June 2024

COMPREHENSIVE PROJECT AND REMOTE WORK MANAGEMENT SYSTEM

Prepared for

University of Sri Jayewardenepura, Faculty of Technology

Presented by

Group 21

COMPREHENSIVE PROJECT AND REMOTE WORK MANAGEMENT SYSTEM

Project Personnel

Academic Supervisor

Ms. Vasana Sahabandu

University of Sri Jayewardenepura

vasanasahabandu@sjp.ac.lk

+94 71 533 0732

Team Members

Udara Nilupul

Software Technology

Faculty of Technology, University of Sri Jayewardenepura

ICT/21/899

ict21899@fot.sjp.ac.lk

+94 76 351 4412

Navod Sandaruwan

Networking Technology

Faculty of Technology, University of Sri Jayewardenepura

ICT/21/914

ict21914@fot.sjp.ac.lk

+94 77 849 5892

Chamika Santhush

Networking Technology

Faculty of Technology, University of Sri Jayewardenepura

ICT/21/915

ict21915@fot.sjp.ac.lk

+94 71 454 2798

Vimukthi Waruna

Networking Technology

Faculty of Technology, University of Sri Jayewardenepura

ICT/21/952

ict21952@fot.sjp.ac.lk

+94 77 491 0856

Table of Contents

INTRODUCTION

Overview

A comprehensive project and remote work management system is a modern solution intended to solve the complexities that arise in managing software development projects. This aims to streamline and enhance the management of software development projects, particularly for teams that operate remotely or are distributed across various locations.

As remote work becomes more prevalent, the need for a unified platform that supports task management, collaboration, and productivity tracking has become critical. The system integrates various functionalities into a single platform including task tracking, milestone management, team collaboration, time tracking, virtual meetings and productivity analysis.

By providing a centralized platform, the system is designed to improve communication, coordination and efficiency among team members involved in the software development project, thereby enabling better communication and improved project outcomes.

(Project manager, Product Owner, Business Analyst, Software Architect, Team Lead, Developers/Programmers, UX/UI Designers, Quality Assurance, Testers)

Purpose

The purpose of this project is to address the challenges faced by software development teams in managing projects and remote work effectively. With the increasing trend of remote work, there is a need for robust tools that can facilitate seamless collaboration, ensure accountability, and provide comprehensive oversight of project progress. We hope to provide these requirements through this system. By developing this system, we aim to,

I. Enhance Team Efficiency - Offer tools for effective task and time management to ensure that team members can work efficiently, even when geographically dispersed.
II. Improve Collaboration - Facilitate seamless communication and collaboration through integrated features like virtual meetings and team messaging, helping teams stay connected and coordinated.
III. Enabling Better Project Oversight - Provide project managers with real-time tracking of project milestones and individual productivity, allowing for more informed decision-making and proactive issue resolution.
IV. Supporting team members who work in different locations. - Create an environment where distributed teams can maintain high levels of productivity and engagement, ensuring that physical distance does not hinder project success.

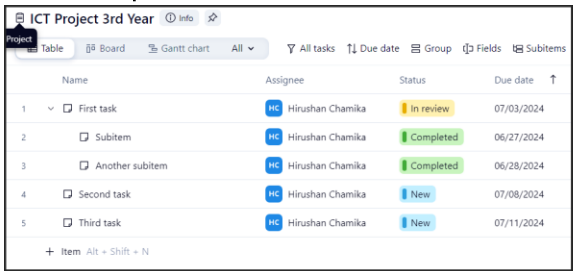
By addressing these key areas, the system aims to significantly improve the management of software development projects and enhance the overall remote work experience.

Project Scope

The scope of this project involves developing a comprehensive Project and Remote Work Management System. The system will integrate various functionalities to support project management and remote work, catering to software development teams. The main features include user Interface (UI), user authentication, user management, project management, task tracking, virtual meetings, document management, chat functionality, and client access to view project progress. The system will be developed using the MERN stack, with additional integrations for video conferencing and authentication.

Feature Comparison with Trello

Core Features	Comprehensive Project and Remote Work Management System	Trello
1.User Account Management	➢ Both systems offer secure user authentication and authorization. ➢ Real-Time Project Management Application includes email verification and role-based access control.	➢ Trello provides integration options for account creation with Google/ Microsoft integration and password recovery.
2.Client Access	➢ Client access to view project Progress. ➢ Chat With team member.	➢ **Client access:** No direct feature, can share boards ➢ **Data visualization:** Via Power-Ups like Charts by Viz drop.

3.Task Management	➢ Create tasks as cards. ➢ Lists to organize tasks into different stages. ➢ Labels, due dates, and checklists within cards. For example:  ICT Project 3rd Year · Info Project · Table · Board · Gantt chart · All · All tasks · Due date · Group · Fields · Subitems Name · Assignee · Status · Due date 1 First task · HC Hirushan Chamika · In review · 07/03/2024 2 Subitem · HC Hirushan Chamika · Completed · 06/27/2024 3 Another subitem · HC Hirushan Chamika · Completed · 06/28/2024 4 Second task · HC Hirushan Chamika · New · 07/08/2024 5 Third task · HC Hirushan Chamika · New · 07/11/2024 + Item Alt + Shift + N	➢ Create tasks as cards. ➢ Lists to organize tasks into different stages.
4.Project Dashboard	➢ **Sidebar with software development phases** ➢ **Navigation bar for functionalities like team, documentation, chat, and meetings**	➢ Sidebar: No ➢ Navigation bar: No, uses boards and lists to organize these functions
5.Team Collaboration	➢ Both platforms support team collaboration through comments, document sharing, and integrations. ➢ Comments and mentions on cards. ➢ Real-Time Project Management Application includes built-in chat and meeting features.	➢ Trello integrates with tools like Slack for enhanced collaboration.
6.User Roles & Permissions	➢ Advanced roles and permissions management	➢ Basic roles and permissions, mainly board-based
7.Task Tracking and Assignment	➢ Real-Time Project Management Application includes basic features with additional role-based assignment.	➢ Trello manages task tracking through cards, assignments, and checklists and integrations like Jira.

8.Notifications and Alerts	➢ Real-time notifications and alerts for task updates, comments, and project progress Notifications for task updates, comments, and integrations.	➢ Notifications for task updates, comments, and integrations
9.Feedback and Issue Tracking	➢ Issue tracking with roles for QA Testers and developers, feedback loops.	➢ Issue tracking via task cards, labels, comments, and integrations like Jira.
10.Implementation Process	➢ The Real-Time Project Management Application has a structured implementation process coordinated by the Team Lead, involving collaboration and review	➢ Trello handles implementation through task cards and assignments without a defined process coordinator.

REQUIREMENT GATHERING

Methodology

- For the requirement gathering of the Comprehensive Project and Remote Work Management System, we employed the following software engineering techniques,

Brainstorming Sessions

- Conducted regular brainstorming sessions to discuss and generate ideas for both functional and non-functional requirements.
- Scheduled regular follow-up sessions to refine and prioritize requirements.
- We used online collaborative tools and mind-mapping software.

Surveys/Questionnaires

- Designed and distributed surveys to gather structured input from members and potential users (industry personnel).
- Then analyzed the survey responses to identify key requirements and preferences.

Prototyping

- Developed low-fidelity prototypes to visualize potential solutions and gather feedback.
- Iterated on designs based on feedback to refine and validate requirements.
    - Shared initial designs with team members and some potential users for feedback.
    - Made necessary adjustments based on the feedback received.
    - Repeated the cycle until the design met user needs and expectations.

User Stories

- Created initial user stories to capture requirements from the perspective of end-users.
- These user stories are then organized, prioritized and continuously improved to ensure they are ready for development. That is, Managed and refined the user stories in a product backlog.
- Example User Story: "As a team member, I want to log my hours so that I can track my productivity."

This comprehensive approach ensures that all relevant requirements are gathered, refined, and validated effectively, setting a solid foundation for the system's development.

SYSTEM REQUIREMENTS

Functional Requirements

Functional Requirement 1 - User Registration

- Users can create an account by providing their personal details (name, email address, and password).
- The system validates inputs, checks for existing accounts, hashes the password, generates a verification code, and stores user details in the database.
- Outputs - Confirmation message or error message.
- Preconditions - The user must not already have an account with the provided email address.
- Postconditions - A new user account is created, and the user is directed to the login page.

Functional Requirement 2 - User Login

- Registered users can log in using their email address and password.
- The system validates inputs, retrieves stored password hash, and compares it with the provided password.
- Outputs - Successful login message and redirection to the dashboard or error message.
- Preconditions - The user must have a registered account.
- Postconditions - The user is logged into the system and redirected to their dashboard.

Functional Requirement 3 - Project Creation

- Project Managers can create a new project by providing the project name and allocating team members.
- The system validates team members' email addresses, checks for existing accounts, and creates the project.
- Outputs - Confirmation message or error message.

- Preconditions - The Project Manager must add at least three members (including themselves).
- Postconditions - A new project is created.

Functional Requirement 4 - Project Requirements Adding/Allocating

- Business Analysts can add new project requirements and share them with the Product Owner through documents.
- The system validates the entered requirements, uploads the documents, and notifies the Product Owner.
- Outputs - Confirmation message or error message.
- Preconditions - The Business Analyst and Product Owner must be assigned to the project.
- Postconditions - New requirements are added to the project's requirements page, and the Product Owner is notified.

Functional Requirement 5 - Adding Reviews and Prioritizes Project Requirements

- Product Owners can review and prioritize project requirements provided by the Business Analyst and share them with team members through documents.
- The system allows the Product Owner to download, review, and upload prioritized requirements, validates the data, and notifies team members.
- Outputs - Confirmation message or error message.
- Preconditions - The Product Owner must be assigned to the project, team members must be assigned, and new requirements must have been added by the Business Analyst.
- Postconditions - Prioritized requirements are added to the project's requirements page, and team members are notified.

