SOFTWARE REQUIREMENTS SPECIFICATION

For

ToDo List

Prepared by:-
Ashik S
Dinesh V
Aravind M
Vignesh G

Academic Year: 2023-2024

1. Introduction

1.1 Purpose

The primary aim of this document is to outline the essential requirements for the Full Stack web project, "To-Do List."The purpose of the "To-Do List" project is to create a user-friendly digital environment that facilitates the efficient management of tasks and to-do items. This project is dedicated to simplifying task organization and tracking by leveraging web technologies. It aims to empower users to effortlessly add, modify, prioritize, and mark tasks as completed, thereby enhancing their productivity and time management.

1.2 Document Conventions

➔ Entire document should be justified.
➔ Convention for Main title
- Font face: Arial
- Font style: Bold
- Font Size: 14

➔ Convention for Sub title
- Font face: Arial
- Font style: Bold
- Font Size: 12

➔ Convention for body
- Font face: Arial
- Font Size: 11

1.3 Scope of Development Project

The scope of the To-Do List web project encompasses the development of a user-friendly task management system. Users will be able to create, view, edit, and delete tasks, complete with details like titles, descriptions, due dates, priorities, tags, and attachments. The project will involve both frontend and backend development, with a responsive UI/UX built using modern web technologies. User authentication and registration will be implemented securely, along with personalization options and task notifications. The application may also include collaboration features and search/filtering capabilities. Robust security measures, testing, and documentation are essential components, followed by deployment to a web server or cloud platform. Scalability, monitoring, and support considerations are optional but vital for long-term success, ensuring the project can adapt and evolve as needed in the future.

Furthermore, the project will prioritize data integrity and user experience by incorporating a well-structured database system for efficient task storage. Backend development will employ a chosen framework, implementing RESTful APIs to handle task operations and ensuring data validation and error handling. The application's security will be a paramount concern, with measures in place to safeguard against common web vulnerabilities. Testing will encompass unit and integration tests to guarantee functionality, while comprehensive documentation will

guide both users and developers. The project's deployment will involve setting up the application on a web server or cloud platform, and provisions for domain configuration and SSL for secure access will be made as necessary. The scope will also encompass maintenance and support plans, ensuring the project's continued excellence in performance and user satisfaction. Future enhancements will be considered to keep the project adaptable and aligned with evolving user needs and technological advancements.

1.4 Definitions, Acronyms and Abbreviations

JAVA -> platform independence
SQL-> Structured query Language
ER-> Entity Relationship
UML -> Unified Modeling Language
IDE-> Integrated Development Environment
SRS-> Software Requirement Specification
ISBN -> International Standard Book Number
IEEE ->Institute of Electrical and Electronics Engineers

1.5 References

➔ **Books**

- "Full Stack Web Development with Vue.js and Node" by Aneeta Sharma, Santhosh Kumar, and Vikram Kriplaney
- "Learning JavaScript Design Patterns" by Addy Osmani - Understanding design patterns is essential for creating maintainable and scalable web applications.
- "Node.js Design Patterns" by Mario Casciaro - This book explores design patterns specifically for Node.js, which is useful if you're using Node.js in your project's backend.

