

12-2024

PROJECT TRACKING WITH MOBILE DEVICES

Mike Son

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd>



Part of the [Computer and Systems Architecture Commons](#), and the [Other Computer Engineering Commons](#)

Recommended Citation

Son, Mike, "PROJECT TRACKING WITH MOBILE DEVICES" (2024). *Electronic Theses, Projects, and Dissertations*. 2041.

<https://scholarworks.lib.csusb.edu/etd/2041>

This Project is brought to you for free and open access by the Office of Graduate Studies at CSUSB ScholarWorks. It has been accepted for inclusion in Electronic Theses, Projects, and Dissertations by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

PROJECT TRACKING WITH MOBILE DEVICES

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Mike Son
December 2024

PROJECT TRACKING WITH MOBILE DEVICES

A Project
Presented to the
Faculty of
California State University,
San Bernardino

by

Mike Son

December 2024

Approved by:

Dr. Haiyan Qiao, Advisor, Computer Science and Engineering

Dr. George M Georgiou, Committee Member

Dr. Jennifer Jin, Committee Member

© 2024 Mike Son

ABSTRACT

This innovative browser-based web application provides comprehensive access to project information, modernized communication, and effective work management from any device. Through a simple interface, it enables real-time collaboration, work delegation, and progress monitoring, making project management simpler everywhere. A standout feature of this application is its provision of a dedicated API (Web Programming Interface) for mobile devices, enabling seamless integration and synchronization between the web application and mobile platform apps. This guarantees a consistent and integrated user experience across all devices, allowing for quick task updates, and information sharing. The web application emphasizes security, with secure encryption and effective authentication processes in place to protect sensitive data and user privacy.

ACKNOWLEDGEMENTS

I want to express my greatest appreciation to my project advisor, Dr. Haiyan Qiao, for her valuable help, support, and guidance throughout this process. This knowledge, support, and constructive feedback have been very important for defining the project's direction and quality.

I would like to express my appreciation to the members of my project committee, Dr. George M Georgiou and Dr. Jennifer Jin, for their insightful comments and suggestions, which have significantly enhanced the outcome of this work.

Finally, I would want to express my sincere appreciation to my family and friends for their constant support and encouragement during all of this.

TABLE OF CONTENTS

| | |
|-----------------------------------|-----|
| ABSTRACT | iii |
| ACKNOWLEDGEMENTS..... | iv |
| LIST OF TABLES | ix |
| LIST OF FIGURES | x |
| CHAPTER ONE: INTRODUCTION | 1 |
| Background..... | 1 |
| Significance | 2 |
| Purpose | 3 |
| CHAPTER TWO: SYSTEM ANALYSIS..... | 4 |
| Proposed Solution | 4 |
| Software Requirements | 6 |
| Development Requirements..... | 6 |
| Software Stack..... | 7 |
| Hardware Requirements | 8 |
| Development Hardware | 8 |
| Server Hardware | 8 |
| CHAPTER THREE: SYSTEM DESIGN..... | 10 |
| Requirement Gathering..... | 10 |
| System Analysis and Design | 11 |
| Role and Access Control..... | 11 |
| Functional Requirements | 12 |
| Non-functional Requirements..... | 13 |

| | |
|--|----|
| System Modeling..... | 13 |
| Use Case Diagrams..... | 14 |
| Sequence Diagrams..... | 19 |
| Database Design | 22 |
| Workflow Entity | 23 |
| Project Entity | 24 |
| Project Comment Entity | 26 |
| Task Entity | 28 |
| Task Comment Entity..... | 30 |
| Task Status Entity | 32 |
| User Entity | 34 |
| User Role Entity | 37 |
| User Token Entity..... | 38 |
| Priority Entity..... | 40 |
| Linking Tables (Many to Many Relationships)..... | 42 |
| Wireframe Creation..... | 45 |
| Web Application Wireframes..... | 46 |
| Mobile Application Wireframe..... | 51 |
| CHAPTER FOUR: SYSTEM IMPLEMENTATION..... | 60 |
| The Three-Tier Architecture..... | 60 |
| Presentation Tier..... | 60 |
| Business Logic Tier..... | 60 |
| Data Access Tier..... | 61 |
| Advantages of a Shared Library | 61 |

| | |
|---|----|
| MVC 5 and .NET MAUI..... | 61 |
| Visual Studio 2022 for Development..... | 62 |
| Project Solution Structure | 62 |
| Documents..... | 63 |
| PrjTrakCommon..... | 63 |
| PrjTrakLib..... | 64 |
| PrjTrakMobile | 64 |
| PrjTrakWeb | 64 |
| UnitTestPrjTrak | 64 |
| Git as Source Control..... | 65 |
| Version Control and Backup | 65 |
| Collaboration and Scaling | 65 |
| Implementing Web Application | 65 |
| Implementing Database Design..... | 68 |
| Implementing Web API | 69 |
| Implementing Mobile Apps..... | 70 |
| CHAPTER FIVE: WEB API, SECURITY AND ENCRYPTION..... | 72 |
| How Apps Communicate | 72 |
| Web API Implementation in Details | 73 |
| Security in Web API..... | 77 |
| Authentication | 77 |
| Authorization | 77 |
| Hypertext Transfer Protocol Secure (HTTPS)..... | 78 |
| Web API Encryption Techniques | 78 |

| | |
|---|-----|
| CHAPTER SIX: TESTING AND DEPLOYMENT | 80 |
| Importance of Testing | 80 |
| Unit Testing..... | 80 |
| Web API Testing..... | 85 |
| Deployment..... | 86 |
| Web Application Deployment | 86 |
| Android Application Deployment | 87 |
| iOS Deployment | 88 |
| CHAPTER SEVEN: CONCLUSION AND FUTURE ENHANCEMENTS | 90 |
| Conclusion | 90 |
| Future Enhancements..... | 91 |
| Single Sign-On (SSO) | 91 |
| Attachment Support | 91 |
| Advanced Analytics and Reporting | 91 |
| CHAPTER EIGHT: SCREENSHOTS OF THE SYSTEM | 92 |
| Web Application Screenshots | 92 |
| iOS Simulator Screenshots – iPhone 15 Pro | 97 |
| Android Screenshots – LG V60 Real Phone..... | 103 |
| REFERENCES | 109 |

LIST OF TABLES

| | |
|---|----|
| Table 1. Workflow Entity Description | 24 |
| Table 2. Project Entity Description..... | 25 |
| Table 3. Project Comment Entity Description | 27 |
| Table 4. Task Entity Description | 29 |
| Table 5. Task Comment Entity Description | 31 |
| Table 6. Task Status Entity Description..... | 33 |
| Table 7. User Entity Description | 35 |
| Table 8. User Role Entity Description..... | 38 |
| Table 9. User Token Entity Description | 40 |
| Table 10. Priority Entity Description..... | 41 |
| Table 11. Project Assignee Description..... | 43 |
| Table 12. Project Watcher Description | 44 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1. Web and Mobile Integrated Concept | 4 |
| Figure 2. Integrated Project Tracking System Architecture | 5 |
| Figure 3. Workflow Management Use Case Diagram..... | 14 |
| Figure 4. Project Management Use Case Diagram | 15 |
| Figure 5. Project Priority Use Case Diagram | 16 |
| Figure 6. Task Management Use Case Diagram..... | 17 |
| Figure 7. Task Status Use Case Diagram | 18 |
| Figure 8. Role Management Use Case Diagram | 19 |
| Figure 9. Mobile App User Login Sequence Diagram..... | 20 |
| Figure 10. Request Data Sequence Diagram | 21 |
| Figure 11. Submit Data Sequence Diagram | 22 |
| Figure 12. Workflow Data Diagram..... | 23 |
| Figure 13. Project Data Diagram | 25 |
| Figure 14. Project Comment Data Diagram..... | 27 |
| Figure 15. Task Entity Data Diagram..... | 28 |
| Figure 16. Task Comment Data Diagram | 31 |
| Figure 17. Task Status Data Diagram | 33 |
| Figure 18. User Data Diagram..... | 35 |
| Figure 19. User Role Data Diagram | 37 |
| Figure 20. User Token Data Diagram..... | 39 |
| Figure 21. Priority Data Diagram | 41 |

| | |
|---|----|
| Figure 22. Project Assignee Data Diagram | 42 |
| Figure 23. Project Watcher Data Diagram | 44 |
| Figure 24. Web App Login Page..... | 46 |
| Figure 25. Web App Overview Dashboard Wireframe | 47 |
| Figure 26. Web App Project Details Wireframe | 48 |
| Figure 27. Web App Projects Wireframe | 49 |
| Figure 28. Web App User Role Wireframe | 50 |
| Figure 29. Web App User Wireframe..... | 51 |
| Figure 30. Mobile Login Wireframes..... | 52 |
| Figure 31. Mobile First Workflow Wireframe..... | 53 |
| Figure 32. Mobile Mid Workflow Wireframe | 53 |
| Figure 33. Mobile Last Workflow Wireframe | 54 |
| Figure 34. Mobile Project Details Wireframe | 55 |
| Figure 35. Mobile Projects Wireframe | 56 |
| Figure 36. Mobile Project Form Wireframe | 57 |
| Figure 37. Mobile Settings Wireframe | 58 |
| Figure 38. Mobile Profile Wireframe | 59 |
| Figure 39. Project Solution Structure..... | 63 |
| Figure 40. Web App MVC Structure | 66 |
| Figure 41. Business and Data Layer | 67 |
| Figure 42. Database Tables | 69 |
| Figure 43. Web API Controller..... | 70 |

| | |
|---|----|
| Figure 44. .NET MAUI Mobile Structure | 71 |
| Figure 45. Web Service Request Actions | 74 |
| Figure 46. Web Service Response Status | 74 |
| Figure 47. Web Request Class Diagram | 75 |
| Figure 48. Web Response Class Diagram | 75 |
| Figure 49. Web API Function Call by Mobile App..... | 76 |
| Figure 50. Example of User Token Data | 77 |
| Figure 51. Web API by Mobile App using Encoding Technique | 79 |
| Figure 52. Unit tests within a Visual Studio project..... | 81 |
| Figure 53. Project Entity Unit Test Result..... | 81 |
| Figure 54. Unit Test Project Entity by CRUD | 83 |
| Figure 55. Unit Test Create Project Entity | 83 |
| Figure 56. Unit Test Read Update Delete Project Entity..... | 84 |
| Figure 57. ProjectCRUD Unit Test Result | 85 |
| Figure 58. Web API Sample Test Result | 86 |
| Figure 59. Web Application Deployment | 87 |
| Figure 60. Android Deployment | 88 |
| Figure 61. Pair To Mac Menu | 88 |
| Figure 62. Pair To Mac Screen..... | 89 |
| Figure 63. iOS Simulator After Pairing..... | 89 |
| Figure 64. Web App Login | 92 |
| Figure 65. Web App Overview | 92 |

| | |
|---|-----|
| Figure 66. Web App Create New Task | 93 |
| Figure 67. Web App Add Project | 94 |
| Figure 68. Web App Admin Manage Users | 94 |
| Figure 69. Web App Admin Manage Workflows | 95 |
| Figure 70. Web App Admin Manage Roles | 95 |
| Figure 71. Web App Admin Manage Priorities..... | 96 |
| Figure 72. Web App Admin Manage Task Status..... | 96 |
| Figure 73. iOS Login Form | 97 |
| Figure 74. iOS Home Screen..... | 98 |
| Figure 75. iOS Password Manager Project View..... | 99 |
| Figure 76. iOS Project List..... | 100 |
| Figure 77. iOS Add Project..... | 101 |
| Figure 78. iOS Settings Screen | 102 |
| Figure 79. Android Login Form | 103 |
| Figure 80. Android Home Screen | 104 |
| Figure 81. Android Project View | 105 |
| Figure 82. Android Project List | 106 |
| Figure 83. Android Add Project..... | 107 |
| Figure 84. Android Settings Screen..... | 108 |

CHAPTER ONE

INTRODUCTION

Background

The strict security rules set by the Criminal Justice Information Services (CJIS) make it important for law enforcement organizations to set up an on-site project tracking system. By employing their own internal systems, these agencies can employ greater control and maintenance over vital information within the secure boundaries of their offices. This strategic approach minimizes the risk of data breaches and unauthorized access, addressing significant concerns associated with cloud-based applications. As a senior software engineer at the San Bernardino Sheriff's department, I am responsible for managing and safeguarding sensitive information and ensuring prompt emergency responses. My motivation for creating this project tracking system comes from the need for on-site infrastructure, with all servers and databases located within the organization to enhance data security and control. By doing so, I can address the delay, unreliability, and security concerns often associated with cloud hosting vendors. On-site hosting will prevent potential data breaches, enhance system performance, and ensure full compliance with CJIS standards by strengthening our security protocols. This solution is essential for maintaining the department's efficiency and security.

Significance

The capacity of managers to view real-time updates on progress, resources, and availability is essential in a project tracking system designed specifically for law enforcement. This will significantly enhance both operational efficiency and accountability. A system like this also helps with strategic decisions by giving leaders a full picture of all the projects and resources that are currently being used. This makes it easier to better divide up resources, set priorities for tasks, and step in quickly when projects are running late or over budget. For law enforcement organizations, a project tracking system is fundamentally an essential tool that helps them keep up high standards of safety, efficiency, and accountability in their work.

This project management system is distinguished from others such as Jira [1], Basecamp [2], and Zoho Projects [3] by its ability to incorporate modular workflows that can be specifically customized for various divisions. A workflow is a series of defined stages that represent the sequence of tasks needed to complete a project. It presents an outline of how work expands from beginning to end, ensuring that all tasks are accomplished in the correct sequence. Beginning with the Information Technology (IT) department, a software development team workflow can include stages like Pending, Design, Development, Testing, Production, while a server team workflow can have Pending, Provisioning, Deployment, and Monitoring. These workflows allow for clear visibility into project

progression, ensuring each stage is properly completed before moving on to the next.

In contrast, other popular project management software, such as Jira, Basecamp, and Zoho Projects, do not provide this level of detailed, customizable workflow management for different types of projects out of the box. The level of adaptability in this project management ensures that every team inside the organization, including IT, network, dispatch center, and others, may operate with a system customized to their requirements while remaining effectively integrated with other departments.

Purpose

Monitoring project progress, allocating resources, defining and keeping track of deadlines, and making sure that project goals are in line with the organization's objectives. This helps team members communicate clearly, gives real-time updates on the project's progress, and finds possible problems or risks that could delay the project. This project tracking system targets to improve responsibility, and decision making by providing a central location for all project related information. This eventually leads to successful and timely project completion.

CHAPTER TWO

SYSTEM ANALYSIS

Proposed Solution

The application will be a hybrid platform that combines web and mobile functionalities. It will be constructed utilizing .NET MAUI [4] for the mobile app, ASP.NET MVC 5 [5] for the web application and Web API [6]. The technology would allow instant exchange of data and ensure compatibility across various device types, such as smartphones, tablets, and computers.

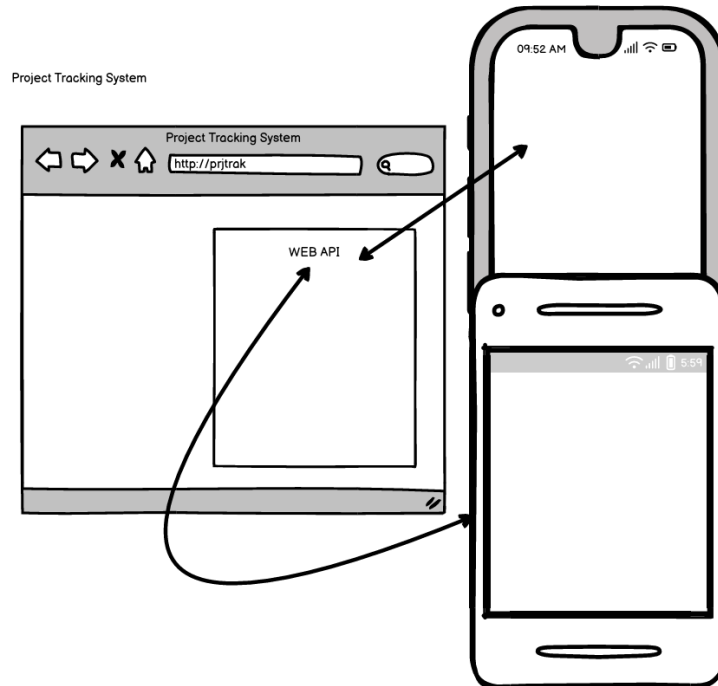


Figure 1. Web and Mobile Integrated Concept

Figure 1 illustrates a project tracking system with web and mobile apps connected via Web API for seamless data interchange.

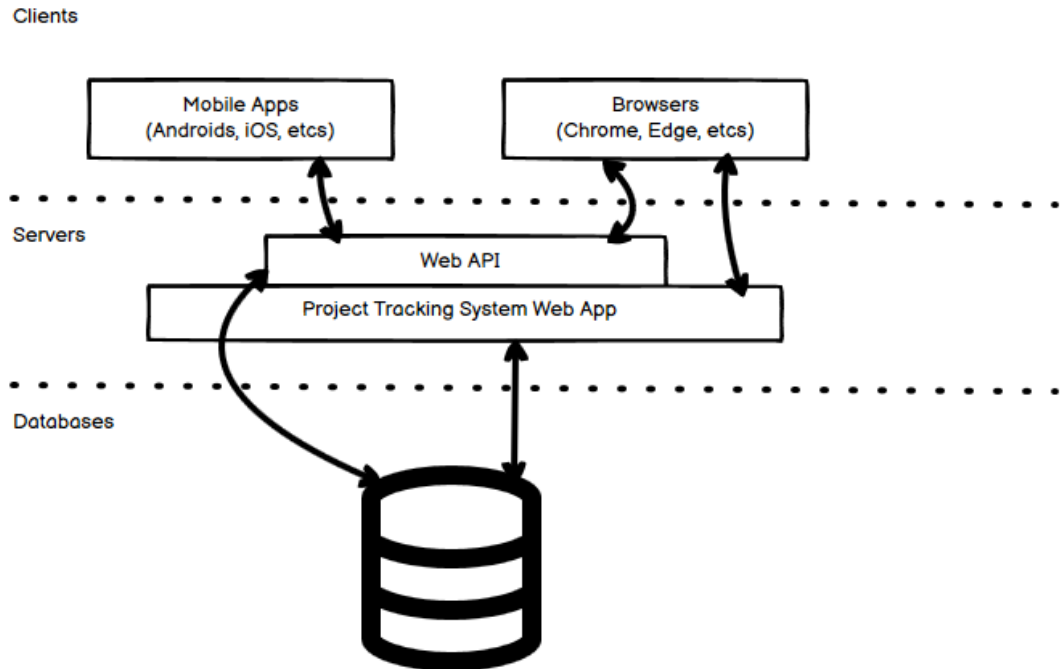


Figure 2. Integrated Project Tracking System Architecture

Figure 2 shows the project tracking system where Android and iOS (iPhone Operating System) apps and Chrome and Edge browsers use a Web API to interface with a central server. The back-end database stores project data, and this setup provides a single access point. The system synchronizes data across several client types in real time, making project modifications available to

everyone. This design offers flexible, scalable interactions across platforms, essential for modern project management.

Software Requirements

Development Requirements

1. .NET MAUI:

- Purpose: To develop cross-platform mobile applications
- Version: .NET 8
- Supported Platforms: iOS, Android, macOS, and Windows

2. ASP.NET MVC 5:

- Purpose: To handle the web API and backend logic
- .NET Framework

3. Visual Studio 2022:

- Purpose: Integrated development environment (IDE) for .NET applications
- Features: Supports .NET MAUI and MVC 5 development with built-in templates and debugging tools

4. SQL Server:

- Purpose: Database server to manage application data
- Version: SQL Server 2022

5. SQL Server Management Studio (SSMS):

- Purpose: A graphical user interface that is necessary for designing, managing, and administering SQL Server databases
- Version: Latest

Software Stack

1. Languages:

- C# (for both .NET MAUI and MVC 5 development)
- HTML, CSS, JavaScript (for front-end web development)

2. Frameworks and Libraries:

- .NET MAUI for mobile client development
- jQuery and Bootstrap for front-end scripting and styling

3. APIs and Services:

- RESTful APIs using ASP.NET Web API
- Security: All communications between the mobile client, web client, and the server will be secured using HTTPS to prevent interception and eavesdropping.
- Data Transmission: HTTPS ensures that all data transmitted over the network is encrypted using TLS (Transport Layer Security), which provides confidentiality and integrity of data.
- Data Protection: Tokens and sensitive data are encrypted in transit and optionally at rest, using industry-standard cryptographic protocols.

Hardware Requirements

I will utilize .NET MVC and .NET MAUI to target iOS and Android platforms. My primary development will be focused on the Windows platform, while a Mac Mini will be utilized to run the iOS app emulator. Optional testing devices for the web application can consist of any operating system that supports modern web browsers. For Android, testing can be done on any device running Android version 12 and above, while for iOS, optional devices include iPhone 12 and later models, although these are not required.

Development Hardware

1. Windows PC:

- Operating System: Windows 10 or higher
- Processor: Intel Core i5 or better
- RAM: 8 GB minimum
- Storage: SSD with at least 256 GB of space
- Capable of running Visual Studio and multiple emulators/simulators

2. Mac Mini (for iOS development):

- Processor: Intel chip or better
- RAM: 8 GB minimum
- Version: macOS Monterey or newer
- Necessary to run iOS simulators through Visual Studio

Server Hardware

1. Server (Virtual Machine or Dedicated)

- Processor: Quad-core Xeon or better, RAM: 16 GB minimum
- Storage: 256 GB SSD primary storage
- High-speed, reliable internet connection for hosted services
- Operating System: Windows Server 2019 or better.