Functional Requirement 6 - Requirements Clarification (Review the prioritized requirements)

- Team members can review prioritized project requirements shared by the Product Owner and request clarifications.
- The system allows downloading the prioritized requirements document, asking questions, uploading these questions, and notifying relevant parties.
- Outputs - Confirmation message or error message.
- Preconditions - The Product Owner must have shared the prioritized requirements, and team members must have access to the project’s requirements page.
- Postconditions - Questions are added to the project's requirements page, and the Business Analyst, Product Owner, and Team Lead are notified.

Functional Requirement 7 – User Interface Adding

- UI/UX Designers can upload their completed design to the project's Design page and share it with the Team Lead.
- The system allows uploading design files, validates the format and data, and notifies the Team Lead.
- Outputs - Confirmation message indicating successful addition or error message indicating any issues.
- Preconditions - Problems must be classified, requirements specified, and Designers must have access to the project’s Design page.
- Postconditions - The new design is added to the Design page, and the Team Lead is notified.

Functional Requirement 8 – System Development Initiation

- Developers can start the system development using GitHub after the system design is approved.
- The system allows linking a central GitHub repository, tracking development progress, and displaying it on the Project Implementation page.
- Outputs - Confirmation message, link to the central GitHub repository, and ongoing updates on development progress.
- Preconditions - The system design must be approved, and developers must have access to the project's GitHub repository.

- Postconditions - The system development process is initiated, and the progress is tracked and displayed on the system dashboard.

Functional Requirement 9 – System Testing Initiation

- QA Testers can conduct manual and automated testing on new features and fixes after development tasks are completed.
- The system allows access to the latest code, tracks the execution of test cases, logs issues, and notifies developers of any bugs found.
- Outputs - Test reports, a list of logged issues, and notifications to developers about the issues found.
- Preconditions - Development tasks must be completed and submitted for code review, and QA Testers must have access to the project's codebase and test environment.
- Postconditions - The testing process is initiated and tracked, issues are logged and assigned to developers, and the system provides visibility into the testing progress and results.

Functional Requirement 10 – Task Tracking and Assignment

- Team Leaders and project members can assign and track tasks for developers, UI/UX designers, QA Testers, and other roles.
- The system allows creating tasks with detailed information, tracking progress, and sending notifications about new tasks, deadlines, and status changes.
- Outputs - Task assignment confirmations, progress updates, notifications, and task status reports.
- Preconditions - Team Leaders and project members must have access to the task management feature, and team members must have defined roles and permissions.
- Postconditions - Tasks are assigned and tracked effectively, real-time updates on task progress and status are provided, and team members are notified of their tasks and deadlines.

Functional Requirement 11 – Virtual Meetings

- Users can schedule virtual meetings using Zoom, with meeting details displayed on the calendar and notifications sent to participants.
- The system allows scheduling Zoom meetings, generates meeting links, saves meeting details, and sends notifications and reminders.
- Outputs - Meeting confirmation notifications, calendar updates, and reminder notifications.
- Preconditions - The system must be integrated with the Zoom API, and users must have the necessary permissions to schedule meetings.
- Postconditions - Meetings are scheduled and displayed on the calendar, and participants receive notifications and reminders about the meetings.

Functional Requirement 12 – Team Member Chat

- Team members can communicate directly with each other in real-time using the chat feature.
- The system delivers messages, stores chat history, and provides notifications for new messages.
- Outputs - Real-time chat messages, notifications for new messages, and stored chat history.
- Preconditions - Users must be logged into the system and have the necessary permissions to access the chat feature.
- Postconditions - Messages are delivered to the intended recipients, and chat history is stored and accessible for future reference.

Functional Requirement 13 – Milestone Management

- Project Managers can create, track, and manage project milestones to ensure the project stays on schedule and key objectives are met.
- The system validates milestone details, tracks progress, updates status based on task completion, and sends notifications and reminders.
- Outputs - Milestone creation confirmation, milestone progress updates, and notifications/reminders.
- Preconditions - The Project Manager must have the necessary permissions to create and manage milestones, and milestone details must be accurately provided.
- Postconditions - Milestones are created, tracked, and managed within the system, and project team members are informed about milestones and deadlines.

Functional Requirement 14 – Productivity and Performance Analysis

- Project managers and team leads can monitor and evaluate team productivity and performance using analytical tools.
- The system aggregates and analyzes data from task completion times, milestone achievements, logged work hours, and user activity, and visualizes metrics through charts, graphs, and reports.
- Outputs - Productivity and performance reports, charts, and notifications for identified issues or achievements.
- Preconditions - The system must have access to relevant data on tasks, milestones, and work hours, and users must have completed enough tasks to generate meaningful analysis.
- Postconditions - Productivity and performance metrics are generated and available for review, and project managers and team leads are informed of any significant issues or achievements.

Functional Requirement 15 – Document Management

- Users can upload, manage, and share project-related documents, ensuring team members have access to necessary documentation.
- The system categorizes and stores documents, manages version control, enables downloads and viewing, and allows updates and deletions with appropriate permissions.
- Outputs - Confirmation messages for successful uploads, updates, or deletions; error messages for issues; and notifications for document updates.
- Preconditions - Users must have the necessary permissions to upload, manage, or view documents, and documents must be in acceptable formats (e.g., PDF, DOCX, XLSX).
- Postconditions - Documents are securely stored and accessible to authorized users, and changes to documents are tracked and versioned.

Functional Requirement 16 – Client Access to View Project Progress

- Clients can view the project's progress through a user-friendly interface with visual representations such as pie charts, progress bars, and detailed reports.
- The system aggregates data on task completion and milestones, calculates overall project progress, and generates visual representations.
- Outputs - Visual progress indicators (pie charts, progress bars), detailed progress reports, and summary notifications for the client.
- Preconditions - The system must have up-to-date data on task completion and milestones, and clients must have valid credentials to access the project progress page.
- Postconditions - Clients can view real-time progress of the project and receive accurate and understandable visual representations of the project status.

Non-Functional Requirements

1. **Performance**
   - The system should handle a large number of concurrent users without significant performance degradation.
   - Response time for any user action should not exceed 2 or 3 seconds under normal load conditions.

2. **Scalability**
   - The system should be able to scale horizontally to accommodate growing user and data volumes.
   - The database should be designed to handle data growth efficiently.
   - Support for adding new features and modules with minimal disruption.

3. **Security**
   - Data should be encrypted both at rest and in transit using industry-standard encryption methods.
   - The system should support multi-factor authentication (MFA) for user logins.
   - Regular security audits and vulnerability assessments should be conducted.

4. **Usability**
   - The user interface should be intuitive and easy to navigate for users with varying levels of technical expertise.
   - Provide consistent user experience across different devices (responsive design).
   - Online help and documentation should be available to assist users.

5. **Reliability**
    - The system should have an uptime of 99.9% to ensure availability.
    - Automated backups should be performed daily, with the ability to restore data within 4 hours in case of failure.

6. **Maintainability**
    - The system should be modular, allowing for easy updates and maintenance.
    - Code should follow standard coding conventions and be well-documented.

7. **Compatibility**
    - The system should be compatible with all major web browsers (Chrome, Firefox, Safari, Edge).
    - It should also be responsive and work on various devices, including desktops, tablets, and smartphones.

