GOVERNMENT POLYTECHNIC NAGPUR

DEPARTMENT OF INFORMATION TECHNOLOGY

SOFTWARE ENGINEERING [IT206G]

MINI-PROJECT

TO-DO WEB APPLICATON

Under the guidance of *Prof. Lokesh Bhadekar*

Group Members

Sujal Lokhande [2207034]

Rachana Vazirani [2207046]

Abhishek Sarate [2207050]

Prutha Shesh [2207053]

COMMUNICATION PHASE

Effective communication was pivotal throughout our ToDo List Web Application project. It ensured seamless collaboration, stakeholder involvement, and issue resolution.

Throughout the initial meetup with the customer, the following problem statement for the software was drawn:

To create a web application that allows users to create a To-Do List where they can easily manage their tasks in a check list format.

Which was then further elaborated and classified into functional and non-functional requirements in the latter sessions:

Functional requirements

1. **Interactive UI**
   The UI should be user-friendly and responsive, allowing seamless interaction with tasks.

2. **User Registration**
   New users should be able to easily create accounts, providing necessary information for registration.

3. **User Login**
   Registered users should securely log in to access their personalized task lists.

4. **Task According to each user**
   Users should only have access to their own tasks, ensuring privacy and organization.

5. **Database connectivity**
   The application should connect to a database to store and retrieve user data and task information efficiently.

Non-Functional requirements

1. **Secure**
   Data security measures should be implemented to protect user information from unauthorized access.

2. **Prevention of Unauthorized Access**
   Access controls should be in place to prevent unauthorized users from accessing sensitive data.

3. **Reliable**
   The application should operate consistently and reliably, ensuring uninterrupted access to tasks.

4. **Collaborative Build Environment for Further Development**
   The development environment should support collaboration among team members for continuous improvement and feature enhancements.

CONCLUSION

Effective communication and thorough requirements gathering were instrumental in our project's success. By engaging stakeholders and addressing their needs, we developed a ToDo List Web Application that meets user expectations and quality standards.

In response to the established problem statement and all requirements defined by the stakeholders, a Software Requirements Specification document was presented to the stakeholders by our team.

SOFTWARE REQUIREMENTS SPECIFICATION

Title: To Do List Web Application

Table of Contents:

Introduction

1.1 Purpose

To create a web application that allows users to create a To-Do List where they can easily manage their tasks in a checklist format.

1.2 Scope

The application will keep track of user tasks.
It will allow users to add new tasks, modify existing tasks and remove them when they are completed.
The application will not enable users to add any task description for details regarding their tasks.

1.3 Definitions, Acronyms, and Abbreviations

Definitions:
Username - Here, it conveys a unique variable for identifying users.
Password - Password is used for user's privacy.
Tasks - A set of work instructions that a user needs to complete within a certain amount of time.
Superuser - Highly authorized admin to keep track of the database.

Acronyms and abbreviations:
UI- User Interface
DOM- Document Object Model

1.4 References

E-References:
1. www.djangoproject.com
2. www.github.com/loksujal123/SE-project.git
3. www.geeksforgeeks.org

Books used:
HTML and CSS: Design and Build Websites by Jon Duckett
The Django book by Adrian Holovaty, Jacob Kaplan-Moss, et al.

1.6 Overview

This SRS consists of a detailed description of all functionalities provided by our project, any constraints that may apply, and an overall description of what the project will and will not do.

Overall Description

2.1 Product Perspective

The ToDo list web application is a user-friendly tool designed to practice task management as a digital platform for creating, editing, and deleting tasks in a checklist format. It offers simplicity and accessibility through a web browser interface.

2.2 Product Functions

1. User registration (signup)
2. User login
3. Adding and Updating tasks
4. A superuser login for tracking Integrity and Database.
5. Easy admin panel.
6. Fast and easy interaction throughout the program.

2.3 User Characteristics

Goal-oriented: Users have specific tasks they need to accomplish and are looking for a tool to help them prioritize and manage their tasks efficiently.

Privacy-conscious: Users value the privacy and security of their task lists. They expect the application to provide secure login features and protect their personal information.

[Such users may include students, professionals, or anyone with a need to organize their daily activities.]

2.4 Constraints

- Simple hashing algorithm that can lead to unauthorized data viewing.
- Technologies used here is less interactive with DOM of webpage.
- Descriptive task allocation not possible.
- Too simple UI interactions

2.5 Assumptions and Dependencies

Assumptions:

It is assumed that the user should be aware of the following:

-Basics flow of account creation and authentication rights.
-Copyrights for designs of UI.
-Basic navigation of simple webapps.
-Awareness for proper inputs throughout the usage of application.

Dependencies:

Internet Connectivity: Users must have reliable internet access to use the ToDo list web application as the application's functionality and accessibility depend on a stable internet connection.

Web Browser Compatibility: The application's usability relies on compatibility with commonly used web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. Compatibility issues with less popular browsers may affect user experience.

Device Compatibility: The application is assumed to be compatible with various devices, including desktop computers, laptops, tablets, and smartphones. However, limitations may arise in terms of screen size, resolution, and processing power on certain devices.

Specific Requirements

3.1External Interfaces

- Back-end framework, i.e. Django (Python framework for websites).
- Admin panel UI from model classes (provide by Django).

- SQL database.
- Google form for collecting feedback from users.

3.2 Functions

User Registration and Verification: Allow users to create an account with a unique username and password.
Verify users securely upon login to access their task lists.
Implement password hashing and encryption for enhanced security.

Task Management: Enable users to create new tasks.
Provide functionality to edit existing tasks. Allow users to mark tasks as completed or delete them when no longer needed.

