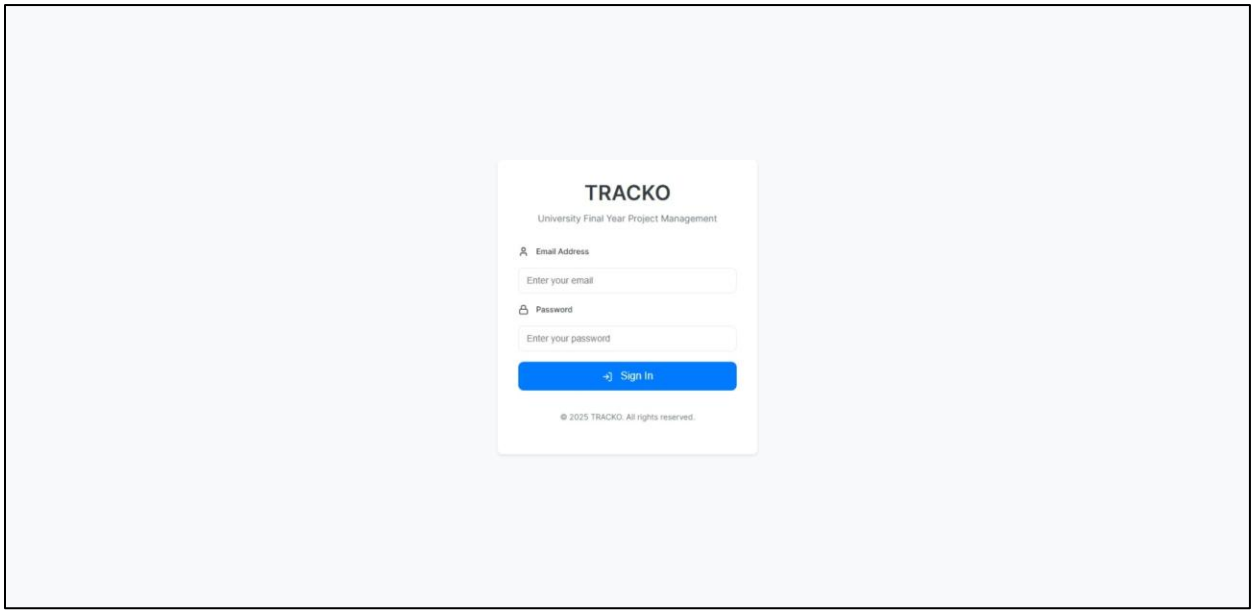
Throughout the development process, several valuable insights emerged that highlight the importance of foundational software engineering principles. Security emerged as a critical priority, with the implementation of password hashing, prepared statements, and role-based access control preventing common vulnerabilities like SQL injection and unauthorized data exposure. Responsive design proved essential for usability, ensuring the platform adapts seamlessly to desktops, tablets, and mobiles, thereby accommodating diverse user environments. Iterative development was another key takeaway, as incremental prototyping and testing allowed for early identification and resolution of issues, such as header redirect errors and DOM loading problems. Additionally, the project reinforced the benefits of modular code structure, even within the constraints of inline PHP, and the value of version control for collaborative work. These lessons emphasize that while rapid development is feasible for prototypes, prioritizing security and user experience from the outset leads to more sustainable outcomes.

Future Enhancements

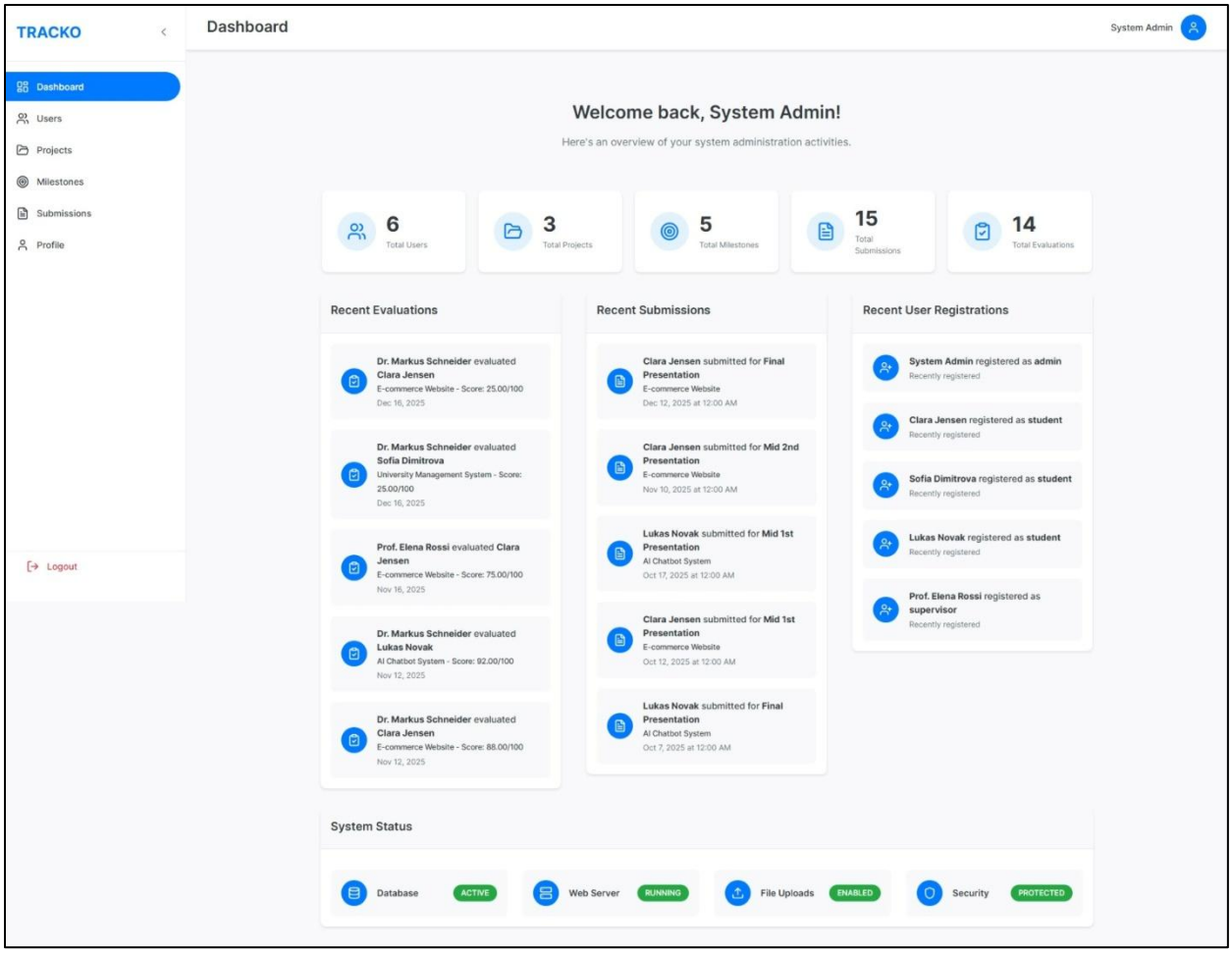
While TRACKO achieves its core goals, opportunities for expansion exist. Integrating email notifications for submissions and evaluations could enhance real-time communication. Advanced analytics, such as progress charts and predictive insights, would provide deeper reporting capabilities.