SYSTEM DESIGN

System architecture diagram

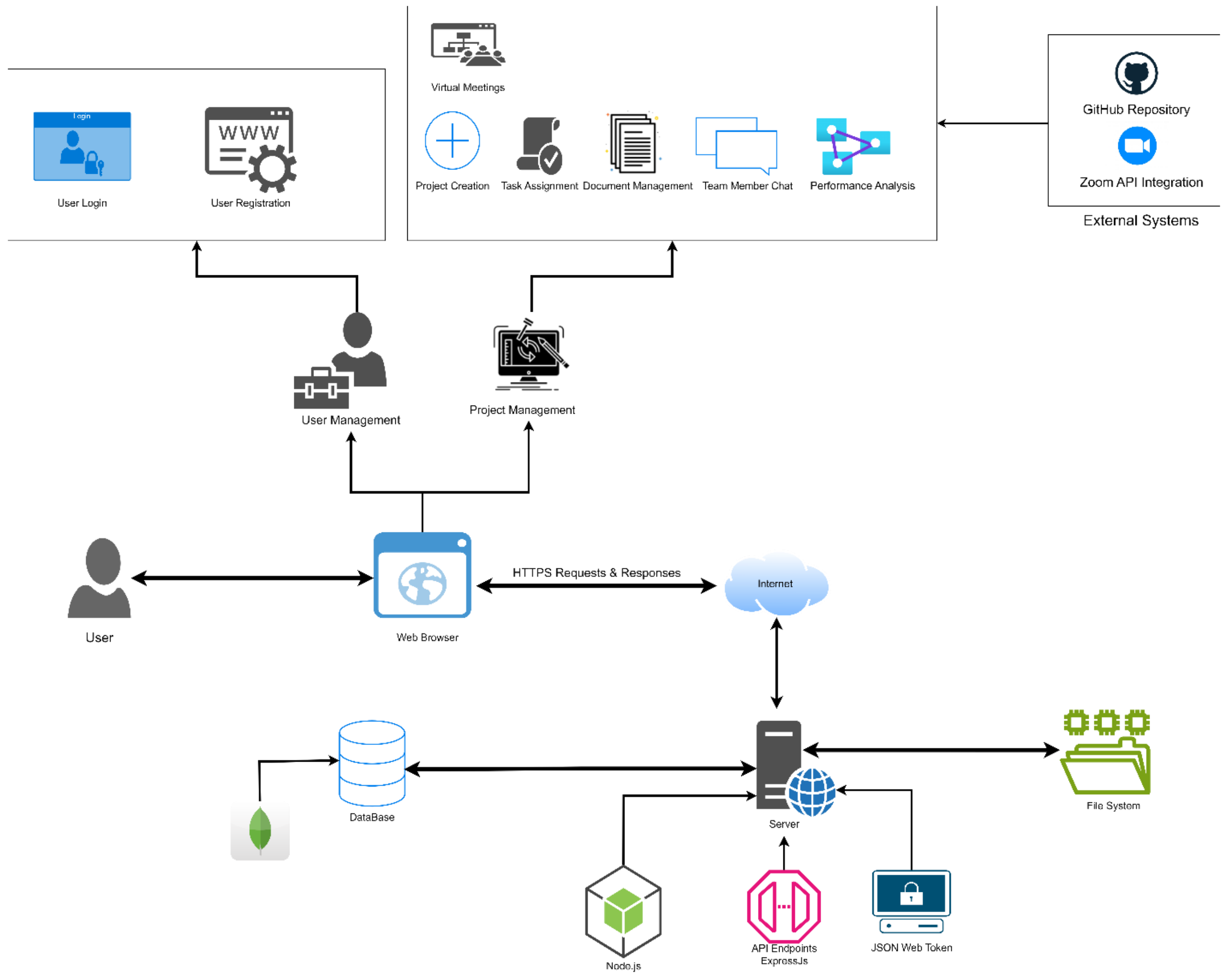


Figure 1 - system architecture diagram

- As in Figure 1, Users Interact with the system via a web browser. They can log in, register, and perform various project management tasks. Through a web browser, they can access the web application and can travel through various user interfaces. The user interface is where the user accesses the system's features through HTTPS requests and responses over the internet.

- Here User Login & Registration Allows users to create an account and log into the system. Also, the user management functionality handles the management of user accounts, including creating, updating, and deleting user profiles.
- Project Management Includes various functionalities like project creation, task assignment, document management, team member chat, performance analysis, and scheduling virtual meetings.
- Also, External Systems including GitHub Repository Integrates with GitHub for version control and project repository management and Zoom API Integration Allows for scheduling and managing virtual meetings directly within the system.

- In the backend The Server works as the central hub and that handles requests from the web browser and communicates with other backend services like ,
  - Node.js for the runtime environment used to execute server-side code.
  - Express.Js use as the web application framework and used to create API endpoints and manage HTTP requests and responses.
  - Also, JSON Web Token (JWT) is Used for secure user authentication and authorization.
- When considering Data Management, Database Stores all the data required for the system and the File System Manages file storage for documents and other files uploaded or generated by the system.

<u>This is how Communication Flow works</u>

- First Users interact with the system via the web browser, sending HTTPS requests to the server. The server processes these requests using Node.js and Express.js, handling authentication with JWT. Also, the server communicates with the database to store and retrieve data and with the file system for file management. The system integrates with external services like GitHub for repository management and zoom for virtual meetings.

Use Case Diagram

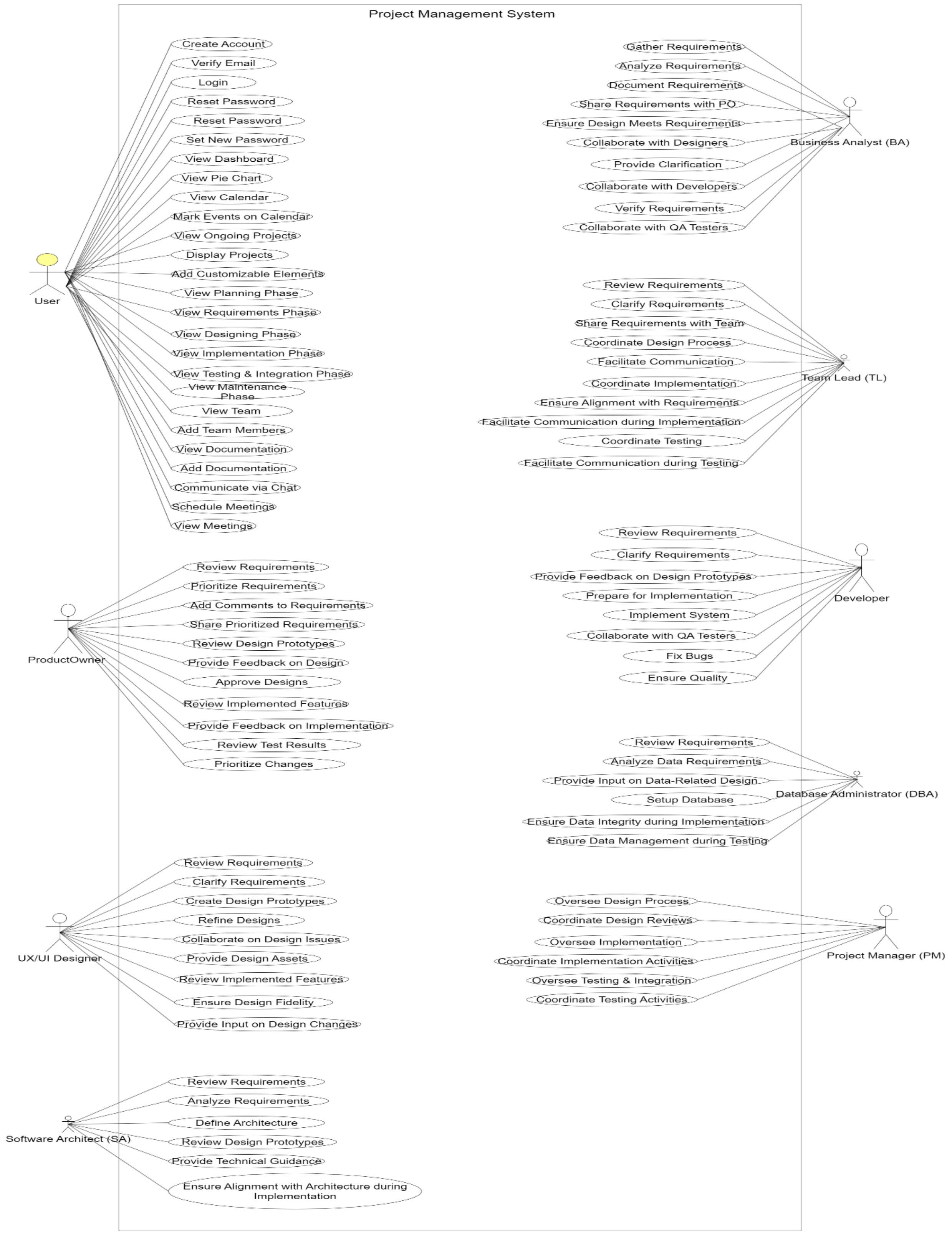


Figure 2- *https://drive.google.com/file/d/1vsI7BILmAXI4_Ir-Ay_t3AQLRcNPrsDn/view?usp=sharing*

The use case diagram for the software development process outlines the interactions between various actors and the functionalities within a project management system. Key actors include Users, Business Analysts (BA), Product Owners (PO), Team Leads (TL), UX/UI Designers, Developers, Software Architects (SA), Database Administrators (DBA), and Project Managers (PM). Each actor has specific roles: Users manage accounts and view dashboards, BAs gather and analyze requirements, POs review and prioritize requirements, TLs coordinate team efforts, UX/UI Designers create design prototypes, Developers implement the system, SAs define the system architecture, DBAs ensure data integrity, and PMs oversee project phases. The use cases span account management, dashboard interaction, project phase viewing, team communication, documentation, and meeting scheduling. This structured approach ensures effective collaboration, clarity, and accountability throughout the software development lifecycle, leading to successful project execution.