CHAPTER THREE

SYSTEM DESIGN

As the sole developer on this project, utilizing the waterfall methodology offers a unique and sequential development process. The advantage of this strategy is in its ability to facilitate careful preparation and proper implementation at every phase, as a result decreasing the likelihood of task duplication and the need for rework. Additionally, it promotes in maintaining a singular concentration on one phase at a time, which is important while overseeing various aspects of development separately. The process will include system requirement gatherings, analysis and design, and wireframe creation.

Requirement Gathering

Requirement gathering is a crucial phase in software project development. The initial version of the project tracking system will primarily be used by the software development team, with the workflow focused on software development processes.

The system organizes tasks within projects, grouped into workflows, each with its own hierarchy and ordering to enhance project management efficiency. It supports multiple user roles: admin, manager, team member, stakeholder, and system user. A stakeholder is anyone who has an interest in the project or is a watcher monitoring the project's progress and outcomes. Each with specific access rights, enabling appropriate interactions with the system. Critical features

include role specific task and project management, prioritization, progress status updates, and a commenting feature for interactive communication across tasks and projects.

The system will be available on both a web interface browser and mobile apps, guaranteeing seamless integration and rapid synchronization over a reliable Web API.

System Analysis and Design

The core functionality of the system revolves around the concept of workflows. Each workflow is a collection of projects, and each project contains multiple tasks. This structure allows for organized management of projects and the tasks within them, enabling clear oversight and efficient execution.

Role and Access Control

The system will include multiple user roles, each with unique tasks and levels of access:

1. Admin: Holds the ultimate control over the system. Administrators have the authority to create and oversee all workflows, projects, and tasks. In addition, they have the ability to assign roles and oversee access for all other users.
2. Manager: Responsible for overseeing all projects. Managers can create projects, assign tasks to team members, and track progress. They also have the authority to modify task details and priorities.

3. Team Member: Executes the project assigned to them. They can update the status and details of their tasks.
4. Stakeholder: Has read only access to specified projects and workflows. Stakeholders can view progress and receive updates but cannot make changes to the project or task details.
5. System User: Non-active user account used programmatically for automation, not associated with real individuals. It performs system tasks like batch jobs and maintenance without standard login capabilities, ensuring secure operations.

Functional Requirements

1. Workflow Management: Administrators may create new workflows and modify existing ones. Workflows serve as huge containers for projects. All users can view workflows to which they have access, depending on their role and admin-set permissions.
2. Project Management: Managers can create projects. They can set project priorities, and assign team members to projects. Projects can be updated by managers and team members with appropriate roles.
3. Task Management: Within each project, managers and team members with sufficient privileges can create or update task. Team members assigned to a task can update the task status (e.g., To Do, Doing, Done) and provide detailed progress reports.

4. Role Management: Administrators determine which responsibilities users have and give those roles to users.
5. Comment System: Users can add comments to both projects and tasks, facilitating real time communication and collaboration among team members, and stakeholders.

Non-functional Requirements

1. Usability: The web application interface will be made to be intuitive and simple enough that people of all technical skill levels can use it.
2. Performance: The system will be adjusted for performance, with quick load times and rapid data processing suitable for multiple users and large amounts of data without lag.
3. Security: Protect sensitive data and prevent unauthorized access. This will include user authentication, data encryption, and secure API connections.
4. Scalability: The system will be scalable, designed to accommodate an increasing number of users, projects, and tasks without a decrease in performance.

System Modeling

A method for visually representing, evaluating, and enhancing the suggested processes inside a system. Using this approach makes it easier to understand how the system works and interacts, which facilitates clear communication between developers, designers, and stakeholders. For this

project, I am utilizing Unified Modeling Language (UML), primarily Use Case and Sequence diagrams to present the system's model.

Use Case Diagrams

In this system, traditional delete functions are largely unnecessary due to the use of an 'active' flag for all entities. Instead of permanently deleting data, the active status is toggled, preventing irreversible data loss and eliminating the need for data recovery processes. This approach retains the full historical record, enabling comprehensive reporting and analytics, which is essential for maintaining historical accuracy and ensuring compliance with legal regulations.

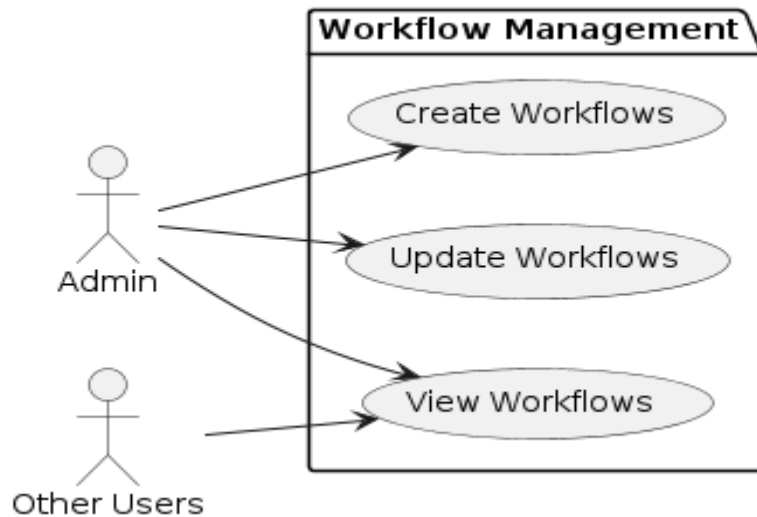


Figure 3. Workflow Management Use Case Diagram

Figure 3 illustrates the roles of the administrators and other users on workflows. Other users refer to the manager, team member, and stakeholder roles. The administrator has the ability to create, update, and view workflows, while other users can only view workflows.

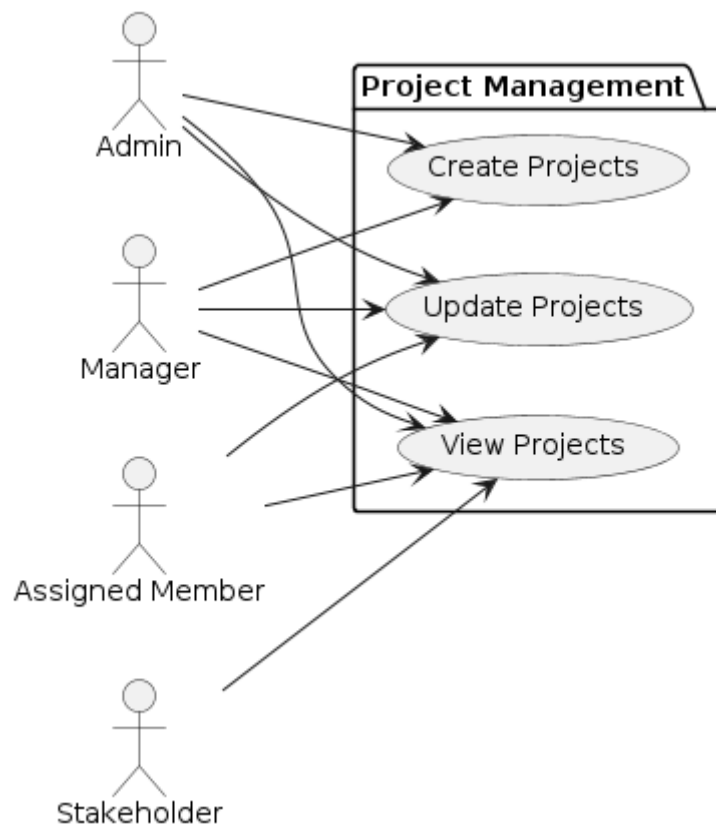


Figure 4. Project Management Use Case Diagram

The use case of project management is shown in Figure 4. The administrator and manager can create, update, and view projects, while assigned

members can update and view the project, and other stakeholders can only view it.

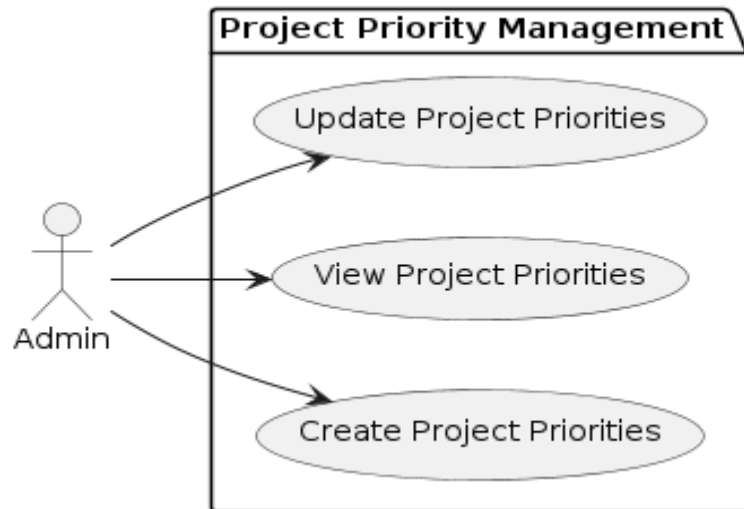


Figure 5. Project Priority Use Case Diagram

Figure 5 illustrates the admin manages project priorities by creating, viewing, and updating. An example of project priority levels includes Low, Medium, High, and other classifications.

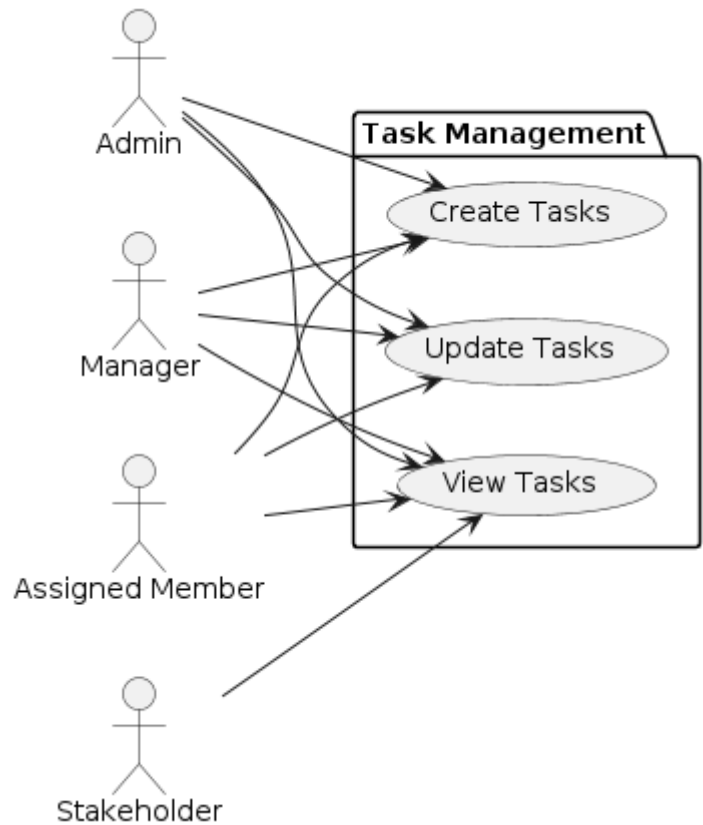


Figure 6. Task Management Use Case Diagram

Figure 6 illustrates that both admins and managers can create, update, and view tasks, while assigned members have the same abilities. Other stakeholders, however, are limited to viewing tasks only.

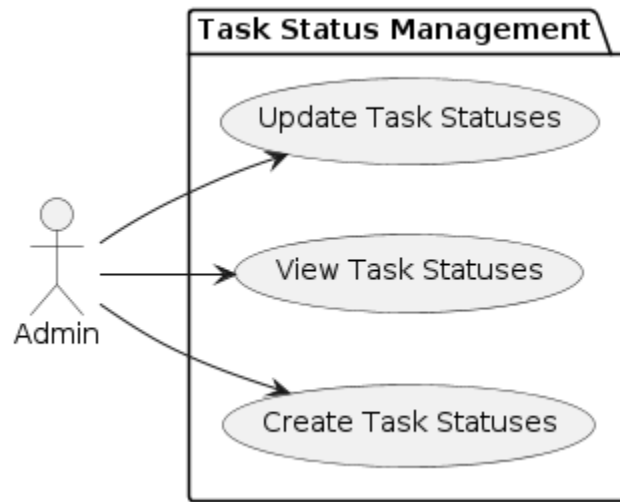


Figure 7. Task Status Use Case Diagram

Figure 7 illustrates the admin manages task statuses by creating, viewing, and updating. An example of task status levels includes To Do, In Progress, Completed, and other states.

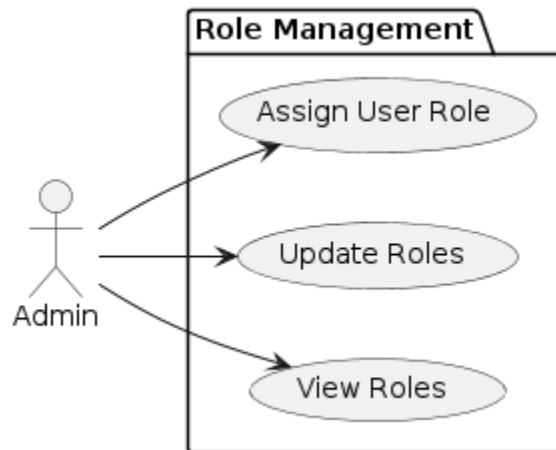


Figure 8. Role Management Use Case Diagram

Figure 8 illustrates that an admin capabilities includes assigning roles to users, updating role descriptions, and viewing the role descriptions of each user.

Sequence Diagrams

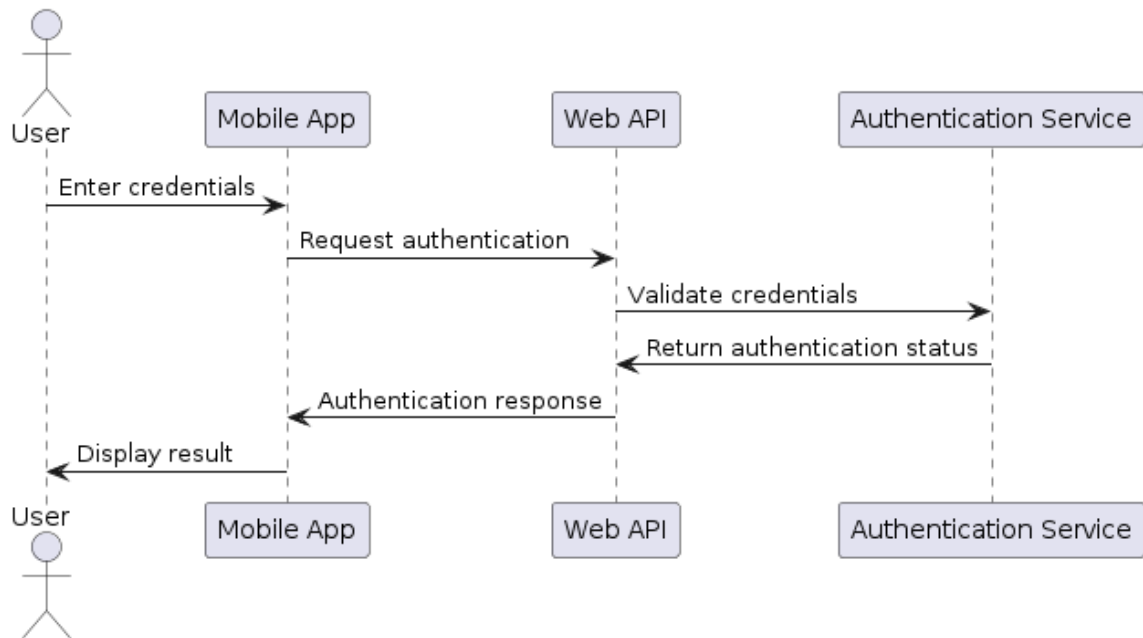


Figure 9. Mobile App User Login Sequence Diagram

Figure 9 illustrates the sequence of interactions when a user logs into the web application through a web app API.

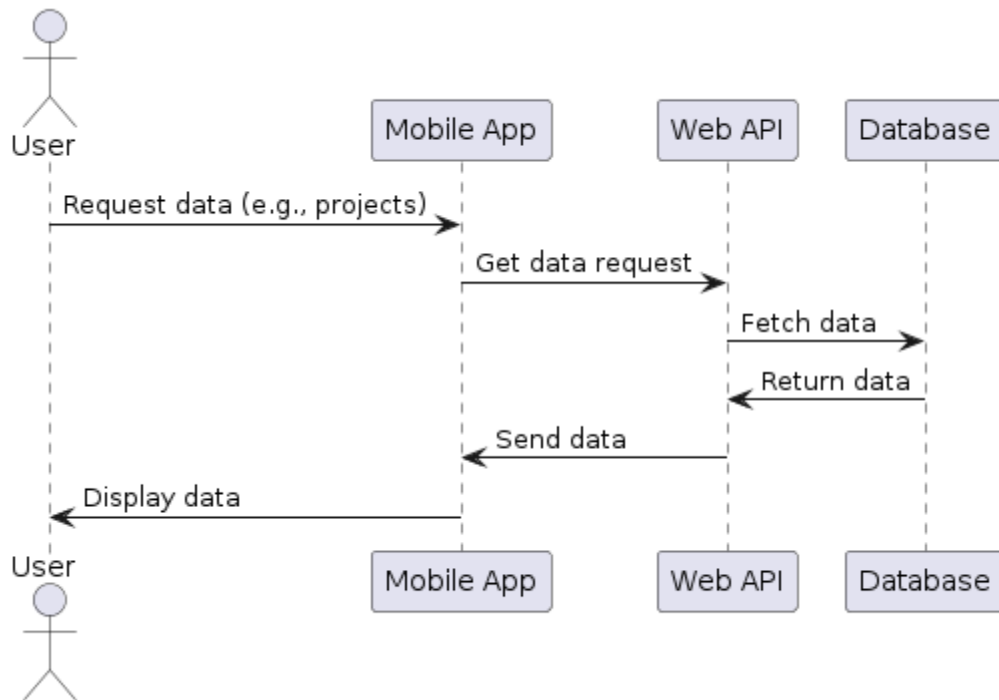


Figure 10. Request Data Sequence Diagram

Figure 10 illustrates the step-by-step process of retrieving data from the server via a web application API then displaying it in the web application.

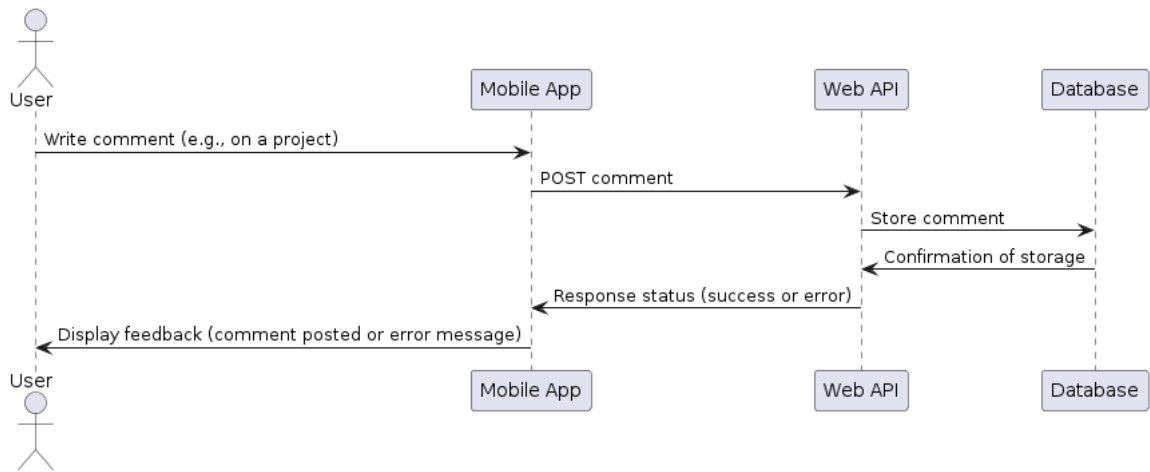


Figure 11. Submit Data Sequence Diagram

Figure 11 illustrates the process of a user submitting a comment on a project via a mobile application interface.

Database Design

A well-designed database is important for a project tracking system because it protects the integrity of the data, speeds up the system, and makes it easier to run complex queries. A well-structured database allows for flexible growth, speeds up data retrieval, and makes it easier to connect to other systems. This makes it easier to handle and keep track of resources, tasks, and projects. Microsoft SQL Server 2022 and the Visual Management Studio tool are used to create the database structure for this project.

Workflow Entity

Workflows are steps that a project goes through from the beginning to the end. It's different for every department of the organization. For a software template project, the work flow is split into several stages. Pending, where projects are waiting for approval and resource distribution. Design, which includes making specific designs and user interfaces; Development, which is where the real coding of changes happen. Testing, using different testing methods to make sure the software meets all requirements. Production is the last stage. This is where the software is put into a working environment.


| workflow | |
|---|-----------|
|  | code |
| | name |
| | is_active |
| | order_id |

Figure 12. Workflow Data Diagram

Figure 12 illustrates the workflow table comprises four columns specifically designed to monitor and categorize various workflows within a system.

Table 1. Workflow Entity Description

| Column Name | Data Type | Description |
|-------------|---------------------|---|
| code | Text | A unique code representing the workflow, up to 50 characters. |
| name | Text | The name of the workflow, can store up to 255 characters. |
| is_active | Yes/No (Boolean) | Boolean value indicating if the workflow is active (1) or inactive (0). |
| order_id | Number | A numeric value used to determine the order of workflows. |

Table 1 describes the structure of a database table intended to store information about workflows.

Project Entity

The 'project' table stores all key information related to various projects within the system. It is designed to capture essential details for each project.

Figure 13 illustrates the database relationships between the 'project' table and three other tables: 'priority,' 'workflow,' and 'user.'

