SOFTWARE REQUIREMENTS SPECIFICATION

For

To-Do List

Done by:-

Abarna S
Avanthika Y
Deepthi Lakshitha S
Sumithra J M

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 To-Do List web project's scope includes creating an approachable task management system. Tasks with attributes like titles, descriptions, due dates, priorities, tags, and attachments can be created, viewed, edited, and deleted by users. Both frontend and backend development will be done for this project, and cutting-edge web technologies will be used to create a responsive UI/UX. Task notifications, customisation choices, and user login and registration will all be implemented securely. Additionally, the programme could provide search and filtering options as well as collaborative features.Thorough security protocols, testing, and documentation are crucial elements, which should be followed by deployment to a cloud platform or web server. Though optional, scalability, monitoring, and support factors are essential for long-term success since they guarantee the project's ability to change and grow as circumstances demand.

In addition, the project will incorporate an organized database system for effective task storage, placing a high priority on data integrity and user experience. A selected framework will be used in backend development, along with RESTful APIs to manage task operations, data validation, and error handling. The security of the application will be of utmost importance, and precautions will be taken to prevent common web vulnerabilities. Unit and integration tests will be used in testing to ensure functionality, and thorough documentation will serve as a reference for developers and users alike. When the project is deployed, the application will be set up on a web server or cloud platform, and any necessary domain configuration and SSL setup will be completed for secure access. Plans for upkeep and support will also be included in the scope, guaranteeing the project's continuous excellence in performance and user satisfaction. To maintain project adaptability and alignment with changing user needs and technological advancements, future enhancements will be taken into consideration.

1.4 <u>Definitions, Acronyms and Abbreviations</u>

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 <u>References</u>

➔ **Books**
- "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.
- "Full Stack Web Development with Vue.js and Node" by Aneeta Sharma, Santhosh Kumar, and Vikram Kriplaney

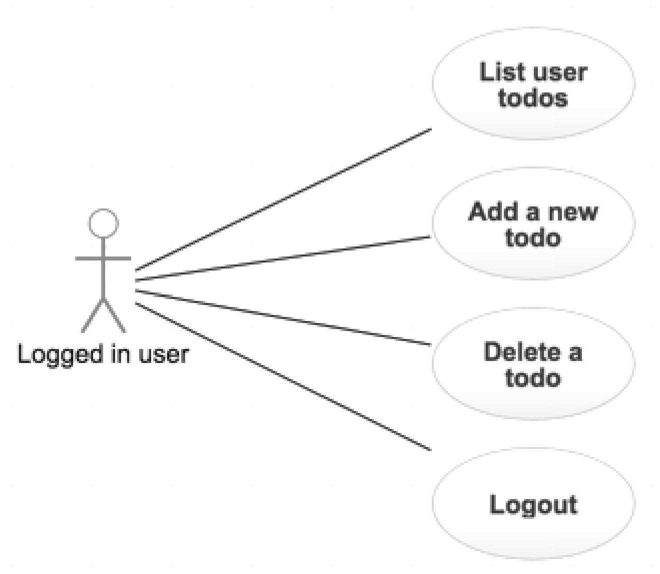
➔ **Websites**
- **https://www.w3schools.com**
- **https://react.dev/learn**