Sequence Diagram

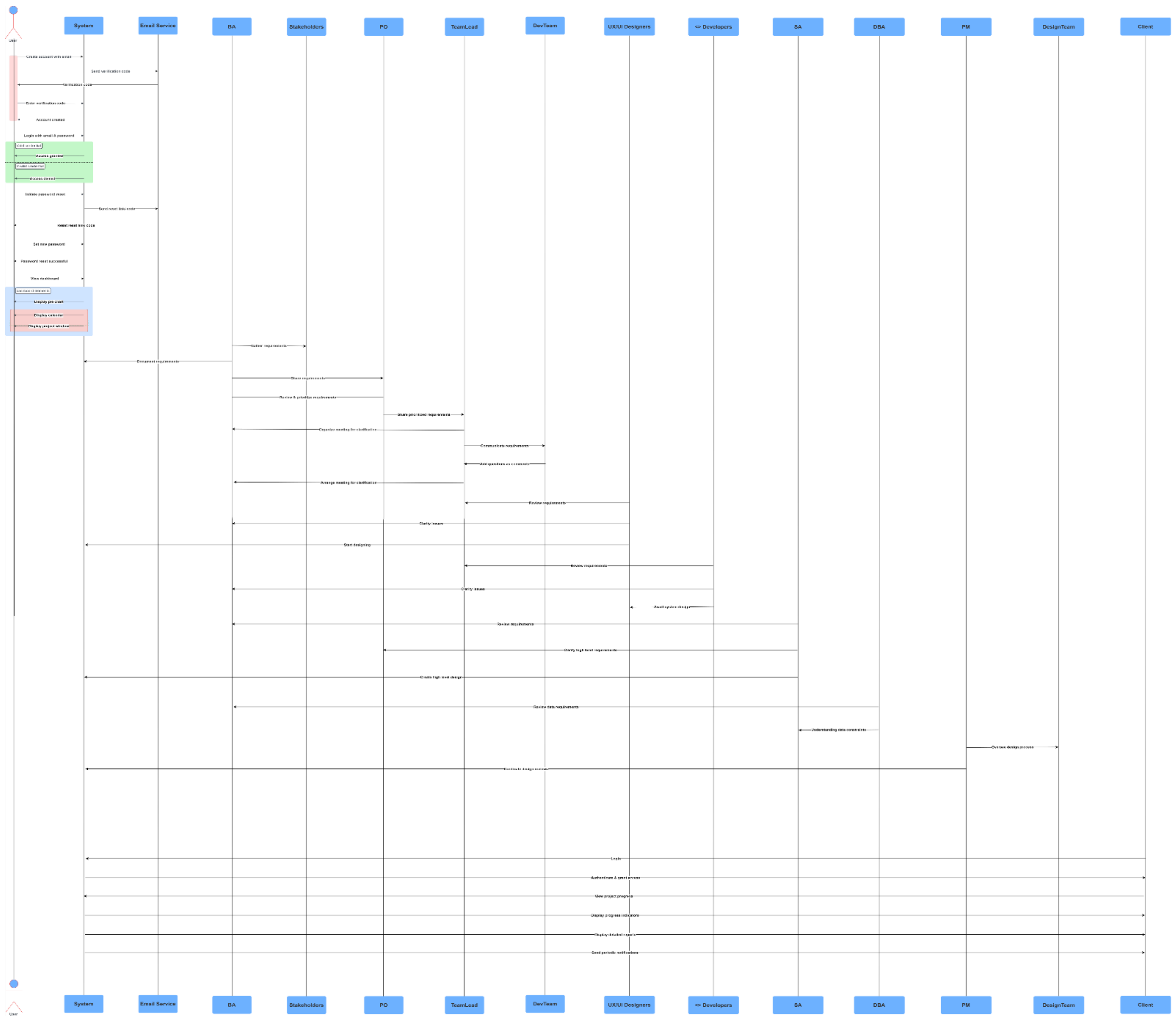


Figure 3 - https://drive.google.com/file/d/1VjdjsVBmFGXyQ7-rvNuVwkqSjyaMg8gb/view?usp=sharing

The sequence diagram (Figure 4) for the Real-Time Project Management Application illustrates the dynamic interaction between various components and actors within the system. It highlights the flow of events for core functionalities such as user account management, project dashboard navigation, requirements gathering, design management, implementation, testing and integration, and client access.

- Key elements depicted in the sequence diagram include the processes of user registration, login, password recovery, and dashboard interactions.
- The diagram also details the collaborative efforts of roles such as Business Analysts, Product Owners, Team Leads, UX/UI Designers, Developers, Software Architects, Database Administrators, and QA Testers in gathering requirements, managing designs, overseeing implementation, and ensuring rigorous testing.
- Additionally, it shows how clients can access real-time project progress through visual representations like charts and reports.

This sequence diagram serves as a critical blueprint for understanding the interactions and workflows essential to the successful operation of the application.

ER Diagram

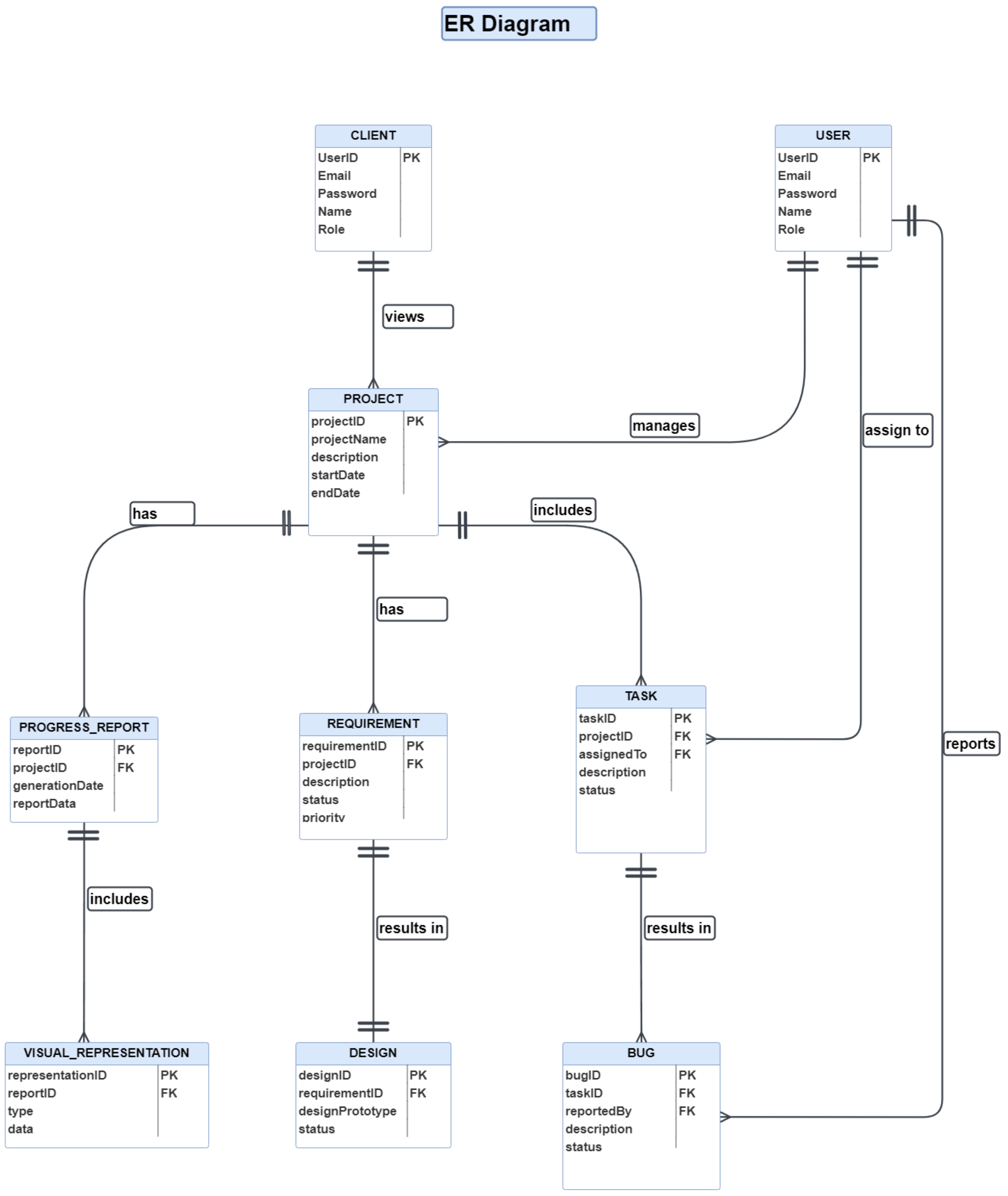


Figure 5 - https://drive.google.com/file/d/1RdAXDYqnyXqBxG78KubLN66C_wG7FbZ5/view?usp=sharing

The Entity-Relationship (ER) diagram (Figure 6) for our Real-Time Project Management Application illustrates the core components and their relationships within the system. **Key entities include User, Authentication, Dashboard, Project, Requirement, Design, Implementation, Testing, and Client.**

The diagram details how User information is managed,

- Email verification and password recovery through the Authentication entity.
- The Dashboard entity displays user-specific data such as time tracking, event calendars, and project overviews.
- The Project entity outlines the lifecycle phases, supported by Navigation features like Team, Documentation, Chat, and Meetings.
- The Requirement entity captures and manages project requirements, highlighting roles such as Business Analyst, Product Owner, Team Lead, UX/UI Designers, Developers, Software Architect, and Database Administrator.
- Design and Implementation entities ensure project goals are met through coordinated processes, while Testing ensures quality standards.
- Lastly, the Client entity provides real-time data visualization of project progress.

This comprehensive ER diagram encapsulates the dynamic interactions and data flow within our application, ensuring efficient project management and user engagement.