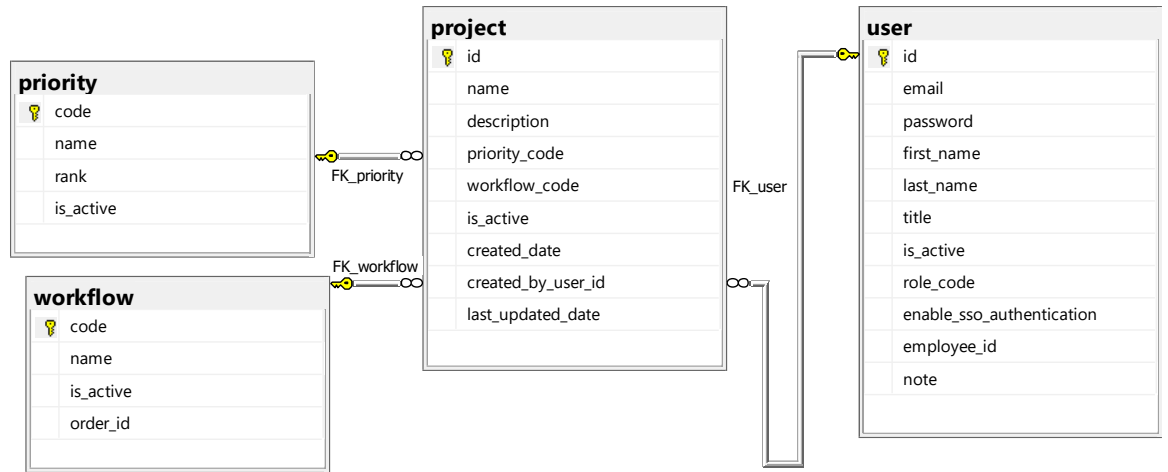


Figure 13. Project Data Diagram

Table 2. Project Entity Description

| Column Name | Data Type | Description |
|---------------|-----------|--|
| id | Number | Unique identifier for each project. Auto-increments. |
| name | Text | The name of the project. |
| description | Text | A detailed description of the project. |
| priority_code | Text | A code indicating the project's priority, linked to another table for priority codes. |
| workflow_code | Text | A code representing the project's workflow status, linked to another table for workflow definitions. |

| Column Name | Data Type | Description |
|--------------------|------------------|---|
| is_active | Yes/No (Boolean) | Project is active (1) or inactive (0). |
| created_date | Date and Time | The date and time the project was created. |
| created_by_user_id | Number | The ID of the user who created the project. |
| last_updated_date | Date and Time | The date and time the project was last updated. |

Table 2 describes the structure of a database table designed to store information about projects.

Project Comment Entity

The project_comment table is specifically designed to handle and store comments that are associated with particular projects within the system. It's very important for communication and teamwork on projects because it lets team members, stakeholders, or anyone else who needs to leave comments or notes directly about projects.

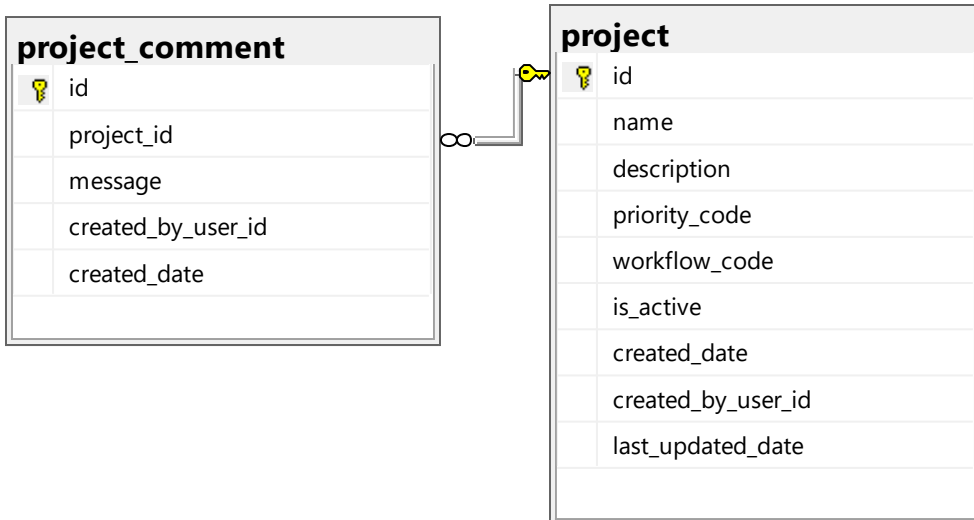


Figure 14. Project Comment Data Diagram

Figure 14 shows the database relationships between two tables: project and project_comment.

Table 3. Project Comment Entity Description

| Column Name | Data Type | Description |
|-------------|-----------|---|
| Id | Number | A unique identifier for each comment. |
| project_id | Number | The identifier of the project to which the comment is associated, linking comments directly to their respective projects. |
| message | Text | The text of the comment. |

| Column Name | Data Type | Description |
|--------------------|---------------|---|
| created_by_user_id | Number | The ID of the user who created the comment. |
| created_date | Date and Time | The date and time when the comment was created, providing a timestamp for the commentary. |

Table 3 describes the structure of a database table intended to store information about project comments.

Task Entity

The task table is designed with a specific structure to effectively manage and track individual tasks that are associated with various projects within the system.

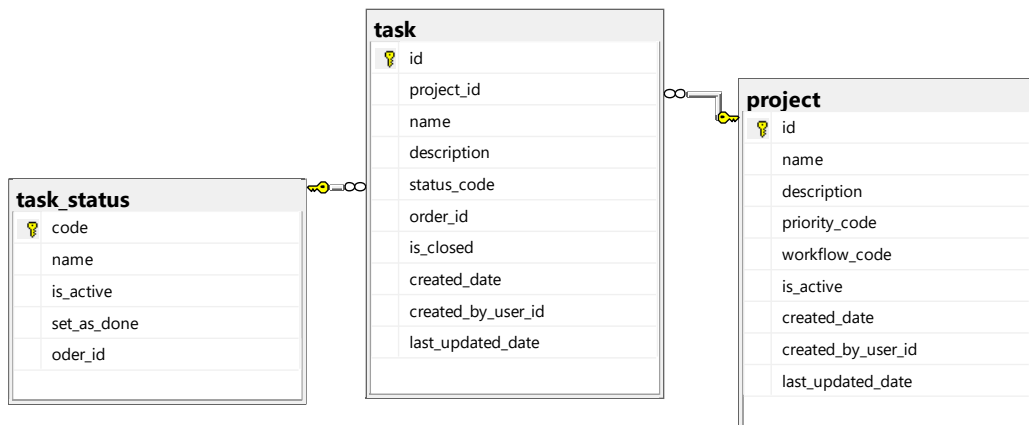


Figure 15. Task Entity Data Diagram

Figure 15 shows the database relationships between three tables: project, task, and task_status.

Table 4. Task Entity Description

| Column Name | Data Type | Description |
|-------------|-----------|---|
| id | Number | A unique identifier for each task. |
| project_id | Number | The identifier of the project to which the task is associated, linking tasks to their respective projects. |
| name | Text | The name of the task, capable of storing up to 4000 characters, which describes the task succinctly. |
| description | Text | A detailed description of the task, which can include extensive information such as objectives, requirements, and specific details. |
| status_code | Text | A code that describes the current status of the task, linked to a status table that defines various states a task can be in |

| Column Name | Data Type | Description |
|--------------------|------------------|---|
| | | (like “pending”, “in progress”, “completed”). |
| order_id | Number | A numeric value that can be used to order or prioritize tasks within a project. |
| is_closed | Yes/No (Boolean) | A Boolean field indicating whether the task is closed (completed or cancelled). |
| created_date | Date and Time | The date and time when the task was created, providing a timestamp for the start of the task. |
| created_by_user_id | Number | The ID of the user who created the task |
| last_updated_date | Date and Time | The date and time the task was last updated. |

Table 4 describes the structure of a database table intended to store information about tasks.

Task Comment Entity

The task_comment table has been specifically designed to efficiently manage and store comments that are associated with individual tasks within the system. It lets stakeholders and team members leave comments or notes on a

task, which makes it easier to share ideas and get feedback in a detailed way and on a regular basis.

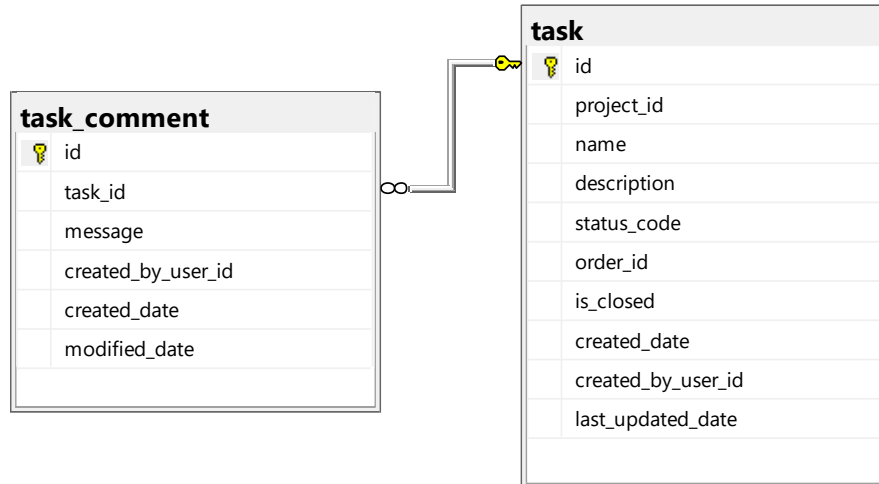


Figure 16. Task Comment Data Diagram

Figure 16 shows the database relationships between two tables: task, and task_comment.

Table 5. Task Comment Entity Description

| Column Name | Data Type | Description |
|-------------|-----------|---|
| id | Number | A unique identifier for each comment. |
| Task_id | Number | Linking comments to their respective tasks. |

| Column Name | Data Type | Description |
|--------------------|------------------|---|
| Message | Text | The text of the comment. |
| Created_by_user_id | Number | The ID of the user who created the comment |
| Created_date | Date and Time | The date and time when the comment was created. |
| Modified_date | Date and Time | The date and time when the comment was last modified. |

Table 5 describes the structure of a database table intended to store information about task comments.

Task Status Entity

The task_status table is set up to describe and keep track of the different states that system tasks can be in. The table makes it easy to keep track of task progress and phases.

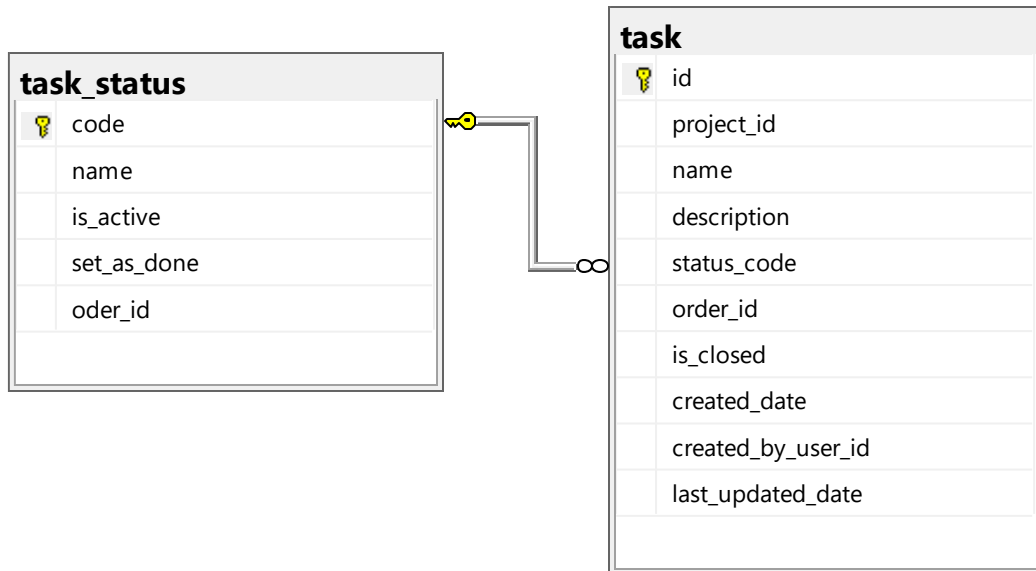


Figure 17. Task Status Data Diagram

Figure 17 shows the database relationships between two tables: `task`, `task_status`.

Table 6. Task Status Entity Description

| Column Name | Data Type | Description |
|-------------------|-----------|---|
| <code>code</code> | Text | A unique code for each status, allowing up to 255 characters. |
| <code>name</code> | Text | The name of the status, also allowing up to 255 characters. |

| Column Name | Data Type | Description |
|--------------------|---------------------|--|
| is_active | Yes/No (Boolean) | A Boolean field indicating whether the status is currently active and can be applied to tasks. |
| set_as_done | Yes/No (Boolean) | A Boolean field indicating whether a task assigned this status should be considered completed or closed. |
| order_id | Number | A numeric value used to determine the order or priority of statuses. |

Table 6 describes the structure of a database table intended to store information about task statuses.

User Entity

The user table plays a critical role in managing user administration and ensuring system security. The system captures crucial information about each user, such as personal identification and their role within the system.

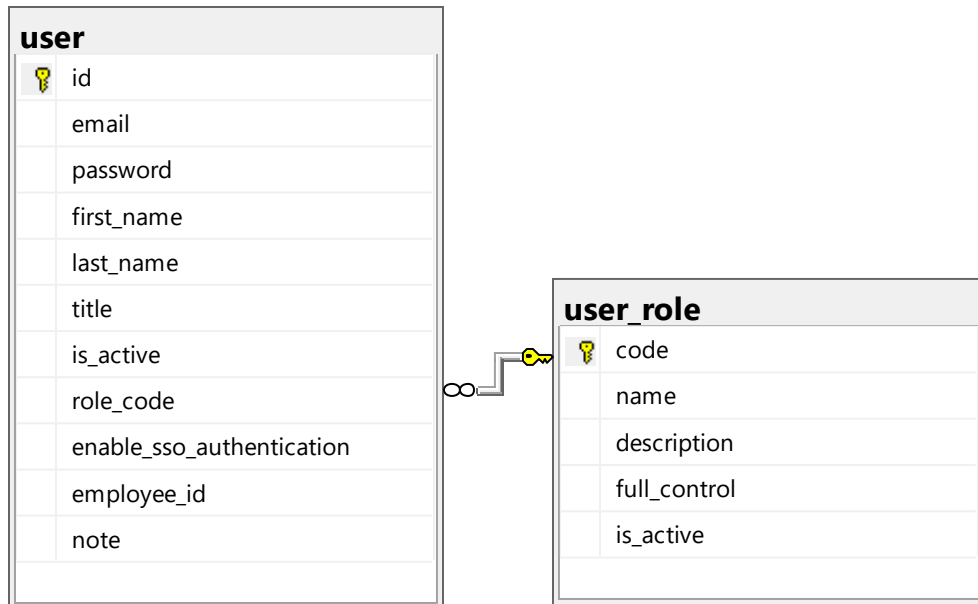


Figure 18. User Data Diagram

Figure 18 shows the database relationships between two tables: user, and user_role.

Table 7. User Entity Description

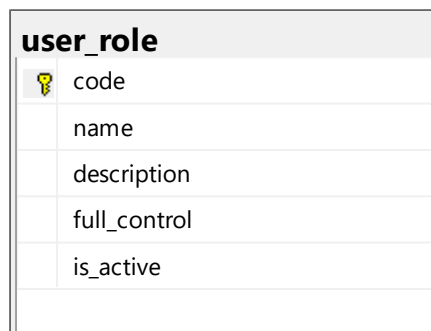
| Column Name | Data Type | Description |
|-------------|-----------|---|
| id | Number | A unique identifier for each user. |
| email | Text | The email address of the user, serving as username for system access. |

| Column Name | Data Type | Description |
|---------------------------|---------------------|---|
| password | Text | The user's password, stored in a hash format. |
| first_name | Text | The user's first name, allowing up to 255 characters. |
| last_name | Text | The user's last name, allowing up to 255 characters. |
| title | Text | The professional or job title of the user (255 characters). |
| is_active | Yes/No (Boolean) | A Boolean field indicating whether the user's account is active. |
| role_code | Text | A code that identifies the user's role within the system, linked to a roles table that defines various permissions and access rights. |
| enable_sso_authentication | Yes/No (Boolean) | A Boolean field indicating whether the user can authenticate via Single Sign-On (SSO), enhancing security and user convenience. |
| employee_id | Text | A unique identifier for the employee within the organization. |
| note | Text | A field for additional information or notes about the user. |

Table 7 describes the structure of a database table intended to store information about users.

User Role Entity

The user_role table is a key element in the system's access control management. It serves the purpose of defining specific roles that can be assigned to users. The roles are identified by a distinct code and name, and thorough descriptions are provided to clarify the specific permissions and responsibilities associated with each role.




| user_role | |
|---|--------------|
|  | code |
| | name |
| | description |
| | full_control |
| | is_active |

Figure 19. User Role Data Diagram

Figure 19 illustrates the user role database table.

Table 8. User Role Entity Description

| Column Name | Data Type | Description |
|--------------|---------------------|---|
| code | Text | A unique code for each role, allowing up to 255 characters. |
| Name | Text | The name of the role, allowing up to 255 characters. |
| Description | Text | A detailed description of the role. |
| Full_control | Yes/No (Boolean) | A Boolean field indicating whether the role has full control over the system. |
| Is_active | Yes/No (Boolean) | A Boolean field indicating whether the role is currently active. |

Table 8 describes the structure of a database table intended to store information about task user roles.

User Token Entity

The user_token table was created to handle the management of authentication tokens for users. It tends to be used in situations where mobile applications need to establish connections with web APIs. The tokens play a

crucial role in ensuring the security of communication between the user's device and the system.

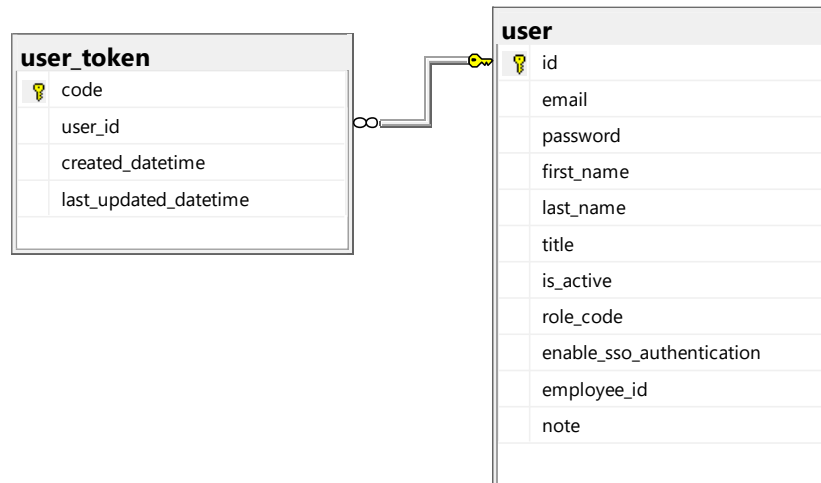


Figure 20. User Token Data Diagram

Figure 20 shows the database relationships between two tables: **user_token**, and **user**.

Table 9. User Token Entity Description

| Column Name | Data Type | Description |
|-----------------------|---------------|---|
| code | Text | A unique token generated for a user session, allowing up to 255 characters. This token is used to authenticate API requests made by the user. |
| user_id | Number | The identifier of the user to whom the token is assigned. |
| created_datetime | Date and Time | The date and time when the token was created. This timestamp helps track when the token was issued. |
| last_updated_datetime | Date and Time | The date and time when the token was last updated or refreshed. |

Table 9 describes the structure of a database table intended to store information about user tokens.

Priority Entity

The purpose of the priority table is to establish and manage the priority levels assigned to projects within a given system. The allocation of resources and attention among different projects can be determined by each priority level.

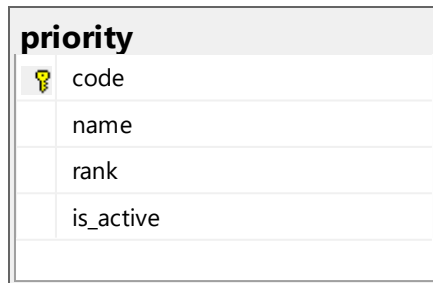


Figure 21. Priority Data Diagram

Figure 21 illustrates the project priority database table.

Table 10. Priority Entity Description

| Column Name | Data Type | Description |
|-------------|------------------|--|
| code | Text | A unique code for each priority level, allowing up to 50 characters. |
| name | Text | The name of the priority level, allowing up to 1000 characters. |
| rank | Number | A numeric value that represents the rank or order of the priority. |
| is_active | Yes/No (Boolean) | The priority level is currently active or not. |

Table 10 describes the structure of a database table intended to store information about project priorities.

Linking Tables (Many to Many Relationships)

The project_assignee table has been specifically designed to effectively manage the many-to-many relationship between projects and assignees within a system. Its primary purpose is to enable the assignment of multiple users (assignees) to multiple projects.

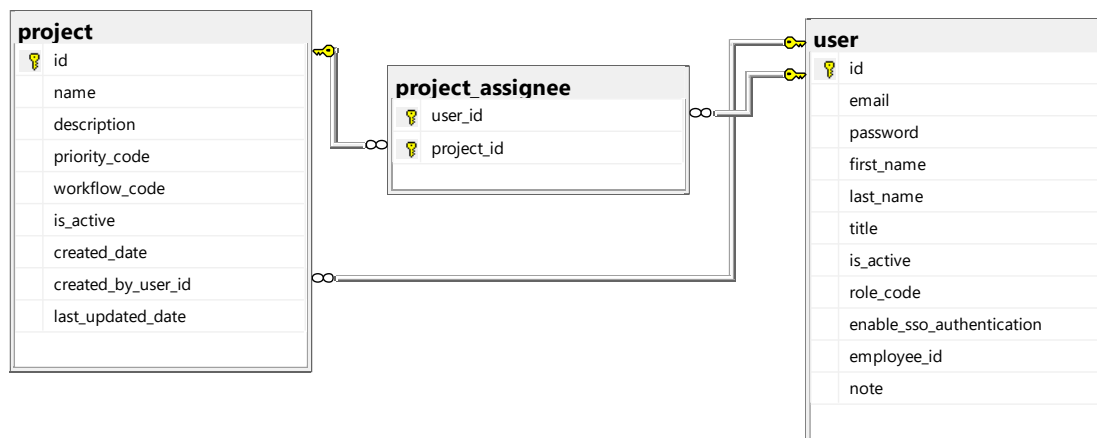


Figure 22. Project Assignee Data Diagram

Figure 22 shows the database relationships between three tables: project, project_assignee, and user.