2. <u>Overall Descriptions</u>

2.1 <u>Product Perspective</u>

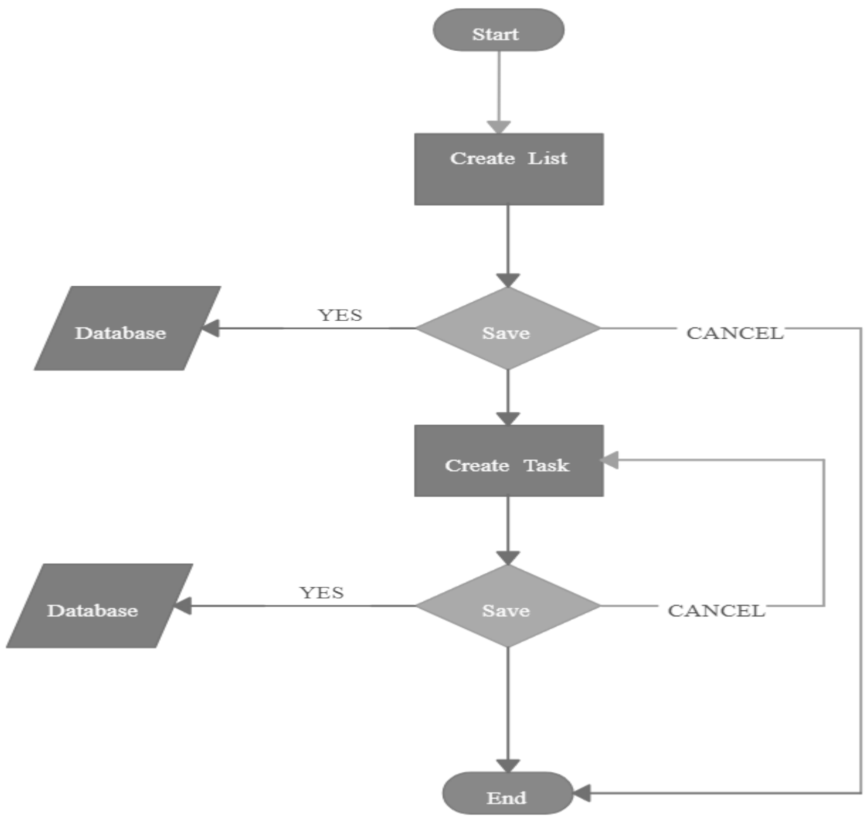
<u>Use Case Diagram of ToDo List</u>

For both individuals and teams, the To-Do List Web Application provides an easy-to-use and effective task management solution. Users may easily create, edit, and delete tasks after registering and logging in to their accounts. It is possible to arrange tasks according to their priority, titles, descriptions, and due dates. Users of the application can personalize their profiles and get timely task notifications. Admin users have more rights for account management, activity monitoring, and system upkeep. In order to improve productivity and organization, users can also view task histories, categorize tasks, and search for particular items. From a product standpoint, the main focus is on smooth task management, customization, and effective control for both regular and administrator users.



2.2 **Product Function**

Entity Relationship Diagram of ToDo List



The Entity Relationship Diagram (ERD) for the To-Do List Web Application shows the key data structures and connections in the system.

The main entities are User and Task. Users, identified by a User ID, have attributes such as Name, Email, Password, Profile Picture, and User Type (admin or regular). The User entity links to Task through a one-to-many relationship, meaning a user can have multiple associated tasks.

Tasks, identified by a Task ID, have features like Title, Description, Due Date, Priority Level, and Completion Status. In turn, the Task entity connects to the User entity through a one-to-many relationship, signifying each task links to one user. This ERD provides a comprehensive model of the data framework underpinning the To-Do List Web Application, enabling efficient task organization 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. Users:

Regular users are individuals employing the application to enhance the efficiency of managing their personal tasks within the framework of a to-do list app. They are able to perform the following tasks:

a. Create, modify, and remove tasks.
a. Assign due dates and priorities to tasks.
b. Categorize tasks using tags or categories.
c. Personalize profile settings (e.g., profile picture, display name).
d. Access and manage their own tasks.
e. Receive notifications for upcoming tasks or deadlines.
f. Conduct searches for specific tasks.
g. Utilize sorting and filtering options based on various criteria.
.

2. Admin:

Admin users within the Todo List app hold elevated privileges and command over the system. This empowers them to perform crucial tasks such as:

a. All privileges of regular users.
a. Manage user accounts and control access rights.
b. Monitor user activities and task histories.
c. Add or remove users.
d. Access and adjust application settings.
e. Review reports and analytics on user behavior.
f. Execute system maintenance and updates.
g. Provide customer support and address user issues.

