# SOFTWARE REQUIREMENTS SPECIFICATION

**For**

## To Do List

**Prepared by:**
Afrid Ariff H
Samsson R
Ananth P
Crispin R

**Academic Year: 2023-2024**

# 1. Introduction

## 1.1 Purpose

The main objective of this document is to illustrate the requirements of the project To-Do List. The document gives the detailed description of the both functional and non-functional requirements proposed by the client.This application aims to help users manage their tasks and enhance their productivity by providing a user-friendly interface for creating, editing, and organizing to-do items.

## 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 Sub title

- 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

The To-Do List Application will be a cross-platform software solution available on web browsers and mobile devices. It will allow users to create and manage tasks, set priorities, due dates, and reminders. The application will also support user account management and synchronization of tasks across devices.

## 1.4 Definitions, Acronyms and Abbreviations

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

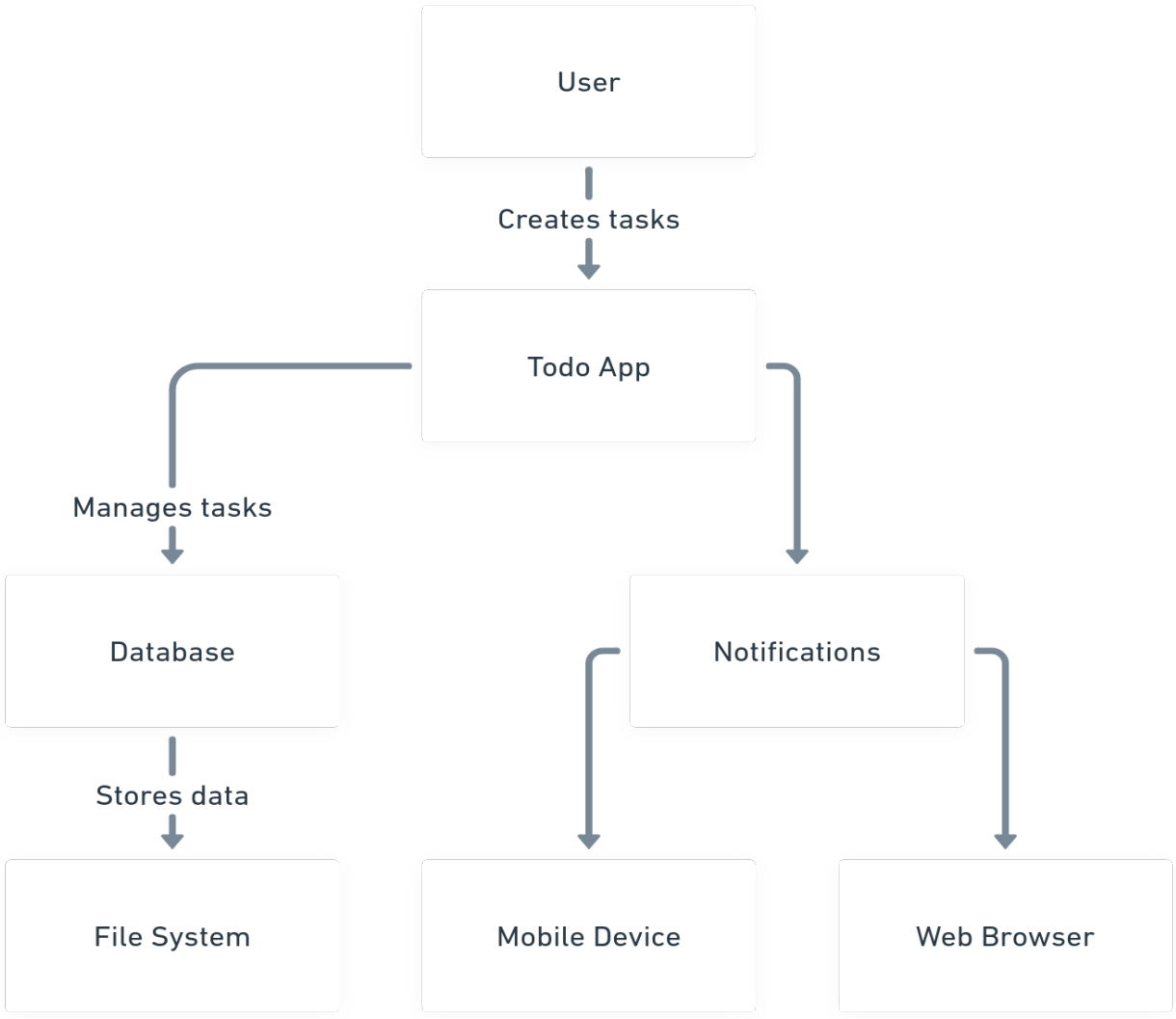
  "Learning React" by Alex Banks and Eve Porcello:

  - This book offers a hands-on introduction to React, covering the core concepts and practical examples to build React applications.

## 2. Overall Descriptions

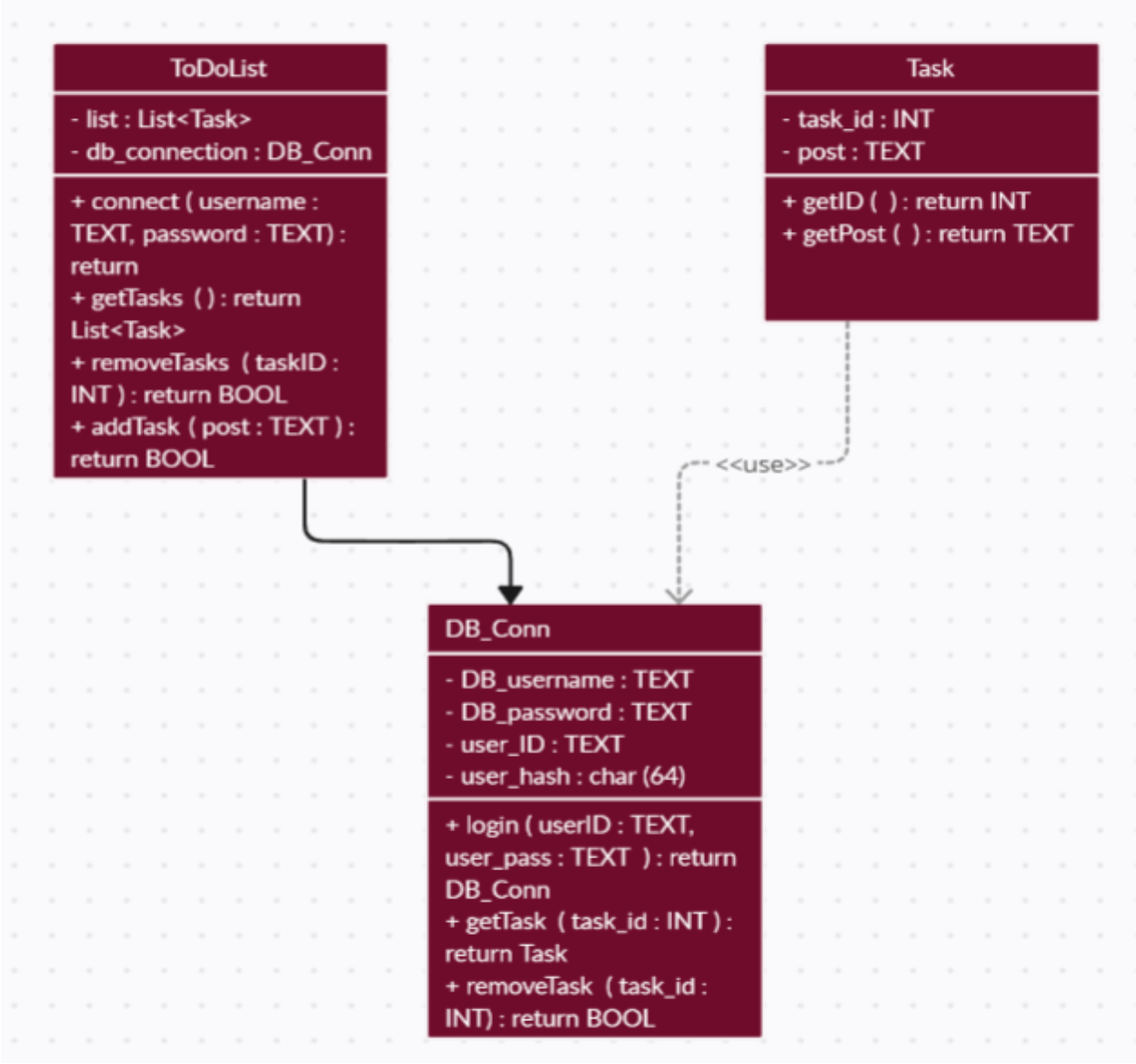
### 2.1 Product Perspective

Use Case Diagram of To Do List



**2.2 Product Function**

Entity Relationship Diagram of To Do List




1. Task Creation and Management:

- Users can create tasks with titles, descriptions, due dates, and priorities.
- Users can organize tasks into categories or lists.
- Tasks can be edited, deleted, or marked as complete.

2. Smart Task Scheduling:
- The app can suggest optimal due dates and times based on task priorities and deadlines.
- Users can set recurring tasks (daily, weekly, monthly).
- Tasks can be sorted automatically by urgency or importance.

3. Task Reminders:
- Users can set reminders for tasks with notifications or alerts.
- Reminders can be location-based (e.g., remind me when I arrive at a specific place).

4. Collaboration and Sharing:
- Users can share tasks or task lists with others.
- Collaborators can edit or comment on shared tasks.

5. Integration with Calendars and Other Apps:
- The app can sync with users' calendars (e.g., Google Calendar, Apple Calendar).
- Integration with email clients and messaging apps for task-related communication.

**2.3 User Classes and Characteristics**

**2.4 Operating Environment**

The product will be operating in windows environment. The Library Management System is a website and shall operate in all famous browsers, for a model we are taking Microsoft Internet Explorer,Google Chrome,and Mozilla Firefox.Also it will be compatible with the IE 6.0. Most of the features will be compatible with the Mozilla Firefox & Opera 7.0 or higher version. The only requirement to use this online product would be the internet connection.
The hardware configuration include Hard Disk: 40 GB, Monitor: 15” Color monitor, Keyboard: 122 keys. The basic input devices required are keyboard, mouse and output devices are monitor, printer etc.

**2.5 Assumptions and Dependencies**

The assumptions are:-

- User Availability: It is assumed that users will have access to the internet and suitable devices (e.g., smartphones, computers) to use the application.

- Data Privacy Regulations: It is assumed that the development team will comply with relevant data protection regulations (e.g., GDPR) in handling user data.

- Platform Compatibility: The application assumes compatibility with specific web browsers (e.g., Chrome, Firefox) and mobile/desktop platforms (iOS, Android, Windows, macOS) as outlined in the project plan.

- Third-Party Services: The application assumes the availability and proper functioning of third-party services and APIs (e.g., email services, calendar integrations) that it relies on.

The dependencies are:-
The To-Do List Application relies on several external components and services:

- Server Infrastructure: The application will be hosted on cloud-based server infrastructure to ensure scalability and reliability.

- Third-Party Libraries: Various third-party libraries and frameworks will be utilized for features such as user authentication, data encryption, and UI components.

- Notification Services: External notification services will be integrated to provide timely reminders and notifications to users.