Table 11. Project Assignee Description

| Column Name | Data Type | Description |
|--------------------|------------------|---|
| user_id | Number | The identifier of the user who is assigned to a project. This field links each entry back to a specific user in the user table. |
| project_id | Number | The identifier of the project to which a user is assigned. This field links each entry back to a specific project in the project table. |

Table 11 describes the structure of a database table intended to store information about project assignees.

The project_watcher table is designed to handle the many-to-many relationship between projects and watchers. The current configuration enables multiple users, referred to as "watchers," to follow multiple projects.

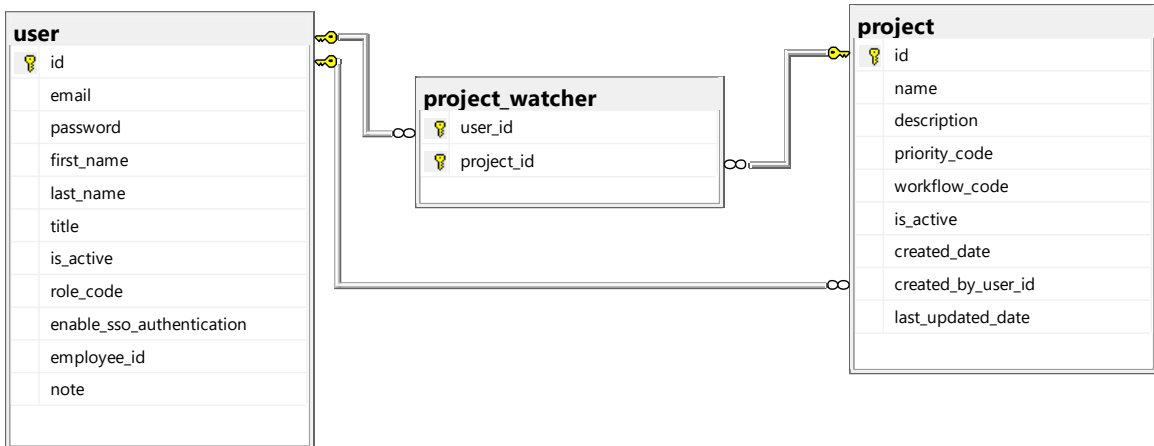


Figure 23. Project Watcher Data Diagram

Figure 23 shows the database relationships between three tables: project, project_watcher, and user.

Table 12. Project Watcher Description

| Column Name | Data Type | Description |
|-------------|-----------|---|
| user_id | Number | The identifier of the user who is watching a project. This links the entry to a specific user in the user table. |
| Project_id | Number | The identifier of the project that is being watched by a user. This links the entry to a specific project in the project table. |

Table 12 describes the structure of a database table intended to store information about project watchers.

Wireframe Creation

Creating wireframes is a crucial step in software development as they provide a visual representation of the user interface (UI) before actual development begins. Early visualization helps define the structure, layout, and interactions of different components, ensuring that all stakeholders have a clear understanding of the expected features and user flow. In the waterfall method, wireframes play a vital role in capturing requirements early, minimizing costly changes later on. They facilitate clear communication between designers, developers, and clients, eliminating confusion and ensuring that everyone is aligned on expectations. This leads to a more efficient development process and contributes to the overall success of the project.

This project will involve the development of two distinct wireframes: one for a web application and another for a mobile application. Each wireframe will have its own unique design, with the mobile version offering fewer capabilities compared to the web version.

Web Application Wireframes

Figure 24 displays the login form for the Project Tracking System's web application, where users must enter their email and password to securely authenticate their accounts.

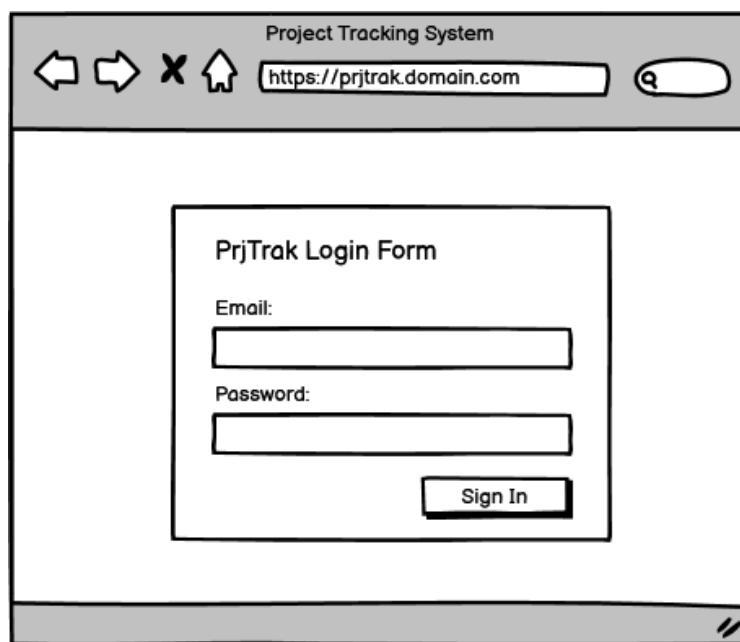


Figure 24. Web App Login Page

Figure 25 below presents the project workflow, allowing users to easily navigate through projects organized by different states, including Pending, Design, Development, Testing, and Production. It provides an organized view, facilitating easy monitoring and status updates.

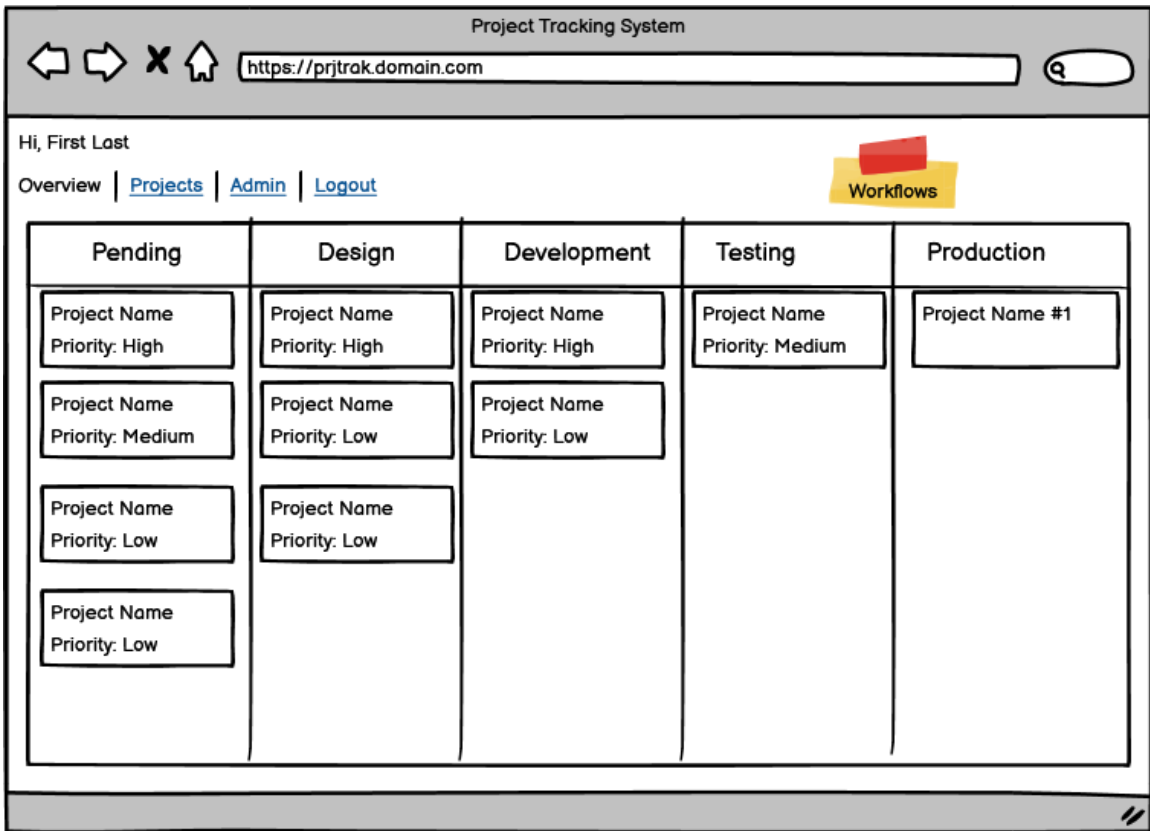


Figure 25. Web App Overview Dashboard Wireframe

Figure 26 below is an overview of a project, including sections for project description, comments, and tasks. It enables users to easily add tasks and monitor their progress through different stages, promoting teamwork and project management.

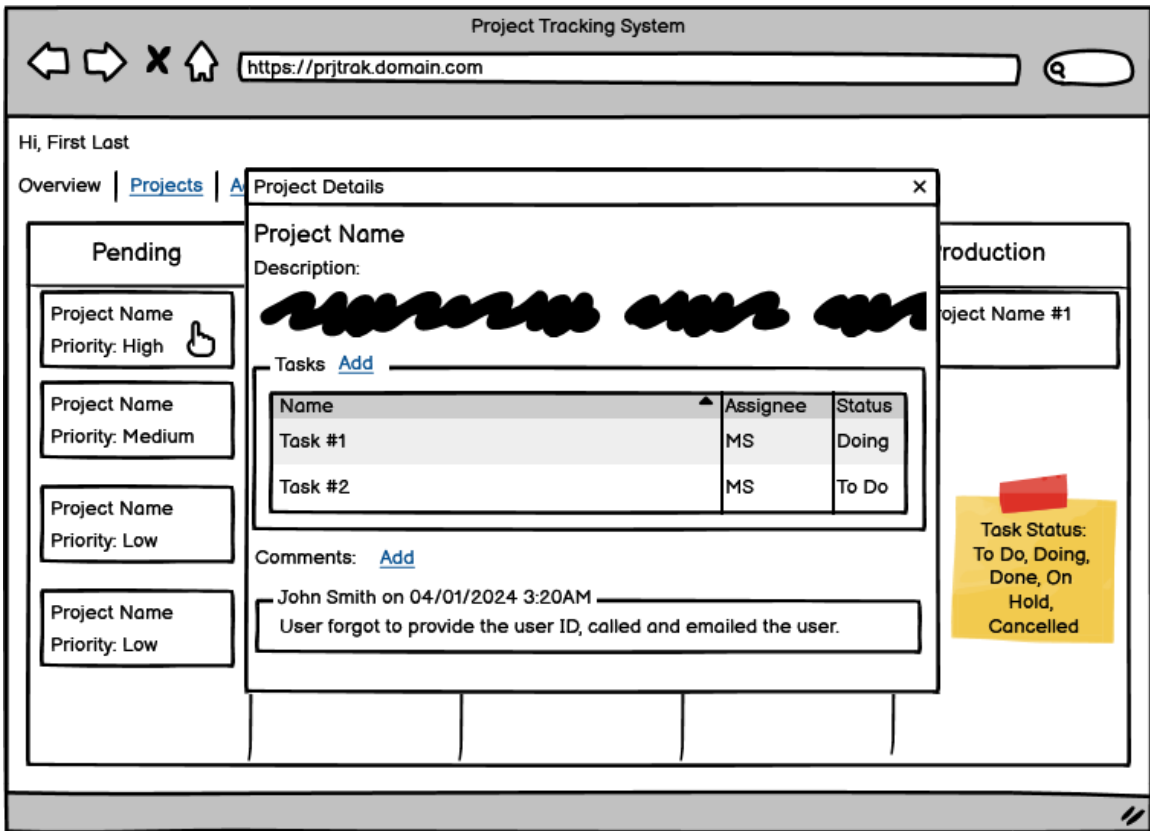


Figure 26. Web App Project Details Wireframe

Figure 27 below provides a brief summary of projects, including information such as the project name, status, assignee, active status, and etc. The purpose of this listing is to provide managers with a convenient way to evaluate the progress and urgency of the projects.

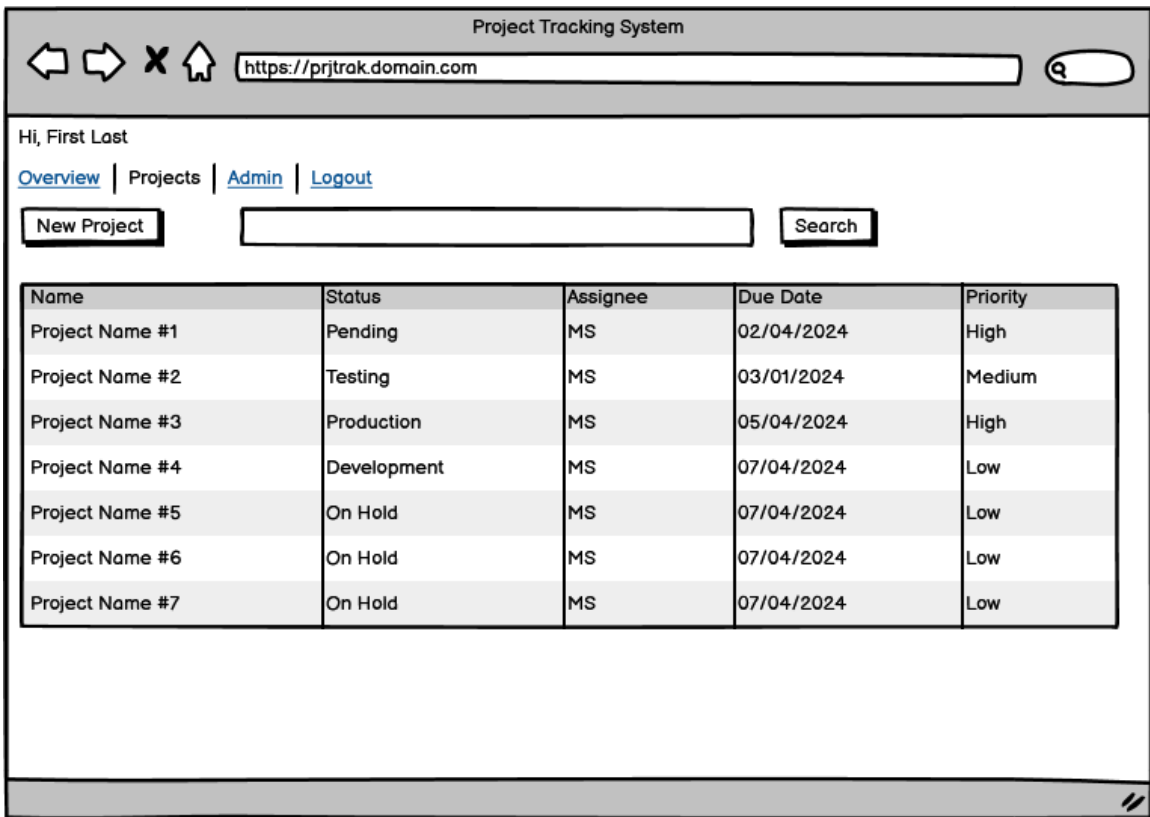


Figure 27. Web App Projects Wireframe

Figure 28 below is designed to prioritize role management by providing an architecture for defining and modifying user roles. This includes the ability to set permissions and descriptions for each role. Ensuring appropriate access to system functions is essential for the proper operation of the system.

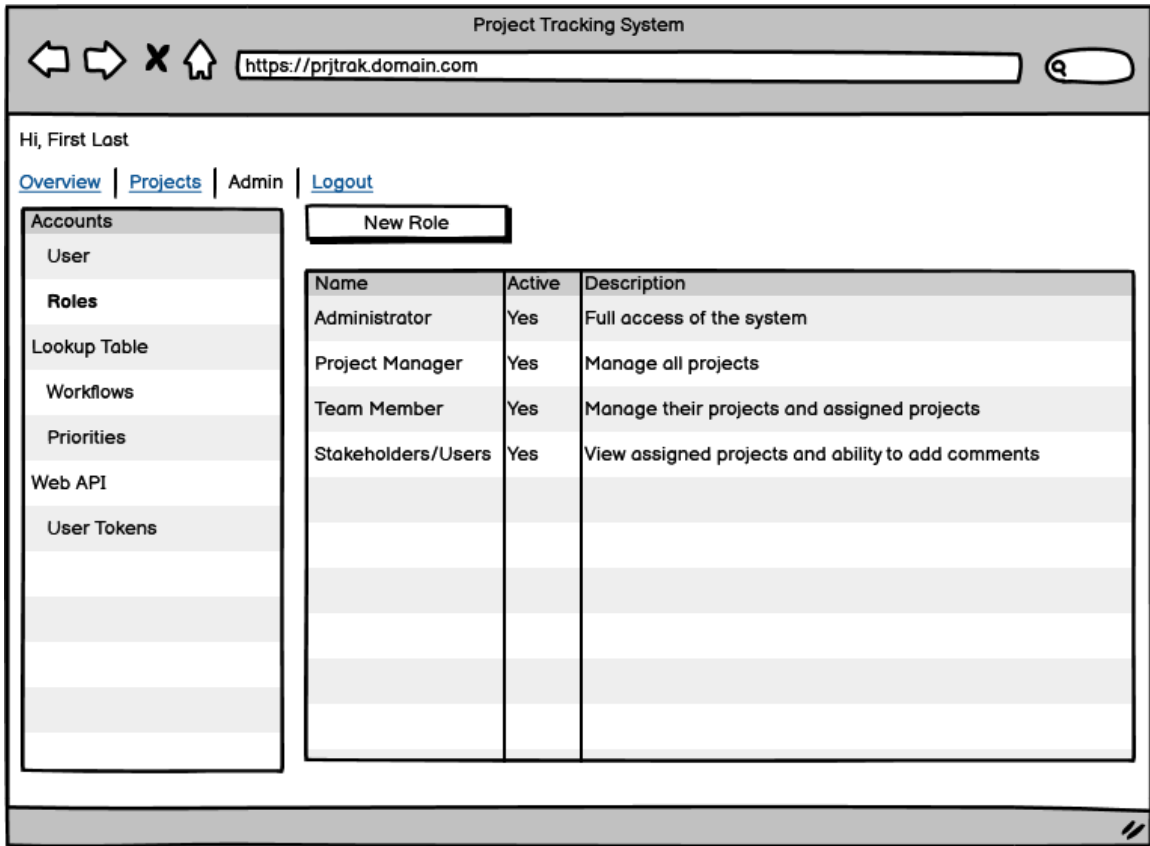


Figure 28. Web App User Role Wireframe

Figure 29 below provides functionality for managing user accounts within the system. The features of this system include the ability to add new users and edit existing user details. These functionalities are vital for managing user roles and access rights.

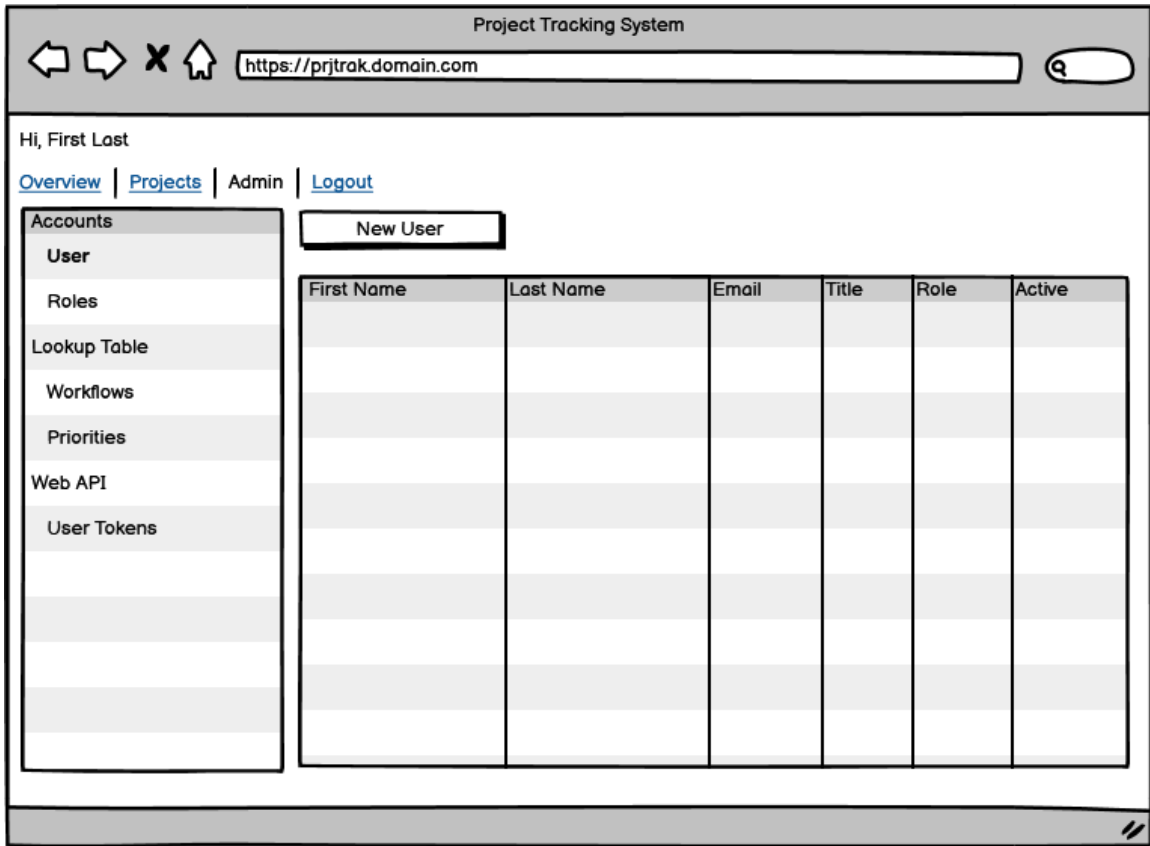


Figure 29. Web App User Wireframe

Mobile Application Wireframe

Figure 30 illustrates the login page, specifically targeted to the needs of mobile users. The user interface consists of two fields, one for email and one for password, and a button labeled “Sign In”.