➔ **Websites**

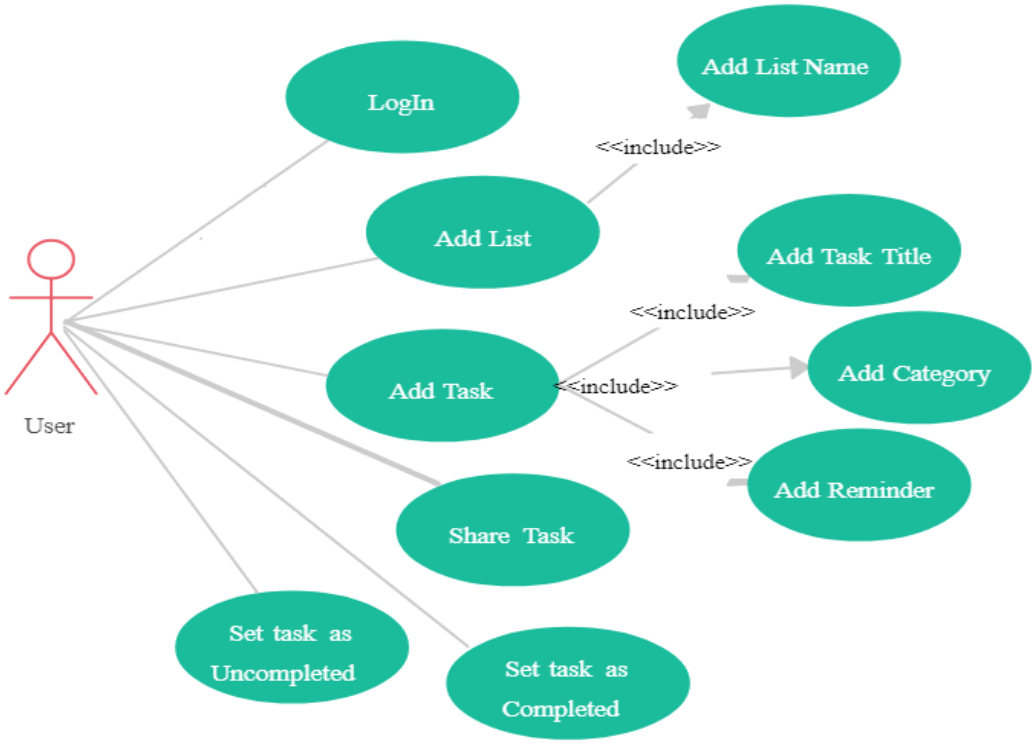
- https://www.w3schools.com
- https://react.dev/learn