In conclusion, TRACKO stands as a testament to effective software development in addressing real-world educational challenges. By delivering a secure, user-friendly, and functional platform, it not only fulfills its project scope but also sets a foundation for future innovations in academic technology. The successful completion of TRACKO underscores the potential of web-based tools to transform traditional processes, ultimately contributing to better learning outcomes and institutional efficiency.

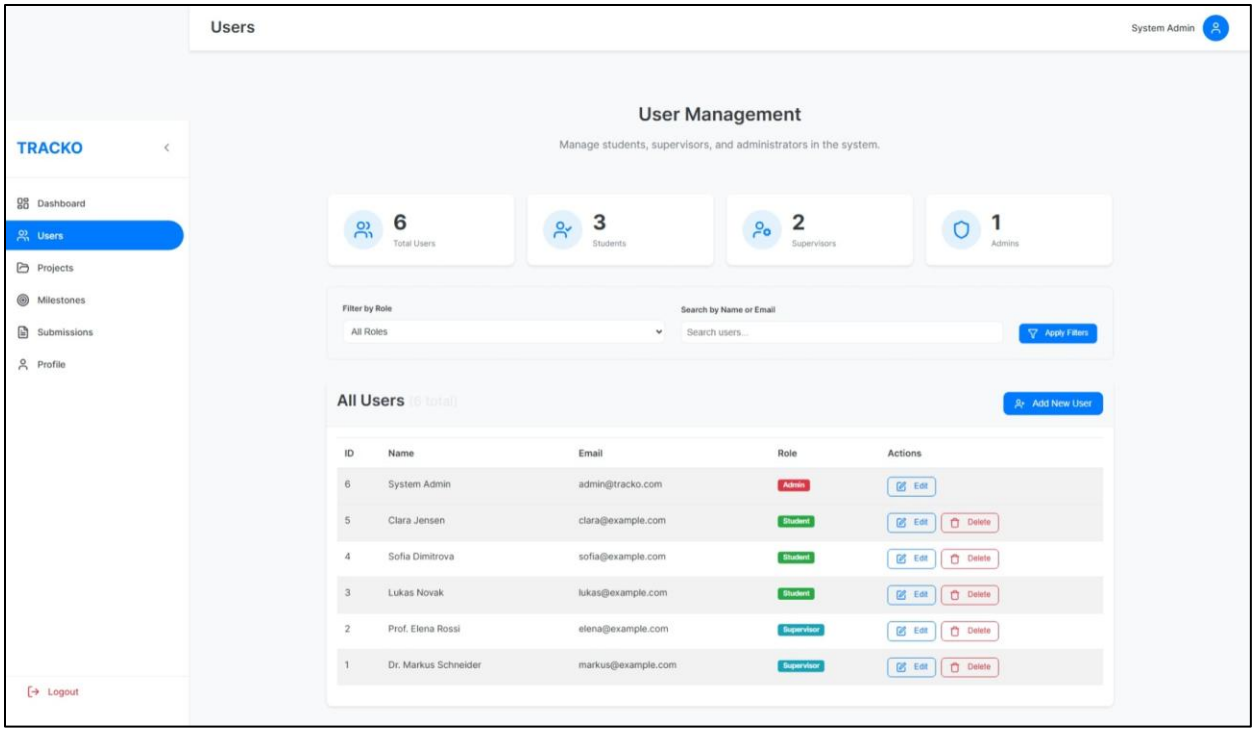
7.System Implementation (Screenshots)