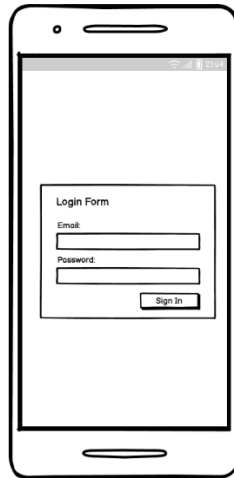


Figure 30. Mobile Login Wireframes

Figure 31 below illustrates projects that are in the 'Pending' stage on a mobile interface. Users have the ability to easily navigate through project names and descriptions. The user can navigate to another workflow by using the ">" icon.

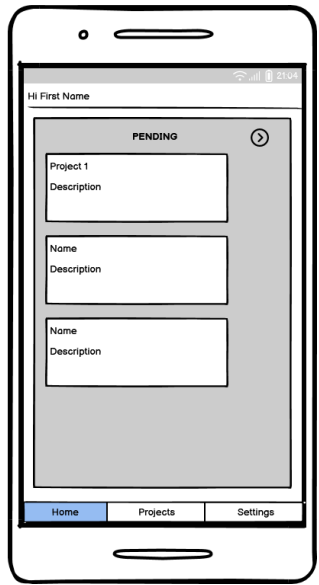


Figure 31. Mobile First Workflow Wireframe

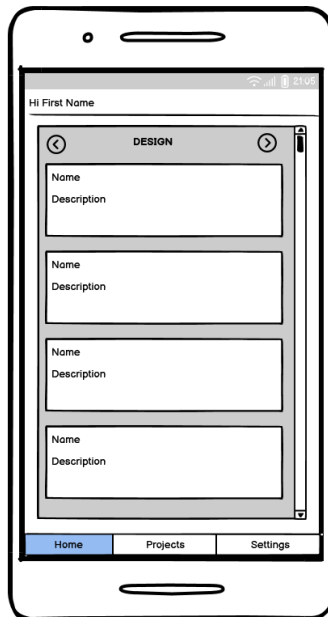


Figure 32. Mobile Mid Workflow Wireframe

Figure 32 illustrates the 'Design' stage displays a list of projects along with their names and brief descriptions. The user allows to navigate to previous or next workflow using "<" and ">" icon.

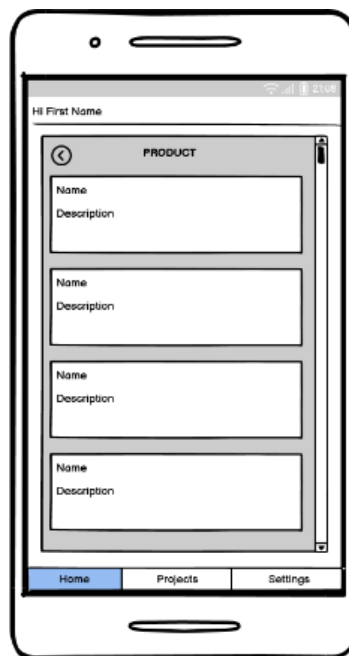


Figure 33. Mobile Last Workflow Wireframe

Figure 33 illustrates the 'Production' stage displays a list of projects along with their names and brief descriptions. This is the last workflow, the user can only navigate previous workflow using "<" icon.

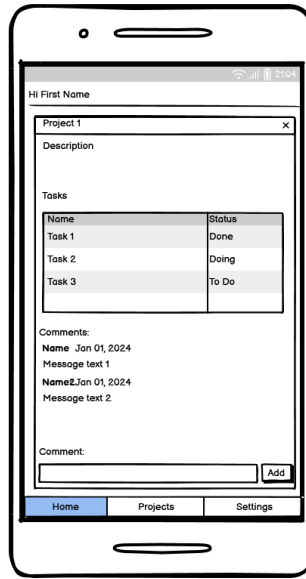


Figure 34. Mobile Project Details Wireframe

Figure 34 illustrates when the user clicks on the project to display its details and associated information. The project's comments and tasks are included.

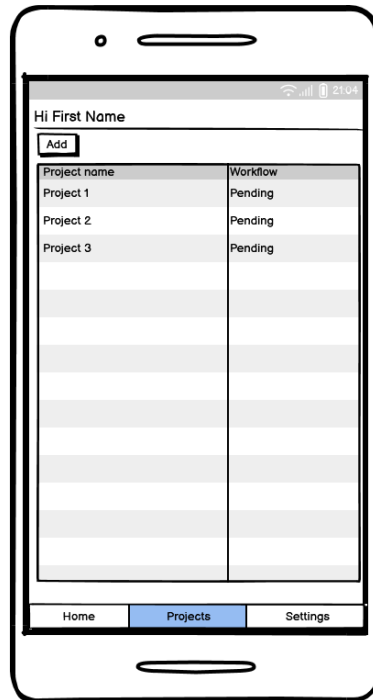


Figure 35. Mobile Projects Wireframe

Figure 35 illustrates that when a user clicks on the Projects tab, it displays all its projects. This includes active and non-active projects. The user can click the "Add" button to add more projects to the list. The user also has the ability to edit the project by clicking on it.

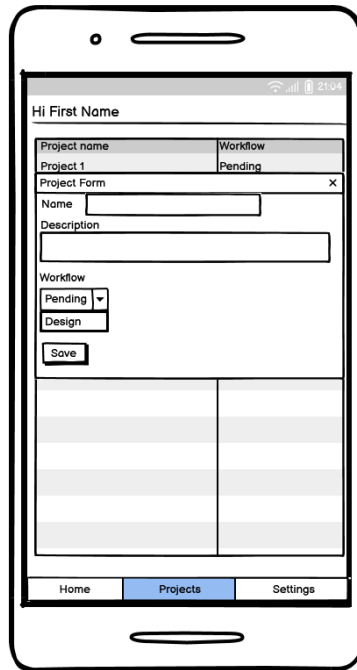


Figure 36. Mobile Project Form Wireframe

Figure 36 illustrates the user ability to add or edit the project. When the user clicks on the "Add" button, they are able to add a new project to the list. A new project form shows up, allowing the user to enter information and click "Save" to save.

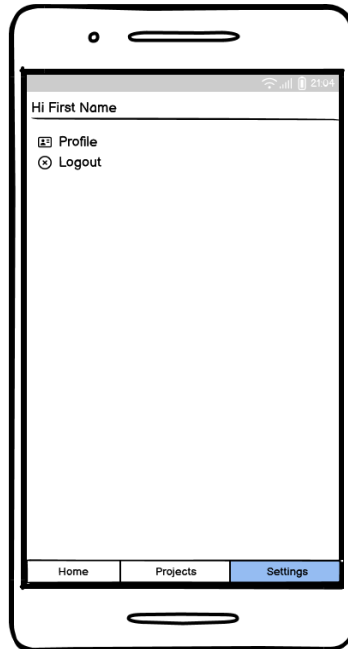


Figure 37. Mobile Settings Wireframe

Figure 37 illustrates that the user can either edit the profile or click logout. When you click "Profile," a profile form will appear. To sign out of the system, click Logout, then go back to the login form.

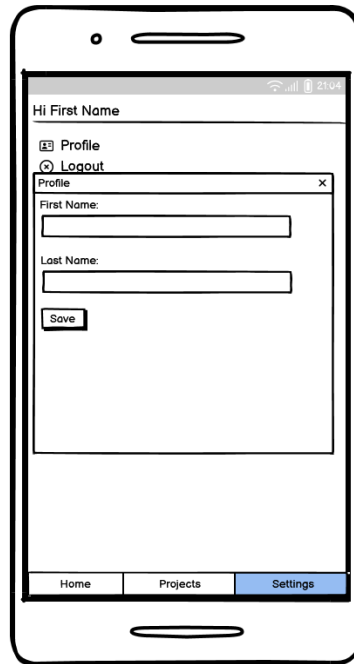


Figure 38. Mobile Profile Wireframe

Figure 38 illustrates that the user can change their first and last name, and when they click "Save," it updates their profile information.

CHAPTER FOUR

SYSTEM IMPLEMENTATION

This chapter outlines the practical implementation of a robust Project Tracking System that leverages both web and mobile platforms using MVC 5 and .NET MAUI. The system is built following a three-tier architecture [13], ensuring a clear separation between the presentation, business logic, and data access layers. This layered approach enhances maintainability and scalability while also simplifying the development process.

The Three-Tier Architecture

The presentation tier, the business logic tier, and the data access tier are the three logical and physical computing tiers that make up the three-tier architecture [8]. This is an established methodology for developing software.

Presentation Tier

This layer is where the user interacts with the app. This includes the user interface (UI) components of the project tracking system that were made with MVC 5 for the web app and .NET MAUI for the mobile app. This layer interacts to the business logic layer to carry out user requests and display the outcomes.

Business Logic Tier

This tier is the application's control center and is in charge of its main functions. It handles the data coming in and going out from the presentation

layer, uses the right business rules, and sends and receives data from and to the data access layer.

Data Access Tier

The strength of an application depends on how well it handles data. This tier interacts directly with the system and is in charge of storing and retrieving data. It hides the database processes within the business logic layer, making it easy to store and get the data you need without exposing the database details.

Advantages of a Shared Library

It is very important that the project tracking system works the same way on both web application and mobile devices. There is a shared library that holds models, utilities, and business logic that are used by both systems. This method not only keeps things from being duplicated, but it further reduces the conflicts between the two systems. Validation rules set up in a shared library make sure that the integrity of the data is kept whether it is entered on web or a mobile device.

MVC 5 and .NET MAUI

Model View Controller design, built-in Web Application Program Interface Controller, view engine, easy routing, and an extensible and pluggable framework are just some of the powerful features that MVC 5's web application for this project.

By using .NET MAUI, developers can make C# apps that work on Windows, iOS, Android, and more. In order to expedite the development of a mobile native application, I can optimize code reuse and ensure maintainability. For mobile devices, .NET MAUI makes this project tracking system possible.

Visual Studio 2022 for Development

Install Visual Studio 2022 and select the required resources for ASP.NET, web development, and mobile development with .NET, which includes .NET MAUI. Visual Studio 2022 provides powerful debugging tools, seamless integration with version control systems, and an extensive range of extensions that improve productivity. By employing these capabilities, the development process can be greatly optimized, including coding, testing, deployment, and maintenance. This environment effectively supports the implementation of the three-tier design, allowing developers to manage multiple projects within one solution, ensuring reliable and consistent performance on both platforms.

Project Solution Structure

Figure 39 below is a detailed analysis of each element in the project solution structure of the Project Tracking System:

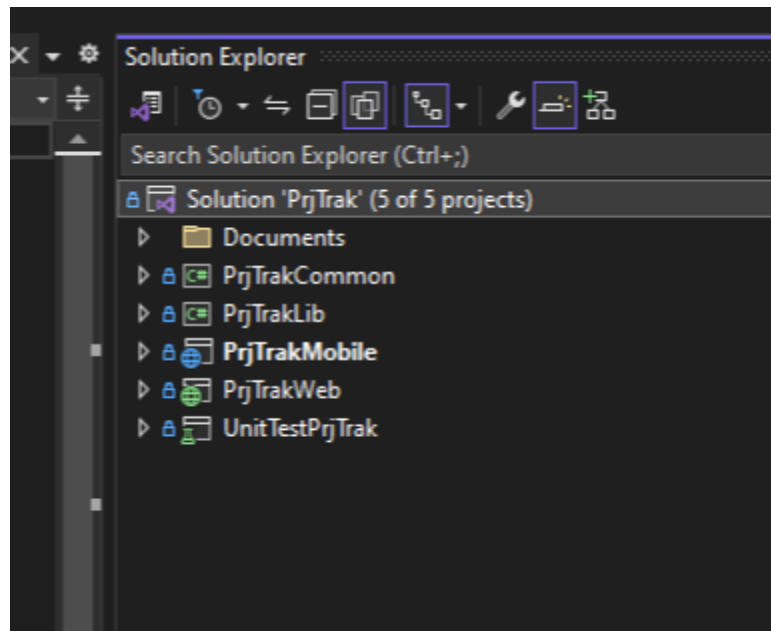


Figure 39. Project Solution Structure

Documents

This folder is necessary for organizing and accessing project documentation. User requirements, wireframes, design specifications, and other development guidance documents are included. Keeping these documents in the project structure allows team members to easily reference the information they need to maintain consistency and project standards.

PrjTrakCommon

This project serves as a web and mobile standards library. It defines Project, Task, User, Comment, Token, Workflow, etc. Centralizing these models ensures that all portions of your program use the same data model, reducing redundancy and simplifying data structure modifications.

PrjTrakLib

This library powers business logic and data layer operations in the web application and Web API. Reuse and maintenance are simplified by abstracting business rules and data processing code from user interfaces. Integration of business logic and data access in one library simplifies operations.

PrjTrakMobile

This .NET MAUI project is for mobile apps. The cross-platform mobile app uses .NET MAUI to give a native Android and iOS experience. The mobile app uses Web API services and PrjTrakCommon for data integrity.

PrjTrakWeb

This project includes both the MVC 5 web application and the Web API. MVC 5 is used to build the web frontend, focusing on rendering views, handling user inputs, and interacting with the server-side via controllers.

UnitTestPrjTrak

This project unit tests all system components. It checks the PrjTrakCommon, PrjTrakLib, Web API endpoints, and mobile-specific functions for functionality. Effective unit testing guarantees that each aspect of the application works as planned and helps identify and fix issues early in development.

Git as Source Control

For this project, I will be using Git as my source control. Git is an ideal choice for source control in any development project, even by a sole developer. There are many benefits that make it more useful and flexible, especially when you need to work on multiple computers, like a desktop and a laptop.

Version Control and Backup

Git has a strong technique for keeping track of versions. I can then keep track of changes, go back to earlier versions if something goes wrong, and see how my project has changed over time. On top of that, it works well as a backup method. The code in your Git repository is safe and can be quickly restored if your hardware fails or is damaged.

Collaboration and Scaling

Even if I am the only developer working on your project right now, it could grow and need more help in the future. It's easy to make your project bigger so that more people can work on it with Git. Other coders can join the project, pull the repository, add to it through their own branches, and use pull requests to have their work added to the main branch. As a result of this, Git is a great tool for both working alone and with a group.

Implementing Web Application

Efficiently implementing a web application depends on the strategic utilization of the MVC (Model-View-Controller) architecture. This architecture

clearly defines the roles of controllers, views, and models, making development and maintenance more streamlined. Controllers act as the middlemen between the user interface and the data model, handling user inputs, executing business logic, and selecting the appropriate views for rendering responses.

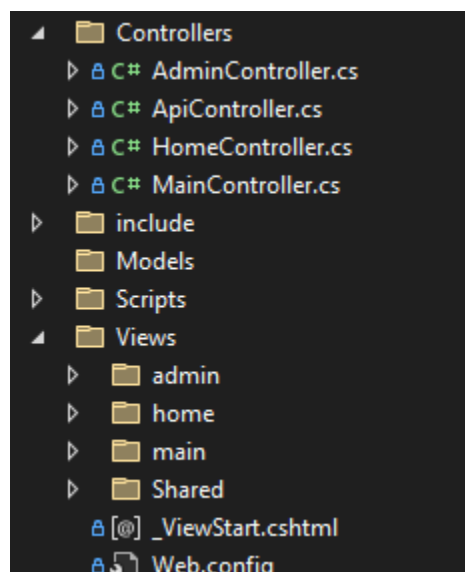


Figure 40. Web App MVC Structure

Figure 40 shows the MVC structure. The project structure is well organized, with controllers aligned to different areas of user interaction.

The AdminController is responsible for managing administrative functionalities and ensuring a secure environment for system oversight. The ApiController plays a vital role in connecting your mobile app with the backend, ensuring smooth data exchanges for mobile users. The HomeController is

responsible for handling user-specific views and interactions, ensuring a personalized user experience. Finally, the MainController is available to all users, handling general requests and serving as the starting point for unauthenticated users, displaying the public facing components.

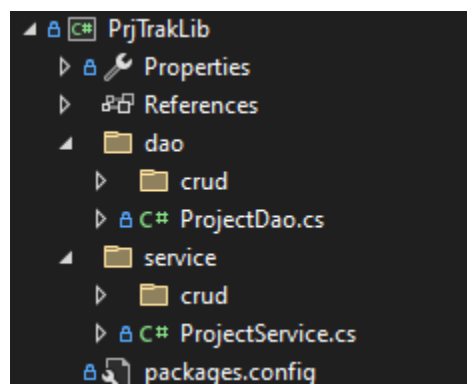


Figure 41. Business and Data Layer

Figure 41 demonstrates a strong three-tier architecture, effectively separating responsibilities into data access object (DAO) and service layers to maintain a clear separation of layers.

Through ProjectDao.cs, the DAO layer effectively handles all direct data interactions, focusing on CRUD (create, read, update, delete) operations to streamline database management and improve security measures. Furthermore, the service layer, as demonstrated by ProjectService.cs, effectively encapsulates the business logic, ensuring its independence from the data access layer. This

modular design enables the service layer to be easily reused in various applications or seamlessly adapted to new technologies, greatly enhancing the system's maintainability, and scalability.

Implementing Database Design

The database design for this project tracking system has been carefully designed using SQL Management Studio v19.1 to ensure efficient performance and reliable data management.

Figure 42 below illustrates the utilization of twelve tables: `dbo.project` for project details, `dbo.task` for task specifics, `dbo.user` for user information, `dbo.project_assignee` to link users to projects, `dbo.task_comment` for task-related discussions, `dbo.user_role` for defining user permissions, `dbo.task_status` to track task progress, `dbo.project_comment` for project discussions, `dbo.project_watcher` to notify users of updates, `dbo.user_token` for secure sessions, `dbo.workflow` to outline process steps, and `dbo.priority` to establish task urgency.

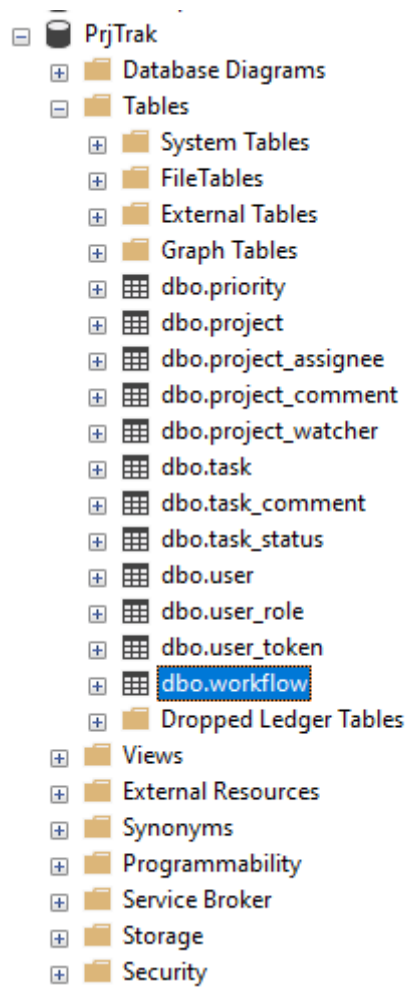


Figure 42. Database Tables

I carefully structure each table to capture and handle all relevant data.

Refer to the previous chapter for details.

Implementing Web API

This project emphasizes the Web API's importance by implementing it through the ApiController, which handles all mobile HTTP POST [7] requests.

```
0 references | Mike, 2 days ago
public class ApiController : Controller
{
    [HttpPost]
    0 references | Mike, 2 days ago
    public JsonResult Mobile(string data)
    {
        var wsResponse = new WResponse();
        wsResponse.setStatus(WStatus.Error);
        try
        {
```

Figure 43. Web API Controller

This functionality in Figure 43 is necessary for secure and efficient data transmission between the mobile app and the server. This allows mobile app interactions to be dynamic and responsive. The next chapter will explain how the ApiController facilitates these activities and its key role in the system's architecture.

Implementing Mobile Apps

Figure 44 below illustrates a modern way to use .NET MAUI Mobile, a framework that lets developers use C# to make interactive user interfaces for mobile apps that work on multiple platforms. The mobile application handles the presentation layer, which interacts with the backend services via Web API. The Web API acts as a bridge between the mobile app and the server, handling all the business logic, database operations, and data storage.

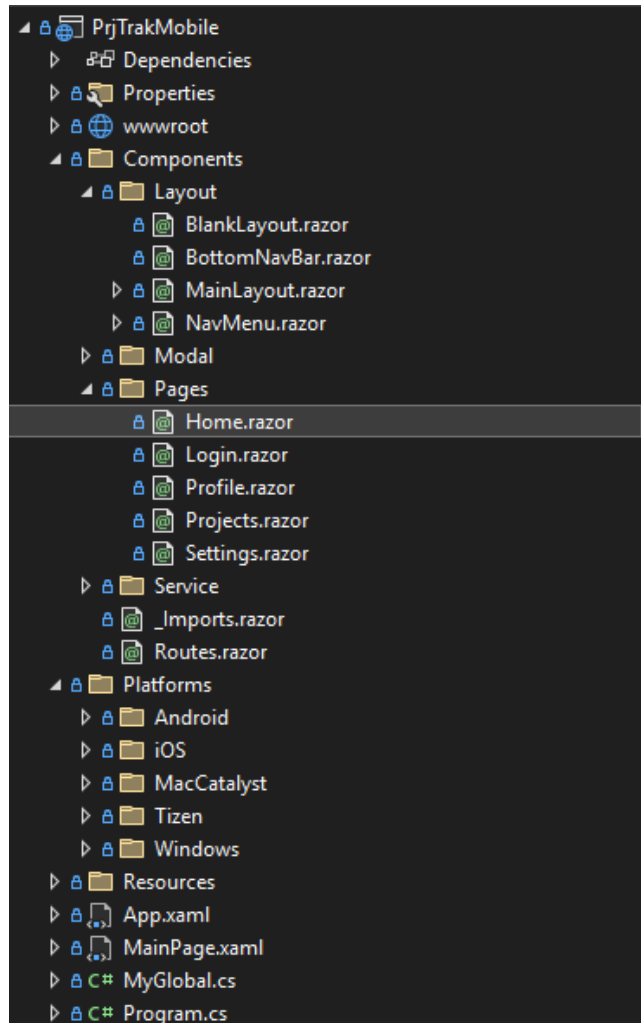


Figure 44. .NET MAUI Mobile Structure

The next chapter will cover the connection between the Web API and mobile apps.

