

DEADLINE DEALER

# SOFTWARE ENGINEERING PROJECT

Phase 3

## Shruti Bansal Tejaswi Manavala Narayanan

Namit Arjaria Bharathi Subramanian Nagarajan

INDEX

* [Introduction 2](#_TOC_250005)
  + Problem Statement
* [Background Research 3](#_TOC_250004)
* [Proposed System 4](#_TOC_250003)
  + Goals and Objectives
* [Project Planning 5](#_TOC_250002)
  + Project Setup.
  + Stakeholders and Their Roles
  + Resource Descriptions
  + Assumptions
* [System Analysis and Design 9](#_TOC_250001)
  + Overall Description
  + Users and Roles
  + User Stories
* [Diagrams](#_TOC_250000)
  + Sequence Diagrams 20
  + Use Case Diagrams 22
  + Class Diagram 24
  + State Diagram 25
* References 26

# Introduction

## Problem Statement

In today's fast-paced world, procrastination, and inefficient task management torment nearly everyone, leading to a plethora of problems. One significant issue is the consistent missing of deadlines. This leads to stress, a sense of failure, and negatively impacts credibility.

The constant pressure of deadlines often leads to compromising the quality of work. When tasks are rushed to meet a set time limit, individuals may prioritize completion over thoroughness, resulting in errors and subpar outcomes. This cycle of last-minute efforts turns into a lack of motivation to consistently work towards deadlines throughout the entire project duration.

Moreover, the current task management landscape often suffers from a disconnect between setting deadlines and effectively tracking progress. Physical planners or to-do lists can become cumbersome and may not be useful to efficiently break down complex tasks into manageable sub-parts. This hinders organization and makes it difficult to gauge the progress of the task being performed. People will struggle to visualize their workload and become overwhelmed and maybe even afraid to start the task.

Therefore, a solution is needed to address the shortcomings of traditional task management methods. An ideal system should combat procrastination and promote a consistent work ethic, while simultaneously mitigating the negative effects of last-minute efforts on the quality of work. This necessitates a platform that fosters effective organization and progress tracking, allowing individuals to visualize their workload and efficiently manage sub-tasks within the framework of set deadlines.

# Background Research

Studies suggest that procrastination is associated with symptoms of depression, anxiety, and stress as well as loneliness and reduced life satisfaction. Procrastination is also associated with prevalent general physical health problems, cardiovascular disease, and unhealthy lifestyle behaviors.

Students engaged in university studies have high levels of freedom and low structure, which places high demands on their capacity to self-regulate. These high demands on self-regulation may explain the high prevalence of procrastination among university students and make persons who are prone to procrastinate more vulnerable to the negative consequences of procrastination while at the university.

Gamification principles have proven effective in various fields, including education and fitness. Rewarding participants with anything at all, ranging from points to edible treats, proved to increase their interest in the activity and had a high chance of making them want to continue performing it.

Therefore, we have concluded that the gamification of the mundane tasks of managing deadlines and completing work would persuade people to stick by their self-set deadlines, and thereby increase their productivity.

# Proposed System

## Goals and Objectives

With Deadline Dealer, we aim to ease the process of managing and meeting deadlines using a gamified environment to make task scheduling fun, while helping users combat procrastination and achieve their goals on time.

Objectives:

* Increase Motivation and Reduce Procrastination:

By incorporating game features like points, Deadline Dealer aims to transform tedious tasks into engaging challenges. This fosters a sense of accomplishment and motivates users to consistently chip away at their workload, overcoming the urge to procrastinate until the last minute.

* Improve Task Quality Through Progress Tracking:

The application will provide a visual representation of progress through features like sub-task completion checklists. This allows users to monitor their workflow, identify areas requiring more focus, and ensure tasks are completed to a high standard without the pressure of last-minute cramming.

* Streamline Organization and Break Down Complex Tasks:

Deadline Dealer will facilitate the segmentation of large projects into manageable sub-tasks. This empowers users to visualize their workload and prioritize tasks effectively, eliminating feelings of overwhelm that often hinder productivity.

* Promote Time Management Skills:

The core functionality of setting deadlines and tracking progress cultivates a sense of time awareness. Users learn to estimate time for different tasks, improving their ability to plan and manage their schedules effectively.

# Project Planning

## Project Setup

### Languages:

* JavaScript – Scripting language for Web pages. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles.

### Frameworks:

* ReactJS – A JavaScript library for building user interfaces.

### Backend:

* Supabase – Supabase is an open-source Firebase alternative. It provides a full Postgres database (an open-source database) for every project. It also manages Authentication and storage, working in real-time.

