SOFTWARE REQUIREMENTS SPECIFICATION

For

To-Do List

Prepared by:-

Prince N Nasario

Regis R

Rajesh R

Isaac Ebanazer R

Academic Year: 2023-2024

1. Introduction

1.1 Purpose

The main objective of this document is to illustrate the requirements of the project Library Management system. The document gives the detailed description of the both functional and non-functional requirements proposed by the client.The purpose of this project is to provide a friendly environment to maintain the details of books and library members.The main purpose of this project is to maintain easy circulation system using computers and to provide different reports. This project describes the hardware and software interface requirements using ER diagrams and UML diagrams.

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 of Development Project

The project aims to create a user-friendly web or mobile application for managing tasks and to-do lists. It will include user registration and login, task creation with details like due dates and priorities, organization into categories or lists, sorting and filtering options, and optional features like reminders, sharing, and collaboration. The application will ensure data security and may incorporate accessibility and internationalization features. The development process will involve planning, design, coding, testing, deployment, and ongoing maintenance. The choice of technologies will depend on the developer's skills and project requirements, with an emphasis on creating an intuitive and visually appealing user interface.

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

1.5 References

Books

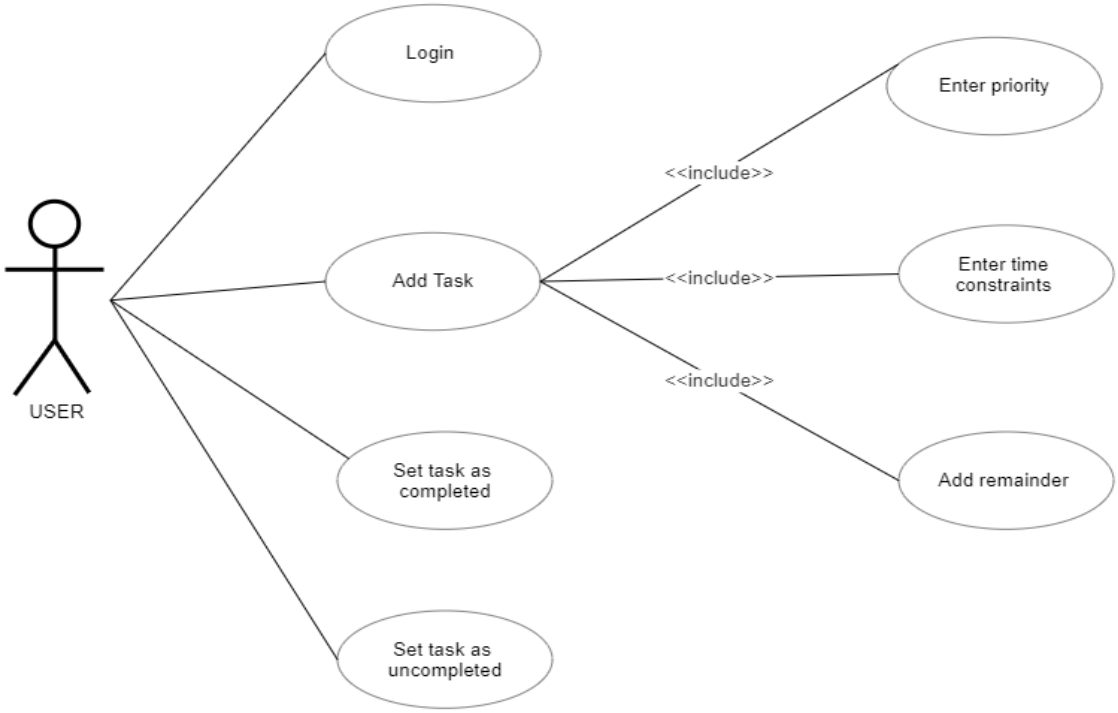
- Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices (ACM Press) by Michael Jackson
- Software Requirements (Microsoft) Second EditionBy Karl E. Wiegers
- Software Engineering: A Practitioner's Approach Fifth Edition By Roger S. Pressman