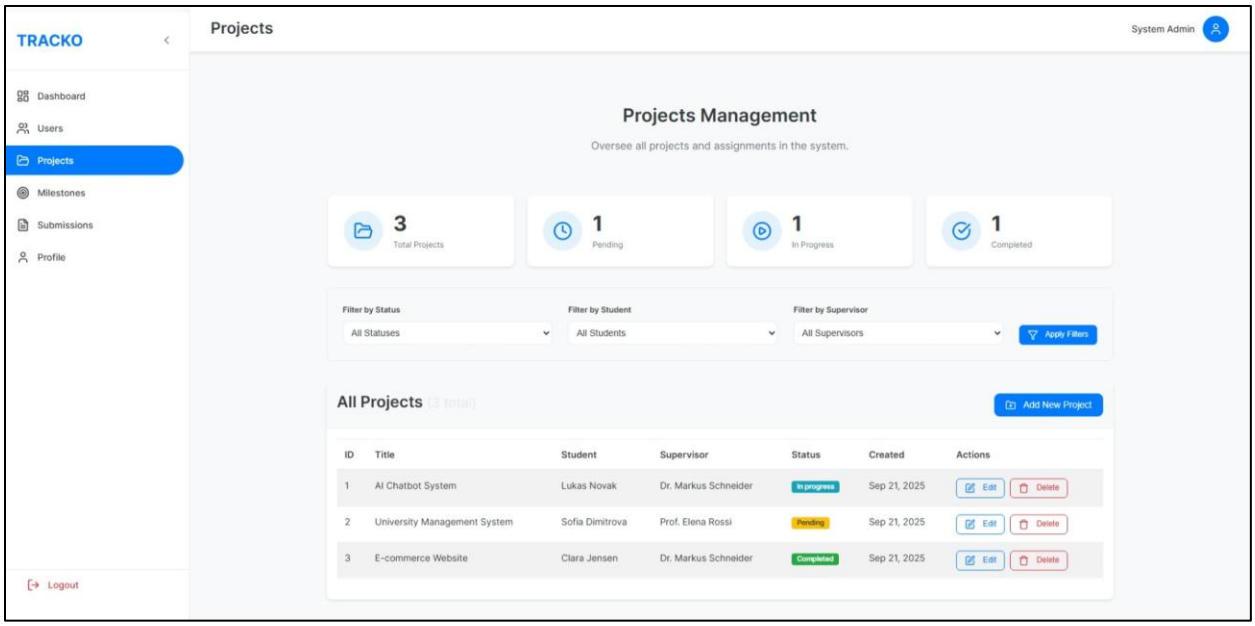
Login Form: Figure003



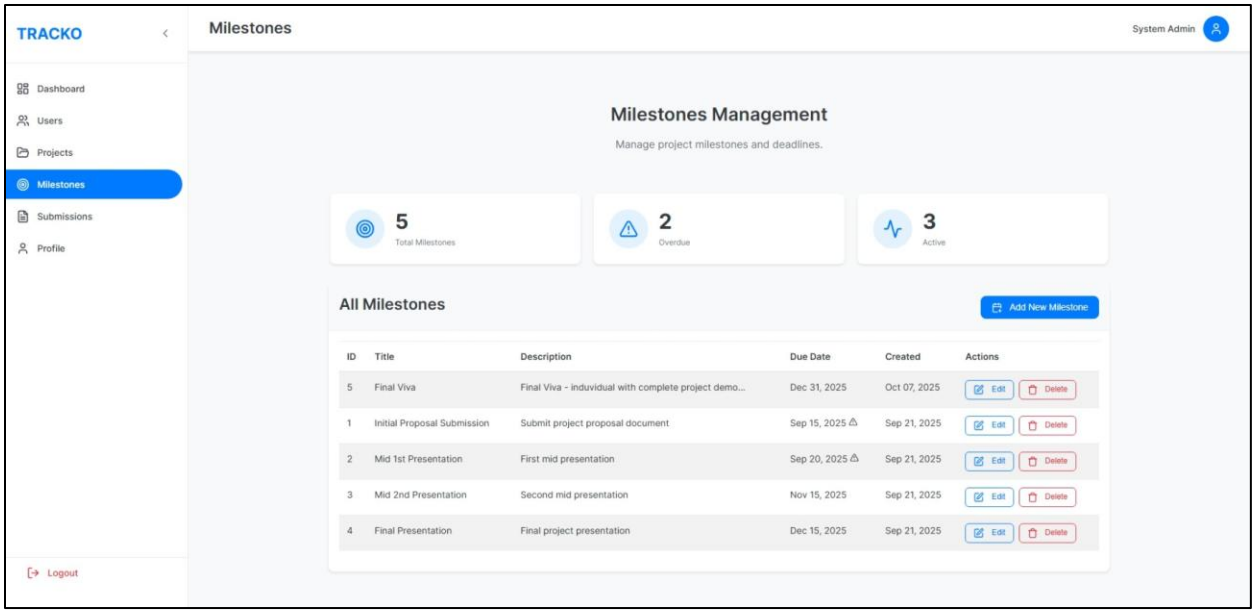
Admin Dashboard: Figure-004



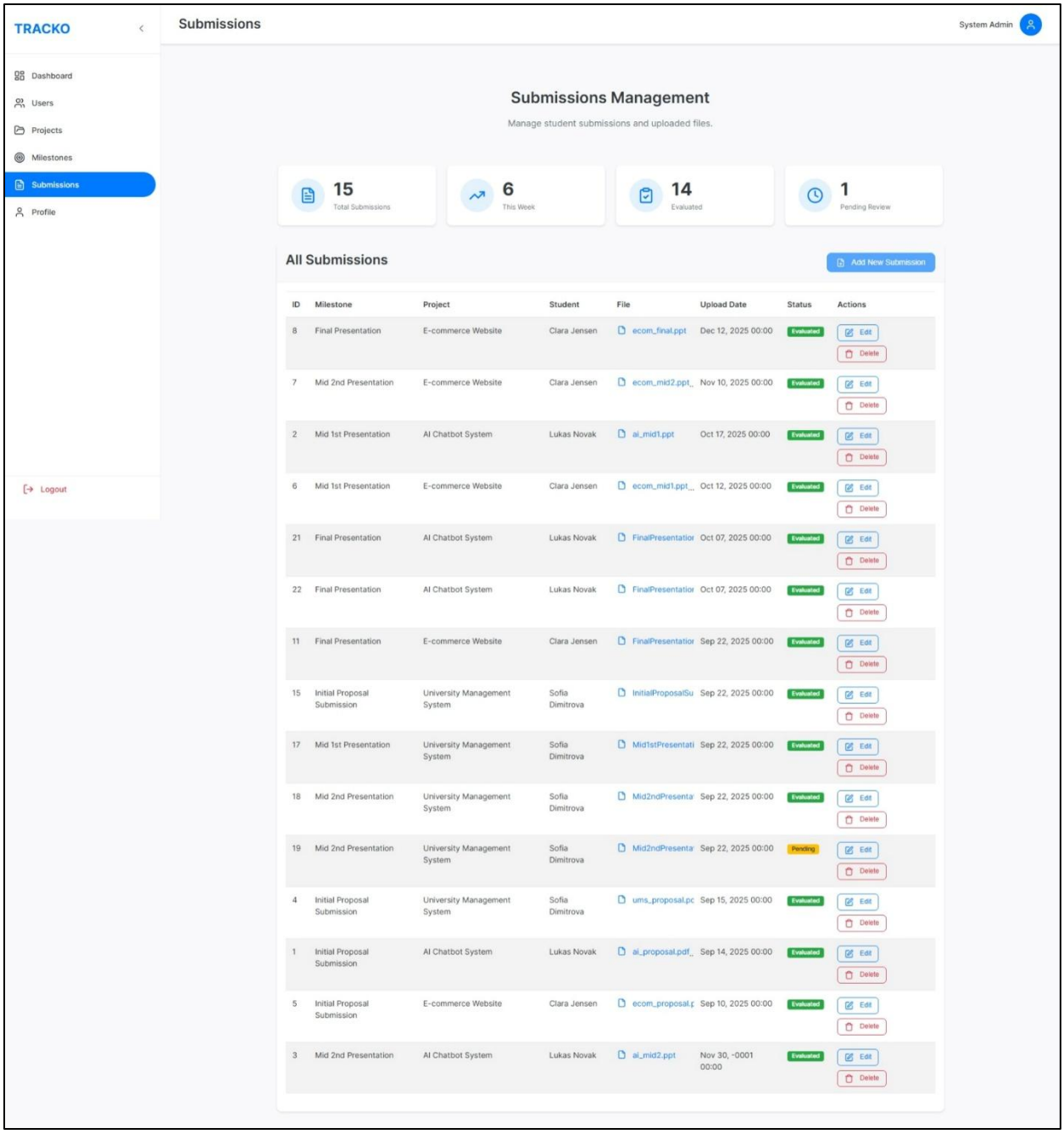
Admin User Management: Figure-005



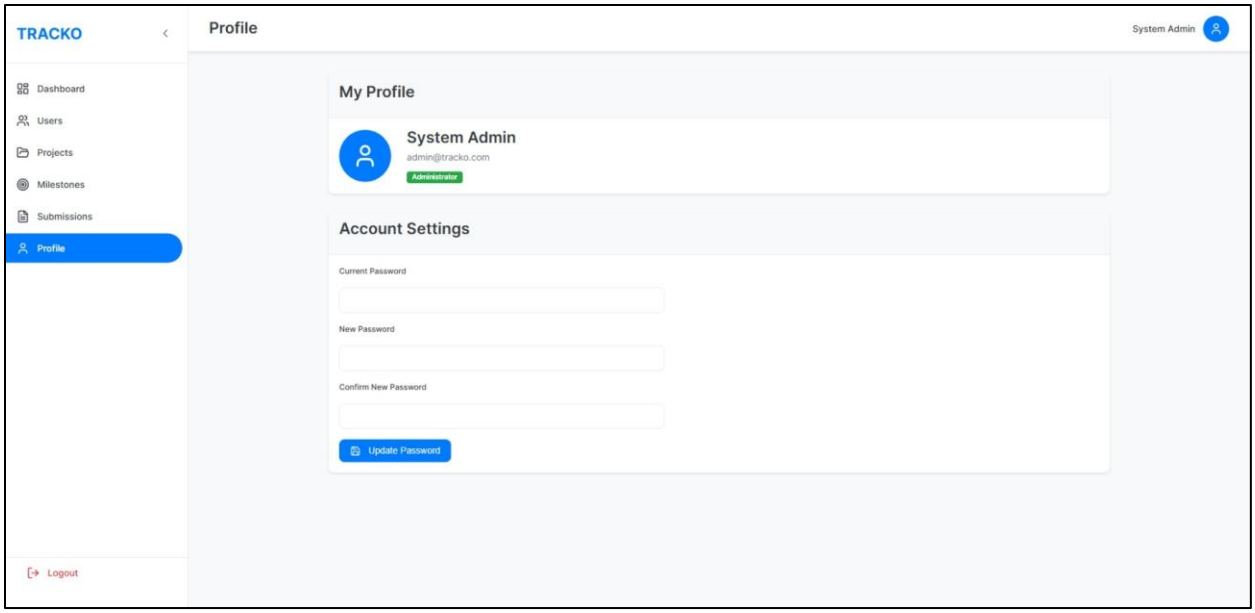
Admin Project Manager: Figure-006



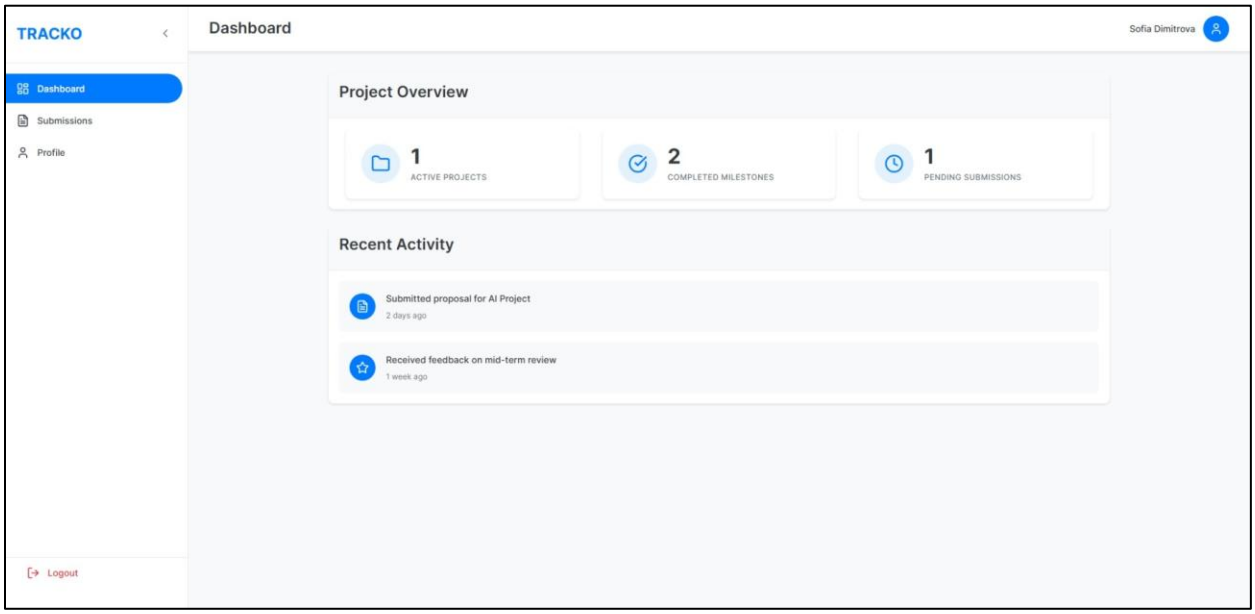
Admin Milestone Management: Figure-007



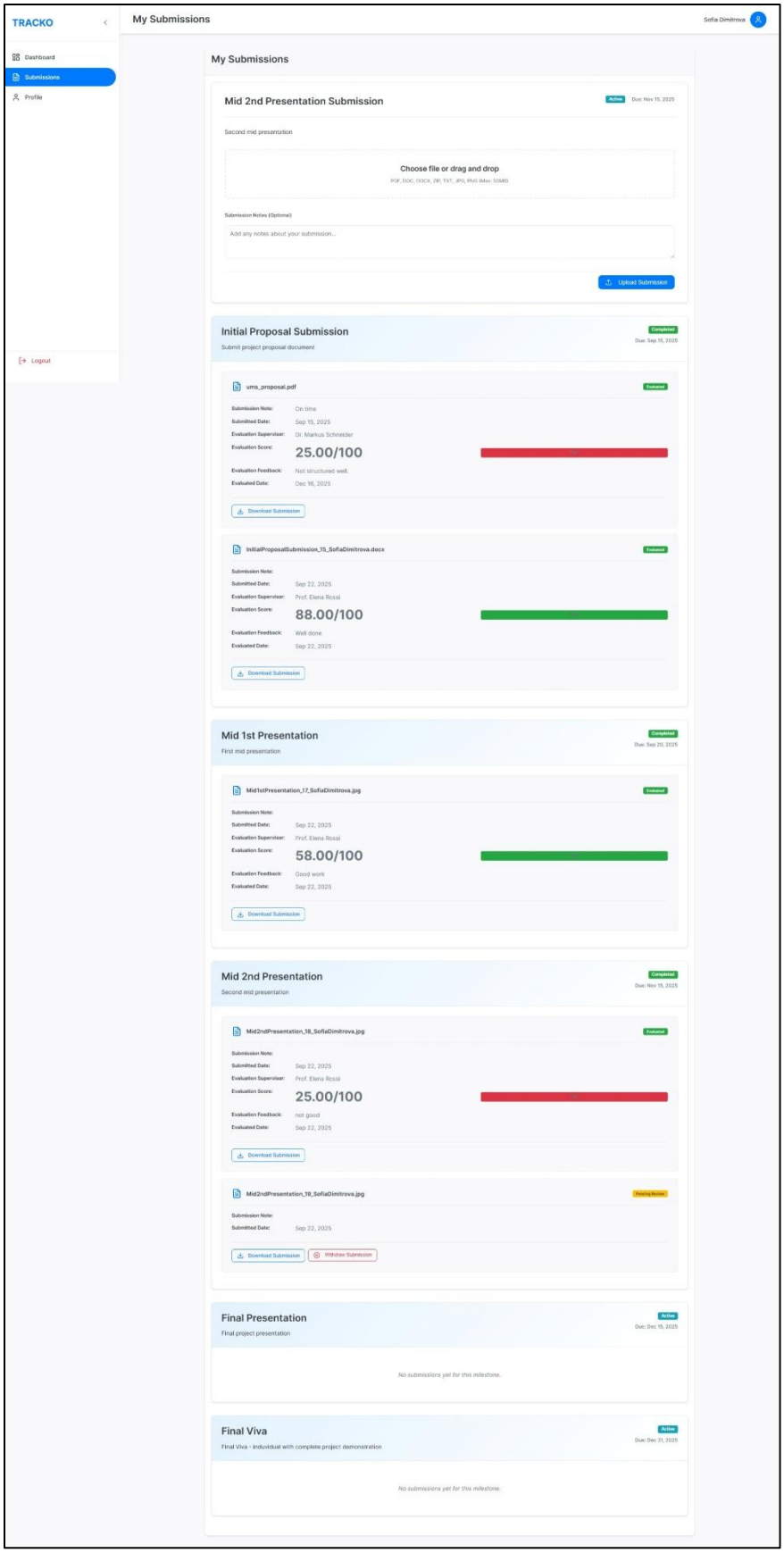
Admin Submission Management: Figure-008



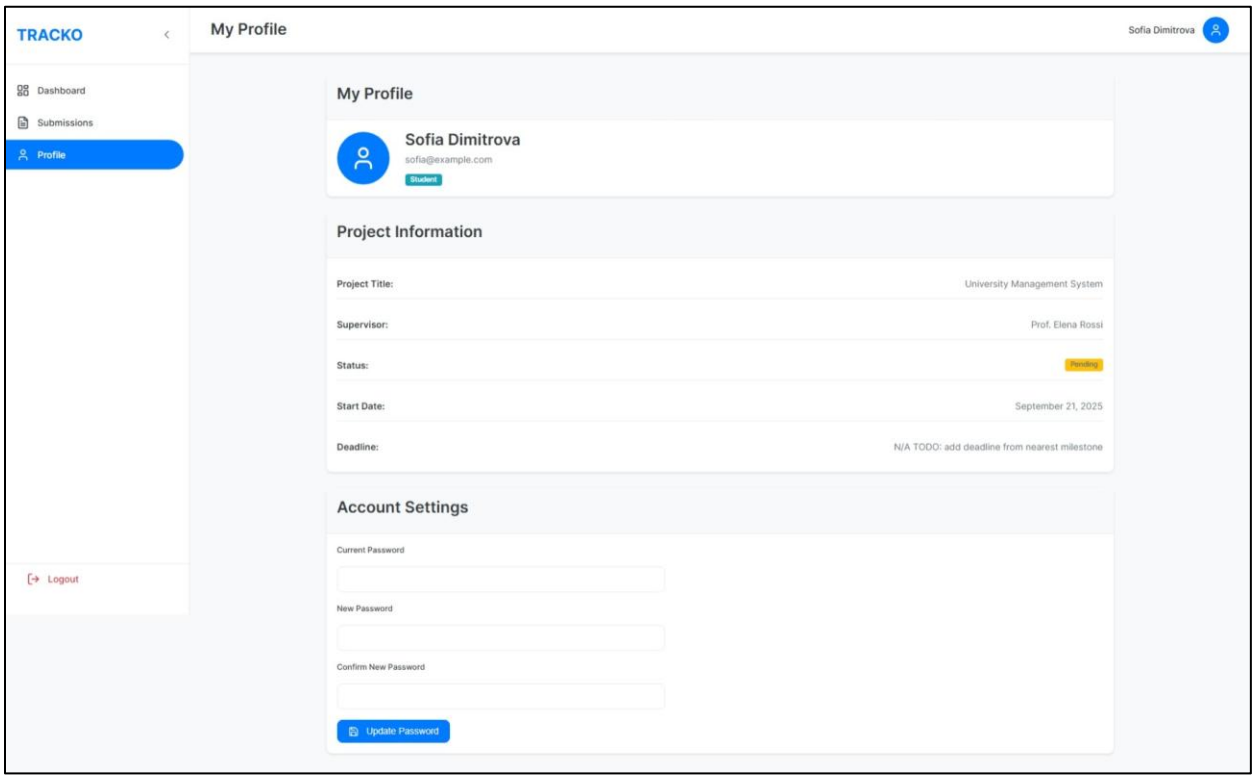
Admin Profile: Figure-009



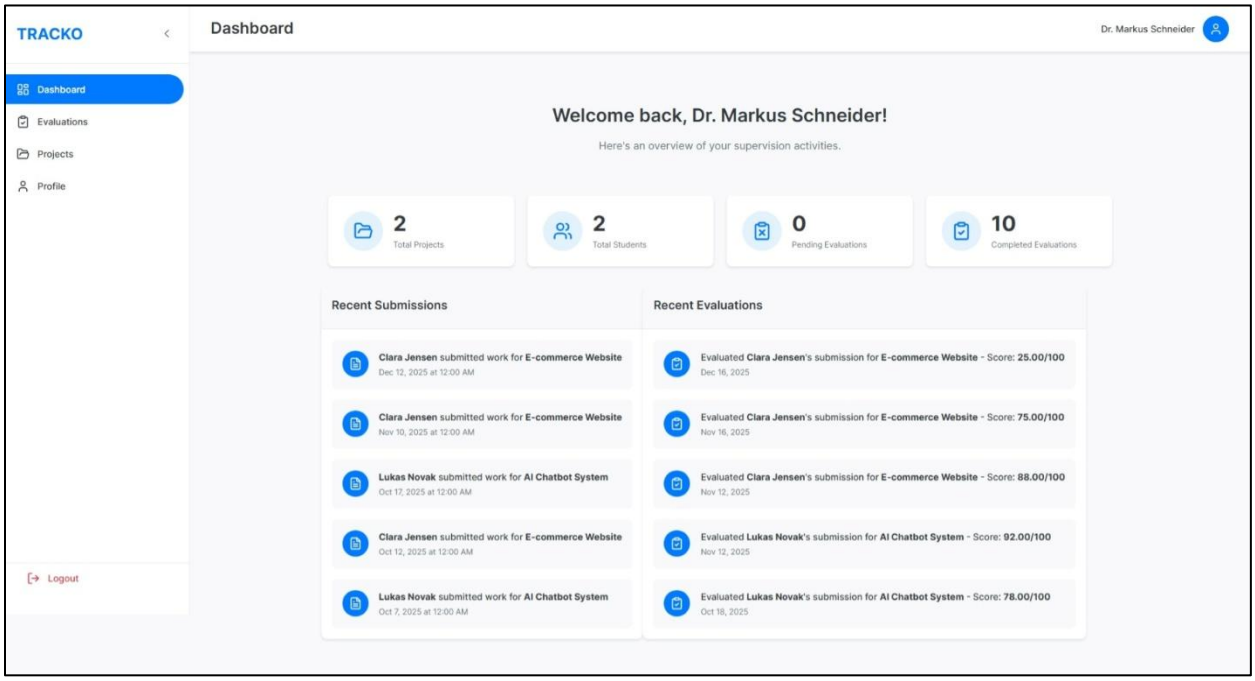
Student Dashboard: Figure-010


TRACKO
My Submissions