CHAPTER FIVE

WEB API, SECURITY AND ENCRYPTION

How Apps Communicate

In this project tracking system, the Web API serves as an essential interface between the mobile applications and the server, allowing for easy communication and data management. The Web API, which uses Hypertext Transfer Protocol (HTTP) techniques, enables mobile apps to effectively submit requests and get responses. This interaction is required for tasks such as updating project statuses, adding comments, and retrieving the most recent project details, ensuring that mobile users can manage their work efficiently in real time.

When a user changes a task in their mobile app, the app makes a POST request to the Web API. The server then processes this data. Based on this input, the server may change database records and send a response back to the mobile app, either confirming the change or giving it updated data. The Web API handles this back-and-forth interaction and makes sure that the data being sent is correct and secure.

Web API Implementation in Details

I designed two custom classes, `WRequest` and `WResponse`. `WRequest` represents the standardized request format that the web app uses to communicate with the server. `WResponse` is the standard format for responses sent from the server back to the client. By standardizing the way requests and responses are handled with `WRequest` in Figure 47 and `WResponse` in Figure 48, I can ensure that the Web API remains clear and maintainable. This structure not only makes the development process more manageable but also enhances the ability to scale and extend the API as needed.

I have also implemented two enumeration types named `WAction` in Figure 45 and `WStatus` in Figure 46. This can be accomplished by creating a static class and employing a constant variable. To use its intellisense feature, I use the enumeration type, which provides me with a list of all accessible actions when I write "`WAction`" and "`WStatus`." I eventually convert these enumerated types into a string for data transmission.


```
public enum WAction
{
    Sign_In,
    Sign_Out,
    Ping,

    Get_Overview,
    Get_All_Projects,
    Get_Project_By_ID,
    Get_Project_Details_By_ID,

    Add_Project_Comment,

    Save_My_Profile,
    Save_Project,
}
```

Figure 45. Web Service Request Actions

```
public enum WStatus
{
    OK,
    Unauthorized,
    Error,
}
```

Figure 46. Web Service Response Status



Figure 47. Web Request Class Diagram

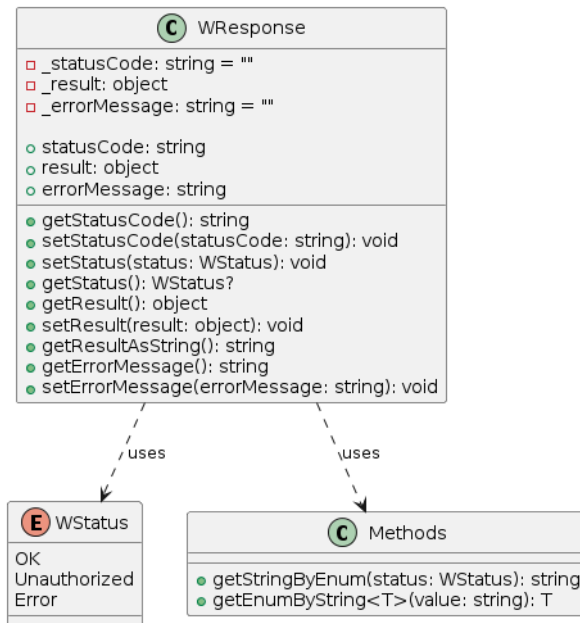


Figure 48. Web Response Class Diagram

The mobile application uses an instance of `WRequest` to request server side operations. The app would set the "**token**" to authenticate the request, "**action**" to specify what operation is to be performed, and "**args**" to provide necessary parameters like identifiers, user inputs, or any data required by the server to fulfill the request.

The Web API returns an instance of `WResponse` in Figure 49 as a result of a request made by a mobile application. The `StatusCode` of the `WResponse` would indicate the outcome of the request, such as success (OK), unauthorized access (Unauthorized), or a server error (Error). The `result` field would carry the data returned from the server if the request was successful, or could be null in case of an error. The `ErrorMessage` would provide a detailed explanation in the event of an error, helping to diagnose the issue or inform the user of what went wrong. This structure ensures that responses from the server are standardized and can be programmatically managed by the mobile application.

```
public static WResponse getWebApi(WAction action, params object[] args)
{
    var wsResponse = Methods.getWebApi(_tokenId, action, args);
    if (wsResponse.getStatus() == WStatus.Unauthorized) {
        setTokenId("");
        MyGlobal.gotoLogin();
    }
    return wsResponse;
}
```

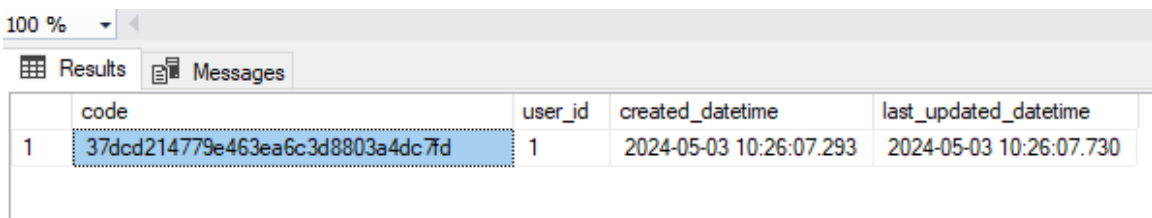
Figure 49. Web API Function Call by Mobile App

Security in Web API

Securing a Web API is important since it frequently handles sensitive data and must be secured from unwanted access and intrusions. Authentication and authorization are security mechanisms that ensure data integrity and confidentiality during transmission.

Authentication

For this specific project, I employ token generation. Each time a user successfully authenticates with the Web API, a token code is generated and stored in the database in Figure 50. The token code is returned to the user, without any additional information, as the user_token table includes the user's information.



The screenshot shows a database query results window with a zoom level of 100%. The window has two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with the following data:

| | code | user_id | created_datetime | last_updated_datetime |
|---|----------------------------------|---------|-------------------------|-------------------------|
| 1 | 37dcd214779e463ea6c3d8803a4dc7fd | 1 | 2024-05-03 10:26:07.293 | 2024-05-03 10:26:07.730 |

Figure 50. Example of User Token Data

Authorization

Once the user is authorized, the token code is sent back to the mobile app. The mobile app then stores this token code in a global variable and utilizes

it for following requests when communicating with the Web API. The authorization process verifies the validity of the token and checks if the last access occurred within the past 15 minutes. If the user's last access occurred more than 15 minutes ago, the server will remove the token and force the user to go through the login process once again. The variable "15 minutes" is set in the Web API server as a security measure. This variable can be changed. The authorization procedure verifies that the authenticated user holds the necessary permissions to carry out the requested operations prior to executing the business logic.

Hypertext Transfer Protocol Secure (HTTPS)

The mobile apps and the Web API use HTTPS [10] to secure the data. It is the act of putting information in a locked, safe box instead of a plain envelope. This private package ensures that no one can access your data during transmission over the internet.

Web API Encryption Techniques

Encrypting data before sending it over the network and decrypting it at the destination are key components of the encryption technique. I will not be able to tell you what the best technique is or how to implement it, but what I will show you is how to get it done.

In this project, I will use a commonly known base64 encoding. Figure 51 demonstrates the encoding process, which encodes the data before and decode

the data after the transaction, which makes it harder for the human to look at, which is better than a plain text string.

```
public static WResponse getWebApi(string wsUrl, string token, WAction action, int timeOutSeconds, params object[] args)
{
    var tokenId = token;
    var url = wsUrl;
    var wsResponse = new WResponse();
    var wsRequest = new WRequest();
    wsRequest.setToken(tokenId);
    wsRequest.setActionEnum(action);
    wsRequest.setArgs(args);
    wsResponse.setStatus(WStatus.Error);
    try
    {
        // {"token":"","action":"Sign_In","args":["admin@domain.com","Mike"]},
        string data = Methods.serializeToJson<WRequest>(wsRequest);
        //eyJ0b2t1b2I6IiIsImFjdGlvb2I6IiNpZ25fSW4iLCJhcmdzIjpbImFkbWluQGRRvWVpbiS5jb20iLCJNaWtLIi19
        data = Methods.getBase64String(data); // Encoding data client side
        string rawData = JsonConvert.SerializeObject(new { data = data });
        string result = getWebServiceData(url, rawData, timeOutSeconds);
        // {"result":"eyJzdGF0eXNDb2RlIjo1T0s1LCJyZXN1bHoiOiRmLyc3QgQWRtaW4iLCJ0ZXN0IHRva2VuIiw1RmFsc2UiXSwiZXJyb3JnZXNzYwdlIjo1In0="}
        string[] info = Methods.split(result, "\\"); result = info[1]; result = result.Remove(result.Length - 2);
        result = Methods.getStringByBase64(result); // Decoding data from the server side
        // {"statusCode":"OK","result":["First Admin","37dcd214779e463ea6c3d8803a4dc7fd","False"],"errorMessage":""}
        var myResult = Methods.deserializeFromJson<WResponse>(result);

        if (myResult != null) { wsResponse = myResult; }
    }
    catch (Exception ex)
    {
        wsResponse.errorMessage = ex.ToString();
    }
    return wsResponse;
}
```

Figure 51. Web API by Mobile App using Encoding Technique

To secure your data, you can implement your own encryption using existing industry cryptography methods. You can simply replace those two functions with your encryption techniques. For example, the user token can be used as a key because it generates a unique token each time a user authenticates. You can also hash the token key and use it as an encryption key. This is just a suggestion; it is obvious that you don't want to reveal the method how you generated your encryption key to others.

CHAPTER SIX

TESTING AND DEPLOYMENT

As with any software project, testing and release are very important stages. Making sure that each component works right before the system goes live is the main focus. Effective testing methods identify and resolve issues prior to software release, enhancing the reliability of the user experience.

Importance of Testing

Testing is very important because it detects problems that could impact how the system works before it goes live. Multiple testing techniques concentrate on various components of the system. I'll discuss Unit Testing, and Web API Testing.

Unit Testing

This requires testing each component or module of the application separately to verify that they are functioning correctly. Unit testing should thoroughly cover all CRUD (Create, Read, Update, Delete) operations within the data models for the project tracking system. This is important to ensure data integrity and accurate execution of business logic.

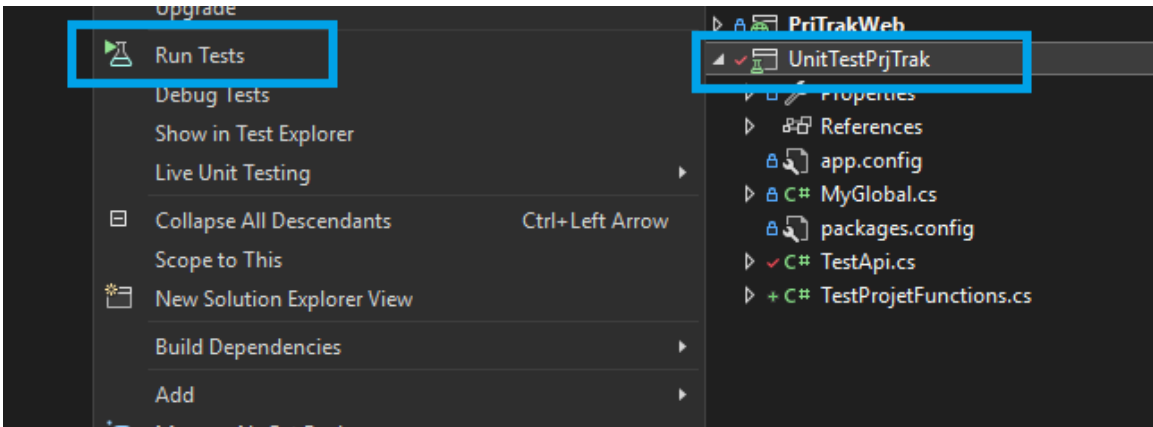


Figure 52. Unit tests within a Visual Studio project

Figure 52 demonstrates the testing process by right-clicking "UnitTestPjTrak," a context menu display. Selecting "Run Tests" runs the project's unit tests. This tool lets developers easily run all unit tests in a project for immediate feedback on code integrity and functionality in development environments.

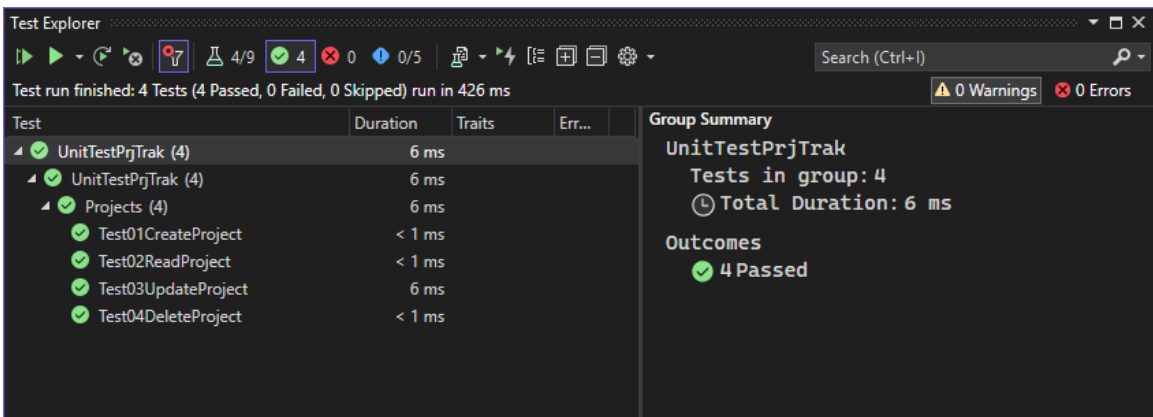


Figure 53. Project Entity Unit Test Result

Figure 53 illustrates unit tests for "Project Entity" utilizing CRUD operations are shown in the image. I executed in the order provided by their numerical prefixes (01, 02, 03, 04), ensuring creation before reading, reading before updating, and updating before deleting. The sequence of actions is as follows: Test01CreateProject, Test02ReadProject, Test03UpdateProject, and Test04DeleteProject. The Test Explorer shows that all four tests passed without problems and were executed, indicating that the Project Entity's CRUD operations are likely implemented correctly and perform well under test conditions.

It is necessary to perform this action for every entity within your system to guarantee the successful completion of all tests. Each time a modification is made to any entity, the unit tests must be repeated to verify the functionality of the new implementation. I won't go into depth about each entity in the Project Tracking System, but I will describe how to create the unit test for Project Entity.

It depends on how you write your test cases. There is no right or wrong. I intended to destroy anything created during my unit tests. Figure 54 illustrates that I declare a null object that I will use. In my case, `_project` equals null. I'll be using this object for CRUD operations.

```
private Project _project = null;
[TestMethod]
public void ProjectCRUD()
{
    CreateProject();
    ReadProject();
    UpdateProject();
    DeleteProject();
}
```

Figure 54. Unit Test Project Entity by CRUD

```
private void CreateProject()
{
    var newEntry = true;
    var project = new Project();
    project.setName("name");
    project.setDescription("description");
    project.setCreatedByUserId(MyGlobal.getProjectService().getAllActiveUsers()[0].getId());
    project.setPriorityCode(MyGlobal.getProjectService().getAllPriorities()[0].getCode());
    project.setWorkflowCode(MyGlobal.getProjectService().getActiveWorkflows()[0].getCode());
    project.setIsActive(true);
    project.setCreatedDate(DateTime.Now);
    project.setLastUpdatedDate(DateTime.Now);
    _project = MyGlobal.getProjectService().saveProject(project, newEntry);
    Assert.IsNotNull(_project);
}
```

Figure 55. Unit Test Create Project Entity

Figure 55 displays the ability to create a project object. I related the first active users with the created by user ID. Project priority code with the first active Priority entity and workflow code with the first active Workflow entity.

Assert.IsNotNull(_project): This checks that the _project variable is not empty

after saving the project. This helps make sure that the save process worked correctly.

```
1 reference
private void ReadProject()
{
    Assert.IsNotNull(_project);
    var project = MyGlobal.getProjectService().getProjectById(_project.getId());
    Assert.IsNotNull(project);
}

1 reference
private void UpdateProject()
{
    Assert.IsNotNull(_project);
    Random r = new Random(); int n = r.Next();
    var project = MyGlobal.getProjectService().getProjectById(_project.getId());
    project.setName(n.ToString());
    Assert.AreNotEqual<Project>(project, _project);
}

1 reference
private void DeleteProject()
{
    Assert.IsNotNull(_project);
    var projectId = _project.getId();
    MyGlobal.getProjectService().deleteProjectById(projectId);
    var project = MyGlobal.getProjectService().getProjectById(projectId);
    Assert.IsNull(project);
}
```

Figure 56. Unit Test Read Update Delete Project Entity

Figure 56 illustrates an Assert in each method to make sure that it works during the unit test. If any of the methods fail during the test, the whole ProjectCRUD test unit fails.

| Test | Duration | Group Summary |
|-----------------------|----------|--|
| ✔ UnitTestPrjTrak (6) | 3 sec | TestCRUD Tests in group: 1 ⌚ Total Duration: 2.1 sec Outcomes ✔ 1 Passed |
| ✔ UnitTestPrjTrak (6) | 3 sec | |
| ✔ TestCRUD (1) | 2.1 sec | |
| ✔ ProjectCRUD | 2.1 sec | |
| ✔ WebApi (5) | 916 ms | |

Figure 57. ProjectCRUD Unit Test Result

Figure 57 illustrates a typical result view from a unit test, showing test outcomes and execution times for each test and test group.

Web API Testing

Like Unit Test, Web API testing is important to make sure that web services are secure, reliable, and fast, especially when they deal with CRUD operations, which are what most web apps do. Make sure that the type and content of all the data you enter are checked. Check how the API handles invalid, unexpected, or missing data to see if it delivers correct error messages. Check how the API handles errors like those that happen on the network or database issue. Make sure that it sends back the right status codes and messages.

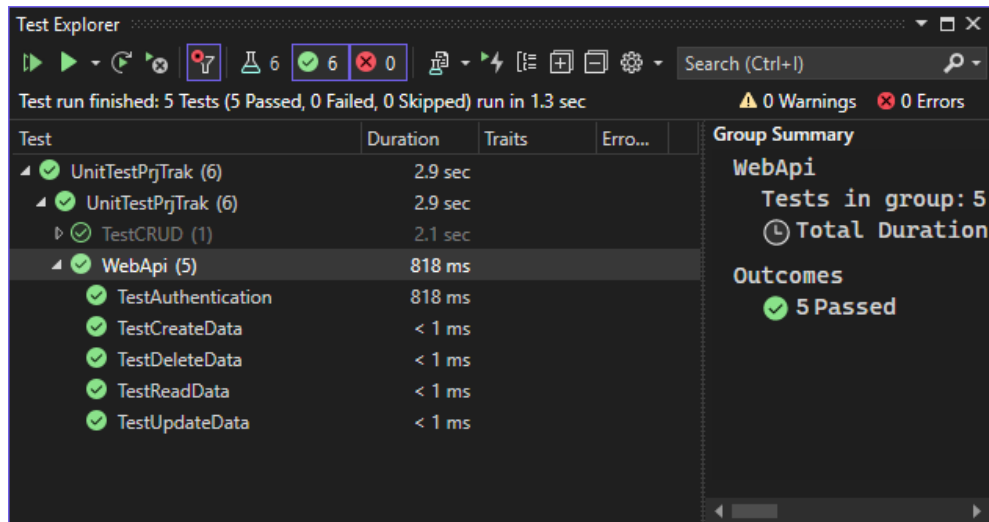


Figure 58. Web API Sample Test Result

Figure 58 illustrates a typical Web API result view from a unit test, showing test outcomes and execution times.

Deployment

The deployment phase involves the transition of the thoroughly tested system from its development environment to a production environment, enabling end users to access and use it.

Web Application Deployment

Using Web Deploy in Visual Studio, I can deploy an MVC 5 web application quickly and with minimal setup. The Microsoft Web Deployment tool makes it easy to release web apps, databases, and site content automatically from within Visual Studio. For this project, I create the project's publish profile by

entering the server information, login information, and the destination URL. After setting everything up, I simply need to right-click the project in Visual Studio and select "Publish," as illustrated in Figure 59.

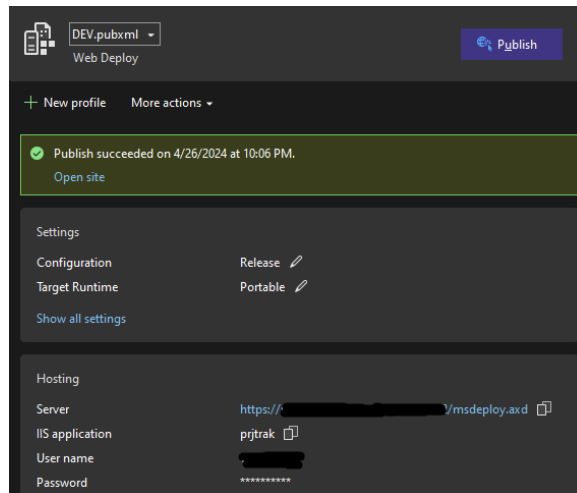


Figure 59. Web Application Deployment

Android Application Deployment

To release a .NET MAUI Android app, build it in Visual Studio and make sure the Android project is set to Release configuration in the project options. Once everything is set up, right-click and choose "Deploy." Once the app is built, look for the APK (Android Application Package) file in the project's bin\Release location as illustrated in Figure 60. Either install this APK directly onto your Android device or test it using a computer emulator.

