SOFTWARE REQUIREMENTS SPECIFICATION

For

TO DO LIST PROJECT

Prepared by:
AJITH.X
BRIGHTLIN ANGEL.S
DEIVANAYAGI.N
KARTHIKA.K

AcademicYear:*2020-2024*

Department of Information Technology

1.1 Purpose

The purpose of this document is to define the requirements for the development of a to-do list application. This application will allow users to create, manage, and track their tasks and to-do lists efficiently.

1.2 Document Conventions

- ➢ Entire document should be justified.
- ➢ Convention for Main title
    - Font face: Times New Roman
    - Font style: Bold
    - Font Size: 14
- ➢ Convention for Subtitle
    - Font face: Times New Roman
    - Font style: Bold
    - Font Size: 12
- ➢ Convention for body
    - Font face: Times New Roman
    - Font Size: 12

1.3 Scope of Development Project

The to-do list application will be a standalone mobile application for iOS and Android platforms. It will include features for task creation, prioritization, due dates, notifications, and task completion tracking.

1.4 Definitions, Acronyms and Abbreviations

REACT -> platform independence
SQL->Structured query Language
ER->Entity Relationship
UML->Unified Modeling Language
IDE->Integrated Development Environment
SRS->Software Requirement Specification

IEEE->InstituteofElectricalandElectronicsEngineers

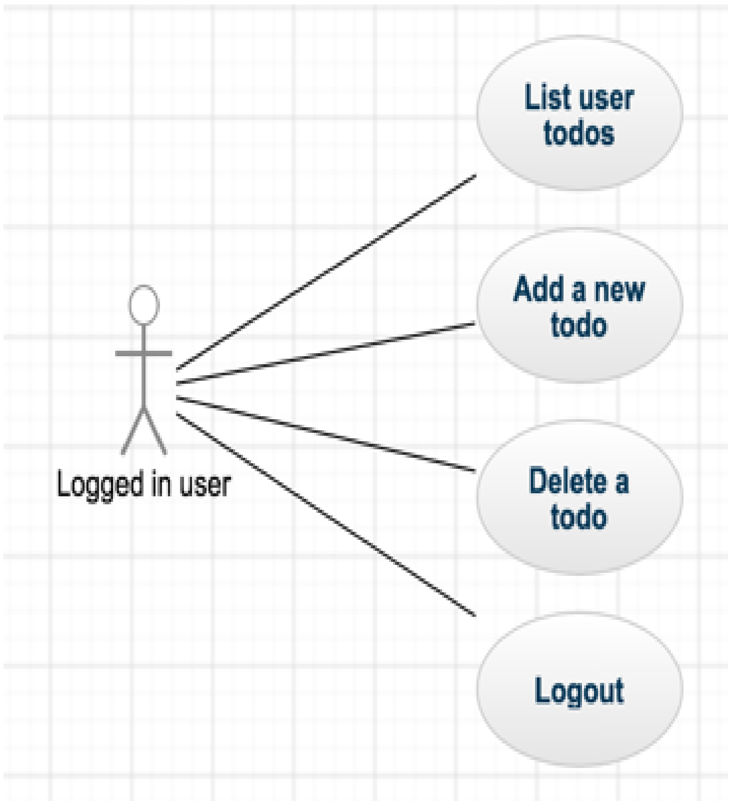
1.5 References

- Books
  - "Getting Things Done: The Art of Stress-Free Productivity" by David Allen - A comprehensive guide to personal productivity and task management.
  - "The Checklist Manifesto: How to Get Things Right" by Atul Gawande - Focuses on the power of checklists in improving efficiency and reducing errors in various fields.
- Websites
  - https://todoist.com
  - https://evernote.com