Websites

2. Overall Descriptions

2.1 Product Perspective

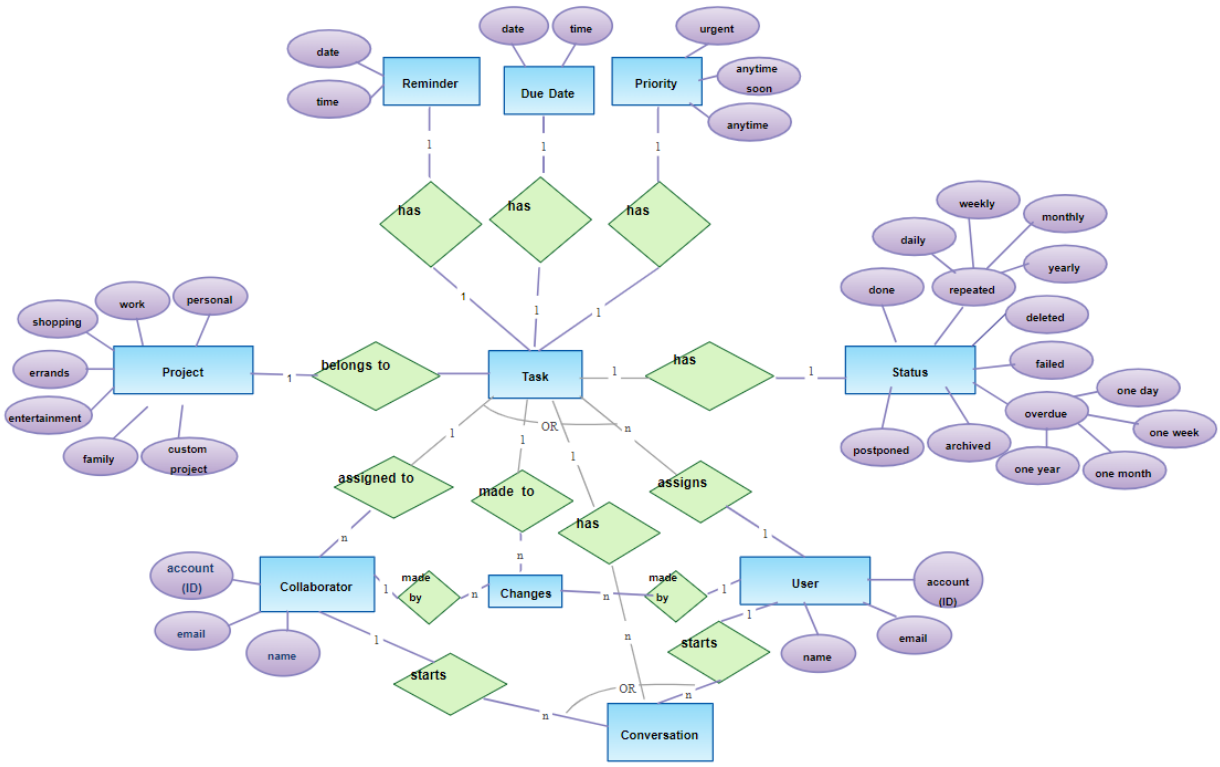
Use Case Diagram of Library Management System



A use case diagram for a to-do list project shows interactions between actors and the system. The user can create, view, update, and delete tasks, set reminders, categorize and prioritize tasks, search, and generate reports. An admin actor may manage accounts or system settings. The diagram provides an overview of the application's functionalities and user interactions.

2.2 Product Function

Entity Relationship Diagram of Library Management System



The entity-relationship (ER) diagram for a to-do list project depicts the entities, attributes, and relationships within the system's data model. The main entities are User, Task, Category (optional), and Reminder (optional). The User entity encompasses attributes such as user ID, username, email, and password. The Task entity includes attributes like task ID, task name, description, due date, and status. Optionally, the Category entity allows tasks to be organized and associated with attributes such as category ID and category name. The Reminder entity, also optional, is associated with a task and includes attributes such as reminder ID, task ID, reminder date/time, and notification preferences. The relationships captured in the diagram include User-Task (one-to-many), Task-Category (optional, one-to-many), and Task-Reminder (optional, one-to-one). These relationships illustrate that a user can have multiple tasks, tasks can be assigned to categories, and tasks can have reminders. The ER diagram provides a visual representation of the data structure and relationships within the to-do list application, aiding in the understanding and design of the system.

2.3 User Classes and Characteristics

The system provides different types of services based on the type of users .In a to-do list app, there are typically different user roles or user types, each with their own distinct

characteristics and responsibilities within the application. Here are some common user roles and their associated attributes:

1. Standard User:
    - Attributes: Standard users are the primary end-users of the to-do list app.
    - Roles and Permissions: Standard users possess the ability to create, view, update, and delete their own tasks. They can set reminders, assign categories, prioritize tasks, and mark them as completed. Depending on the app's functionality, they may also be able to search for tasks, generate reports, and collaborate with other users.
2. Administrator:
    - Attributes: Administrators have elevated privileges and responsibilities within the to-do list app.
    - Roles and Permissions: Administrators have the authority to manage user accounts, including creating and deleting accounts, resetting passwords, and modifying user settings. They may also have access to system-wide settings and configurations. Administrators are often tasked with overseeing categories, reminders, or other administrative functions.
3. Collaborator (optional):
    - Attributes: Collaborators are users who have been invited or granted access to collaborate on specific tasks.
    - Roles and Permissions: Collaborators are granted viewing and editing rights for tasks they have been specifically invited to collaborate on. They may be able to add comments, make changes, or mark tasks as completed. Collaborators typically have restricted permissions compared to standard users or administrators.