### Tools:

* CSS - Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML. CSS describes how elements should be rendered on screen.
* Vite – A frontend build tool, and a build command that bundles code, pre-configured to output highly optimized static assets for production.
* Replit – An online Integrated Development Environment where developers can collaborate and work on the code simultaneously.
* GitHub – A Graphical User Interface (GUI)-based Hosting service for Distributed Version Control System Repositories.

## Stakeholders and Their Roles

* **Students:** The primary stakeholder in this project is the student community. Students find it extremely complex to keep track of the various deadlines they have for different subjects and other non-academic matters. They are the primary audience for whom the project is being catered for, and the gamification of tasks is implemented, keeping in mind the interest and enthusiasm shown by the young community on making mundane tasks into a reward-based system.
* **Professors and other Professionals:** Professors and other working professionals can also hugely benefit in such an easily deployable system. It makes it easy for the professors to relate and connect with the students, if they share a common platform to track tasks, deadlines, etc. It is also a very intuitive application for anyone to use, though the environment is primarily designed to appeal to students. OCJ Students can also be assigned tasks by their respective superiors using the same platform as it is simpler to create tasks rather than coordinate over a multitude of emails.
* **Event Managers:** Another achievable feature of such a user-friendly website would be, to use it to create accounts with preset deadlines, which can be used to help people keep track of ongoing events. Example: Such accounts with deadlines of all important Breeze events would come in as very useful for students in and outside SNU.
* **Career Development Centre:** Upon the initial success of getting a smooth, errorless functioning of the website among students, we envision expanding the usage of such a shared platform, by allowing the CDC system to automatically set workshops and interview deadlines in the respective accounts of students. Other opportunities and entrance exam deadlines can also be shared in the same platform.
* **Other Miscellaneous Users:** Our software can also cater to other users who wish to ease their process of scheduling tasks and managing deadlines, be it at home, at the workplace, or even both combined. Our web app will be able to suit any of their needs, as long as they can categorize their tasks.
* **The Admins:** The Administrators would be in charge of frequently maintaining the application, checking for bugs and fixing them, monitoring the database, and updating the app as per user demand.

## Resource Descriptions

* + **Visual paradigm**: Visual Paradigm is a software design and development tool that provides an integrated environment for various modelling and diagramming tasks. It is widely used by software architects, developers, business analysts, and project managers across different industries. We have used it to create all the diagrams required to visualize the different parts of the project i.e., Sequence Diagram, Activity Diagram, etc.
  + **Replit:** Replit is a cloud-based application delivery and monitoring platform designed to optimize and secure access to enterprise applications and cloud services. It is primarily used by organizations to provide secure remote access to their internal applications, while ensuring high performance and reliability. We have used it to visualize the output of the website and to build prototypes for the website initially.
  + **Jira:** Jira is a popular issue-tracking and project management tool developed by Atlassian. It is a web-based application that provides a centralized platform for teams to plan, track, and manage their work. It offers a range of features and functionalities to support agile methodologies, such as Scrum and Kanban, as well as traditional project management approaches.
  + **GitHub:** GitHub is a web-based hosting service for version control and collaborative development of software projects. It provides a platform for developers to store, manage, and track changes to their source code repositories, collaborate with other developers, and contribute to open-source projects. We have used GitHub to publish our code and create new branches where we can try to implement experimental features and clear bugs.

## Assumptions

For the success of our software project, we make the following assumptions:

1. That users will adopt the usage of the app and use it on a regular basis, that they will regularly log their tasks to be completed, and log the status of completion of their work.
2. That gamification of the process of dealing with deadlines motivates the users to actively engage with the Web app, and the desire to score points in the app serves as enough reward for the users to continue using it.
3. That the tech stack we are using – ReactJS, Vite, Supabase – will continue to be effective, available, and serve its purpose as the web app continues to grow and be used.
4. That our background research about the necessity of this app is correct, and that people will prefer to use this gamified version of task managing over physical methods of maintaining pen-and-paper checklists.