2. Over all Descriptions

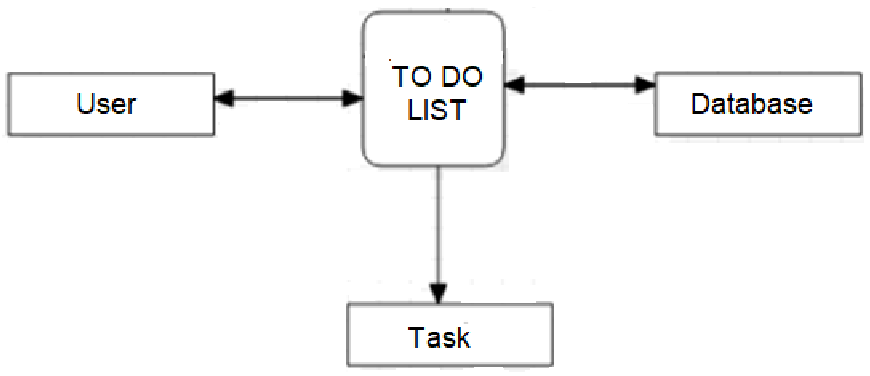
2.1 Product Perspective

Use Case Diagram of To-Do list



2.2 Product Function

Entity Relationship Diagram of To-Do list.



The Intelligent To Do System provides a simple mechanism for users to organize their daily routine by saying something like “Remind me to bring vegetables at 2pm” and it will smartly set the reminder and will prompt it to the user.

- Set a reminder to organize your day
- Set an alarm for meetings, milestones etc.
- Reduce the effort of typing
- Suggests time for the completion of task
- Prioritize the most important task first
- Constantly reminding you to complete pending tasks

2.3 User Classes and Characteristics

In the blogging website many users are using it. In this there will be many members with different scope because each user have different need. So for the users we needs to create the website for all type of users.

1. Individual Users:
   Characteristics:
   - Personal use of the to-do list.
   - Create, manage, and complete tasks.
   - Prefer a simple and intuitive interface.

- May use the application for personal tasks, shopping lists, or daily planning.

2. Family Users:
   Characteristics:
   - Multiple family members share and collaborate on a to-do list.
   - May include tasks related to household chores, groceries, and family events.
   - Need shared lists and collaborative features.
   - Value real-time synchronization and notifications for family members.
3. Students:
   Characteristics:
   - Use the to-do list for managing homework, assignments, and study tasks.
   - May need features like due dates, prioritization, and task categorization.
   - Value mobile access and reminders for deadlines.
4. Professionals:
   Characteristics:
   - Use the to-do list for work-related tasks and projects.
   - Need advanced features like task delegation, project management, and integrations with other tools (e.g., calendars, email).
   - Value productivity-enhancing features and integration capabilities.
5. Small Business Owners:
   Characteristics:
   - Manage tasks related to their business operations.
   - May require task assignment to employees or team members.
   - Need task tracking and progress monitoring.
   - Value collaboration features and reporting.
6. Teams and Workgroups:
   Characteristics:
   - Collaborative use of the to-do list for team projects and tasks.
   - Multiple team members with varying roles and responsibilities.
   - Require shared lists, task assignment, comments, and file attachments.
   - Value real-time updates and communication within the application.
7. Managers and Supervisors:
   Characteristics:
   - Oversee team or departmental tasks and projects.
   - Need access to team performance metrics and task progress.
   - Require reporting and analytics features.
   - Value visibility into team members' workloads and deadlines.
8. Freelancers:
   Characteristics:
   - Manage client projects and assignments.
   - Need to track billable hours and project deadlines.
   - May require invoicing and payment tracking features.
   - Value task management that supports freelancing workflows.

2.4 Operating Environment

The operating environment for a to-do list application includes various factors that influence its functionality, accessibility, and performance. Here are the key components of the operating environment for a to-do list application:

- Platform Compatibility, Web Browser Compatibility (for web-based apps) or Mobile Device Compatibility (for mobile apps)
- Internet Connectivity, Cross-Platform Support, Performance Requirements
- Data Synchronization, Security Measures
- Localization, Accessibility
- Data Backup and Recovery, Notification Services, User Support and Training
- Scalability, User Environment, Cross-Device Compatibility

2.5 Assumptions and Dependencies

Assumptions:

1) User Knowledge: Users are assumed to have a basic understanding of how to use digital to-do list applications, including creating tasks, setting priorities, and managing lists.