Superuser/Admin Functionality: Implement administrative functions for managing user accounts and tasks, monitoring system activity, including user interactions and task completion rates.

3.3Performance Requirements

Page Load Time: The application's main page should load within 2 seconds on standard internet connections.

Server Uptime: The server hosting the application should maintain at least 99.9% uptime over a given period, ensuring high availability for users.

Scalability: The application should be designed to scale horizontally to accommodate increasing user loads without degradation in performance.

Task Creation and Updation: The application should allow users to create new tasks and update existing tasks with minimal latency, aiming for response times of under 500 milliseconds.

Feedback and Notifications: Feedback messages for task operations should be displayed instantly to users, with response times of under 200 milliseconds.

Any notifications for upcoming task deadlines or reminders should be delivered promptly, ensuring timely alerts for users.

Database Performance: Database queries for retrieving user-specific data should execute within 300 milliseconds.

3.4 Logical Database Requirements

The data entities stored in the database will be as follows:

Username: Stores a unique identity to identify user when he/she claims to login to application.
Frequency of username is once per user with no additional data entities. It relates with other data in database to verify and authenticate user.

Email: Used for tasks updates and alerts. It is defined only once for a user. Only used for authentication of user, not to verify login.

Password: This piece of data is encrypted and hashed using a simple unsalted algorithm and then forwarded to for salt hashing to store in database. This data is very confidential and is only known to the user.
Only stored for once per user.

Superuser: Highly privileged (admin) for managing data of users. Not more than 5 times for the application. The Superuser would relate with all rows in the database (for all users).

3.5 Design Constraints

Testing on smaller screens is not possible.

3.6 Software System quality attributes

Usability: Users would be able to easily navigate through the application and perform tasks such as adding, modifying, and removing tasks.

Reliability: The application will accurately keep track of users' tasks and will be robust enough to handle errors or unexpected inputs gracefully.

Performance: The application would respond promptly to user actions such as adding or removing tasks and be able to handle a reasonable number of concurrent users without degradation in performance.

Security: User authentication mechanisms will ensure that only authorized users can access and modify their tasks and users' data is abstracted from other users.

Maintainability: The codebase of the application would be well-organized and documented to facilitate future updates and enhancements and to make it easier for developers to understand and modify the application as needed.

Compatibility: The application would be compatible with a wide range of web browsers and devices to ensure a seamless user experience.

Appendices

- GDPR (General Data Protection Regulation)
- User Guidelines – Pre-defined instructions regarding ethical and proper use of software
- Data Model – Pre-defined regulatory standards and procedures.

Index

Model used for project development:

Along with the requirements stated by the customer, the customer also specified that the project must be completed in the least time possible. For this, they were willing to provide with any increased cost of development.

Hence, our team concluded that the Rapid Application Development (RAD) model would be the most suitable model for the development of this project.

PLANNING PHASE

After all requirements are specified and approved by all stakeholders, we moved towards the planning phase of software development which included:

Scheduling
Tracking
Estimation

Scheduling and Tracking

For our project scheduling, we established the sequential tasks required for project completion along with the time required for each task to be completed.

Also, all tasks were divided amongst all team members, keeping in mind their knowledge, ease, and efficiency regarding their specific tasks.

Appropriate milestones were established on regular intervals to monitor the progress of software development.

With all time specifications of individual tasks and their respective checkpoints, we were able to create the following table defining all sequential tasks and their scheduled time.

Tasks Specifications for GANTT Chart:

WORK TASKS	Planned Start	Actual Start	Planned Complete	Actual Complete
1. Project Initiation 1.1. Meet with customer 1.2. Identify needs and project constraints 1.3. Establish problem statement 1.4. Presenting SRS Milestone: project statement defined.	w1d1 w1d2 w1d2 w1d4	w1d1 w1d2 w1d2 w1d4	w1d1 w1d2 w1d3 w1d4	w1d1 w1d2 w1d3 w1d4
2. Development of plan 2.1. Cost & Time estimation 2.2. Identifying & Assigning Tasks 2.3. Scheduling individual tasks Milestone: Project plan established	w1d5 w2d1 w2d2	w1d5 w2d1 w2d2	w1d5 w2d1 w2d2	w1d5 w2d1 w2d2
3. Developing models & basic structure 3.1. Design flow of activities 3.2. Identify no. of modules 3.3. Identify interdependency between modules 3.4. Develop design plan for project Milestone: Models and designs defined	w2d3 w2d3 w2d3 w2d4	w2d3 w2d3 w2d3 w2d4	w2d4 w2d4 w2d4 w3d1	w2d4 w2d4 w2d4 w3d1
4. Construction/Coding 4.1. Cloning of UI (frontend - HTML5, CSS3) 4.2. Database creation (backend) 4.3. Adding security 4.4. Testing Milestone: Implemented final construction	w3d2 w3d5 w4d3 w4d5	w3d2 w3d5 w4d3 w4d5	w3d4 w4d2 w4d4 w5d1	w3d4 w4d2 w4d4 w5d1
5. Deployment of project 5.1. Deployment 5.2. Maintenance 5.3. Feedback Milestone: Deployed finished software (Final Project Completion)	W5d2 W5d3 W6d2	W5d2 W5d3 W6d2	W5d2 W5d5 W6d2	W5d2 W5d5 W6d2

With the help of the above table, we were able to design a GANTT Chart to represent all scheduled tasks and the timeline for each of them:

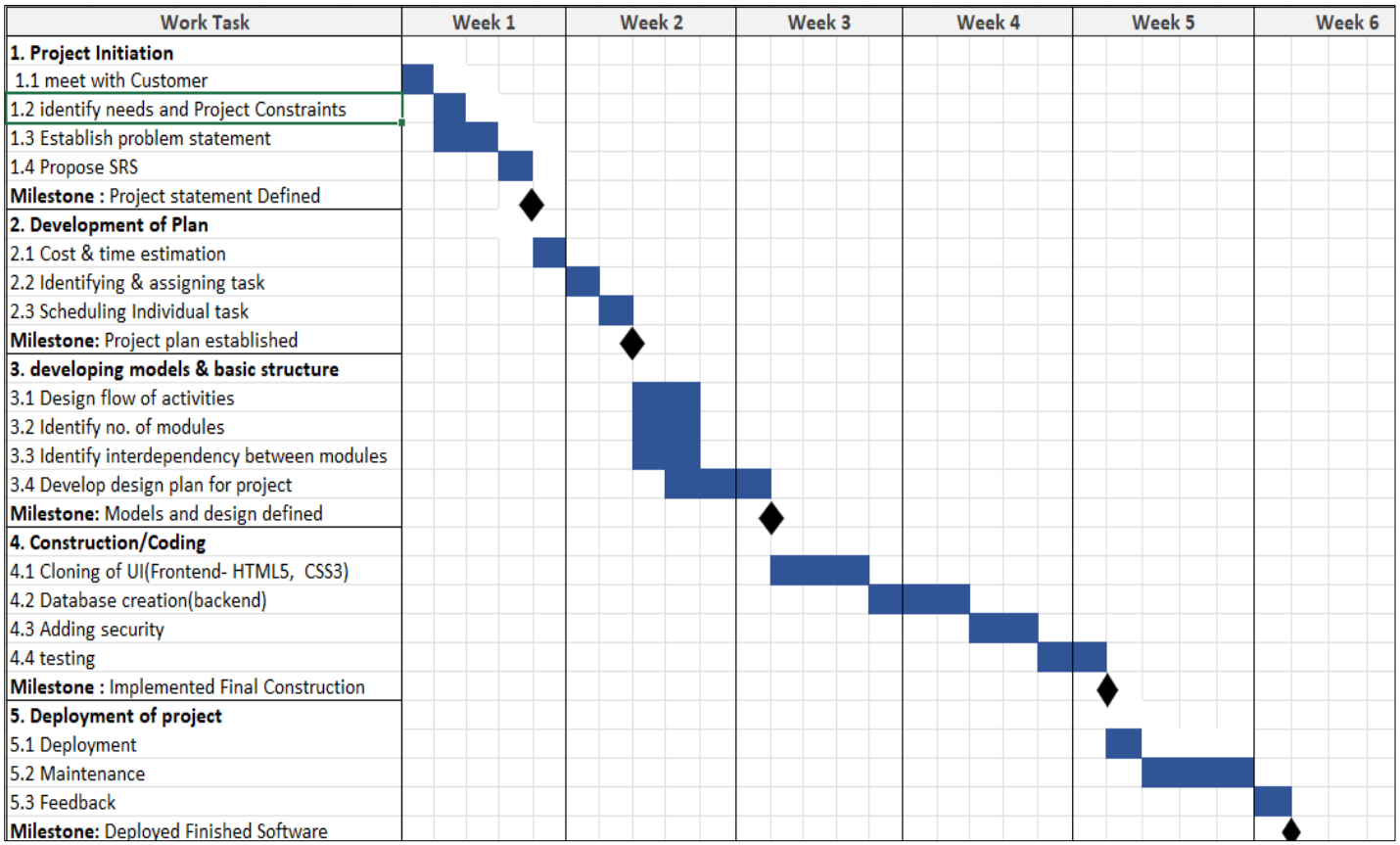


Fig. GANTT Chart

PROJECT ESTIMATION

Time Estimation

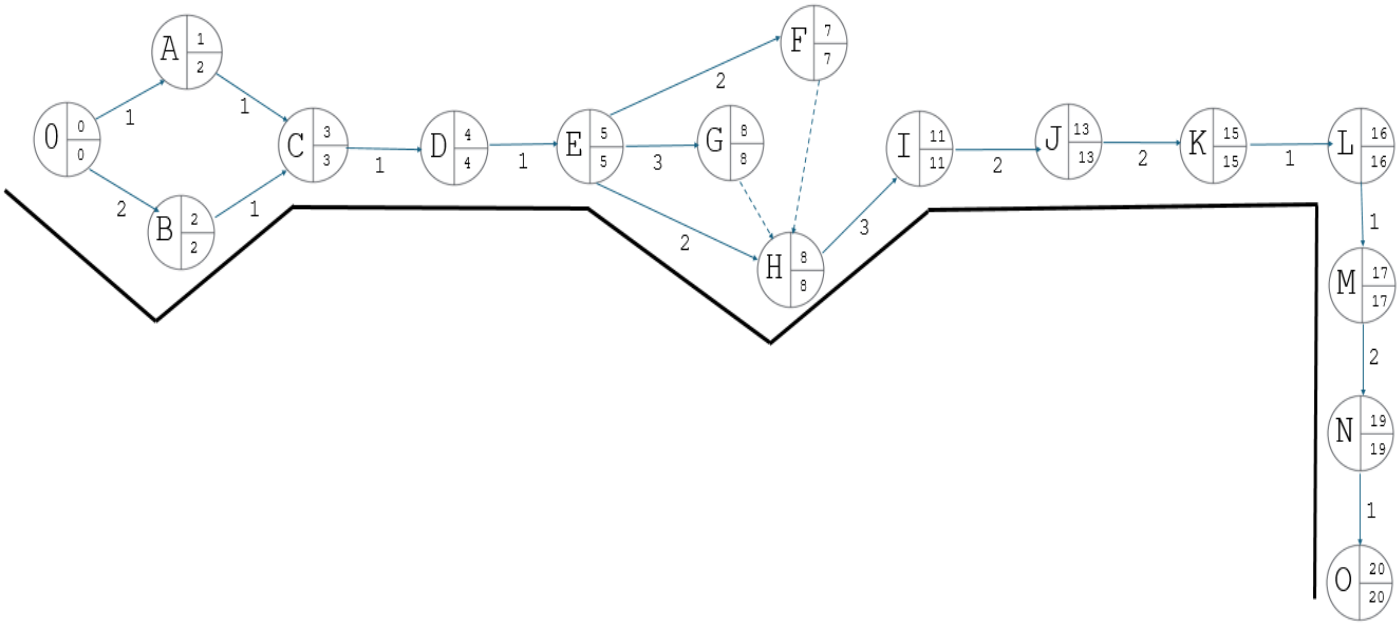
PERT Chart (Project Evaluation Review Technique)