The To-Do List Web Application distinguishes between these user classes to deliver a tailored user experience. Regular users efficiently manage their tasks, while administrators maintain oversight and control, ensuring the platform's smooth operation.

2.4 <u>Operating Environment</u>

The To-Do List web application will work across different platforms and be accessible through current web browsers. It will be compatible with these well-known browsers:

Google Chrome

Mozilla Firefox

Microsoft Edge

Apple Safari, and other modern browsers.

This comprehensive browser support guarantees a consistent user experience, regardless of the chosen platform. The To-Do List app embraces inclusivity and is not restricted to any specific operating system. It is engineered to perform efficiently on various environments, including:
Windows
macOS
Linux
Mobile operating systems like iOS and Android.

<u>Hardware Configuration</u>:

The web application will not have any particular hardware needs for users' devices. Users should be able to access the application using a computer, tablet, or smartphone that has an internet browser and internet connection.

<u>2.5 Assumptions and Dependencies</u>

<u>Assumptions:</u>

- ➢ Robust Development Practices: The development process is anticipated to strictly adhere to industry best practices, ensuring a meticulously crafted and error-free application.
- ➢ Intuitive User Interface: The system will boast an intuitive and user-friendly interface, prioritizing simplicity to enhance usability for all users.
- ➢ Secure Database Storage: Task-related information will be securely stored in a database, combining accessibility with robust security measures.
- ➢ Optimized Performance and Scalability: The system is engineered to handle a growing user base and expanding task volumes, guaranteeing swift data access while maintaining performance.
- ➢ Efficient Search Capabilities: The application will feature advanced search functionalities, empowering users with efficient task management tools.
- ➢ Continuous Availability: The To-Do List application is designed to be available 24/7, providing uninterrupted user access at any time.
- ➢ User Authentication: Access to user accounts and actions within the application will require correct usernames and passwords, ensuring secure and personalized interactions.

<u>Dependencies:</u>

- ➢ Hardware and Software Infrastructure: The application's functionality is contingent upon the availability of requisite hardware and software components essential for web development and hosting.
- ➢ Clear Requirements and Specifications: The successful development and operation of the project hinge on well-defined requirements and specifications that guide the entire process.
- ➢ User Proficiency: Effective utilization of the application, especially for administrators, demands a comprehensive understanding of its features and functionalities.

- ➢ Efficient Report Management: The system must possess the capability to securely store and promptly retrieve reports as needed, facilitating organized data management.
- ➢ Reliable Database Connectivity: The application's seamless operation relies on reliable access to a database, ensuring the accuracy of task-related data updates.
- ➢ Continuous Internet Connectivity: Uninterrupted access to the web-based To-Do List application necessitates users to maintain a stable internet connection for seamless interactions.

2.5 <u>Requirements</u>

<u>Software Configuration:</u>

The To-Do List project will employ cutting-edge web technologies for both its front-end and back-end development, ensuring a modern and efficient application.

Front End:

- The React framework will serve as the foundation for the front-end development, enabling the creation of dynamic and responsive user interfaces.
- Node.js will be utilized to manage packages and dependencies efficiently, fostering a streamlined development process.
- Developers will leverage popular code editors such as Visual Studio Code, Sublime Text, or similar tools to craft and optimize React code, ensuring a smooth development experience.

Back End:

- Node.js will drive the server-side scripting, providing a scalable and event-driven architecture for robust back-end functionality.
- MongoDB has been selected as the back-end database, offering flexibility and scalability in storing and managing task-related data.
- The Node.js runtime environment will be employed to support the development and execution of server-side scripts, ensuring a cohesive and efficient back-end development process.

Operating System Compatibility:

The To-Do List application is designed for compatibility with a diverse range of operating systems, including but not limited to Windows, macOS, and Linux. This cross-platform approach ensures widespread accessibility, allowing users to seamlessly engage with the application regardless of their preferred operating environment.