# System Analysis and Design

The Overall Description

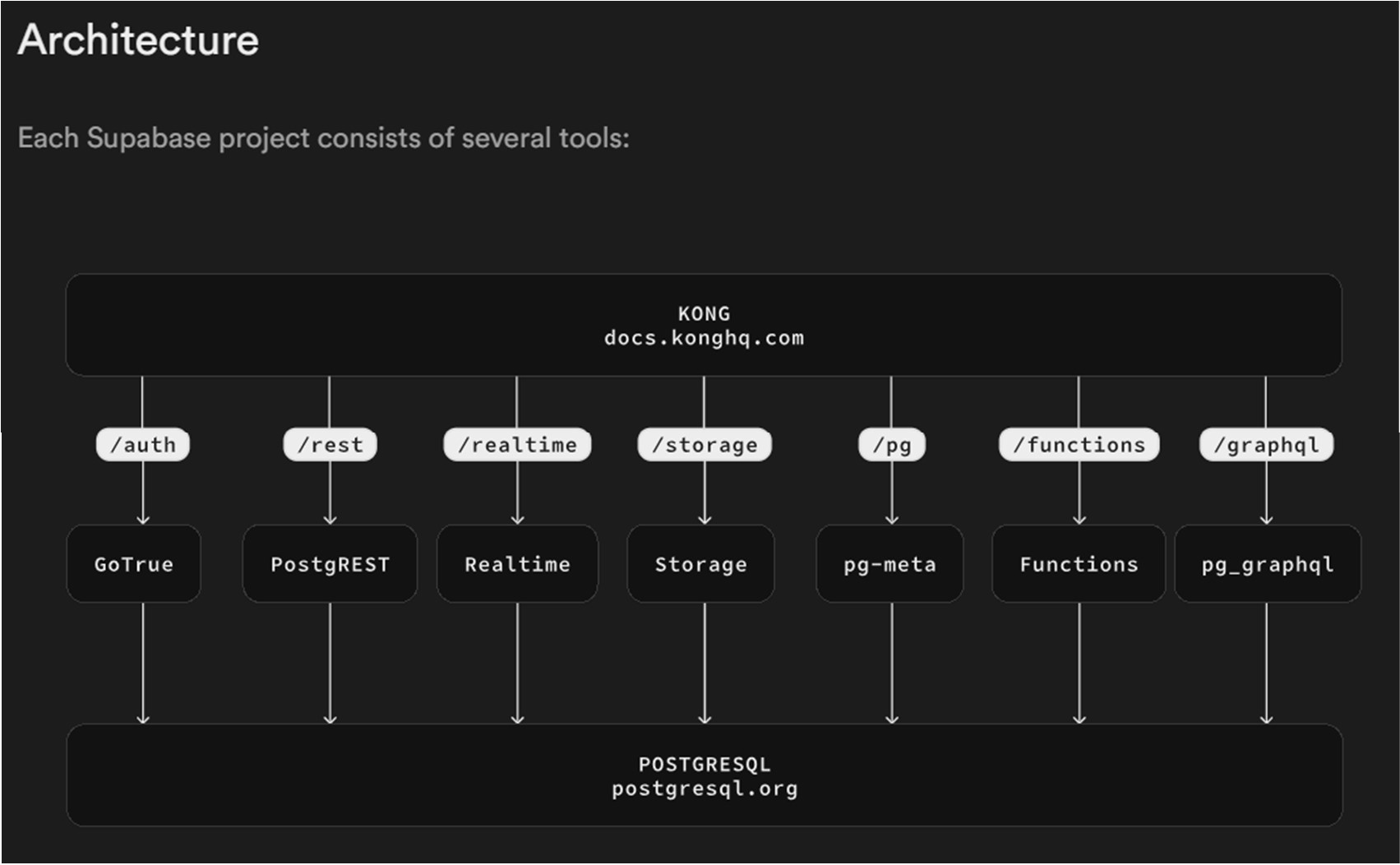
System Overview

* Sign Up:
* Users can easily create an account using their email address. After email verification, users activate their accounts to gain access to the platform. Additionally, they set up a password to secure their account and personalize their experience.
* Task Management:
* The platform empowers users to organise their tasks efficiently by categorising them into customizable categories. These categories can represent various aspects of users' responsibilities, such as subjects for students or departments/teams for managers. Users have the flexibility to create, modify, and delete categories to suit their unique needs.
* Reward and Punishment System:
* A dynamic reward and punishment system is used for timely task completion and consistent progress. Users can earn bonus points for submitting tasks before the deadline and maintaining streaks within their categories. Conversely, late submissions result in point deductions. The system fosters a sense of accountability and motivates users to develop productive habits.
* React-based Dynamic Interface:
* Leveraging the power of React, the platform offers a responsive and interactive user interface. React's component-based architecture enables seamless navigation and intuitive task management. Components are thoughtfully designed and reusable, ensuring a consistent and efficient user experience across the application.
* Enhancement of Time Management Skills:
* By encouraging users to prioritise tasks and adhere to deadlines, the platform serves as a valuable tool for enhancing time management skills. Users learn to allocate their time effectively, avoid procrastination, and maintain a proactive approach to task completion. Real-time feedback and gamified elements further reinforce positive behaviours, fostering a culture of productivity and success.

System Architecture

The system architecture of a React application typically follows a component-based architecture. It focuses on reusability, modularity, and efficient updates through the virtual DOM. Additional libraries and patterns are often used to handle state management, routing, data fetching, and styling.