To estimate the minimum time required for project completion, we have used the PERT Chart which helps in graphically representing the sequence of tasks to be carried out, the relation or dependency between the tasks and identifying simultaneous completion of tasks, if any.

Specification Table for Task Nodes:

Task ID	Task Description	Prec. ID	Succ. ID	Duration
A	Identify needs & project constraints	-	C	1
B	Establish problem statement	-	C	2
C	Propose SRS	A, B	D	1
D	Cost & Time estimation	C	E	1
E	Assign & schedule tasks	D	F, G, H	1
F	Design flow of activities	E	I	2
G	Identify modules	E	I	2
H	Develop design plan	E	I	2
I	Cloning of UI	H	J	3
J	Database creation	I	K	2
K	Adding Security	J	L	2
L	Testing	K	M	1
M	Deployment	L	N	1
N	Maintenance	M	N	2
O	Feedback	N		1

With the help of the above table defining task nodes, their relation between each other and their individual time durations, we were able to design a PERT Chart for our web application.



CPM = 0-B-C-D-E-H-I-J-K-L-M-N-O

= 2+1+1+1+2+3+2+2+1+1+2+1

= 19

COST ESTIMATION:

Size Oriented Metrics

Project Tasks	LOC	Effort	Errors	People
UI Coding	594	2	5	2
Database Creation and Integration	115	3	1	1
URLs Mapping	141	1	2	2
Settings	125	1	0	1
Admin Panel	50	1	0	1

MODELLING PHASE

In the modelling phase, we have designed various charts and diagrams which provide a better understanding for our project creation and the functionalities that our project must provide.

Use Case Diagram

Based on the set requirements and functionalities of our project, we were able to design a Use Case diagram to summarize the details of our system's users (actors) and their interactions with the system.

This diagram was used to only visualize the inputs, outputs, and the functions of the system, without any description of their implementation.