Class Diagram

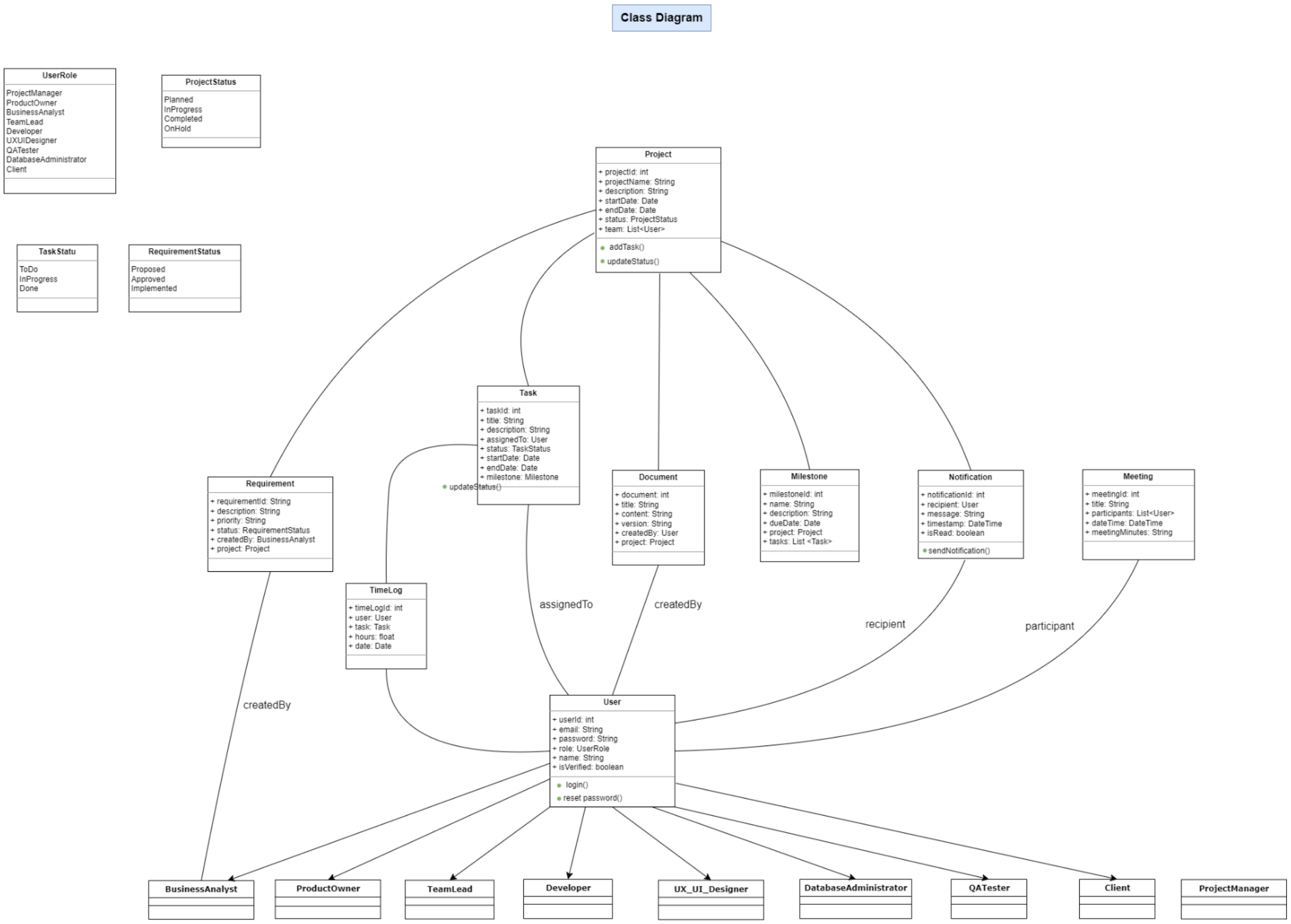


Figure 7 – class diagram (https://drive.google.com/file/d/1R0lkxke8MNyOJdtOOjYMXhvHUMLcO_ty/view?usp=sharing)

The class diagram (Figure 8) for the Real-Time Project Management Application provides a detailed and organized view of the system's architecture. **This diagram represents the core entities, their attributes, methods, and the relationships between them, offering a clear and organized view of the application's object-oriented design.**

Features

- The class diagram for the Real-Time Project Management Application illustrates a robust framework encompassing core components such as User Account Management, Dashboard, Project Dashboard, Requirements Gathering, Design Management, Implementation, Testing & Integration, and Client Access.
- The User class manages user information and authentication processes, while the Dashboard provides user-specific data visualization. The Project Dashboard outlines project lifecycle phases and essential navigation features. Requirements are captured and managed by various role classes including Business Analyst (BA), Product Owner (PO), Team Lead (TL), UX/UI Designers, Developers, Software Architect (SA), and Database Administrator (DBA).
- Design Management ensures alignment with project goals, while Implementation oversees the development process. Testing & Integration focus on maintaining quality standards, and Client Access offers real-time project progress visualization. This comprehensive diagram facilitates clear communication and efficient development of the project management application.

Relationships

- Associations: Represent how classes interact with one another, ensuring data flow and functionality across different modules.
- Generalizations: Show inheritance relationships, indicating shared attributes and methods among classes.

Activity Diagram

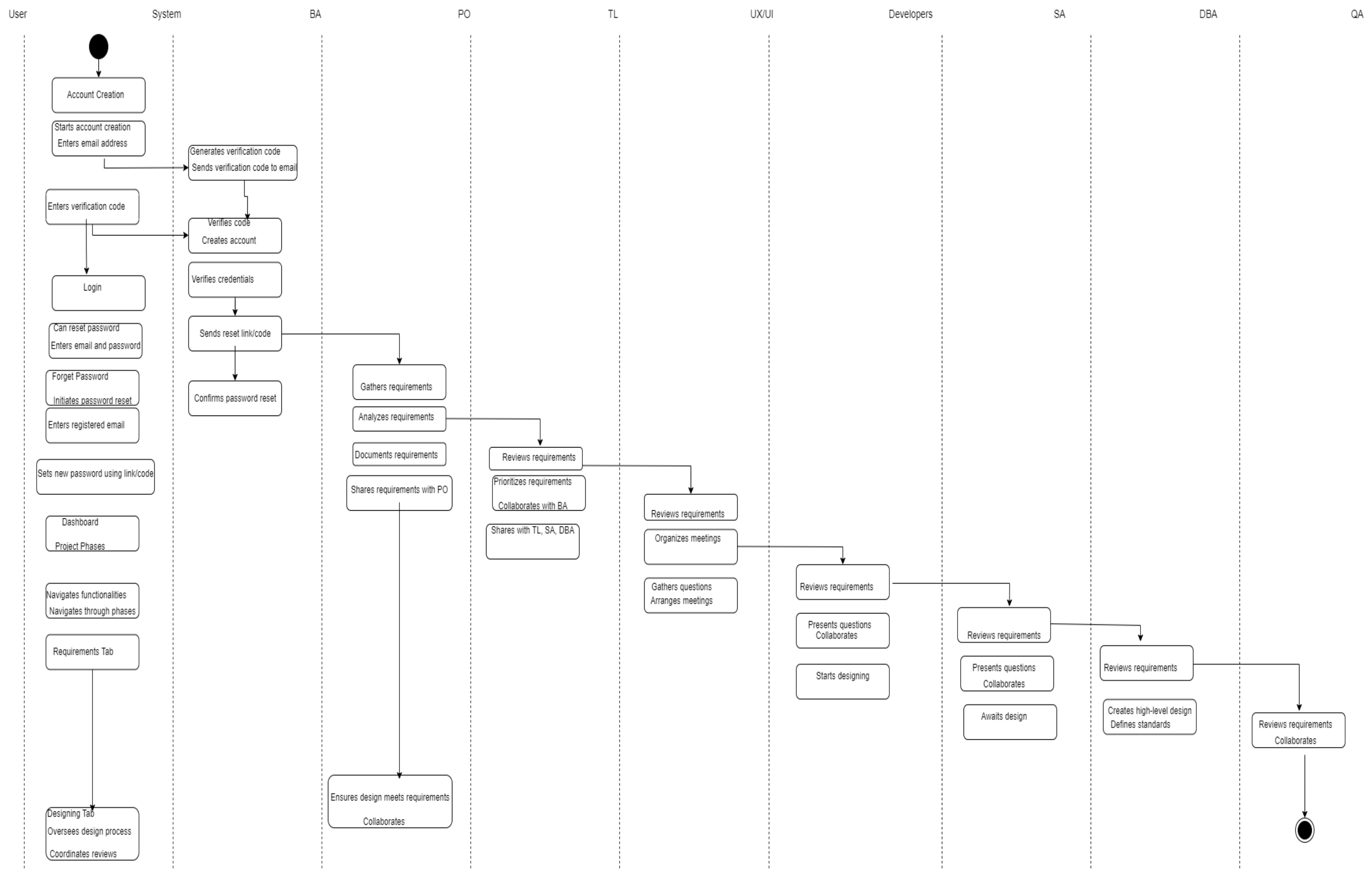


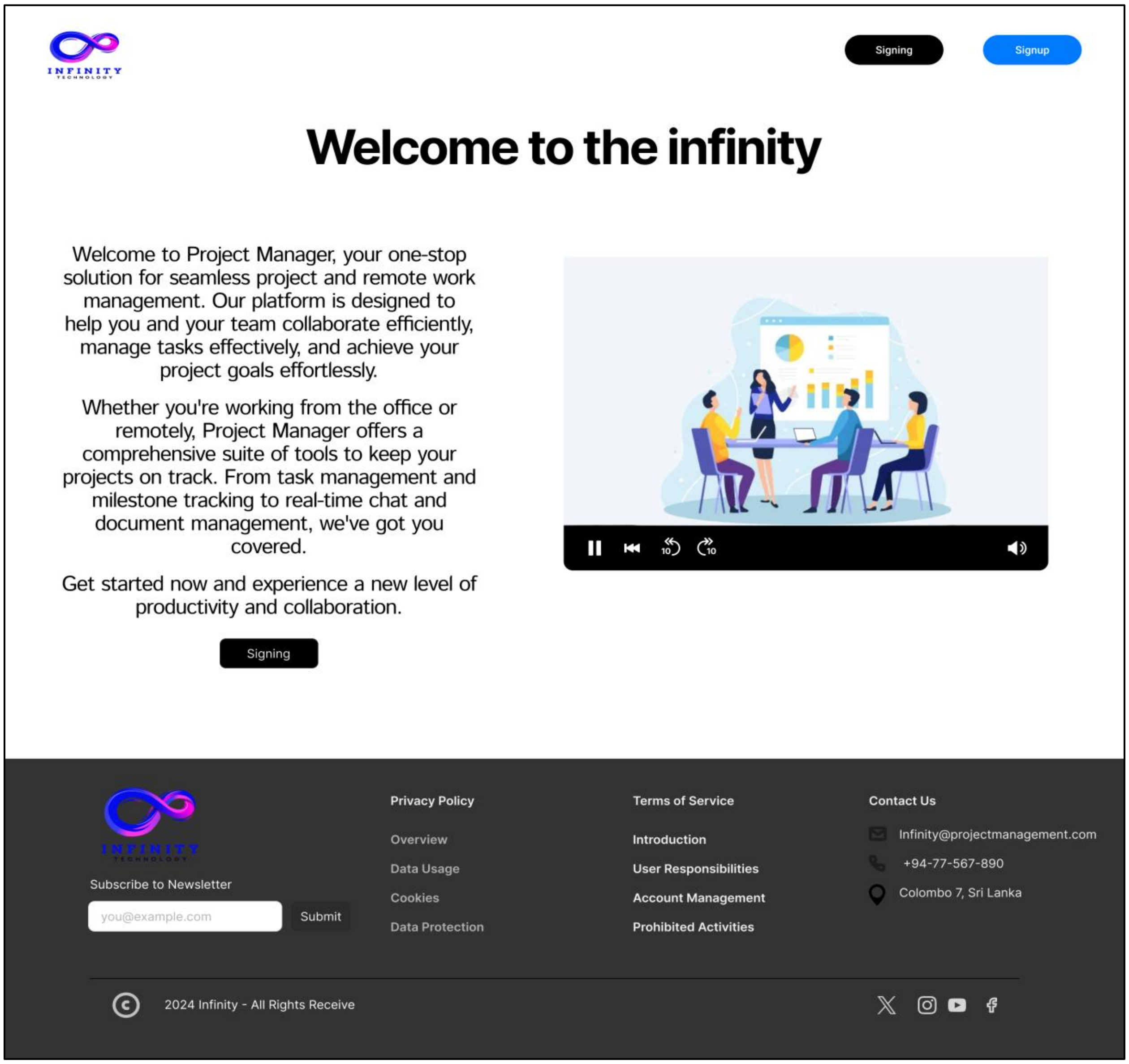
Figure 9 – Activity Diagram (https://drive.google.com/file/d/1905giQBaLLtEYqSLf1ePMKOCgdQQ294B/view?usp=sharing)