It is important to emphasize that the specific user roles and their attributes can vary based on the design and requirements of the to-do list app. The aforementioned user roles provide a general framework of the common responsibilities and permissions associated with different user types in a to-do list application.

2.4 Operating Environment

Creating a web-based to-do list journal demands careful consideration of the operating environment to ensure a user-friendly experience. Prioritize cross-browser compatibility, making certain your website functions consistently on various browsers like Chrome, Firefox, Safari, and Edge. Implement responsive design for adaptability across diverse screen sizes, including desktops, laptops, tablets, and smartphones. Test on different operating systems and browser versions to reach a broad audience.

Accessibility is key; follow guidelines like WCAG for inclusivity. Optimize performance for fast loading times, especially for users with slower internet connections. Select a reliable hosting provider and configure the server environment wisely. Use browser developer tools for debugging, test with popular extensions, and adhere to modern web standards.

Security, content delivery networks (CDNs), DNS configuration, backups, monitoring, updates, and user feedback should all be part of your strategy for a robust to-do list journal website.

2.5 Assumptions and Dependencies

Assumptions and dependencies for a to-do list website refer to the conditions or factors that are considered true or necessary for the successful development and operation of the website. Identifying these assumptions and dependencies is crucial for project planning and risk management. Here are some common assumptions and dependencies for a to-do list website:

Assumptions:

- Internet Access: Users of the to-do list website have reliable internet access, as the application is web-based and requires an internet connection.

- Browser Compatibility: Users have access to modern web browsers and devices, allowing them to use the website without significant compatibility issues.

- User Base: There is a target audience interested in using the to-do list website for personal or professional task management.

- Data Security: Users assume their data is stored securely, and the website employs best practices for data protection.
-
- Stable Hosting: The hosting infrastructure for the website is stable and can handle the expected traffic and data storage requirements.

- Development Resources: There are skilled developers and designers available to create and maintain the website.

- Compliance: The website complies with legal and regulatory requirements, including data privacy regulations (e.g., GDPR) if applicable.

- Feedback and Usage: Users will provide feedback for improving the website, and there will be sufficient usage to justify ongoing development and maintenance.

Dependencies:

- Technology Stack: The choice of programming languages, frameworks, and tools may be dependent on the development team's expertise and preferences.

- Third-Party Services: If the website integrates with third-party services (e.g., cloud storage, authentication providers), it depends on the availability and reliability of these services.

- APIs and Integrations: Dependencies on external APIs or integrations (e.g., weather data, calendar apps) must consider the availability and stability of these services.

- Data Backup and Recovery: Dependent on backup systems and processes to ensure data recovery in case of data loss or server failures.

- Server Infrastructure: Relies on the hosting provider's infrastructure for server availability, scalability, and performance.

- Security Measures: The effectiveness of security measures depends on constant monitoring and updates to protect against emerging threats.

- User Engagement: The success of the website depends on user adoption and continued engagement, which may require marketing efforts and user support.

- Budget and Resources: Availability of budget and resources for ongoing development, maintenance, and scaling of the website.

Identifying these assumptions and dependencies early in the project helps in creating a more accurate project plan, managing risks, and ensuring the successful development and operation of the to-do list website. Regularly reviewing and updating these assumptions and dependencies is also essential as the project progresses and external factors change.

2.6 Requirement

Software Configuration:

The to-do list website is developed using modern web technologies:
Front-End: HTML5, CSS3, JavaScript
Back-End: Node.js, Express.js
Database: MongoDB
Development Tools: Visual Studio Code

Operating Environment:

The website runs on various systems and browsers:

OS: Windows, macOS, Linux
Browsers: Chrome, Firefox, Safari, Edge
Hardware Requirements:

For optimal performance:

Processor: Dual-core or higher
Storage: 20GB+
RAM: 4GB+

3. External Interface Requirement

3.1 GUI

The to-do list website offers a user-friendly graphical interface that empowers both users and administrators with efficient task management capabilities. Here are key features and functionalities:

User-Friendly Interface: The website boasts an intuitive graphical interface that simplifies task creation, updating, and viewing, ensuring a seamless experience for users.

Quick Reports: Users can effortlessly access quick reports displaying task statuses within specific timeframes, providing valuable insights into task management.

Stock Verification: The website includes a robust search feature, enabling users to locate tasks using various criteria, enhancing task discoverability and organization.

Customizability: Administrators have the flexibility to customize the user interface to suit specific preferences and requirements.

Module Integration: All modules seamlessly integrate into the graphical user interface, adhering to established standards, and ensuring a cohesive user experience.