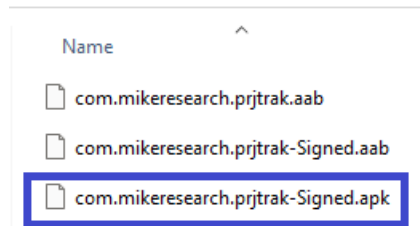


Figure 60. Android Deployment

iOS Deployment

The process of deploying a.NET MAUI iOS app is more complicated than for Android because of limitations on the iOS platform and license issue. To build this, you will need to do so on a Mac or a Mac that is linked to the internet so that you can pair it with your working computer. So that you can use the iPhone simulator, you also need to have the XCode tool on your Mac. Figure 61 demonstrates how to pair to Mac using the Tools menu in Visual Studio.

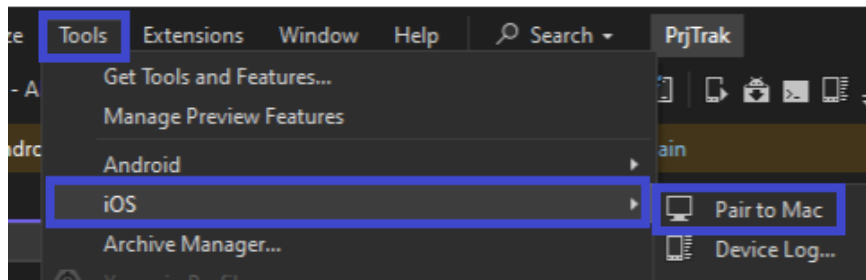


Figure 61. Pair To Mac Menu

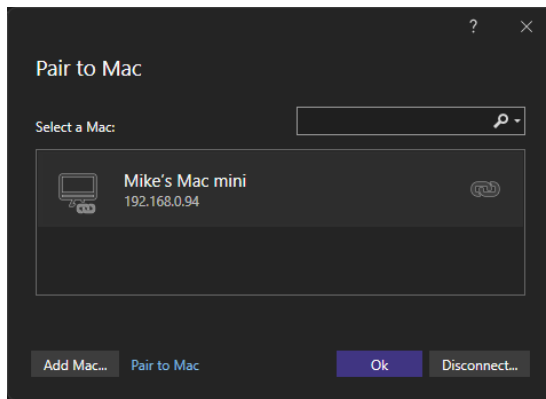


Figure 62. Pair To Mac Screen

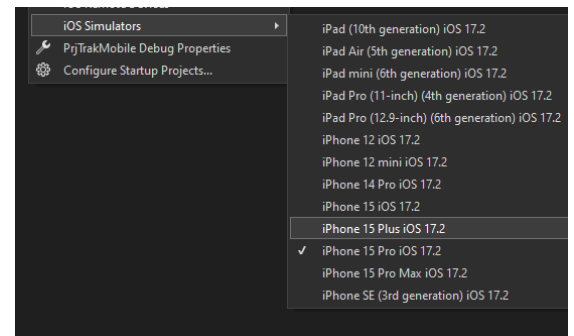


Figure 63. iOS Simulator After Pairing

Figure 62 shows the Pair to Mac dialog window in Visual Studio, which is used to pair a Windows machine with a Mac for iOS development.

Figure 63 shows the iOS Simulator selection menu in Visual Studio, which allows the user to choose a simulator for testing iOS applications.

I can use the simulator for beta testing but am not able to deploy the IPA (iOS App Store Package) file. You must have an Apple ID, enroll in the Apple Developer Program, pay a \$99 annual membership fee, and have a Mac computer to deploy or publish your application.

CHAPTER SEVEN

CONCLUSION AND FUTURE ENHANCEMENTS

Conclusion

The completion of the Project Tracking System has been a valuable learning journey, underscoring the importance of industry-standard practices and modern technologies in software development. Through this project, I successfully built and deployed a fully functional system, which includes web applications, mobile apps, and Web APIs, all integrated within a unified framework. This accomplishment demonstrates my ability to create cross-platform solutions and manage complex project architectures while following to best practices in software engineering.

The use of MVC 5 has effectively structured the web application, with the model-view-controller architecture ensuring a clear separation of layers, enhancing both maintainability and scalability. Similarly, .NET MAUI has greatly streamlined the mobile app development process by enabling the creation of cross-platform apps from a single codebase. This approach has simplified the development and management of iOS and Android applications, while ensuring consistent functionality across devices and accelerating the overall development process.

This project has strengthened my technical proficiency in various areas of software development and broadened my understanding of project

implementation. The knowledge and experience I've gained will significantly influence how I approach future software development projects.

Future Enhancements

As the Project Tracking System evolves, several enhancements can be implemented to improve functionality and increase user engagement.

Single Sign-On (SSO)

Integrating Single Sign-On (SSO) using Active Directory will simplify and secure the login process. With SSO, users can access multiple applications using a single set of credentials, which streamlines access control and enhances the overall user experience.

Attachment Support

Implementing file attachment capabilities will significantly increase the system's utility. Users will be able to share images, documents, and other relevant files directly within tasks and projects, fostering better collaboration and information sharing.

Advanced Analytics and Reporting

Integrating analytical tools like charts, graphs, and dashboards into the system to provide valuable insights on project performance, resource utilization, and employee productivity.

CHAPTER EIGHT
SCREENSHOTS OF THE SYSTEM

Web Application Screenshots

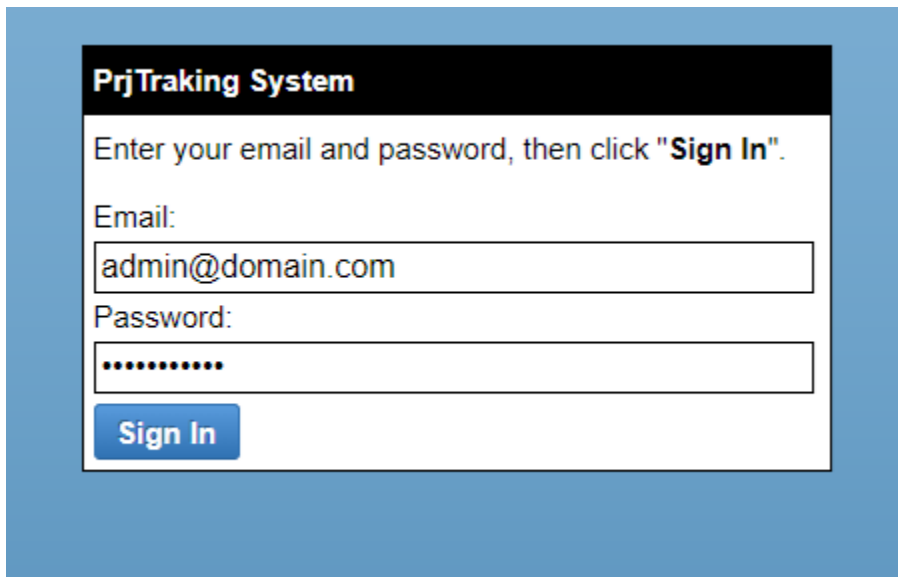


Figure 64. Web App Login

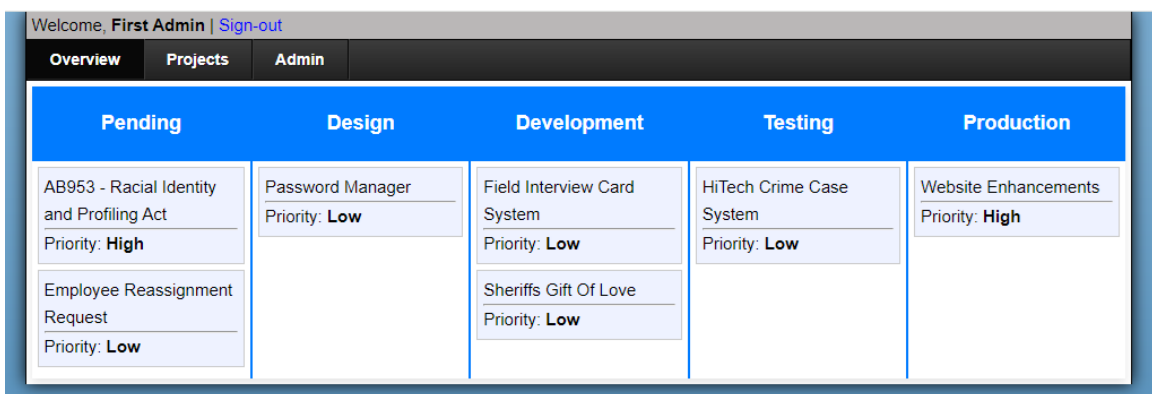


Figure 65. Web App Overview

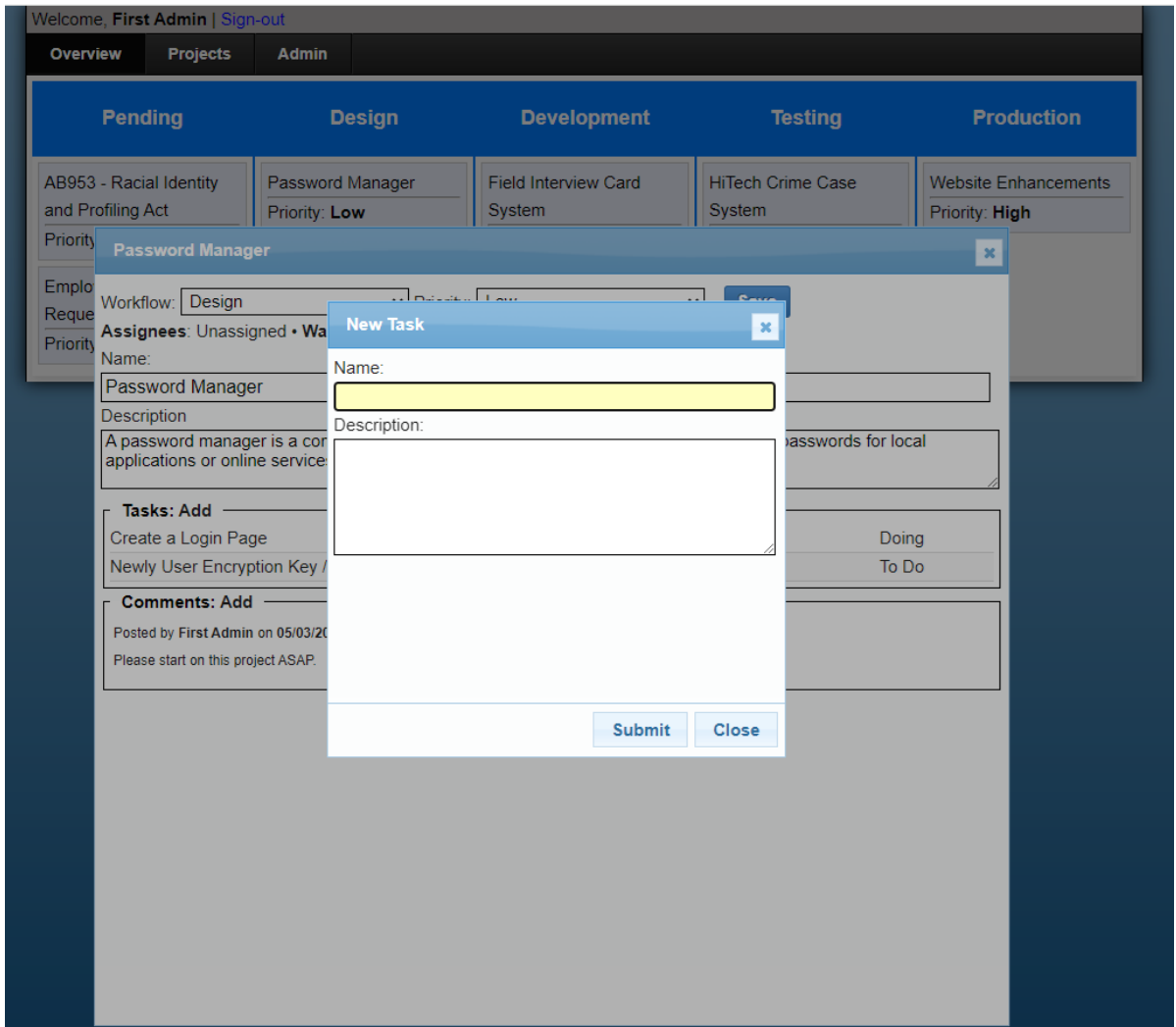


Figure 66. Web App Create New Task

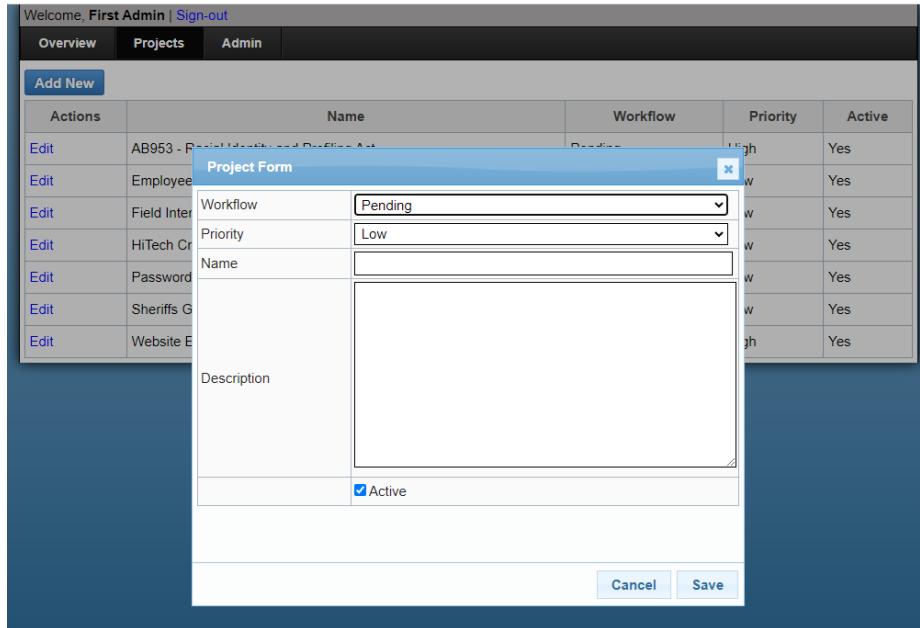


Figure 67. Web App Add Project

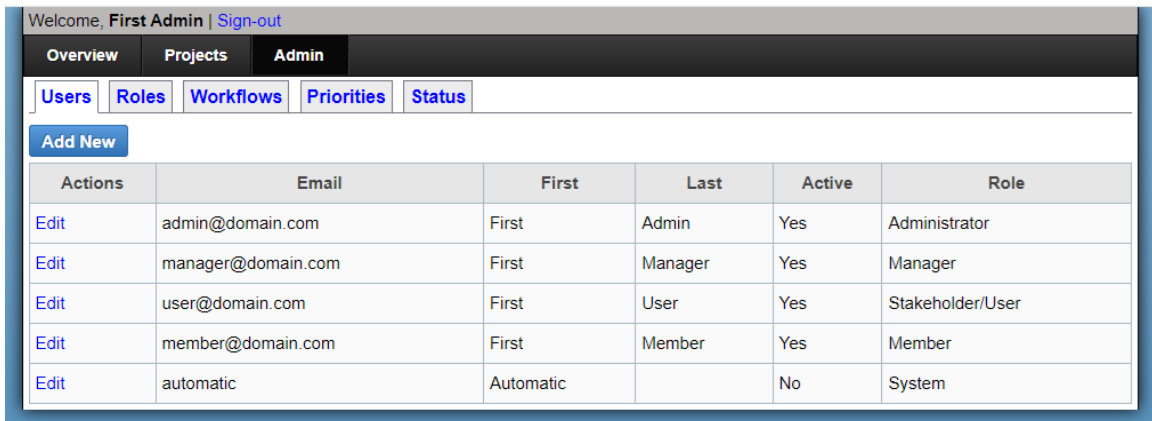


Figure 68. Web App Admin Manage Users

Welcome, **First Admin** | [Sign-out](#)

Overview **Projects** **Admin**

[Users](#) [Roles](#) [Workflows](#) [Priorities](#) [Status](#)

[Add New](#)

| Actions | Name | Active |
|----------------------|-------------|--------|
| Edit | Demo | No |
| Edit | Design | Yes |
| Edit | Development | Yes |
| Edit | Pending | Yes |
| Edit | Production | Yes |
| Edit | Testing | Yes |

Figure 69. Web App Admin Manage Workflows

Welcome, **First Admin** | [Sign-out](#)

Overview **Projects** **Admin**

[Users](#) [Roles](#) [Workflows](#) [Priorities](#) [Status](#)

| Actions | Name | Description | Full Control | Active |
|----------------------|------------------|----------------------------------|--------------|--------|
| Edit | Administrator | Full access to entire system | Yes | Yes |
| Edit | Manager | All projects | No | Yes |
| Edit | Member | Full access to assigned projects | No | Yes |
| Edit | System | System User | No | Yes |
| Edit | Stakeholder/User | See assigned projects only | No | Yes |

Figure 70. Web App Admin Manage Roles

Welcome, **First Admin** | [Sign-out](#)

Overview | **Projects** | **Admin**

[Users](#) | [Roles](#) | [Workflows](#) | [Priorities](#) | [Status](#)

[Add New](#)

| Actions | Name | Rank | Active |
|----------------------|--------|------|--------|
| Edit | High | 2 | Yes |
| Edit | Low | 0 | Yes |
| Edit | Medium | 1 | Yes |

Figure 71. Web App Admin Manage Priorities

Welcome, **First Admin** | [Sign-out](#)

Overview | **Projects** | **Admin**

[Users](#) | [Roles](#) | [Workflows](#) | [Priorities](#) | [Status](#)

[Add New](#)

| Actions | Name | Active |
|----------------------|----------|--------|
| Edit | Canceled | Yes |
| Edit | Doing | Yes |
| Edit | Done | Yes |
| Edit | On Hold | Yes |
| Edit | To Do | Yes |