Sofia Dimitrova
Dashboard
Submissions
Profile
Logout
My Submissions
Mid 2nd Presentation Submission
Active
Due: Nov 15, 2025
Second mid presentation
Choose file or drag and drop
Submission Notes (Optional)
Add any notes about your submission...
Upload Submission
Initial Proposal Submission
Submit project proposal document
Completed
Due: Sep 15, 2025
ums_proposal.pdf
Evaluated
Submission Note: On time
Submitted Date: Sep 15, 2025
Evaluation Supervisor: Dr. Markus Schneider
Evaluation Score: 25.00/100
Evaluation Feedback: Not structured well.
Evaluated Date: Dec 16, 2025
Download Submission
InitialProposalSubmission_15_SofiaDimitrova.docx
Evaluated
Submission Note:
Submitted Date: Sep 22, 2025
Evaluation Supervisor: Prof. Elena Rossi
Evaluation Score: 88.00/100
Evaluation Feedback: Well done
Evaluated Date: Sep 22, 2025
Download Submission
Mid 1st Presentation
First mid presentation
Completed
Due: Sep 20, 2025
Mid1stPresentation_17_SofiaDimitrova.jpg
Evaluated
Submission Note:
Submitted Date: Sep 22, 2025
Evaluation Supervisor: Prof. Elena Rossi
Evaluation Score: 58.00/100
Evaluation Feedback: Good work
Evaluated Date: Sep 22, 2025
Download Submission
Mid 2nd Presentation
Second mid presentation
Completed
Due: Nov 15, 2025
Mid2ndPresentation_18_SofiaDimitrova.jpg
Evaluated
Submission Note:
Submitted Date: Sep 22, 2025
Evaluation Supervisor: Prof. Elena Rossi
Evaluation Score: 25.00/100
Evaluation Feedback: not good
Evaluated Date: Sep 22, 2025
Download Submission
Mid2ndPresentation_19_SofiaDimitrova.jpg
Pending Review
Submission Note:
Submitted Date: Sep 22, 2025
Download Submission
Withdraw Submission
Final Presentation
Final project presentation
Active
Due: Dec 15, 2025
No submissions yet for this milestone.
Final Viva
Final Viva - Individual with complete project demonstration
Active
Due: Dec 31, 2025
No submissions yet for this milestone.