The activity diagram (Figure 10) outlines key workflows and interactions within a system, segmented into distinct phases. The user initiates account creation by entering their email, receiving a verification code, and completing registration upon validation by the system. For login, users enter their credentials, with an option to reset forgotten passwords through a link or code sent by the system. Post-login, users access a dashboard displaying work metrics, upcoming events, and ongoing projects. The project dashboard allows navigation through various project phases and functionalities. The Requirements Tab involves the Business Analyst gathering, analyzing, and documenting requirements, which are then reviewed and prioritized by the Product Owner, and shared with the Team Lead, Software Architect, and Database Administrator. Each role collaborates to refine requirements and proceed with design and implementation phases. The diagram ensures clarity on user-system interactions and role-specific activities, emphasizing collaborative workflows in requirement gathering, design, and implementation processes.

UML Diagrams - https://drive.google.com/drive/folders/1Luf6ytYLgr2z1iAMT9K5cfwL2-oCM-R4?usp=sharing

User Interfaces

Landing page



Welcome to the infinity

Welcome to Project Manager, your one-stop solution for seamless project and remote work management. Our platform is designed to help you and your team collaborate efficiently, manage tasks effectively, and achieve your project goals effortlessly.

Whether you're working from the office or remotely, Project Manager offers a comprehensive suite of tools to keep your projects on track. From task management and milestone tracking to real-time chat and document management, we've got you covered.

Get started now and experience a new level of productivity and collaboration.

Infinity@projectmanagement.com
+94-77-567-890
Colombo 7, Sri Lanka

2024 Infinity - All Rights Receive

Figure 11 - Landing page of the web application

When users come to our "Comprehensive Project and Remote Work Management System" they first visit this Landing page (Figure 12). The landing page welcomes users to the web application, highlighting its tools for seamless project management, team collaboration, task tracking, and real-time communication. Users are prompted to sign in or sign up to enhance their productivity and collaboration.

Home Page

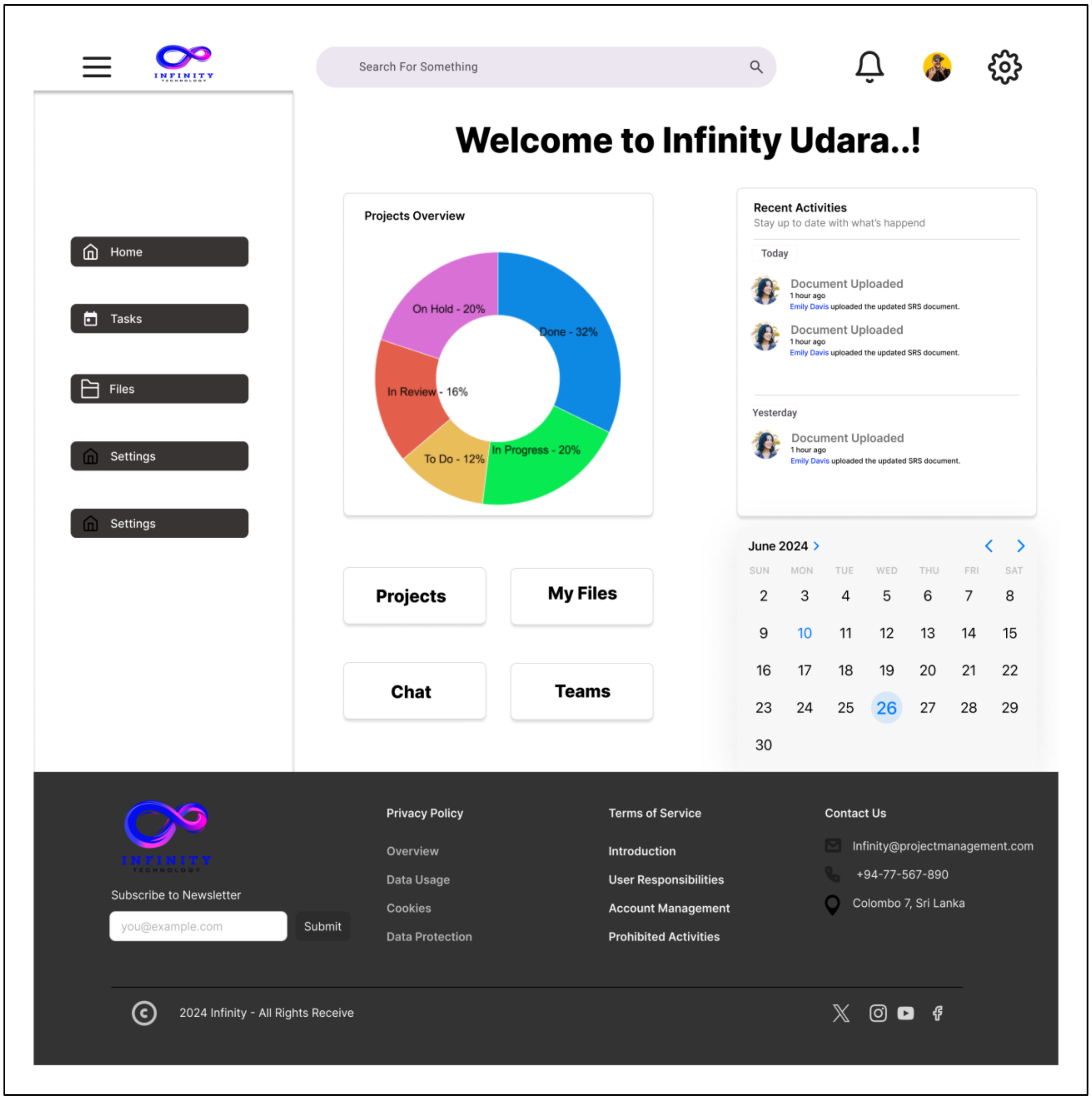


Figure 13 - Home Page of the system

After login into the system users can see the home page (Figure 9) of the Comprehensive Project and Remote Work Management System provides an overview of project statuses with a visual chart and recent activities. The interface also features some Key navigation options, a calendar and quick access to some data.