Figure 72. Web App Admin Manage Task Status

iOS Simulator Screenshots – iPhone 15 Pro

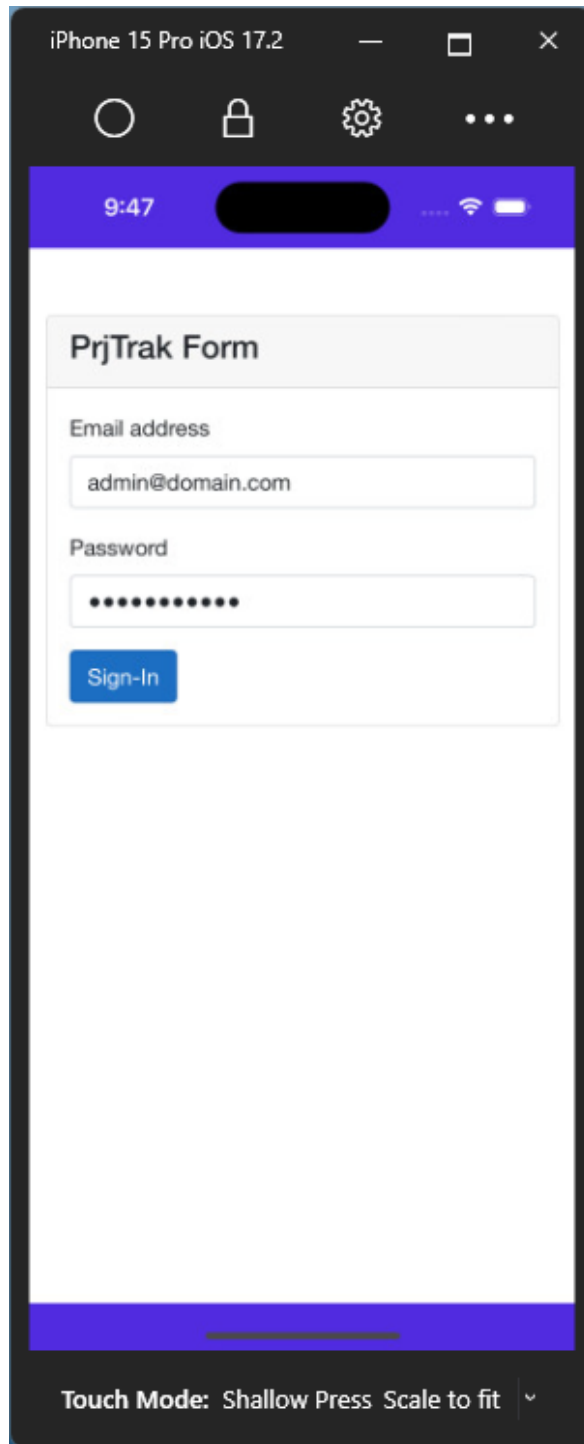


Figure 73. iOS Login Form

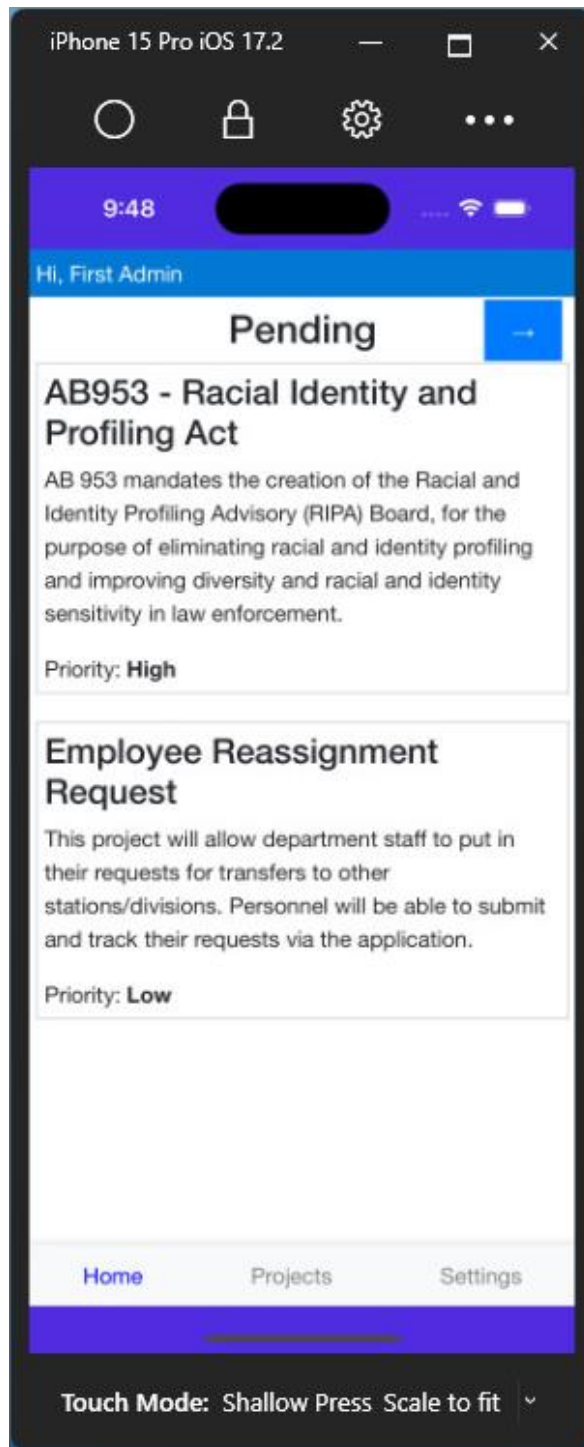


Figure 74. iOS Home Screen

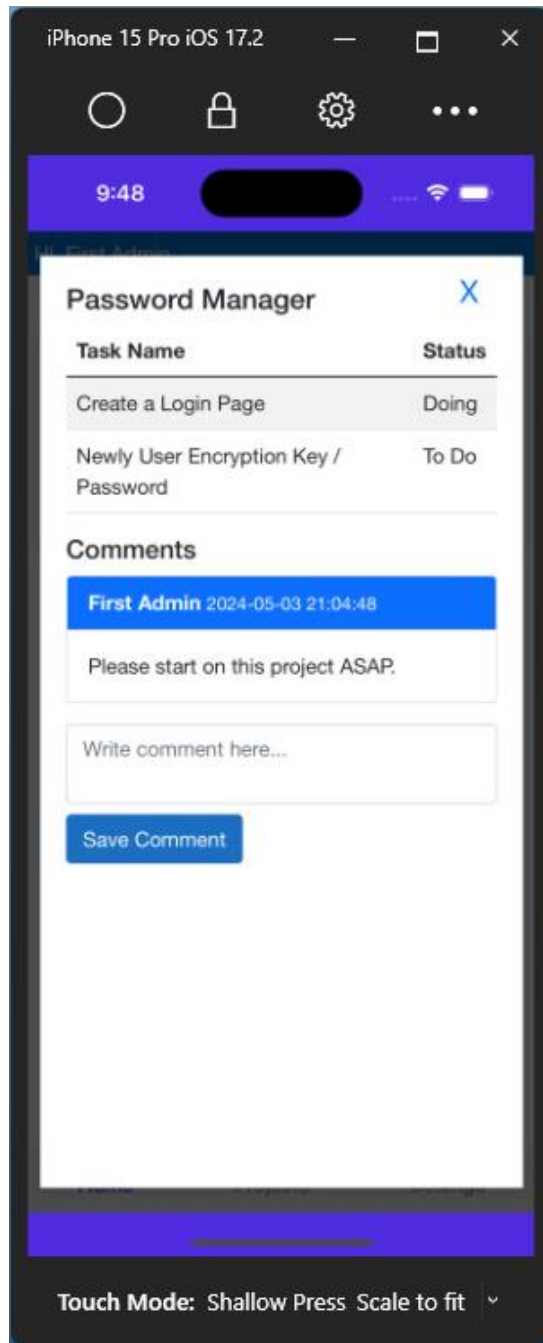


Figure 75. iOS Password Manager Project View

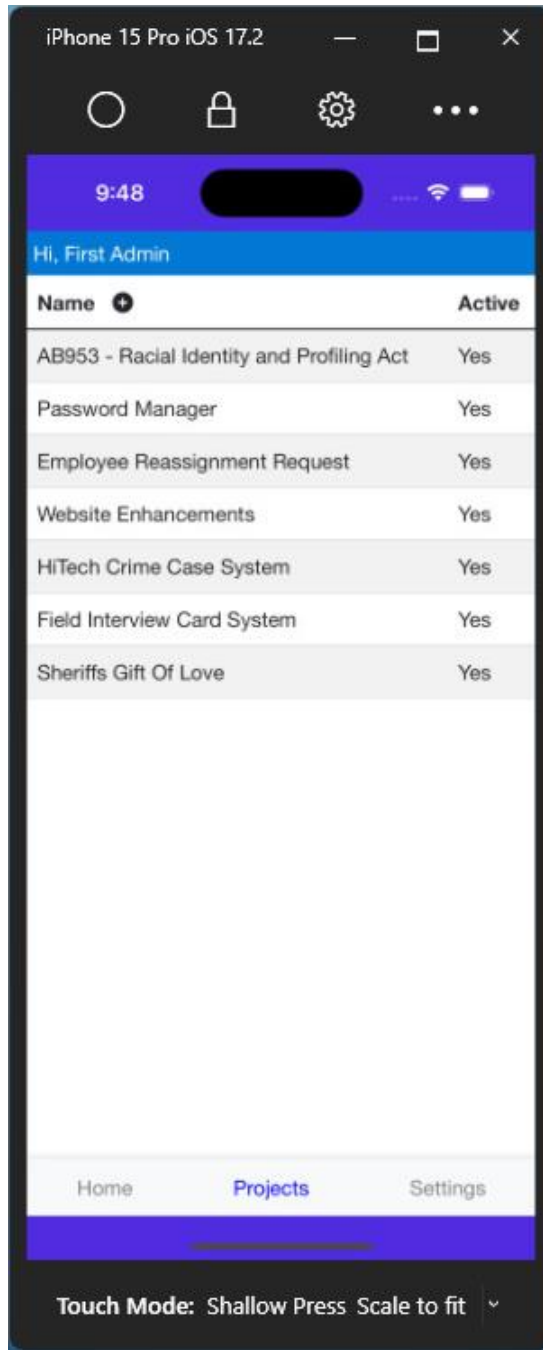


Figure 76. iOS Project List

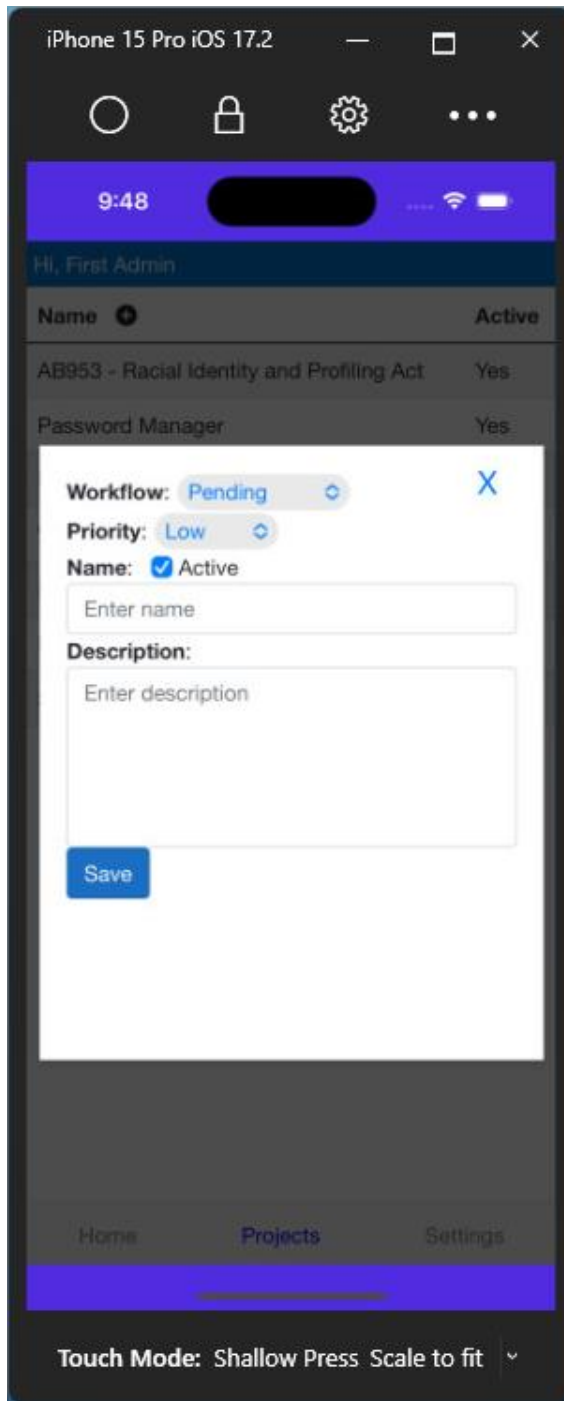
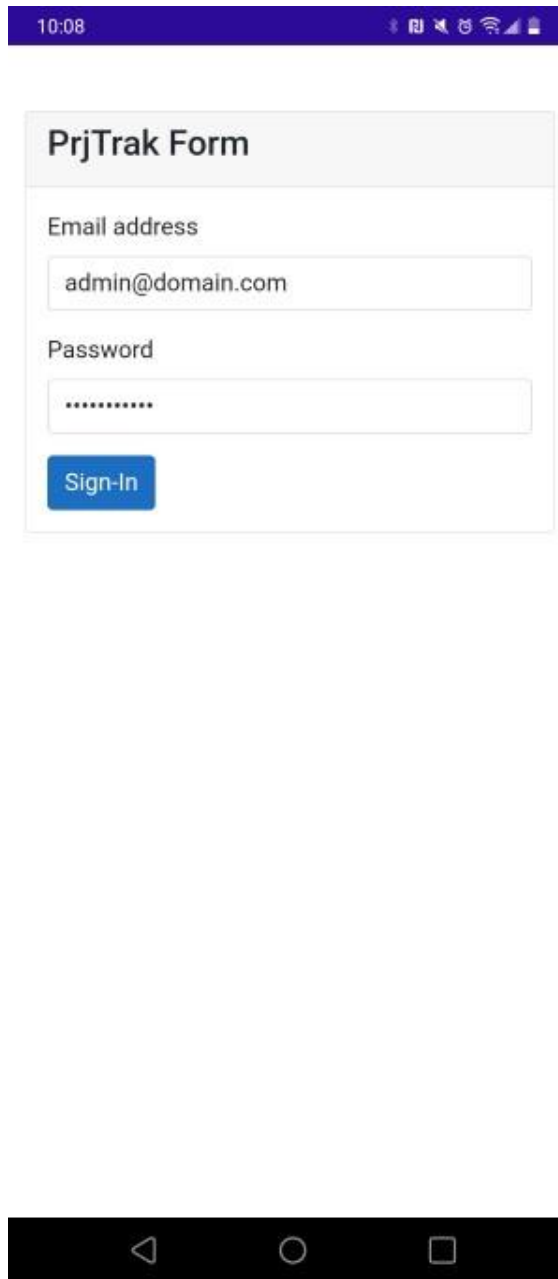


Figure 77. iOS Add Project



Figure 78. iOS Settings Screen

Android Screenshots – LG V60 Real Phone



The screenshot shows an Android mobile interface. At the top, a dark blue status bar displays the time '10:08' and various system icons. Below this is a white rectangular form titled 'PrjTrak Form'. The form contains two input fields: 'Email address' with the text 'admin@domain.com' and 'Password' with a masked password of eight dots. A blue 'Sign-In' button is positioned below the password field. At the bottom of the screen, a black navigation bar contains the standard Android back, home, and recents icons.

Figure 79. Android Login Form

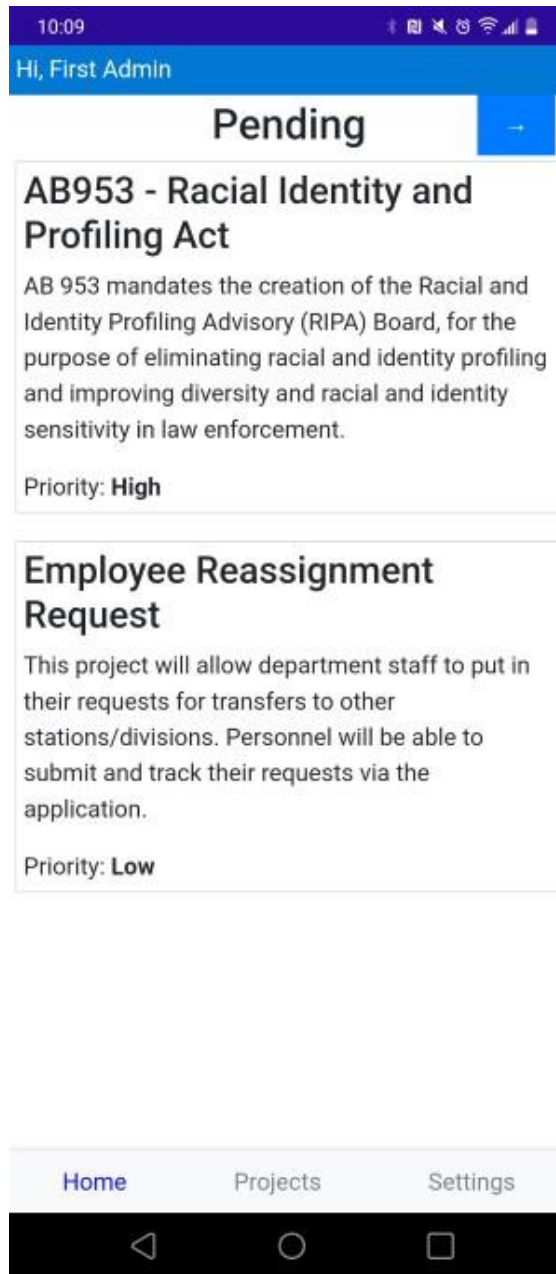


Figure 80. Android Home Screen

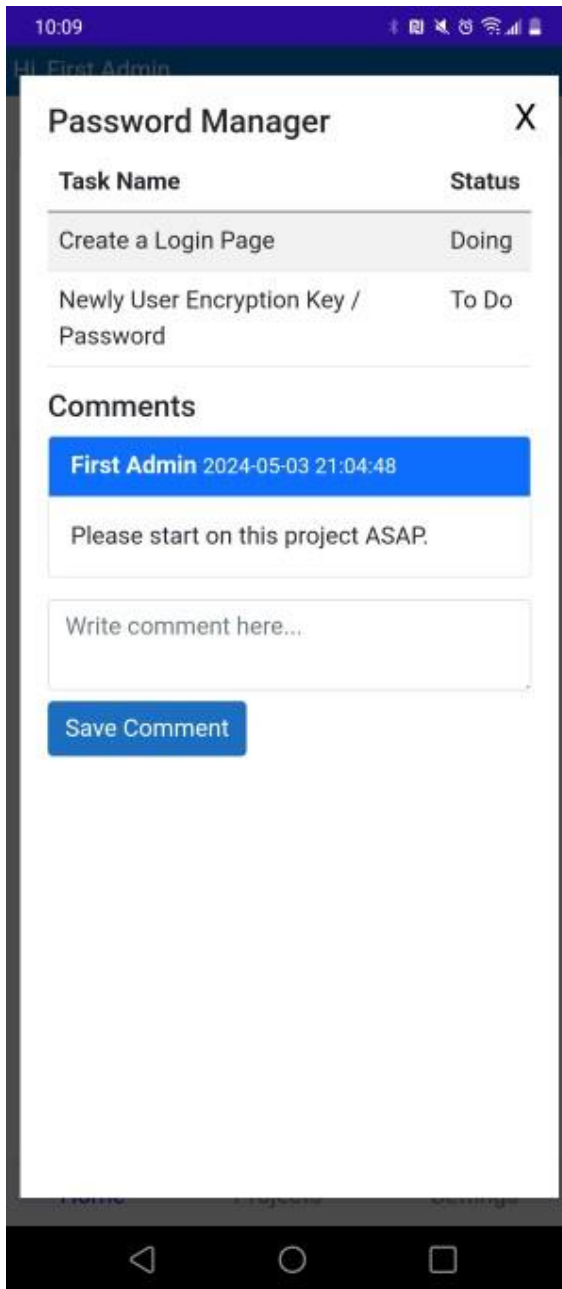


Figure 81. Android Project View



Figure 82. Android Project List

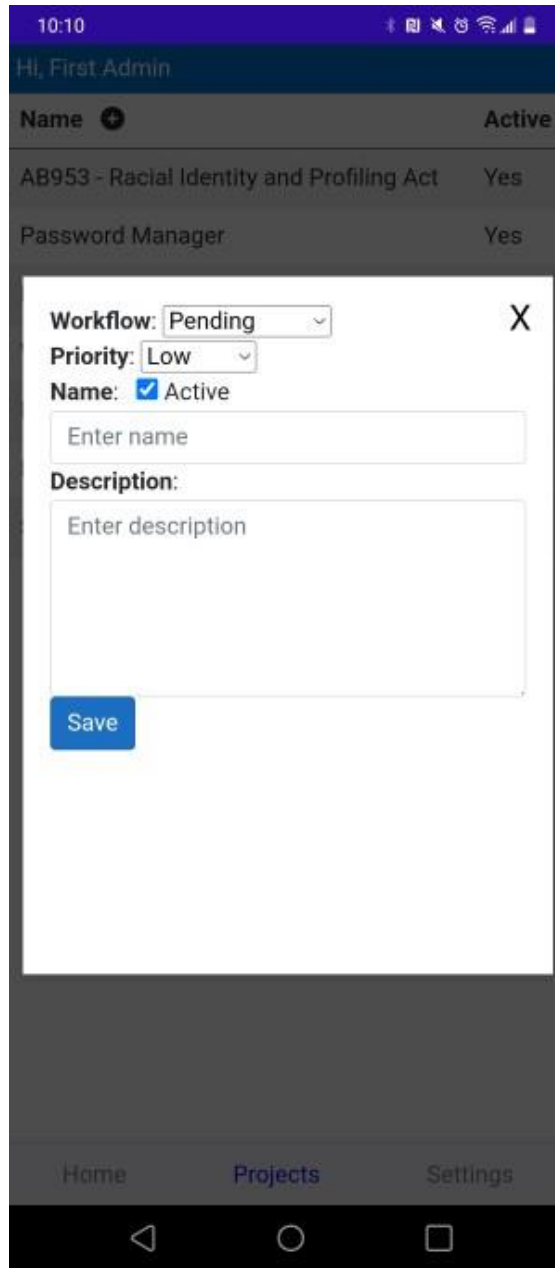


Figure 83. Android Add Project

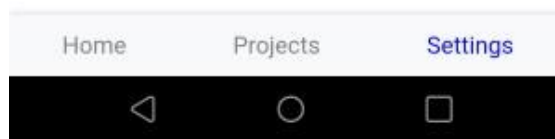


Figure 84. Android Settings Screen

REFERENCES

- [1] Atlassian, "Jira Software," [Online]. Available:
<https://www.atlassian.com/software/jira>.
- [2] Basecamp, "Basecamp: Project Management & Team Communication Software," [Online]. Available: <https://basecamp.com>.
- [3] Zoho, "Zoho Projects: Online Project Management Software," [Online]. Available: <https://www.zoho.com/projects/>.
- [4] Microsoft, "Build cross-platform native apps," [Online]. Available: <https://dotnet.microsoft.com/en-us/apps/maui>.
- [5] Microsoft, "Create mobile apps with .NET," [Online]. Available: <https://dotnet.microsoft.com/en-us/apps/mobile>.
- [6] Microsoft, "Develop RESTful APIs with ASP.NET," [Online]. Available: <https://dotnet.microsoft.com/en-us/apps/aspnet/apis>.
- [7] TutorialTeacher, "Consume Web API POST Method in ASP.NET MVC," [Online]. Available: <https://www.tutorialsteacher.com/webapi/consume-web-api-post-method-in-aspnet-mvc>.
- [8] TechTarget, "Three-tier application," [Online]. Available: <https://www.techtarget.com/searchsoftwarequality/definition/3-tier-application>.
- [9] TutorialTeacher, "Implement POST Method in Web API," [Online]. Available: <https://www.tutorialsteacher.com/webapi/implement-post-method-in-web-api>.
- [10] Microsoft, "Working with SSL in Web API," [Online]. Available: <https://learn.microsoft.com/en-us/aspnet/web-api/overview/security/working-with->

[ssl-in-web-api](#).

[11] Microsoft, "Getting Started with ASP.NET MVC 5," [Online]. Available:

<https://learn.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/getting-started>.

[12] Microsoft, "ASP.NET MVC Controllers Overview (C#)," [Online]. Available:

<https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/aspnet-mvc-controllers-overview-cs>.

[13] IT Path Solutions, "Single Codebase App Development Trends in 2023,"

[Online]. Available: <https://www.itpathsolutions.com/single-codebase-app-development-trends-in-2023/>.

[14] Microsoft, "Cross-Platform Mobile Development in Visual Studio," [Online].

Available: <https://learn.microsoft.com/en-us/visualstudio/cross-platform/cross-platform-mobile-development-in-visual-studio?view=vs-2022>.

[15] Uno Platform, "The Rise of C# Markup for Cross-Platform Development,"

[Online]. Available: <https://platform.uno/blog/the-rise-of-c-markup-for-cross-platform-development/>.

[16] Amazon Web Services, "Three-Tier Architecture Overview," [Online].

Available: <https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/three-tier-architecture-overview.html>.