2.6 <u>Data Requirement</u>

At its core, the application serves as a comprehensive tool for seamlessly generating, overseeing, and monitoring tasks. User input queries trigger dynamic interactions with the database, allowing the application to deliver real-time updates and responses, thereby guaranteeing a frictionless user experience.The meticulously chosen technology stack for this project is tailored to create a resilient and effective To-Do List web application. This stack not only ensures the current robustness of the system but also provides scalability and adaptability for the seamless integration of future enhancements and features.

3. External Interface Requirement

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:

a. User Authentication: Users will be required to authenticate themselves through unique credentials, such as usernames and passwords, to access their accounts and tasks.

a. Validation: The system will validate user credentials to ensure that only authorized individuals can log in and use the application.

User Data Privacy:

b. 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.

c. 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 performance requirements for a to-do list application can vary based on factors such as the scale of the application, the number of users, and the specific features it offers.Here are the performance requirements:

- Responsiveness: The application should provide a smooth and responsive user interface.The time taken to load the application initially and subsequent page loads should be minimal.

- Error Handling: Implements effective error handling to gracefully manage issues like network failures, server errors, or other unexpected scenarios.
- Scalability: The application should be designed to handle a growing number of users and tasks. Ensuring that the system can scale horizontally to accommodate increased load.

5.2 **Safety Requirements:**

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

- Database Backup: Implements regular data backups to prevent data loss in the event of system failures or security incidents.
- Authentication and Authorization: Enforce proper authorization controls to ensure that users can only access and modify their own tasks and data.Encourage users to use strong, unique passwords, and consider implementing multi-factor authentication for added security.

5.3 **Security Requirements:**

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

- Secured Database: Safely store user data, including task details and personal information, using a secure database.
- User Access Control: Regular users should be granted view-only privileges for information, with restricted permissions to make edits or alterations limited solely to their individual data.
- Admin Privileges: Administrators must possess distinct accounts endowed with the authority to modify the database, whereas ordinary members should be barred from direct access to the database.
- User Authentication: Encourage users to use strong, unique passwords, and consider implementing multi-factor authentication for added security.
- User Authorization: Enforce proper authorization controls to ensure that users can only access and modify their own tasks and data.

5.4 **Requirement Attributes:**

- Notification System:Timely delivery of notifications and reminders.
- Open Source: The project ought to be accessible as open source, enabling contributions from the community and fostering transparency.
- Intuitive User Interface: A user-friendly interface that is easy to navigate.
- Setup: Users should have the capability to effortlessly download, install, and configure the system without encountering complications.

5.5 **Business Rules:**

Enforcement of business rules and regulations is necessary to guarantee conformity with project policies, expenses, and discounted offers. Users are required to observe legal guidelines and protocols, refraining from engaging in any unlawful activities.

5.6 <u>User Requirements:</u>

- Customization: Options for users to customize the appearance or organization of their to-do lists.
- User Education: Adequate user interfaces, along with comprehensive user manuals, online assistance, and installation guides, should be furnished to instruct users on the utilization and upkeep of the system.
- Admin Facilities: Administrators should be granted privileges for services like backup and restoration, password retrieval, data transfer, data duplication, automatic recovery, file management, and routine server upkeep.

6. <u>Other Requirements</u>

6.1 <u>Data and Category Requirements:</u>

In the To-Do List project, various user categories exist, such as teaching staff, administrators, and students. Access rights and permissions vary based on the user's category. For instance:

Administrators: These users possess extensive privileges, allowing them to modify data, delete entries, add information, and perform other administrative tasks.

Regular Users: Users, excluding administrators and librarians, are limited to read-only access for retrieving information from the database.

Likewise, diverse categories of tasks or to-do items will be established. The display of relevant data to users depends on the category of tasks. It is essential to structure and code the categories and associated data in a specific format to facilitate effective categorization and retrieval.