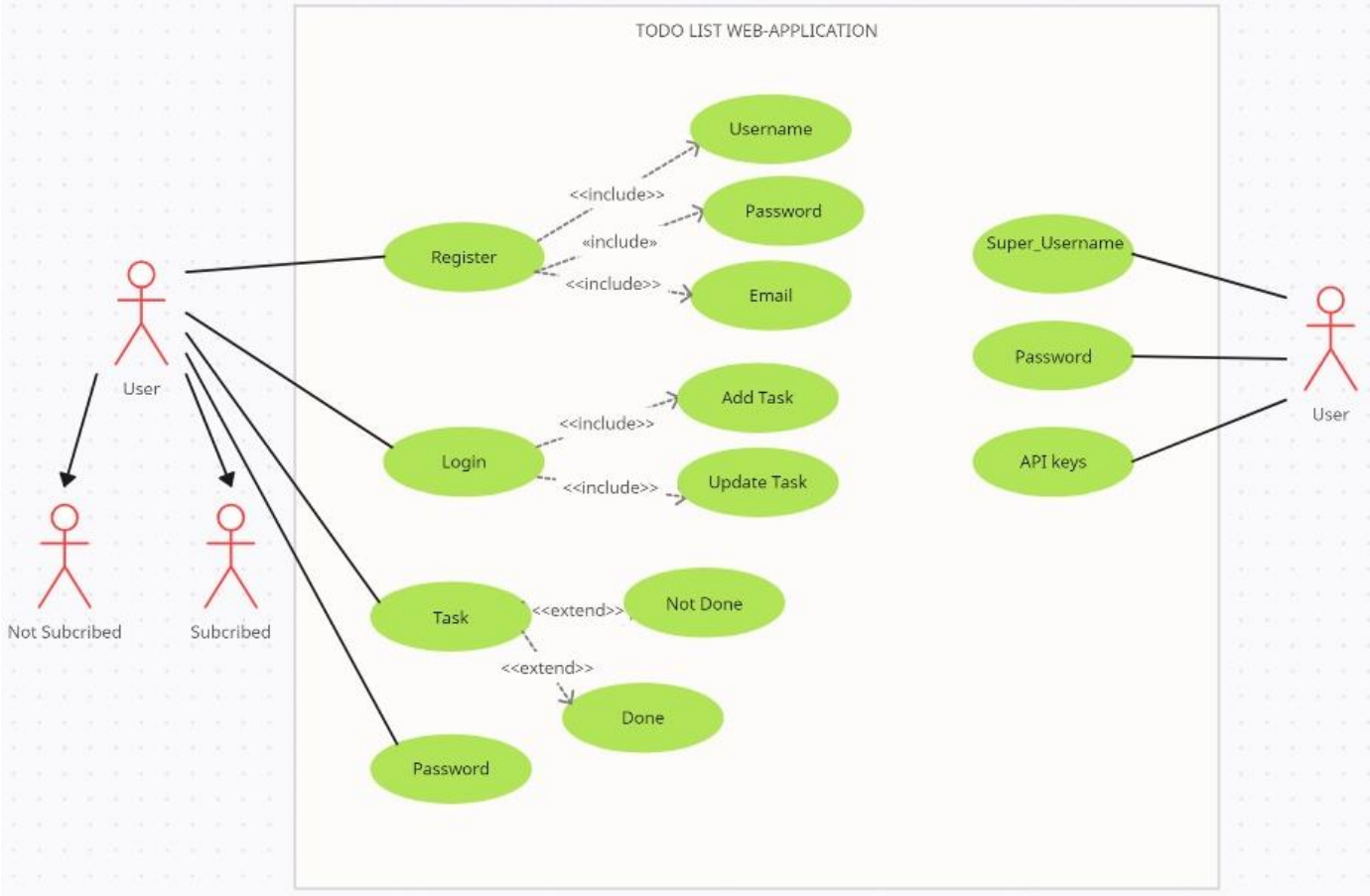


Fig. Use Case Diagram

State Transition Diagram

To describe the behaviour of our system, the following State Transition diagram was designed which consists of the various finite states of the system:

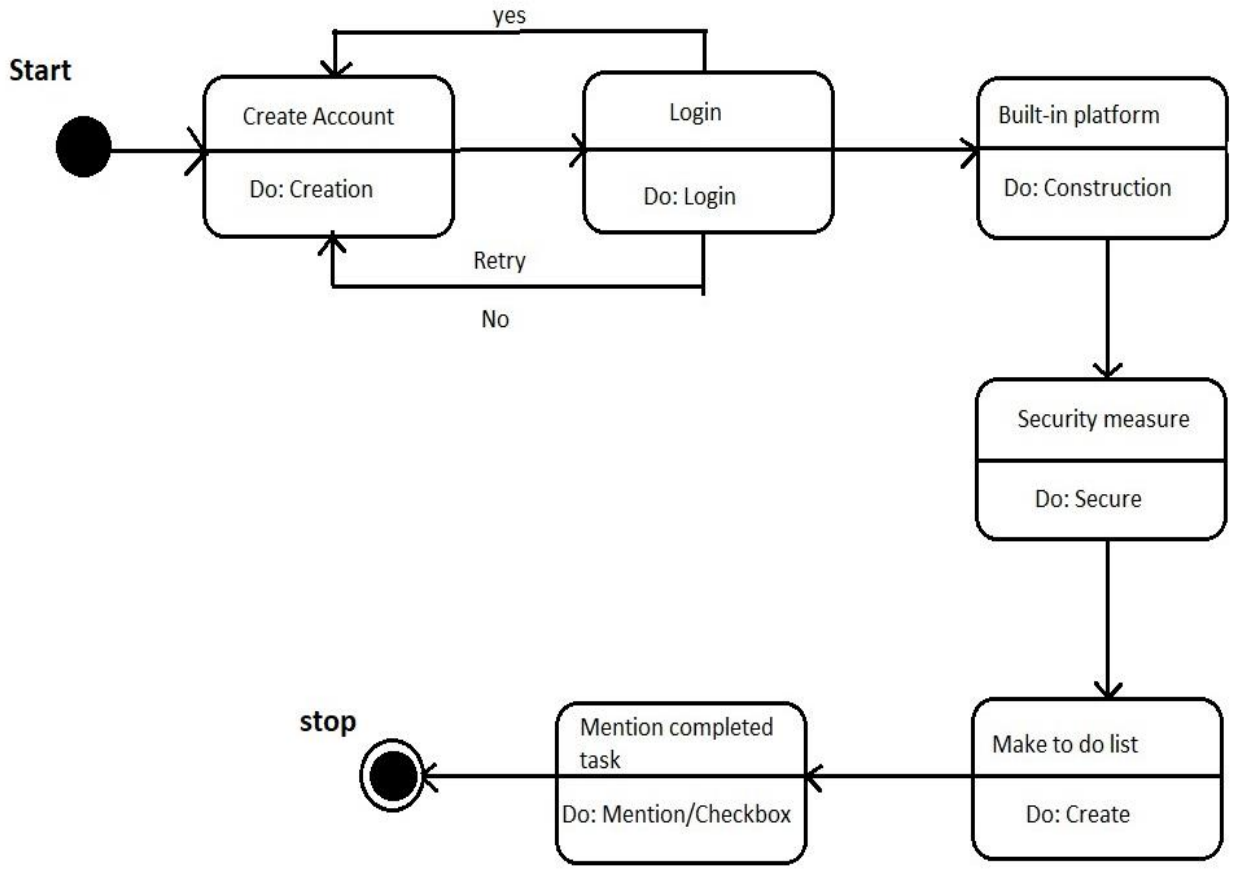


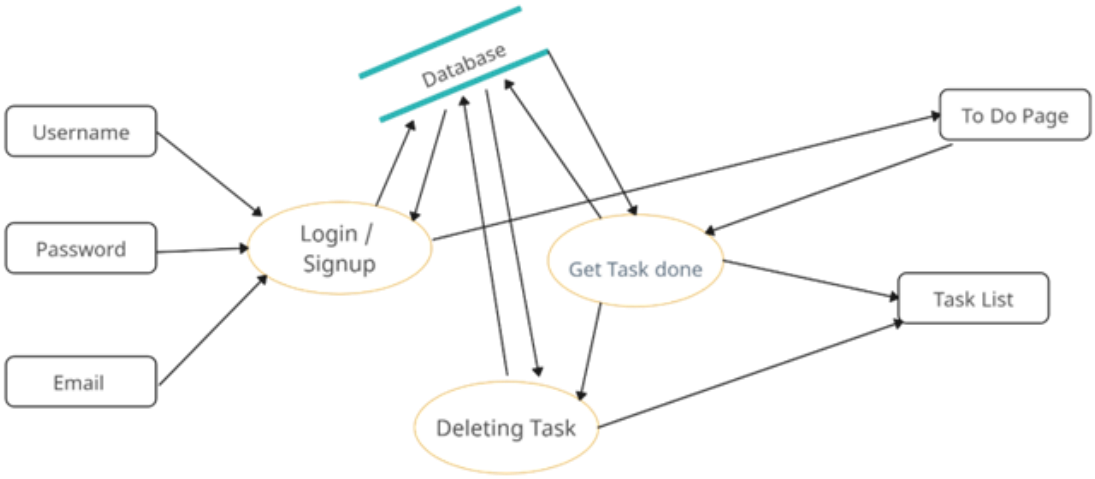
Fig. State Transition Diagram

Data Flow Diagram

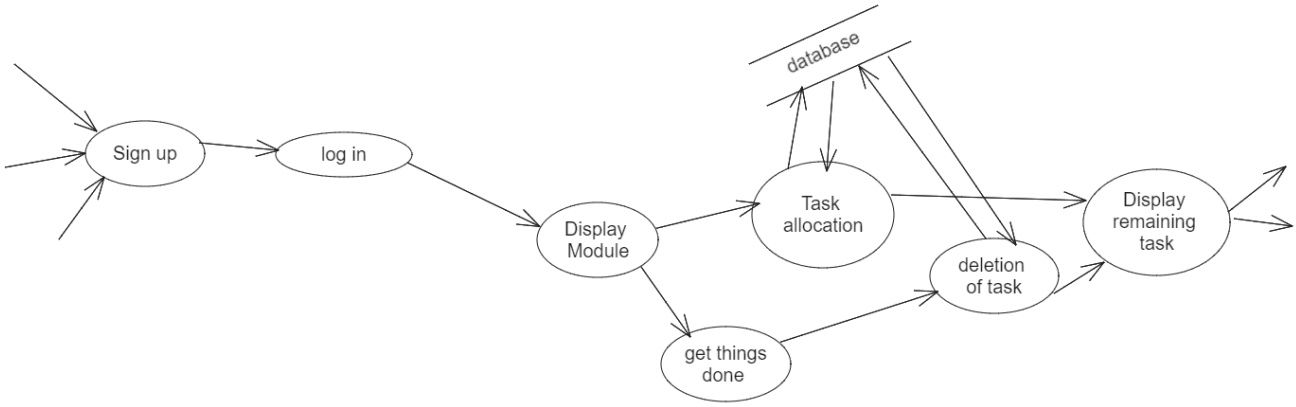
We have used the Data Flow Diagram to graphically represent the flow of data in the system and the processes that are involved in it to transfer data:

Level - 0:

Level - 1:



Level - 2:



Activity Diagram

The Activity diagram was designed to represent the sequential or parallel flow of activities throughout the system.