2) Internet Connectivity: The application assumes that users will have access to the internet for synchronization and updates. However, offline functionality may be provided as an enhancement.

3) Hardware and Device Compatibility: The application assumes compatibility with commonly used devices and operating systems, such as Windows, macOS, iOS, and Android. Compatibility with older devices and operating system versions may not be a priority unless specified.

4) Data Privacy and Security: Users are assumed to trust the application with their task data. Adequate security measures are in place to protect user data from unauthorized access and breaches.

Dependencies:

1) Development Frameworks and Libraries: The project may depend on specific programming languages, frameworks, and libraries for development. Any changes or updates to these dependencies may affect the project.

2) Third-Party Services: If the application relies on third-party services or APIs (e.g., email integration, cloud storage), the project's success may depend on the availability and reliability of these services.

3) Data Hosting and Storage: The project may depend on a cloud hosting service or a specific database system for data storage and retrieval. Ensuring the availability and performance of these services is crucial.

4) External Integrations: Dependencies on external applications or systems, such as calendar apps or task management tools, must be considered. Changes or issues with these integrations can impact the project.

2.6 Requirement

Software Configuration:-

This software package is developed using java as frontend which is supported by sun micro system.

Microsoft SQL Server as the back end to store the database.
Operating System: Windows 7 and above
Language: react JS,HTML,CSS
Database: MS SQL Server (back end)
Hardware Configuration:-Processor:
Pentium(R)Dual-core CPU
Hard Disk: 40GB
RAM:256 MB or more

2.7 Data Requirement

The inputs consist of the query to the database and the output consists of the solutions for the query. The output also includes the user receiving the details of their accounts. In this project the inputs will be the queries as fired by the users like create an account, selecting books and putting into account. Now the output will be visible when the user requests the server to get details of their account in the form of time, date and which books are currently in the account.

3 External Interface Requirement

3.4 GUI

User-Friendly Design:

- ➢ Keep the interface clean and intuitive, with a focus on simplicity and ease of use.
- ➢ Use a clear and legible font for text elements.
- ➢ Choose a visually appealing color scheme that is easy on the eyes.

Task List View:

- ➢ Display a list of tasks with essential details such as title, due date, and priority.
- ➢ Allow users to easily distinguish between completed and pending tasks (e.g., by using different visual cues).
- ➢ Provide options to sort and filter tasks based on various criteria (e.g., due date, priority, category).

Task Creation and Editing:

- ➢ Include a button or action to create new tasks.
- ➢ Offer a user-friendly task creation form with fields for title, description, due date, and priority.
- ➢ Enable users to edit task details easily with an inline or separate editing interface.

Task Categorization:

- ➢ Allow users to categorize tasks into lists, folders, or tags for better organization.
- ➢ Provide options to create, rename, and delete categories.
- ➢ Implement drag-and-drop functionality for moving tasks between categories.

Task Actions:

- ➢ Offer actions like marking tasks as complete or incomplete.
- ➢ Include options to delete or archive tasks.
- ➢ Provide a context menu or action buttons for task-related actions.

Search and Filter:

- ➢ Include a search bar for users to quickly find specific tasks.
- ➢ Implement filters or advanced search options to narrow down task results.

Task Details View:

- ➢ Enable users to view and edit task details in a separate screen or modal.
- ➢ Show additional information, such as task notes or attachments.
- ➢ Provide a space for adding comments or notes to tasks.

Notifications and Reminders:

- ➢ Display reminders and notifications for tasks with due dates.

- ➢ Allow users to set or snooze reminders directly from the interface.

User Preferences:

- ➢ Provide a settings or preferences section for users to customize aspects like notification preferences, time zone, and date format.

Data Synchronization:

- ➢ Ensure that task data is synchronized in real-time or near-real-time across devices.
- ➢ Display synchronization status or indicators.