6.2 <u>Appendix:</u>

The appendix for a to-do list app typically includes supplementary information that supports and complements the main content of the application documentation. Here are some of the following elements that could be included in the appendix:

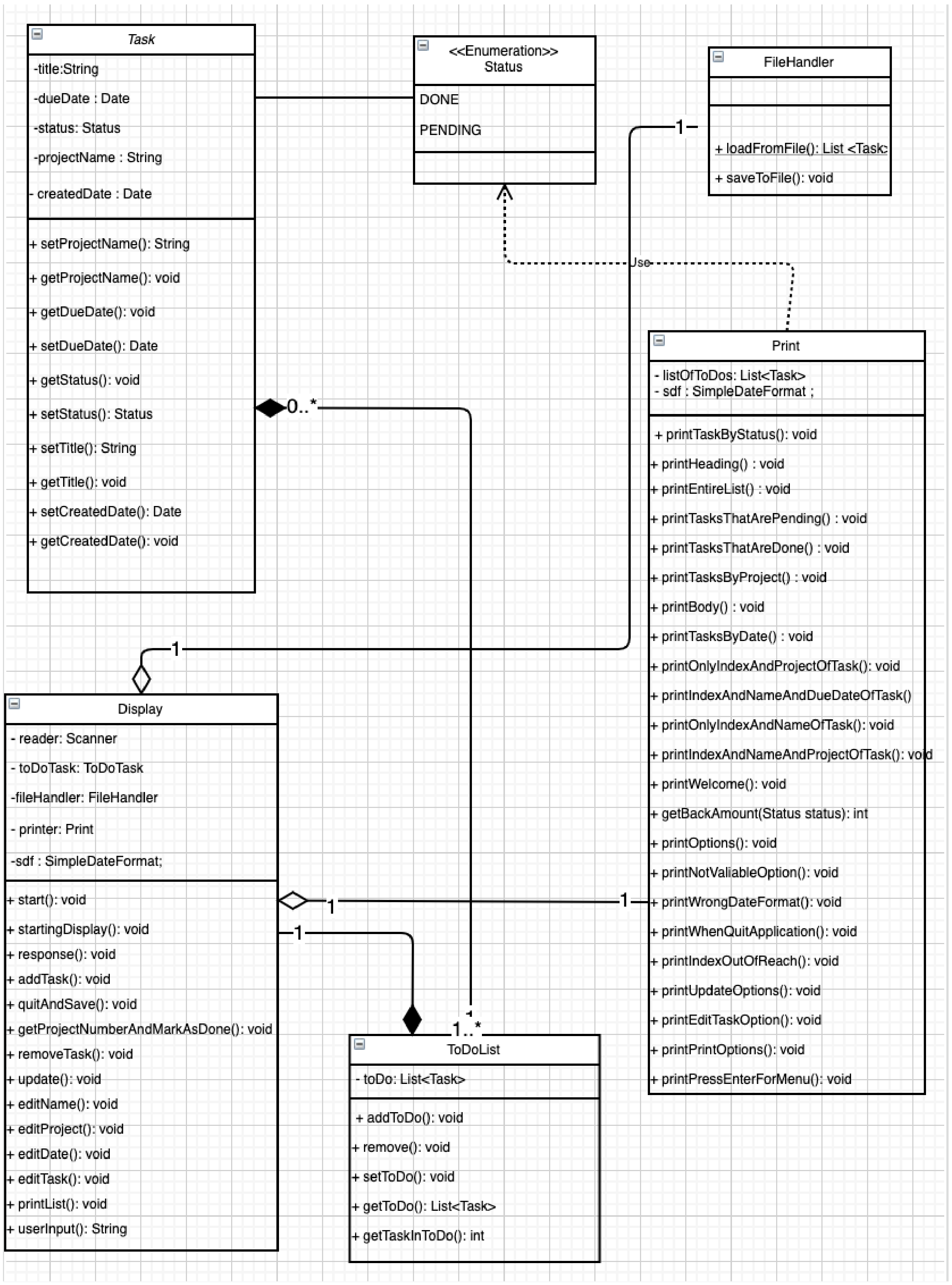
- User Guide: Provide a detailed guide on how users can navigate and utilize the features of the to-do list application.
- Acknowledgments:: Acknowledge and give credit to individuals, teams, or organizations that contributed to the development or support of the application.
- System Requirements: List the minimum hardware and software requirements necessary to run the to-do list application successfully