**2.6 Requirement**
Software Configuration:- Developed by React as Front end and Java Springboot for backend

Operating System: Windows NT, windows 98, Windows XP Language: Java Runtime Environment, Net beans 7.0.1 (front end)

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**
Task Data

- The application shall capture and store task data, including:
- Task titles (up to 255 characters).
- Task descriptions (optional).
- Due dates.
- Priority levels (e.g., high, medium, low).
- Task completion status.
- Each task entry shall have a unique identifier

User Data
The system shall collect and manage user data, including:

- Usernames.
- Email addresses.
- Hashed and salted passwords.
- User preferences and settings.

## 3. External Interface Requirement

### 3.1 GUI

Web Interface:
Users accessing the application via web browsers will experience a responsive, browser-based interface with cross-browser compatibility.

Desktop Application Interface:
Desktop users on Windows and macOS will benefit from a dedicated desktop application offering platform-specific features and performance optimizations.

## 4. System Features

1. Task Creation and Editing:

- Users can create new tasks with titles, descriptions, due dates, and priorities.

Ability to edit existing tasks, including changing titles, descriptions, and due dates.

2. Task Organization:

- Users can organize tasks into categories, lists, or projects for better management.
- Support for creating sub-tasks or checklists within larger tasks.

3. Task Priority and Urgency:

- Ability to assign priority levels (e.g., high, medium, low) to tasks.
- Highlight urgent tasks or those nearing their due dates.

4. Reminders and Notifications:

- Users can set reminders for tasks with customizable notification settings.
- Receive notifications via in-app alerts, emails, or push notifications.

5. Offline Access:

- Ability to use essential features even without an internet connection.
- Data synchronization when the device reconnects.

6. Data Export and Reporting:
- Export tasks and data to various formats (e.g., CSV, PDF).
- Generate reports and statistics on task completion and productivity.

## 5. Other Non-functional Requirement

## 5.1 Performance Requirement

1. Response Time:
- The application should respond to user interactions (e.g., task creation, editing, marking as complete) within 2 seconds or less.

2. Task Loading Time:
- The time taken to load the user's task list should be minimal, even when the user has a large number of tasks. It should not exceed 3 seconds.

3. Notification Delivery:
- Task reminders and notifications should be delivered promptly, with a delay of no more than 15 seconds.

## 5.2 Safety Requirement

Data Safety (Mandatory)

- The application shall implement regular data backups to prevent data loss due to system failures.
- Data recovery mechanisms shall be in place to retrieve user data in case of accidental deletion.

User Authentication (Mandatory)

- Robust user authentication mechanisms shall be used to prevent unauthorized access to user accounts.
- Passwords shall be securely stored using industry-standard hashing algorithms.

## 5.3 Security Requirement

Data Encryption

- All data transmissions between the client and server shall be encrypted using HTTPS.
- User data, including passwords, shall be stored using strong encryption to protect against data breaches.

Access Control

- Access to user data and settings shall be restricted to authorized users.
- Role-based access control (RBAC) shall be implemented for administrators and regular users.

## 5.4 Requirement attributes

Priority

- Each requirement shall be assigned a priority level indicating its importance. Priorities include:
- High (H)
- Medium (M)
- Low (L)

Status

- The status of each requirement shall be tracked throughout the project, including:
- Proposed
- In Progress
- Completed
- Verified

## 5.5 Business Rules

- Users may only register one account per valid email address.
- Task due dates should be in the future, not in the past.
- Tasks marked as "completed" cannot be deleted directly; they can only be archived.

## 5.6 User Requirement

- Users must be able to create, edit, delete, and mark tasks as complete.
- Users should have the option to set reminders for tasks.
- The application shall provide a user-friendly and intuitive interface for task management.

# 6. Other Requirements

## 6.1 Data and Category Requirement

Data Storage (Mandatory)

- Task and user data shall be stored in a secure relational database management system (RDBMS) for efficient retrieval and management.
- Data storage must be scalable to accommodate a growing user base.

Category Management (Desirable)

- Users shall be able to create custom task categories or lists for organizing tasks.

- Each task can belong to one or more categories.

## 6.2 Appendix

## 6.3 Glossary

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

## 6.4 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 classes 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. The relationships are depicted using a role name and multiplicities