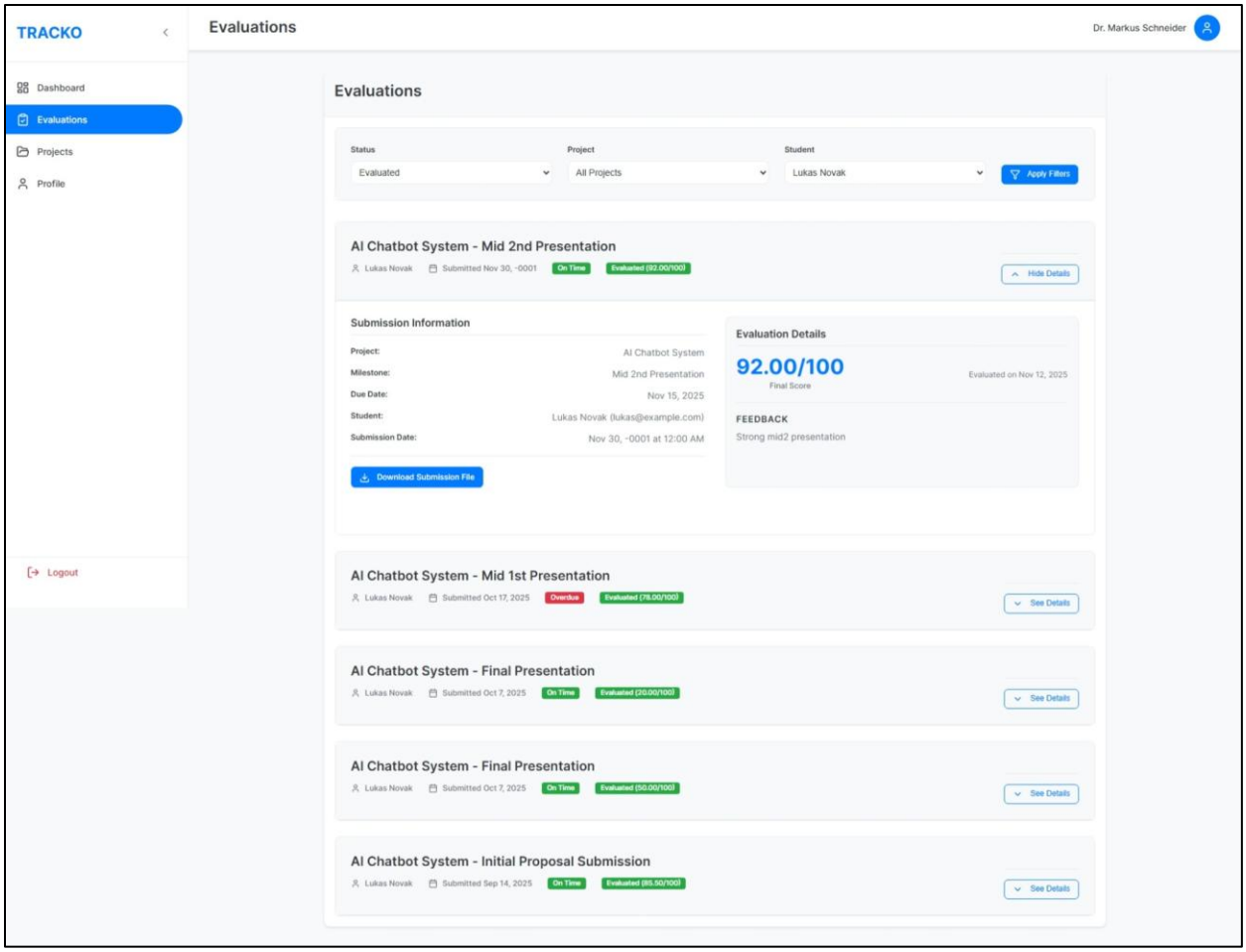
Student Submission: Figure-011



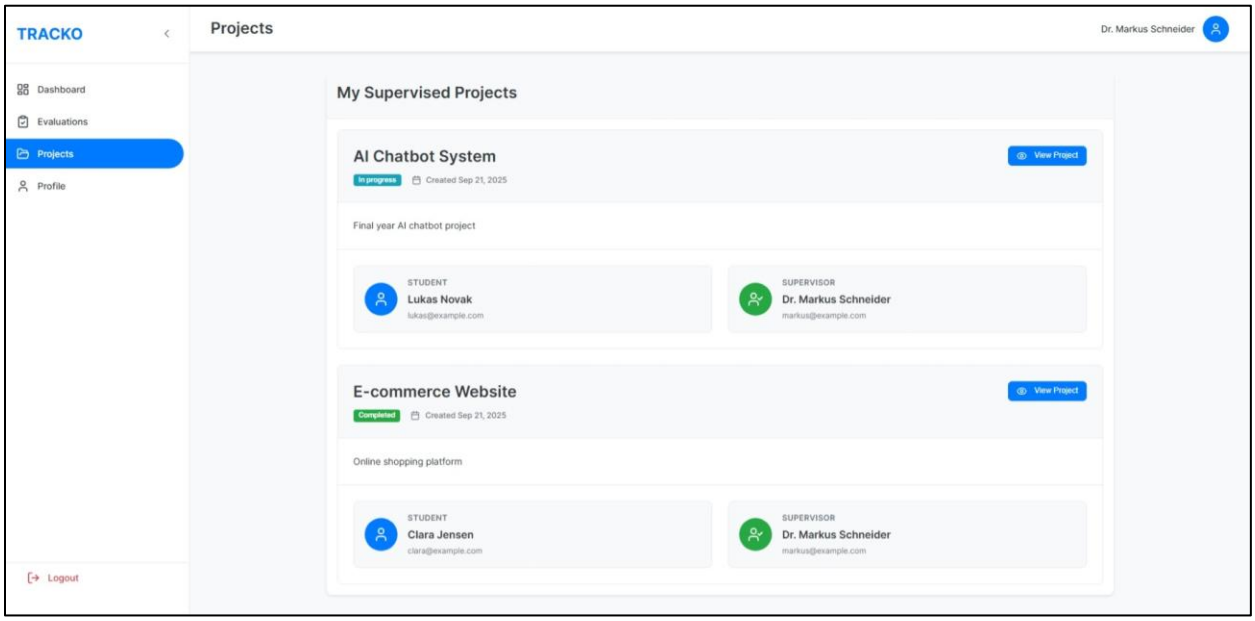
Student Dashboard: Figure-012



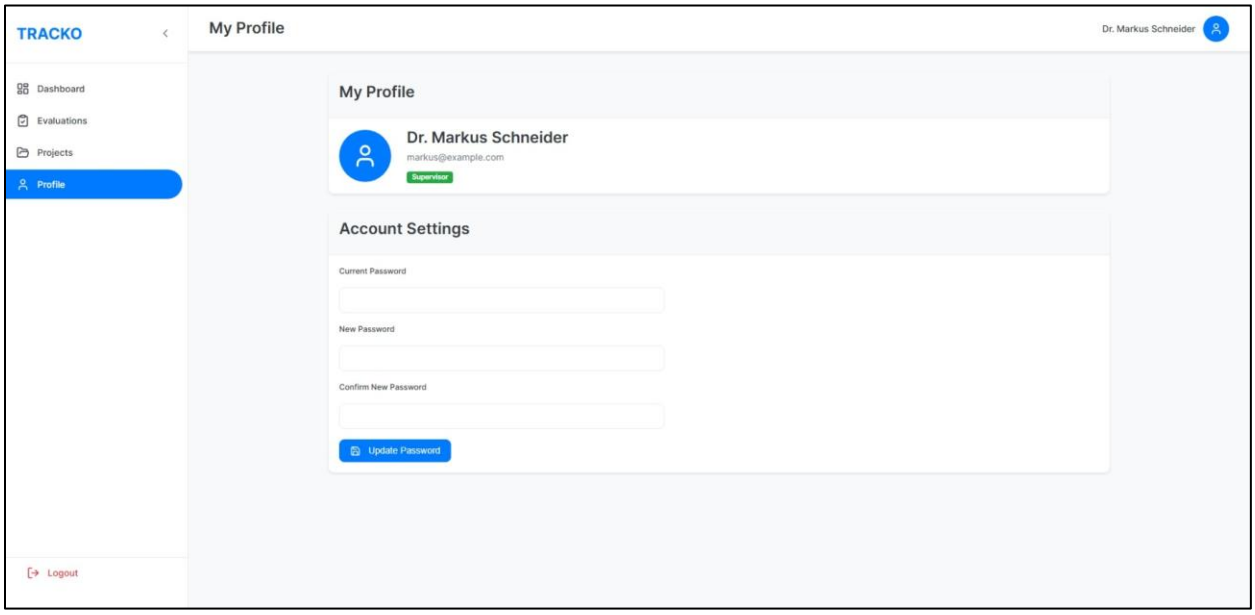
Supervisor Dashboard: Figure-013



Supervisor Evaluation: Figure-014



Supervisor Projects: Figure-015



Supervisor Profile: Figure-016

*** The-end ***