* 1. Components:
     + React applications are built using components, which are reusable and can be either class- based (using ES6 classes) or function-based (using React hooks).
     + Components can be further divided into smaller components, creating a hierarchy of components representing the UI structure.
  2. Virtual DOM (Document Object Model):
     + Instead of directly manipulating the DOM for every state change, React creates a virtual representation of the DOM in memory and updates it according to changes in application state.
     + When the state changes, React calculates the difference between the current virtual DOM and the previous one (reconciliation) and applies only the necessary updates to the actual DOM.
  3. State and Props:
     + State represents the data that a component owns and manages internally. It can change over time in response to user actions or other events.
     + Props (short for properties) are inputs to a component that are passed down from its parent component. Props are immutable and are used to configure a component and provide data to it.
  4. Component Lifecycle:
     + Class-based components have lifecycle methods that allow developers to hook into different stages of a component's life.
     + Function-based components can use React hooks like `useEffect` to achieve similar functionality.
  5. State Management:
     + For managing state across multiple components or complex state scenarios, React developers often use libraries like Redux, MobX, or React Context API.
     + These libraries provide a centralised way to manage state and make it easier to share data between components.
  6. Routing:
     + React applications typically use a routing library like React Router to manage navigation and rendering of different views based on the URL.
     + React Router allows developers to define routes and map them to specific components, enabling single-page applications with multiple views.
  7. Styling:
     + React applications can use various approaches for styling, including traditional CSS files, CSS-in-JS libraries like styled-components, or CSS preprocessors like Sass.
     + Each component can have its own styles scoped to that component, promoting modularity and encapsulation.



Source: https://supabase.com/docs/guides/getting-started/architecture

Components of Supabase’s system architecture:

1. PostgreSQL Database:
   * PostgreSQL is a powerful open-source relational database management system (RDBMS). PostgreSQL offers robust features for storing and managing structured data, including support for ACID transactions, JSONB data type for semi-structured data, and extensive indexing capabilities.
2. Real Time Event System:
   * Supabase includes a real time event system that enables developers to subscribe to changes in the database in real-time. This system is built on PostgreSQL's LISTEN/NOTIFY feature, which allows clients to receive notifications when specified events occur in the database.
   * Developers can subscribe to changes in specific database tables and receive real-time updates via WebSocket connections, enabling features such as live chat, real-time collaboration, and notifications.
3. REST API:
   * Supabase provides a RESTful API layer on top of the PostgreSQL database, allowing developers to interact with the database using HTTP endpoints. The REST API exposes CRUD (Create, Read, Update, Delete) operations for managing data in the database.
4. Authentication and Authorization:
   * Supabase includes built-in authentication and authorization functionality to secure access to web applications. It supports various authentication methods, including email/password, social login (e.g., Google, GitHub), and JSON Web Tokens (JWT). Developers can define access control policies to restrict access to specific data based on user roles and permissions.
5. Storage:
   * Supabase offers cloud storage capabilities for storing files and media assets associated with web applications. It integrates with popular cloud storage providers like AWS S3 and Google Cloud Storage to provide scalable and durable storage solutions.
   * Developers can upload, download, and manage files using the Supabase Storage API, which offers features like signed URLs for secure access and metadata management.
6. Realtime Database Triggers:
   * Supabase allows developers to define database triggers using PostgreSQL's PL/pgSQL language. Triggers are automatically executed in response to specified database events, such as INSERT, UPDATE, or DELETE operations on specific tables.
   * Developers can use database triggers to enforce data integrity constraints, perform data validation, or execute custom business logic.

##### Functional Requirements:

1. User Registration and Authentication:
   * Users must be able to register for an account using their email address.
   * Upon registration, users should receive a verification email to confirm their account.
   * Users must be able to log in securely using their email and password credentials.
   * Forgot password functionality should allow users to reset their password via email verification.
2. Task Management:
   * Users should be able to create, edit, and delete tasks.
   * Tasks must be categorised into customizable categories, allowing users to organise their tasks effectively.
   * Each task entry should include fields for task description, deadline, and category selection.
   * Users should be able to view their tasks sorted by category and deadline.
3. Reward and Punishment System:
   * The system must implement a reward and punishment system to incentivize timely task completion.
   * Users should earn points for completing tasks before the deadline and maintaining streaks within categories.
   * Points earned or deducted should be reflected in the user's profile and visible to the user.
4. Real-time Updates:
   * Users should receive real-time updates on task deadlines, points earned, and other relevant information.
   * Changes made by one user (e.g., task completion) should be immediately reflected to other users viewing the same data.
5. User Profile Management:
   * Users should be able to update their profile information, including email address and password.
   * User profiles should display relevant information such as total points earned and current streak status.