2. Overall Descriptions

2.1 Product Perspective

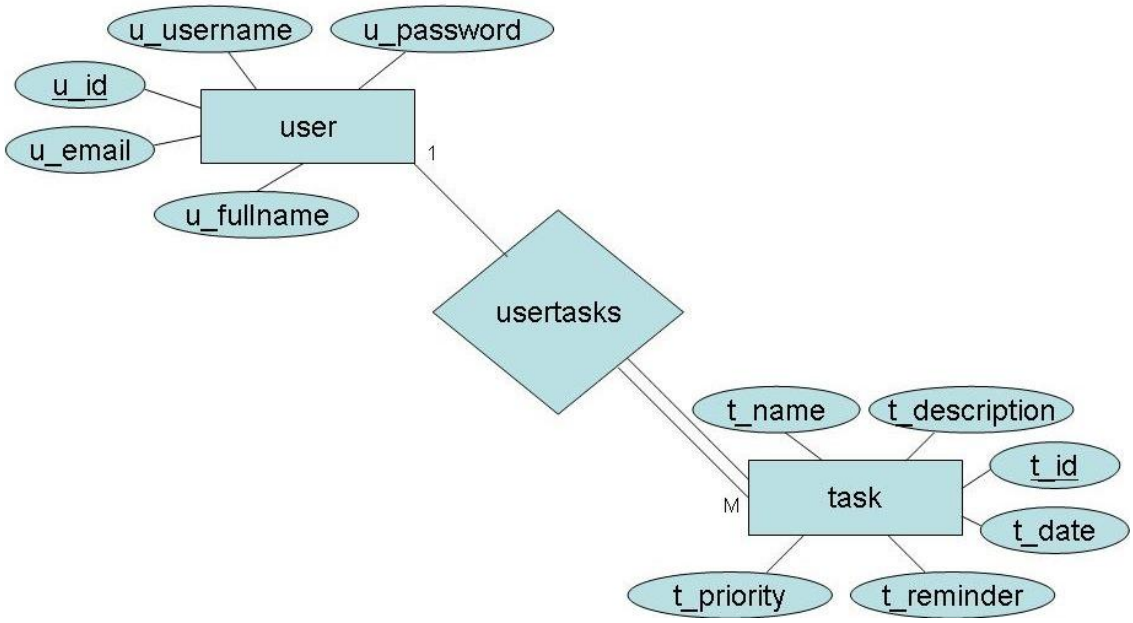
Use Case Diagram of ToDo List

The To-Do List Web Application offers a user-friendly and efficient task management solution for individuals and teams. Users can register and log in to their accounts, creating, editing, and deleting tasks with ease. Tasks can be organized with titles, descriptions, due dates, and priority levels. The application empowers users to customize their profiles and receive timely task notifications. Admin users enjoy elevated privileges for managing accounts, monitoring activities, and system maintenance. Users can also categorize tasks, search for specific items, and view task histories, enhancing their productivity and organization. Overall, the product perspective revolves around seamless task management, personalization, and efficient control for both regular and admin users.



2.2 Product Function

Entity Relationship Diagram of ToDo List



The Entity Relationship Diagram (ERD) for the To-Do List Web Application illustrates the fundamental data structure and relationships within the system. The core entities include User

and Task. Users, identified by a User ID, possess attributes like Name, Email, Password, Profile Picture, and User Type (admin or regular). The User entity is linked to Task through a one-to-many relationship, signifying that a user can have multiple tasks associated with them. Tasks, identified by a Task ID, have properties including Title, Description, Due Date, Priority Level, and Completion Status. The Task entity, in turn, is associated with the User entity via a one-to-many relationship, indicating that each task is linked to a single user. This ERD offers a comprehensive view of the data model underpinning the To-Do List Web Application, facilitating effective task management and user interactions**.**

2.3 User Classes and Characteristics

The To-Do List Web Application serves different user classes, each with specific characteristics and privileges.

1. Regular Users:

- Characteristics: Regular users are individuals who uses application for personal task management.
- Privileges:
  - ❖ Create, edit, and delete tasks.
  - ❖ Set due dates and priorities for tasks.
  - ❖ Categorize tasks with tags or categories.
  - ❖ Customize profile settings (e.g., profile picture, display name).
  - ❖ View and manage their own tasks.
  - ❖ Receive notifications for upcoming tasks or deadlines.
  - ❖ Search for specific tasks.
  - ❖ Sort and filter tasks based on various criteria.

2. Admin Users (Administrators):

- Characteristics: Admin users have elevated privileges and control over the application.
- Privileges:
  - ❖ All privileges of regular users.
  - ❖ Manage user accounts and access rights.
  - ❖ Monitor user activities and task histories.
  - ❖ Add or remove users.
  - ❖ Access and modify application settings.
  - ❖ View reports and analytics on user behavior.
  - ❖ Perform system maintenance and updates.
  - ❖ Provide customer support and resolve user issues.

The To-Do List Web Application distinguishes between these user classes to provide a tailored user experience, ensuring that regular users can efficiently manage their tasks while administrators maintain oversight and control over the platform.

2.4 Operating Environment

The To-Do List web application will operate in a cross-platform environment, accessible through modern web browsers. It will be compatible with the following popular browsers:

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Apple Safari
- Other modern browsers

The application will not be limited to a specific operating system and should function seamlessly on Windows, macOS, Linux, and mobile operating systems.

Users will require an internet connection to access and use the application.

Hardware Configuration (Suggested):

The web application will not impose specific hardware requirements on users' devices.
Users should have access to a computer, tablet, or smartphone with a web browser and an internet connection.

2.5 Assumptions and Dependencies

Assumptions:

- ➢ Error-Free Development: It is assumed that the development process will adhere to best practices, resulting in an error-free application.
- ➢ User-Friendly Interface: The system will be designed to be user-friendly, ensuring ease of use for all users.
- ➢ Database Storage: All task-related information will be stored in a secure and accessible database.
- ➢ Performance and Scalability: The system should be capable of handling a growing number of users and tasks, providing fast access to data.
- ➢ Search Functionality: The application will offer search capabilities and support efficient task management.
- ➢ 24/7 Availability: The To-Do List application is expected to be available 24 hours a day for user access.
- ➢ User Authentication: Users will need correct usernames and passwords to access their accounts and perform actions.