Project Page

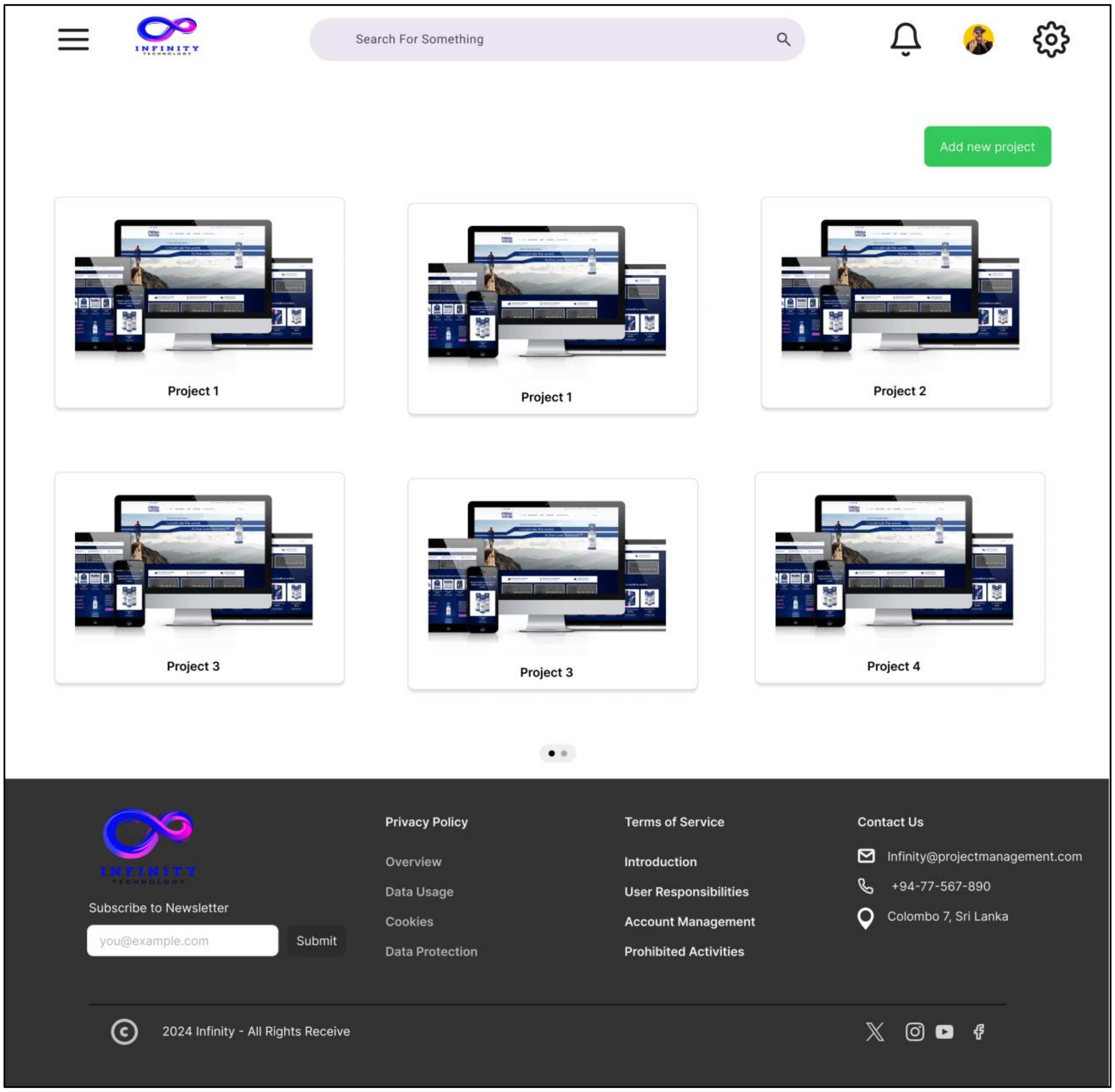


Figure 14 - Project Page of the system

The projects page (Figure 15) displays an overview of existing projects with thumbnail images for easy identification. Users can search for projects using the search bar at the top or add a new project using the "Add new project" button.

SYSTEM DEVELOPMENT, TESTING AND DEPLOYMENT STRATEGIES

Development Process – How We Plan to do Development Process

System Development Strategy

Development Methodology – Agile

Scrum Framework - Use sprints, usually 2-4 weeks long, with daily stand-ups, sprint planning, sprint reviews, and retrospectives. This ensures iterative development, continuous feedback, and flexibility to adapt to changing requirements.

1. **Initial Setup**

   i. Project Initialization

   - We Create a GitHub repository for the project. This will be the central repository where all team members will contribute their code.
   - We Set up the initial project structure using the MERN stack (MongoDB, Express.js, React.js, Node.js).

   ii. Local Development Environment

   - All Team members need to have the necessary tools installed. Node.js, MongoDB, an IDE (like VS Code or WebStorm), Git, and Docker (optional for containerization).
   - They need to Clone the GitHub repository to each team member's local machine.

2. **Development Workflow**

i. Branching Strategy

- We Use Git branches to manage development. Team leader Creates a main branch for stable code and each member creates separate branches for each feature or bug fix.
- This Format needs to use Branch create - Example branches - feature/login, feature/chat, bugfix/ui-bug.

ii. Task Assignment

- Assign tasks to team members based on their roles (developers, UI/UX designers, QA testers). Roles can change between iterations.

iii. Development Process

- Frontend Development – Firs need to Build UI components using React.js and integrate them with the backend.
- Backend Development - Develop API endpoints using Express.js and connect them to MongoDB/MySQL.
- Integration - Regularly merge feature branches into a develop branch to integrate and test changes. This integration done by the Team leader.

iv. Code Reviews

- Create pull requests for merging code into the main branch.
- The Team Leader reviews the code, suggests changes, and approves pull requests.

Testing Strategy

3. Local Testing

i. Manual Testing

- Each developer should manually test their features locally to ensure they work as expected.
- Each member Conducts integration testing by running the entire application on their local server.

ii. Automated Testing

- We plan to write unit tests for individual components and functions using Jest.

    - User Registration process – In this case, we test user creation, validation, and error handling.
    - User Login process – first need to Verify every user logs into the system using the correct email address and password. Also, we consider session management and error responses.
    - Project Creation Process – In this case, we ensure the project creation process works correctly. Also, the project data is correctly saved and retrieved.
    - Task Tracking and Assignment Process - Test task creation, updates, and assignment logic and only the task can be created who has access to it.
    - Milestone Management Process – The Project manager can create milestones and update milestones, and when the milestones are complete need to be correctly shown in the system.
    - Notifications generating and sending Process – we ensure notifications are generated and sent correctly for the particular members.
    - Chat Process –here we ensure test message sending, receiving the correct person, and storing the messages.
    - Virtual Meetings Process – the system should ensure that meetings can be organized correctly and marked on the respective members' calendars.

- Also, we Implement end-to-end tests using Selenium.
    - This End-to-end testing will verify that the entire application workflow works as expected. All functionalities must work correctly without any errors. We perform end-to-end testing using Selenium.

- We hope to set up a CI/CD pipeline using GitHub Actions to automatically run tests on every push or pull request.

4. Local Deployment

i. Local Server Deployment

- First, we Deploy the application to a local server for initial testing. The server is based on Express.js.

5. Integration and Collaboration

i. Combining Development Parts

- Regularly we push and pull code to and from the GitHub repository to keep everyone's code up to date.
- We hope to Resolve merge conflicts by communicating with team members and understanding the changes.

ii. Continuous Integration

- We hope to Use GitHub Actions to automate testing and integration.

Deployment Strategy

6. Deployment to AWS

- Finally, we hope to Deploy the application to an AWS EC2. Then we Perform both manual and automated testing in local and AWS environments to ensure the system's functionality and reliability.

System Development Strategy

Development Tools

- Integrated Development Environment (IDE) - VS Code or JetBrains WebStorm for code development
- Version Control System – GitHub for repository hosting
- JavaScript Build Tools – Webpack, Vite

Frontend Technologies

- Programming Languages - JavaScript, HTML5, CSS3
- Frameworks/Libraries
    - React.js - For building user interfaces
    - Redux - For state management
- UI Component Libraries – Bootstrap

Backend Technologies

- Programming Languages - JavaScript (Node.js)
- Frameworks - Express.js

Database

- MongoDB - NoSQL database for flexible data storage
- MySQL - Relational database for structured data storage

APIs and Integrations

Video Conferencing APIs

- Zoom API or Google Meet API for virtual meeting functionality.

Authentication

- JSON Web Tokens (JWT) for secure authentication and session management.

<u>Testing Strategy</u>

Testing Tools

- Unit Testing - Jest
- Integration Testing – Mocha with Chai
- End-to-End Testing – Selenium

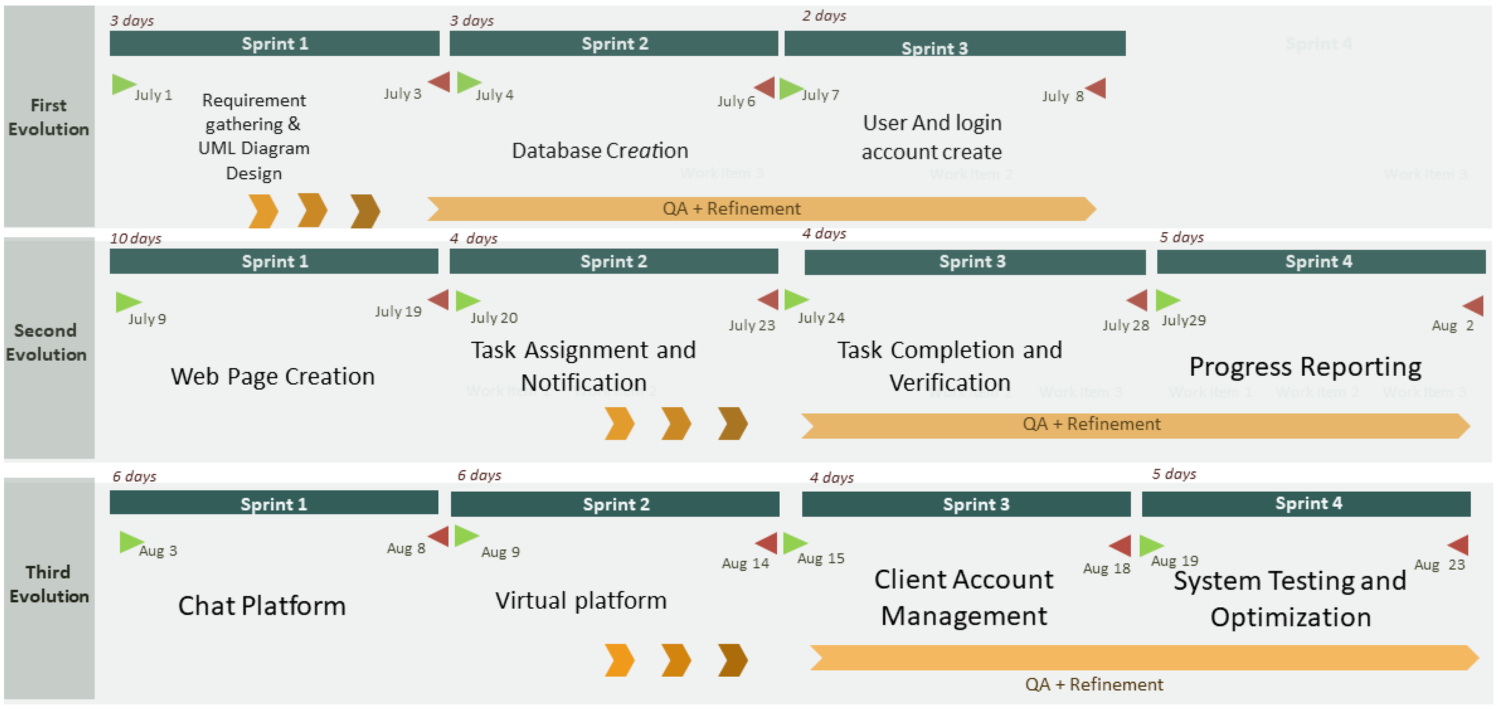
<u>Deployment Strategy</u>

Deployment Tools

Cloud Providers – AWS for hosting and scaling the application.