4 System Features

Here are some common system features for a to-do list application:

- Task Creation
- Task Management
- Task Categories or Lists
- Reminders and Notifications
- Search and Filter
- Task Sorting
- User Authentication
- Data Synchronization

5 Other Non-functional Requirements

5.4 Performance Requirement

Response Time:

- Define the maximum acceptable response time for common user actions (e.g., creating a task, loading task lists).
- For example, tasks should load within 2 seconds.

Concurrency:

- Specify the number of concurrent users the system should support without significant degradation in performance.
- For example, the application should handle at least 500 concurrent users during peak usage.

Scalability:

- Describe how the application should scale as the user base and data volume increase.
- Specify load balancing and resource allocation strategies for scaling horizontally.

5.5 Safety Requirement

The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take the database backup so that the database is not lost. Proper UPS/inverter facility should be there in case of power supply failure.

5.6 Security Requirement

- User Authentication, Authorization and Access Control
- Data Encryption, Session Management
- Input Validation, Security Headers
- Security Patching, Rate Limiting and Abuse Prevention

- Password Storage, Data Privacy Compliance
- Security Logging, Security Incident Response Plan
- Data Deletion, Security Testing

5.7 Requirement attributes

Requirement attributes are characteristics or properties associated with each software requirement. These attributes provide additional context, information, and details about each requirement, helping stakeholders better understand and manage them throughout the project.

5.8 Business Rules

Business rules are specific, detailed statements that define or constrain how a to-do list application should operate from a business or operational perspective. They help ensure that the software aligns with the goals and requirements of the organization or business that is developing or using it..

5.9 User Requirement

User requirements in the context of a to-do list application represent the needs, expectations, and preferences of the individuals who will be using the software. These requirements are critical for ensuring that the application is user-friendly, efficient, and aligns with user expectations. Here are some common user requirements for a to-do list application:

1. Task Creation and Editing:
   - Users should be able to easily create new tasks by specifying a title, description, due date, and priority.
   - Users need the ability to edit task details when necessary.
2. Task Management:
   - Users should be able to view, organize, and manage their tasks efficiently.
   - Tasks should be sortable and filterable based on criteria like due date, priority, and category.
3. Task Prioritization:
   - Users need the option to assign priority levels to tasks (e.g., high, medium, low).
   - The application should highlight or visually differentiate high-priority tasks.
4. Task Categorization:
   - Users should be able to categorize tasks into lists, folders, or tags for better organization.
   - Creating, renaming, and deleting categories should be straightforward.

6 Other Requirements

6.4 Data and Category Requirement

Data and category requirements in the context of a to-do list application outline how tasks and categories should be managed and organized within the software. These requirements are essential for ensuring that the application effectively stores, retrieves, and presents task-related information to users. Here are some data and category requirements for a to-do list application:

Data Requirements:

1. Task Information:
    - Each task must include a title, description, due date, and priority level (e.g., high, medium, low).
    - Tasks may have additional attributes such as labels, tags, or notes.
2. Task Status:
    - Tasks should have a status indicator to differentiate between completed and pending tasks.
    - Completed tasks should be archived or moved to a separate section.
3. User Accounts:
    - User accounts should store user profiles, including usernames, email addresses, and profile pictures (if applicable).
    - Secure storage of user credentials (passwords) is required.
4. Collaboration Information:
    - Data related to task sharing and collaboration, including shared task lists and collaborators' details, should be stored securely.

6.5 Appendix

A:Admin, Abbreviation, Acronym, Assumptions; B: Books, Business rules; C: Class, Client ,Conventions; D: Data requirement, Dependencies; G: GUI; K: Key; L: Library, Librarian; M:Member; N:Non-functional Requirement; O:Operating environment; P:Performance ,Perspective ,Purpose; R: Requirement, Requirement attributes; S:Safety, Scope ,Security, System features; U:User ,User class and characteristics, User requirement;