Dependencies:

- ➢ Hardware and Software: The application's functionality will depend on the availability of specific hardware and software components required for web development and hosting.
- ➢ Requirements and Specifications: The project's development and operation will depend on clearly defined requirements and specifications.

- ➢ User Understanding: End-users, particularly administrators, must have a thorough understanding of the application to use it effectively.
- ➢ Report Storage: The system should have the capability to store and retrieve reports as required.
- ➢ Database Access: The application relies on access to a database to store and manage task-related data. Updates to the database must be accurate.
- ➢ Internet Connection: Users must have a stable internet connection to access and use the web-based To-Do List application.

2.6 Requirement

Software Configuration:

The To-Do List project will be developed using modern web technologies for both the front end and the back end.

Front End:

Language: React framework for front-end development.

Development Environment: Node.js for managing packages and dependencies.

Integrated Development Environment (IDE): Popular code editors like Visual Studio Code, Sublime Text, or similar for writing React code.

Back End:

Language: Node.js for server-side scripting.

Database: MongoDB as the back-end database for storing task-related data.

Development Environment: Node.js runtime environment.

Operating System Compatibility:

The application should be compatible with a range of operating systems, including Windows, macOS, and Linux.

2.7 Data Requirement

The application's primary function is to facilitate the creation, management, and tracking of tasks. Input queries from users will lead to database interactions, and the application will provide real-time updates and responses to ensure a smooth user experience.

The technology stack chosen for the project is well-suited for developing a robust and efficient To-Do List web application, offering scalability and flexibility for future enhancements and features.

3. External Interface Requirement

3.1 GUI (Graphical User Interface)

The To-Do List web application will provide an intuitive and user-friendly graphical interface for both users and administrators. Here are the key aspects of the GUI:

- ➢ User Interface: The software will feature a visually appealing and responsive user interface accessible through web browsers. It should be easy to navigate and interact with tasks.
- ➢ Task Management: Users will be able to create, edit, delete, and prioritize tasks using the graphical interface. Tasks will be displayed in an organized and user-friendly manner.
- ➢ Reports: The application will offer quick access to reports, such as task completion status, deadlines, and productivity insights.
- ➢ Modular Design: All modules within the application will integrate seamlessly into the graphical user interface while adhering to a consistent design template. This ensures a uniform user experience.
- ➢ Login Interface: Users will be able to register and create accounts. The login interface will prompt users to enter their usernames and passwords. Incorrect login attempts will trigger error messages for user feedback.
- ➢ Categories View: The application may provide a categorization feature for tasks, allowing users to organize tasks into different categories or lists. Librarians or administrators may have control over adding, editing, or deleting task categories.
- ➢ User Control Panel: For administrators or power users, a control panel will be available to manage users, tasks, and other resources. This panel will allow tasks to be added, edited, or removed, as well as options for managing user access and permissions.

4. System Features

User Authentication and Validation:

- User Authentication: Users will be required to authenticate themselves through unique credentials, such as usernames and passwords, to access their accounts and tasks.
- Validation: The system will validate user credentials to ensure that only authorized individuals can log in and use the application.

User Data Privacy:

- Account Privacy: User accounts will be designed with privacy in mind, ensuring that each user can only access and manage their own tasks and data.
- Administrator Access: Only administrators will have the authority to access and manage all user accounts and data, ensuring data privacy and security for regular users.

5. Other Non-functional Requirements

5.1 Performance Requirements:

The To-Do List project must meet performance expectations to ensure an efficient and responsive user experience. Here are the performance requirements:

- Fast and Accurate: The system should provide fast and accurate performance, allowing users to create, update, and manage tasks without delays.

- Error Handling: The application should handle both expected and unexpected errors gracefully, preventing data loss and minimizing downtime. For example, invalid username/password errors should be managed effectively.
- Scalability: The system should be capable of handling a large volume of data, accommodating a high number of tasks and users without encountering faults or performance degradation.