##### Non-Functional Requirements:

1. Performance:
   * The system should have minimal response times for user interactions, with pages loading within 2 seconds.
   * The backend should be able to handle concurrent user requests without significant degradation in performance.
2. Security:
   * User data must be stored securely, with sensitive information encrypted both in transit and at rest.
   * Authentication mechanisms should follow industry-standard practices, such as password hashing and salting.
   * Access to sensitive data and functionalities should be restricted based on user roles and permissions.
3. Usability:
   * The user interface should be intuitive and easy to navigate, with clear instructions and prompts.
   * Error messages should be descriptive and provide guidance on how to resolve issues.
   * The system should be accessible to users with disabilities, following WCAG guidelines for web accessibility.
4. Reliability:
   * The system should have high availability, with uptime of at least 99.9%.
   * Regular backups of the database should be performed to prevent data loss in case of system failures or disasters.
5. Scalability:
   * The system should be able to handle an increasing number of users and tasks without significant performance degradation.
   * Scalability should be achieved through horizontal scaling of resources, such as adding more server instances or utilising cloud-based solutions.

## Users and Roles

1. User Roles:

Users can sign up for the DeadlineDealer platform to manage their personal tasks and deadlines. They have access to features such as task creation, deadline setting, and earning rewards.

1. User Registration and Authentication:
   * Users can sign up for a DeadlineDealer account using their email address and set up a password.
   * Upon registration, users receive a verification email to confirm their email address.
   * Users must log in to access the platform's features, using email/password authentication or social login options.
2. User Profiles:
   * Each user has a profile associated with their account, containing information such as their name and email address.
   * Users can view and edit their profile information.
3. Task Management:
   * Users can create, edit, and delete tasks for personal use.
   * They can categorize tasks into customizable categories, such as subjects or projects.
   * Users can set deadlines for tasks and receive notifications for upcoming deadlines.
4. Security and Data Privacy:
   * User data is encrypted and stored securely in the database to ensure data privacy and protection.
   * The system implements best practices for authentication and authorization to prevent unauthorized access to user accounts and sensitive information.
   * Users have control over their data and can delete their account or request data exportation if needed.

## User Stories and Acceptance Criteria

|  |  |  |
| --- | --- | --- |
| ID | Feature Name | Story Points |
| 1 | SignUp in App | 5 |
| 2 | Login to App | 3 |
| 3 | Create Category | 7 |
| 4 | Add Task | 7 |
| 5 | Complete Task | 5 |

#### SPRINT 1

Estimated User Story Points: 8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Added | Description | Status | Story Points | Actual Equivalent Story Points | %  Completed |
| 1 | Onset | As a user, I want to Sign Up to the app and create an account | C | 5 | 4 | 100 |
| Acceptance Criteria | | | Verification | | | |
| 101 | The user should use a valid email ID | | Create test cases to verify that invalid email IDs cannot be used to create accounts | | | |
| 102 | Valid email IDs must be accepted by the app | | Create test cases to make sure that any valid email ID is accepted | | | |

|  |  |  |
| --- | --- | --- |
| ID | Task | Member |
| 1 | Create APIs to handle the signup process | Shruti Bansal |
| 2 | Create the frontend for the SignUp screen | Tejaswi M |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Added | Description | Status | Story Points | Actual Equivalent Story Points | %  Completed |
| 1 | Onset | As a user, I want to Login to the app to start using it | C | 3 | 4 | 100 |
| Acceptance Criteria | | | Verification | | | |
| 103 | The user should use the same email ID that they used to sign up | | Create test cases to verify that email IDs that are not accepted | | | |
| 104 | Registered email IDs must not be shown as invalid | | Test to make sure every email ID stored in the database is accepted | | | |

|  |  |  |
| --- | --- | --- |
| ID | Task | Member |
| 1 | Create APIs to handle the login process | Bharathi |
| 2 | Create the frontend for the login screen | Tejaswi |
| 3 | Use backend APIs to handle Forgot Password | Shruti |

#### SPRINT 2

Estimated User Story Points: 14

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Added | Description | Status | Story Points | Actual Equivalent Story Points | %  Completed |
| 1 | Onset | As a user, I want to add a task to my task list | I | 7 | 6 | 100 |
| Acceptance Criteria | | | Verification | | | |
| 201 | The user should be able to input task details | | Create test cases to verify the task addition function | | | |
| 202 | The user should be able to set task priority | | Ensure task priority can be set successfully | | | |

|  |  |  |
| --- | --- | --- |
| ID | Task | Member |
| 1 | Implementing form inputs for task details such as title, description, priority, etc. | Tejaswi |
| 2 | Implementing database operations to store task details. | Namit |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Added | Description | Status | Story Points | Actual Equivalent Story Points | %  Completed |
| 1 | Onset | As a user, I want to set deadlines for my tasks | I | 7 | 7 | 100 |
| Acceptance Criteria | | | Verification | | | |
| 203 | The user should be able to select a deadline date | | Ensure deadline selection interface functions | | | |
| 204 | The user should receive notifications for deadlines | | Test notification system for deadline reminders | | | |

|  |  |  |
| --- | --- | --- |
| ID | Task | Member |
| 1 | Adding a date picker component to select  deadlines. | Namit |
| 2 | Implementing logic for sending deadline notifications to  users. | Bharathi |