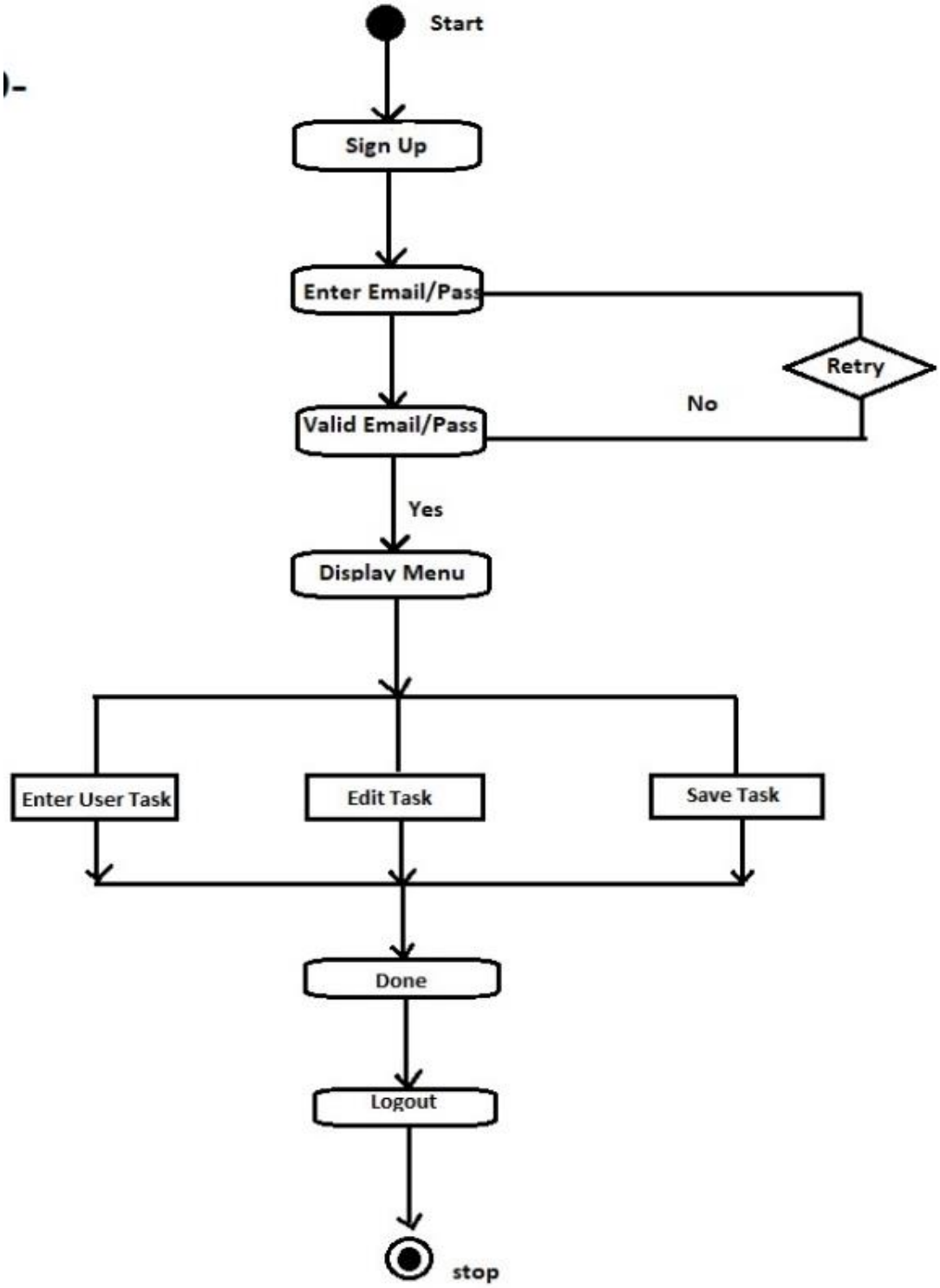


Fig. Activity Diagram

Sequence Diagram

The Sequence diagram was then designed to represent the various actors involved in the functioning of the system and how they interact with each other sequentially for the system to function successfully.

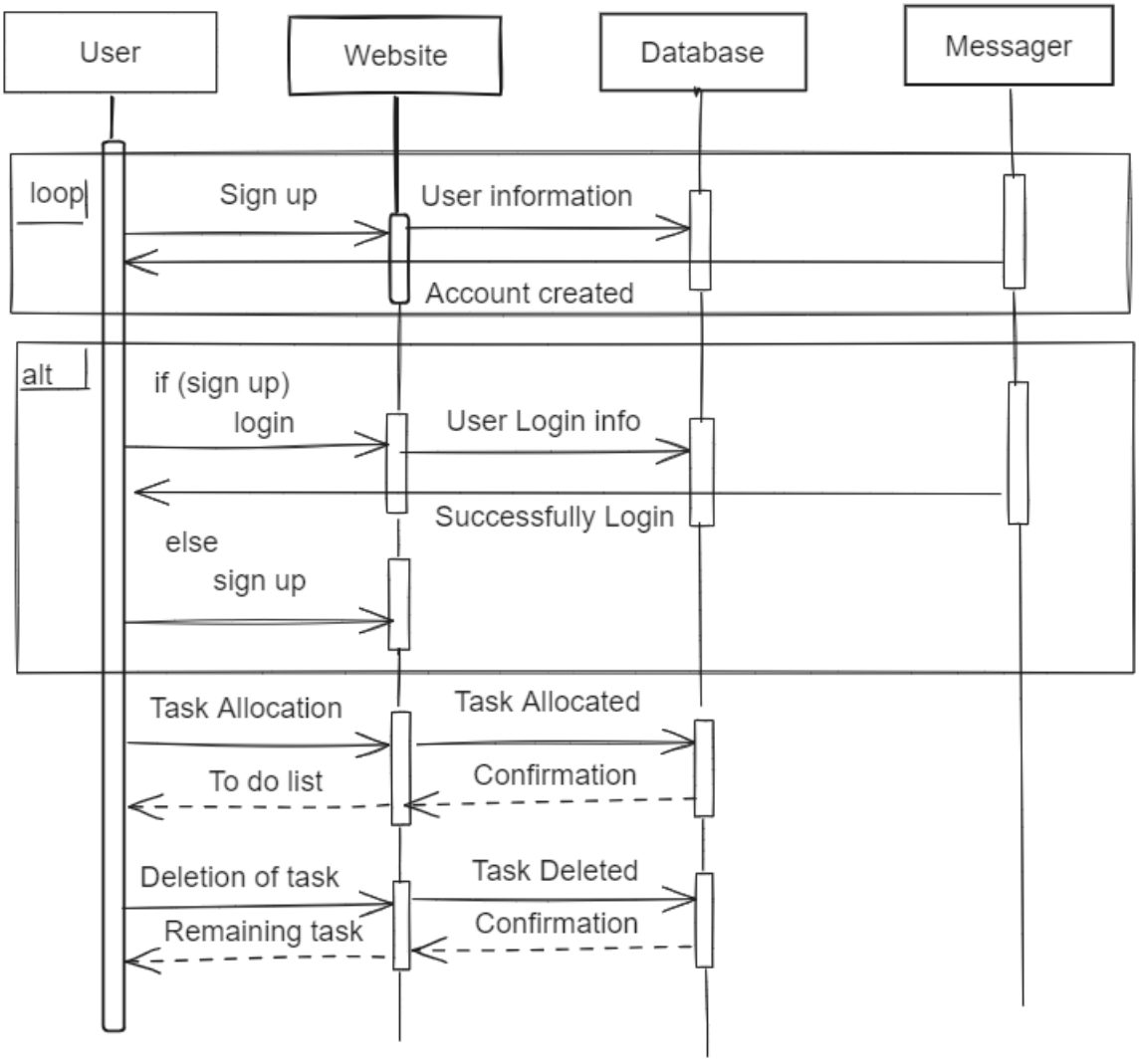


Fig. Sequence Diagram

Entity-Relationship Diagram

We designed the ER model/diagram to represent various entities that would be active throughout the functioning of the software and would interact with each other. The relations between such entities are also shown according to their behaviour and interaction with each other.