5.2 Safety Requirements:

To ensure data safety and system reliability, the project should meet the following safety requirements:

- Database Backup: Regular database backups should be implemented to prevent data loss due to unexpected events like viruses or operating system failures.
- Power Backup: Adequate power backup solutions, such as UPS or inverters, should be in place to ensure system availability during power supply failures.

5.3 Security Requirements:

Security is paramount for user data and system integrity. Here are the security requirements:

- Secured Database: The system should use a secured database to protect user data from unauthorized access.
- User Access Control: Normal users should have read-only access to information, with limited permissions to edit or modify only their personal information.
- User Authentication: Strong user authentication mechanisms should be in place to verify user identities and prevent unauthorized access.
- Password Security: User passwords should be stored securely, and the system should prevent any form of password hacking.
- Admin Privileges: Administrators should have separate accounts with rights to update the database, while regular members should not be able to access the database directly.

5.4 Requirement Attributes:

- Multiple Admins: Multiple administrators will have the authority to make changes to the system, ensuring flexibility and collaboration.
- Open Source: The project should be open source, allowing for community contributions and transparency.
- User-Friendly: The system should prioritize user-friendliness to make it accessible and easy to use for all users.
- Download and Installation: Users should be able to easily download, install, and set up the system without complexity.

5.5 Business Rules:

Business rules and regulations should be enforced to ensure compliance with project policies, costs, and discount offers. Users should adhere to legal rules and protocols, avoiding any illegal activities.

5.6 User Requirements:

- User Knowledge: Users are members and administrators of the system. Members are assumed to have basic computer and internet browsing knowledge, while administrators should have a deeper understanding of system maintenance.
- User Education: Proper user interfaces, user manuals, online help, and installation guides should be provided to educate users on system usage and maintenance.
- Admin Facilities: Administrators should have access to facilities such as backup and recovery, password recovery, data migration, data replication, auto-recovery, file organization, and regular server maintenance.

6. Other Requirements

6.1 Data and Category Requirements:

In the context of the To-Do List project, there are different categories of users, including teaching staff, administrators, students, and more. Depending on the user's category, access rights and permissions are determined. For example:

- Administrators: Administrators have comprehensive rights to modify data, delete entries, append information, and perform other administrative tasks.
- Regular Users: Users, other than administrators and librarians, have read-only access to retrieve information from the database.

Similarly, there will be different categories of tasks or to-do items. Depending on the category of tasks, relevant data should be displayed to users. The categories and associated data should be structured and coded in a particular format to ensure effective categorization and retrieval.

6.2 Appendix:

An appendix is a section of the document where additional information, references, or related content is provided. It often includes abbreviations, acronyms, assumptions, and other relevant information. In the context of the To-Do List project, the following elements could be included in the appendix:

- Abbreviations and Acronyms: A list of abbreviations and acronyms used in the document and the project.
- Assumptions: Assumptions made during project planning or document creation, which can help in understanding the project context.
- Business Rules: Specific rules and regulations that users must adhere to when using the To-Do List application.

6.3 Glossary:

The glossary section provides definitions and explanations of terms, conventions, and acronyms used throughout the document and the project. Here are some examples:

- Administrator: A user with administrative privileges, responsible for managing the system.
- User: A general login ID assigned to most users of the system.
- Client: The intended users and stakeholders of the software.
- SQL: Abbreviation for Structured Query Language, used to retrieve information from a database.
- SQL Server: A database server used to store data in an organized format.
- Layer: A component or section of the project architecture.
- User Interface Layer: The part of the project where users directly interact with the application.
- Application Logic Layer: The web server component responsible for processing computations and logic.
- Data Storage Layer: The section of the project where data is stored and retrieved.
- Use Case: A high-level diagram illustrating the project's functionality and interactions.
- Class Diagram: A type of static structure diagram that describes the system's structure, including classes, attributes, and relationships.
- Interface: A mechanism used for communication between different components or systems.
- Unique Key: A unique identifier used to differentiate entries or records in a database.

6.4 Class Diagram