PROJECT MILESTONES AND TIMELINE

Timeline



First Evolution/Sprint – Core functionality implementation

1. Requirement gathering
    - In this stage we gathering all the functional requirement and non-functional requirement for the first phase. According the that all UML diagram design doing in this period.

2. UML diagram design
    - Create UML diagrams for system architecture, use cases, class diagrams, and sequence diagrams.

3. Project Initiation and User Account Management
    - Implement user registration, login, and profile management

4. Data Base creation
    - Create system database and design user account, implement with database

Second Evolution/Sprint - Advanced Task Management, Reporting, Web Pages, and Chat Platform

1. Task Assignment and Notification
    - Implement task assignment and notifications.

2. Task Completion and Verification
    - Implement task completion by members and verification by leaders.

3. Progress Reporting
    - Implement report generation, filtering, customization, and exporting.

4. Web Page Creation
    - Implement web pages for various system functionalities.

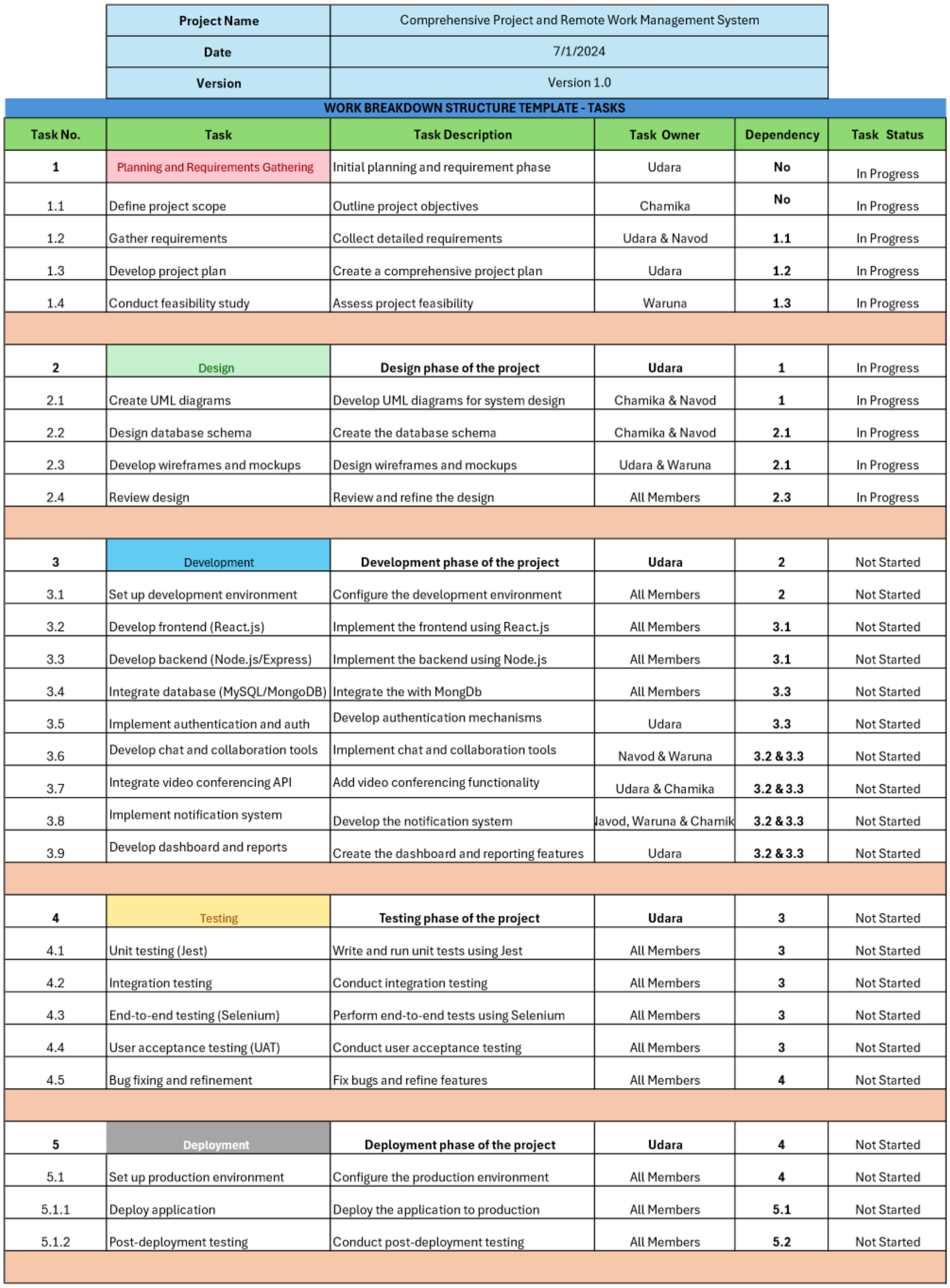
Third Evolution/ Sprint - Client Access and System Optimization

1. Client Account Management
    - Implement client account creation and login.

2. Client Report Access
    - Implement client access to progress reports and export functionality.

3. System Testing and Optimization
    - Conduct unit, integration, performance, and load testing. Implement security measures.
4. Chat Platform
    - Implement a chat platform for communication between project members and leaders

Work breakdown structure



Project Name	Comprehensive Project and Remote Work Management System
Date	7/1/2024
Version	Version 1.0

WORK BREAKDOWN STRUCTURE TEMPLATE - TASKS

Task No.	Task	Task Description	Task Owner	Dependency	Task Status
1	Planning and Requirements Gathering	Initial planning and requirement phase	Udara	**No**	In Progress
1.1	Define project scope	Outline project objectives	Chamika	**No**	In Progress
1.2	Gather requirements	Collect detailed requirements	Udara & Navod	**1.1**	In Progress
1.3	Develop project plan	Create a comprehensive project plan	Udara	**1.2**	In Progress
1.4	Conduct feasibility study	Assess project feasibility	Waruna	**1.3**	In Progress
2	Design	**Design phase of the project**	**Udara**	**1**	In Progress
2.1	Create UML diagrams	Develop UML diagrams for system design	Chamika & Navod	**1**	In Progress
2.2	Design database schema	Create the database schema	Chamika & Navod	**2.1**	In Progress
2.3	Develop wireframes and mockups	Design wireframes and mockups	Udara & Waruna	**2.1**	In Progress
2.4	Review design	Review and refine the design	All Members	**2.3**	In Progress
3	Development	**Development phase of the project**	**Udara**	**2**	Not Started
3.1	Set up development environment	Configure the development environment	All Members	**2**	Not Started
3.2	Develop frontend (React.js)	Implement the frontend using React.js	All Members	**3.1**	Not Started
3.3	Develop backend (Node.js/Express)	Implement the backend using Node.js	All Members	**3.1**	Not Started
3.4	Integrate database (MySQL/MongoDB)	Integrate the with MongDb	All Members	**3.3**	Not Started
3.5	Implement authentication and auth	Develop authentication mechanisms	Udara	**3.3**	Not Started
3.6	Develop chat and collaboration tools	Implement chat and collaboration tools	Navod & Waruna	**3.2 & 3.3**	Not Started
3.7	Integrate video conferencing API	Add video conferencing functionality	Udara & Chamika	**3.2 & 3.3**	Not Started
3.8	Implement notification system	Develop the notification system	Navod, Waruna & Chamik	**3.2 & 3.3**	Not Started
3.9	Develop dashboard and reports	Create the dashboard and reporting features	Udara	**3.2 & 3.3**	Not Started
4	Testing	**Testing phase of the project**	**Udara**	**3**	Not Started
4.1	Unit testing (Jest)	Write and run unit tests using Jest	All Members	**3**	Not Started
4.2	Integration testing	Conduct integration testing	All Members	**3**	Not Started
4.3	End-to-end testing (Selenium)	Perform end-to-end tests using Selenium	All Members	**3**	Not Started
4.4	User acceptance testing (UAT)	Conduct user acceptance testing	All Members	**3**	Not Started
4.5	Bug fixing and refinement	Fix bugs and refine features	All Members	**4**	Not Started
5	Deployment	**Deployment phase of the project**	**Udara**	**4**	Not Started
5.1	Set up production environment	Configure the production environment	All Members	**4**	Not Started
5.1.1	Deploy application	Deploy the application to production	All Members	**5.1**	Not Started
5.1.2	Post-deployment testing	Conduct post-deployment testing	All Members	**5.2**	Not Started

CONCLUSION

Summary

The Comprehensive Project and Remote Work Management System project proposal demonstrates a thorough understanding of the project's goals, the necessary functionalities, and the technical requirements. With a clear outline of features such as task tracking, milestone management, team collaboration, time tracking, virtual meetings, productivity analysis, and user management, the proposal is well-positioned to address the needs of software development teams.

Summary of Key Points

1. Clear Objectives
    - The system aims to streamline project management and remote work coordination, enhancing team productivity and communication.
2. Detailed Functionalities
    - The proposal covers essential features required for effective project and remote work management.
3. Technological Stack
    - The choice of Node.js/Express, React.js, MySQL, and video conferencing APIs is appropriate for building a robust and scalable system.

   Roles and Responsibilities
    - Clearly defined roles ensure that each team member understands their responsibilities and contributes effectively.
4. Visual Aids
    - The inclusion of UML diagrams and wireframes provides a clear visual representation of the system's architecture and design.
5. Project Milestones
    - Well-defined milestones help in tracking the progress and ensuring timely delivery of the project.

***