#### SPRINT 3

Estimated User Story Points: 8

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Added | Description | Status | Story Points | Actual Equivalent Story Points | %  Completed |
| 1 | Onset | As a user, I want to earn points and rewards for completing tasks | I | 5 | 5 | 100 |
| Acceptance Criteria | | | Verification | | | |
| 301 | The user should earn points based on task completion | | Test points calculation and allocation functionality | | | |
| 302 | Rewards should be provided upon reaching certain milestones | | Ensure rewards are granted accurately | | | |

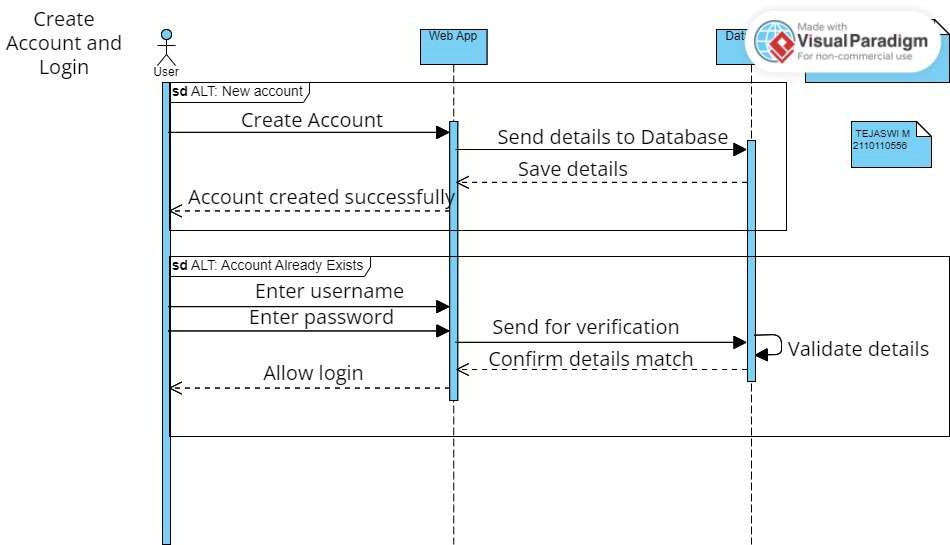
|  |  |  |
| --- | --- | --- |
| ID | Task | Member |
| 1 | Designing and implementing UI elements to display points,  rewards, and achievements. | Shruti |
| 2 | Storing user points and rewards data in the database. | Bharathi |

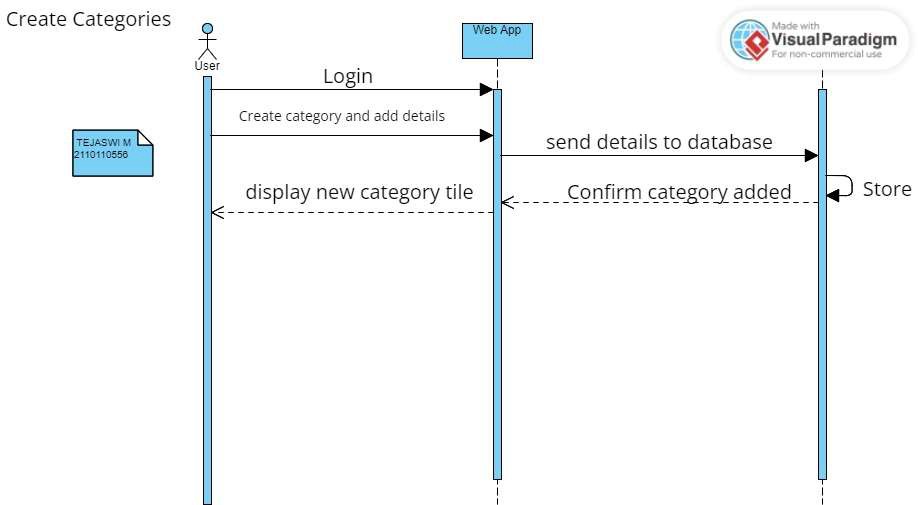
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Added | Description | Status | Story Points | Actual Equivalent Story Points | %  Completed |
| 1 | Onset | As a user, I want to view a checklist for each task | I | 3 | 4 | 100 |
| Acceptance Criteria | | | Verification | | | |
| 303 | The user should see a checklist for each task | | Test checklist display functionality | | | |
| 304 | The checklist should update as tasks are completed | | Verify checklist updates upon task completion | | | |