These entities include:
User - The end user that will access the application for task management.

Website - The UI that the user will interact with.

Database - (supervised by the admin) that will store users' and their task's information.

Task - The tasks that the user will add or modify using the application.

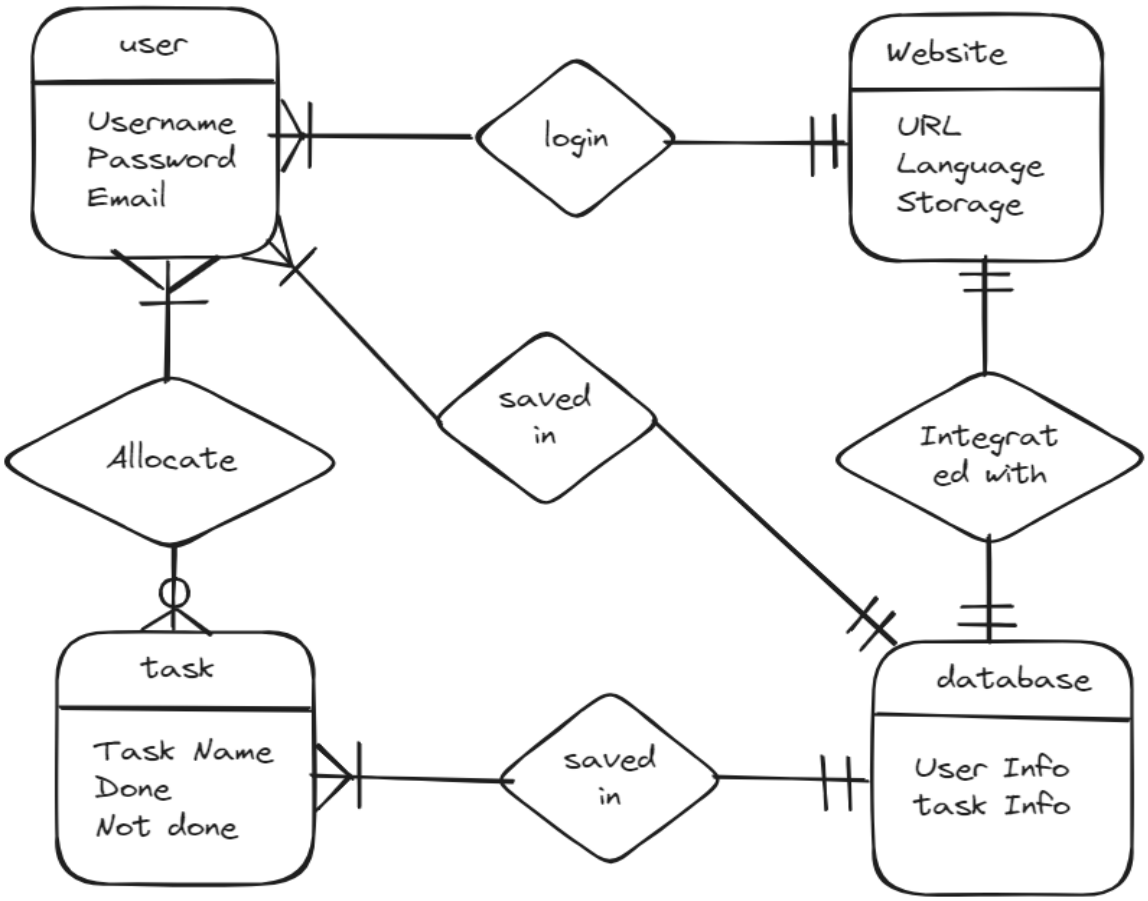


Fig. Entity-Relationship Diagram

RISK:

Security concerns:

1. Lacking of user authentication
2. Password saved in simple hashing algorithm with less frequent iterations.

Risk Mitigation, Monitoring, and Management

Risk is a potential problem, it is a possibility of experiencing loss, if the risk came to occur in a software project it will lead to the failure of the organization, it will lead in economic as well as calendar time loss. Thus, it is very essential to manage the risk before it causes loss.

Risk Mitigation:

The proactive approaches adopted to manage the risk and avoidance of the risk to happen in further processes is termed as mitigation of risk. Using decorators to avoid the unauthorized navigation of ToDo page also improves privacy concern would lead to mitigate the list before it occurs lack in security.

Risk Monitoring:

The information gathered through mitigation leading the manager to manage the risk by establishing the schedule is called as monitoring the list.in the given scenario we can monitor the risk through continuous testing, debugging tools for testing multiple inputs and outputs for multiple iteration.

Risk Management:

When the risk has actually happened and mitigation efforts has failed thus, risk will be managed at last. in the above scenario we can manage the risk by verifying error with global documentation of framework used and requesting the customer for extension of deadline. Also requesting multiple teams to collaborate with team with risk for managing the risk.

Risk Projection by developing Risk Table:

Risks	Category	Probability	Impact
Failure of system if tons of users register at once.	PS	50%	1
Hashing algorithm could be cracked	BU	30%	2
Delayed or failed loading of Icons in UI	TE	20%	3
User logout not registered	TE	10%	2
Invalid tasks status updates due to lack of awareness by admin	BU	10%	4