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:

- Unique Key: A unique identifier used to differentiate entries or records in a database.
- Entity Relationship Diagram (ERD): A visual representation illustrating the key data structures and connections between entities in the To-Do List Web Application.
- User: An individual with an assigned User ID, responsible for task management within the application. Attributes include Name, Email, Password, Profile Picture, and User Type.
- Task: A fundamental element in the application identified by a Task ID, featuring attributes such as Title, Description, Due Date, Priority Level, and Completion Status.
- User Class: Different categories of users in the To-Do List Web Application, including Regular Users and Administrators.
- Administrator: A user with elevated privileges responsible for managing the system.
- Operating Environment: The platforms and browsers compatible with the To-Do List web application, ensuring accessibility across various systems.
- 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.
- Hardware Configuration: The hardware requirements for accessing the To-Do List web application.
- Assumptions: Fundamental beliefs about the development and operation of the To-Do List application.
- Dependencies: External factors or conditions that the To-Do List application relies on for successful operation.
- Software Configuration: The technology stack and configuration chosen for the front-end and back-end development of the To-Do List project.
- Graphical User Interface (GUI): The visual interface of the To-Do List web application, providing an intuitive and user-friendly experience.
- Requirement Attributes: Specific characteristics and attributes associated with the requirements of the To-Do List application.
- Business Rules: Regulations and guidelines users must adhere to while using the To-Do List application.

6.4 Class Diagram


Task
-title:String
-dueDate : Date
-status: Status
-projectName : String
- createdDate : Date
+ setProjectName(): String
+ getProjectName(): void
+ getDueDate(): void
+ setDueDate(): Date
+ getStatus(): void
+ setStatus(): Status
+ setTitle(): String
+ getTitle(): void
+ setCreatedDate(): Date
+ getCreatedDate(): void
<<Enumeration>> Status
DONE
PENDING
FileHandler
+ loadFromFile(): List <Task
+ saveToFile(): void
Use
Print
- listOfToDos: List<Task>
- sdf : SimpleDateFormat ;
+ printTaskByStatus(): void
+ printHeading() : void
+ printEntireList() : void
+ printTasksThatArePending() : void
+ printTasksThatAreDone() : void
+ printTasksByProject() : void
+ printBody() : void
+ printTasksByDate() : void
+ printOnlyIndexAndProjectOfTask(): void
+ printIndexAndNameAndDueDateOfTask()
+ printOnlyIndexAndNameOfTask(): void
+ printIndexAndNameAndProjectOfTask(): void
+ printWelcome(): void
+ getBackAmount(Status status): int
+ printOptions(): void
+ printNotValiableOption(): void
+ printWrongDateFormat(): void
+ printWhenQuitApplication(): void
+ printIndexOutOfReach(): void
+ printUpdateOptions(): void
+ printEditTaskOption(): void
+ printPrintOptions(): void
+ printPressEnterForMenu(): void
0..*
1
1
1
1
1
1..*
Display
- reader: Scanner
- toDoTask: ToDoTask
-fileHandler: FileHandler
- printer: Print
-sdf : SimpleDateFormat;
+ start(): void
+ startingDisplay(): void
+ response(): void
+ addTask(): void
+ quitAndSave(): void
+ getProjectNumberAndMarkAsDone(): void
+ removeTask(): void
+ update(): void
+ editName(): void
+ editProject(): void
+ editDate(): void
+ editTask(): void
+ printList(): void
+ userInput(): String
ToDoList
- toDo: List<Task>
+ addToDo(): void
+ remove(): void
+ setToDo(): void
+ getToDo(): List<Task>
+ getTaskInToDo(): int