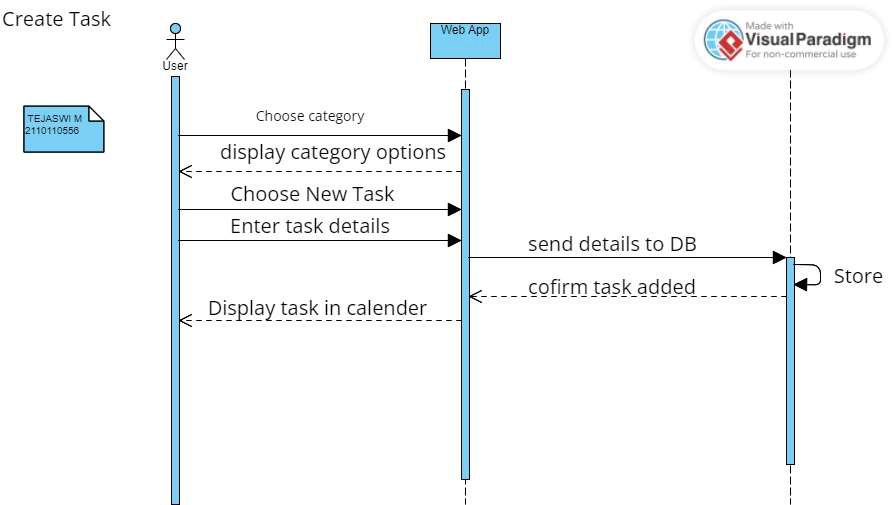
|  |  |  |
| --- | --- | --- |
| ID | Task | Member |
| 1 | Designing and implementing UI components to display task  checklists. | Bharathi |
| 2 | Implementing logic to synchronize checklist updates across  users. | Namit |