6.6 Glossary

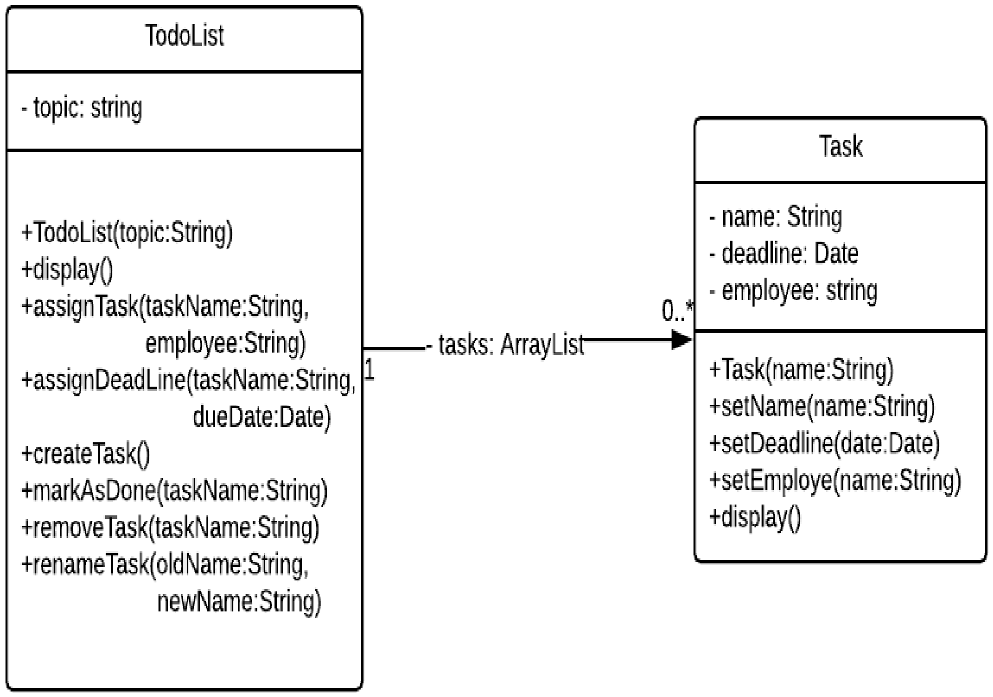
The following are the list of conventions and acronyms used in this document and the project as well:

6.6.1 Administrator: A login id representing a user with user administration privileges to the software.
6.6.2 User: A general login id assigned to most users.
6.6.3 Client: Intended users for the software
6.6.4 SQL: Structured Query Language; used to retrieve information from a database
6.6.5 SQL Server: A server used to store data in an organized format
6.6.6 Layer: Represents a section of the project
6.6.7 User Interface Layer: The section of the assignment referring to what the user interacts with directly
6.6.8 Application Logic Layer: The section of the assignment referring to the Web Server. This is where all computations are completed
6.6.9 Data Storage Layer: The section of the assignment referring to where all data is recorded
6.6.10 Use Case: Abroad level diagram of the project showing a basic overview
6.6.11 Class diagram: It is a type of static structure diagram that describes the structure of a system by showing the system's cases, their attributes, and the relationships between the classes

6.6.12 Interface: Something used to communicate across different mediums
6.6.13 Unique Key: Used to differentiate entries in a database

6.7 Class Diagram

A class is an abstract, user-defined description of a type of data. It identifies the attributes of the data and the operations that can be performed on instances (i.e.objects) of the data. A class of data has a name, a set of attributes that describes its characteristics, and a set of operations that can be performed on the objects of that class. The classes 'structure and their relationships to each other frozen in time represent the static model. In this project there are certain main class Which are related to other classes required for their working. There are different kinds of relationships between the classes as shown in the diagram like normal association, aggregation, and generalization.


TodoList
- topic: string
+TodoList(topic:String)
+display()
+assignTask(taskName:String, employee:String)
+assignDeadLine(taskName:String, dueDate:Date)
+createTask()
+markAsDone(taskName:String)
+removeTask(taskName:String)
+renameTask(oldName:String, newName:String)
1
- tasks: ArrayList
0..*
Task
- name: String
- deadline: Date
- employee: string
+Task(name:String)
+setName(name:String)
+setDeadline(date:Date)
+setEmploye(name:String)
+display()