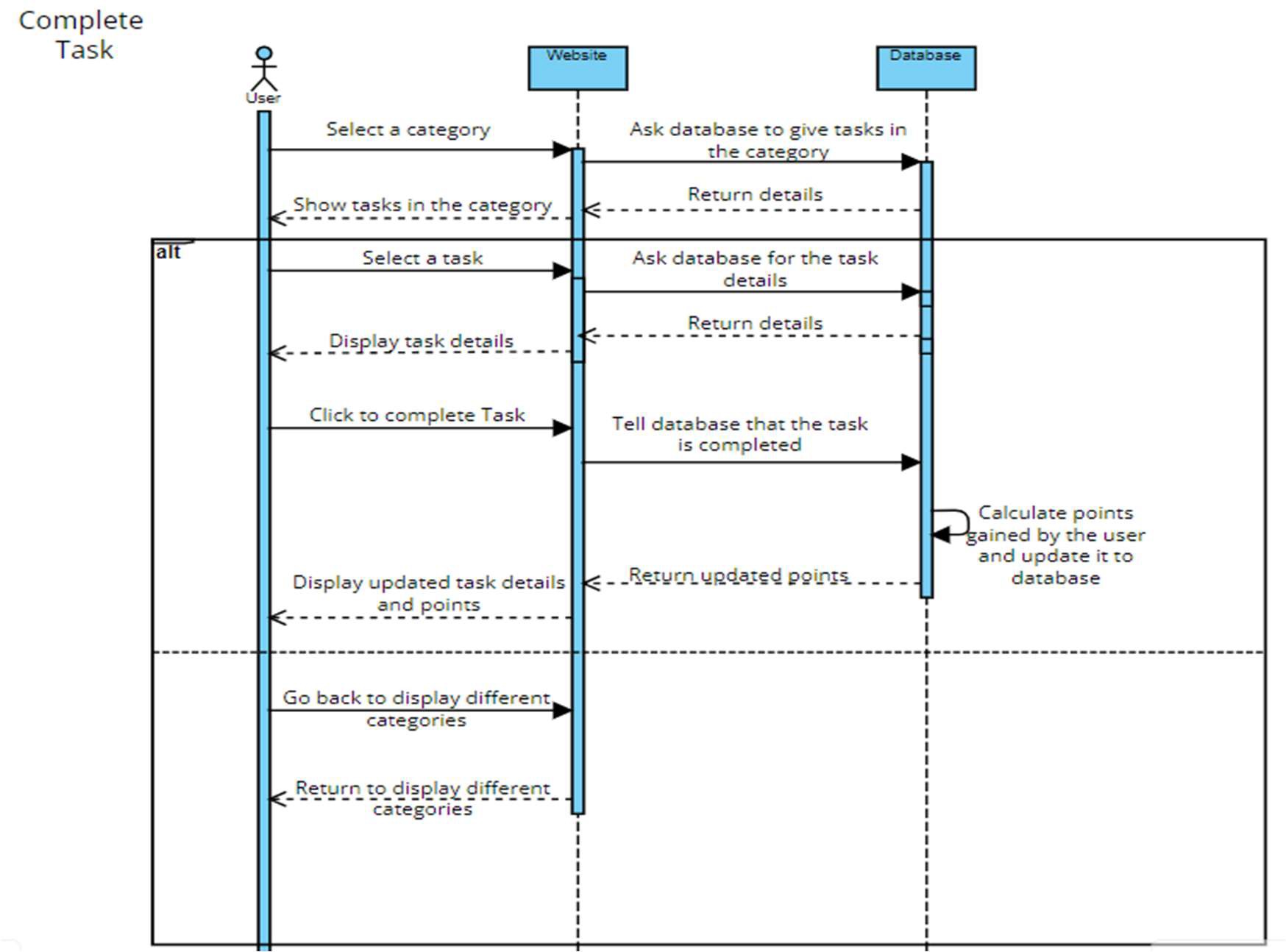
# Diagrams

## Sequence Diagrams

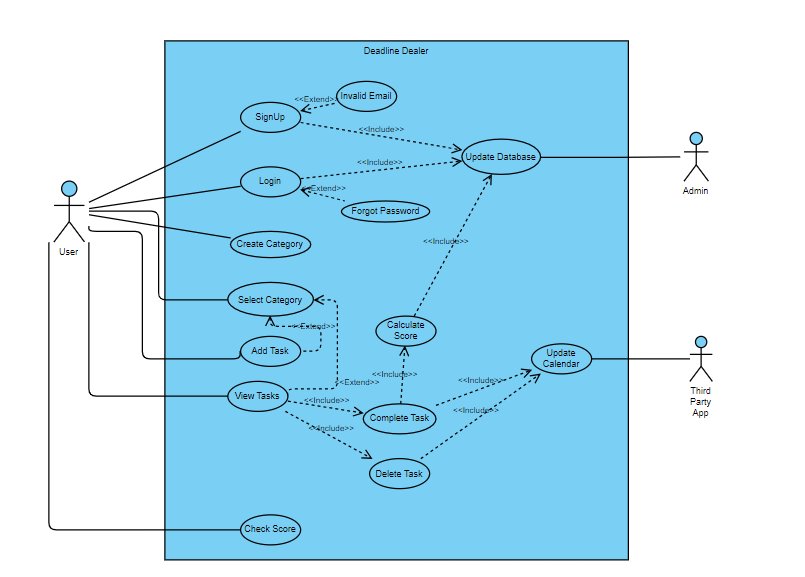








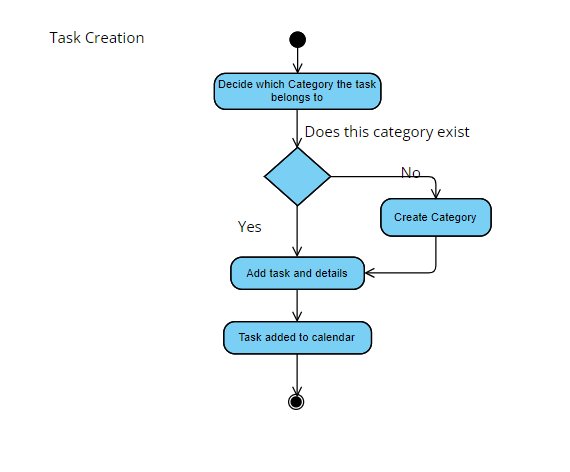
Use Case Diagram



Activity Diagram

A diagram of a web application

Description automatically generated



A diagram of a task completion

Description automatically generated

Class Diagram

A diagram of a calendar event

Description automatically generated

State Diagram

A diagram of a task

Description automatically generatedA diagram of a flowchart

Description automatically generated

# References and Resources

1. https://replit.com/
2. [https://replit.com/@ShrutiBansal5/SelfStudy#README.md](https://replit.com/%40ShrutiBansal5/SelfStudy#README.md)
3. https://developer.mozilla.org/en-US/docs/Web/JavaScript
4. https://react.dev/
5. https://legacy.reactjs.org/docs/getting-started.html
6. https://supabase.com/docs
7. https://developer.mozilla.org/en-US/docs/Web/CSS
8. https://github.com/vitejs/vite
9. https://psycnet.apa.org/buy/1987-05643-001
10. https://[www.ncbi.nlm.nih.gov/pmc/articles/PMC9857662/#:~:text=Cross%2Dsectional%20studie](http://www.ncbi.nlm.nih.gov/pmc/articles/PMC9857662/#%3A~%3Atext%3DCross%2Dsectional%20studie) s%20suggest%20that,loneliness%20and%20reduced%20life%20satisfaction.&text=Procrastinatio n%20is%20also%20associated%20with,13%20and%20unhealthy%20lifestyle%20behaviors.