Simplified Design: The design philosophy prioritizes simplicity, ensuring that different interfaces maintain a uniform and user-friendly appearance.

User Management Integration: The user interface effectively interacts with the user management module, dedicating a segment to user login and logout operations.

Login Interface:
For unregistered users, registration is a straightforward process. Once registered, users can log in by entering their username and password. Any incorrect entries trigger an error message for user correction.

Search Functionality:
Users can search for tasks by specifying keywords, such as task names, to quickly locate and access the desired tasks.

Category Management:
Category view displays task categories and offers librarians the ability to add, edit, or delete categories, ensuring efficient organization.

Administrative Control Panel:
The control panel empowers administrators to manage users, resources, and lending options. This includes adding/removing users, editing resource details, and configuring lending options.

This user-centric and feature-rich interface provides an efficient and customizable environment for effective task management.

System Features:

Ensuring the security and privacy of user accounts is paramount in our to-do list website. We achieve this by implementing the following features:

User Authentication: Users will undergo a secure authentication process, primarily using their unique user IDs and passwords. This ensures that only authorized individuals can access their accounts.

Administrator Monitoring: Our system incorporates robust administrator monitoring capabilities. Administrators have the authority to oversee and manage user accounts effectively. This includes updating account statuses, issuing alerts if users attempt to exceed task limits defined by the website policy, and applying fines to users who miss task deadlines.

Accountability: To maintain user privacy and confidentiality, our system strictly enforces that users can only access and manage their own accounts. Other users' accounts remain inaccessible to prevent unauthorized access. The responsibility of managing all user accounts rests solely with the administrator.

These features collectively provide a secure and accountable environment for users to manage their tasks while safeguarding their personal information and adherence to website policies.

5. Other Non-functional Requirements

5.1 Performance Requirements

Our to-do list website will serve as the primary task management system across various user segments. Therefore, it must meet the following performance criteria:

Speed and Accuracy: The system should deliver fast and precise performance, ensuring swift task management.

Error Handling: The system must handle both anticipated and unanticipated errors effectively to prevent data loss and minimize downtime. It should include built-in error testing to detect invalid login attempts, protecting user information.

Scalability: The system should gracefully handle a substantial amount of data, accommodating a large number of tasks and users without errors.

5.2 Safety Requirements

To ensure system safety and reliability, we have established safety requirements:

Database Backup: Regular database backups are crucial to prevent data loss in the event of virus attacks or operating system failures.

Power Supply: Adequate power backup solutions, such as UPS or inverters, should be available to maintain system availability during power supply failures.

5.3 Security Requirements

Our commitment to user data security is reflected in the following security requirements:

Secured Database: The system utilizes a secured database to safeguard user information.

Access Control: Normal users are restricted from editing or modifying data beyond their personal information. Different user types have access constraints.

User Authentication: Robust user authentication mechanisms are in place to ensure that only authorized individuals can access the system. Measures are implemented to prevent password hacking.

Admin and Member Accounts: The system enforces strict separation between admin and member accounts, with only administrators having rights to update the database.

5.4 Requirement Attributes

Multiple Admins: Multiple administrators can create changes to the system. Members and other users are restricted from making changes.

Open Source: The project is designed to be open source, encouraging transparency and collaboration.

User-Friendly: The database maintains a user-friendly interface for all users.

Easy Installation: Users should find it easy to download and install the system.

5.5 Business Rules

Users are expected to adhere to business rules, including project costs and available discount offers. Illegal activities and protocol breaches are prohibited, and both admins and members must comply with established rules and regulations.

5.6 User Requirements

Users of the system include members and administrators, each with specific roles. Members are assumed to have basic computer and internet browsing knowledge, while administrators possess advanced system knowledge to handle potential issues like disk crashes and power failures. The system provides comprehensive user interfaces, user manuals, online help, and installation guides for user education.

Administrators offer various facilities, including backup and recovery, password recovery, data migration, data replication, auto-recovery, file organization, and routine system maintenance.

6. Other Requirements

6.1 Data and Category Requirements

Distinct user categories (e.g., students, staff) determine access rights, with administrators having more extensive privileges. Various categories of tasks and their associated data should be systematically organized and coded.

6.2 Appendix

A: Account, Abbreviation, Assumptions;
B: Backend, Business Rules;
C: Categories, Client, Conventions;
D: Data Requirements, Dependencies;
G: GUI (Graphical User Interface);
K: Key Features;
L: List, Login;
M: Member, Mobile App;
N: Non-functional Requirements;
O: Operating Environment;
P: Performance, Purpose;
R: Requirement, User Roles;
S: Security, Sync, System Features;
U: User, User Interface, User Requirement.

6.3 Glossary

The following list includes conventions and acronyms used throughout this document and the project:

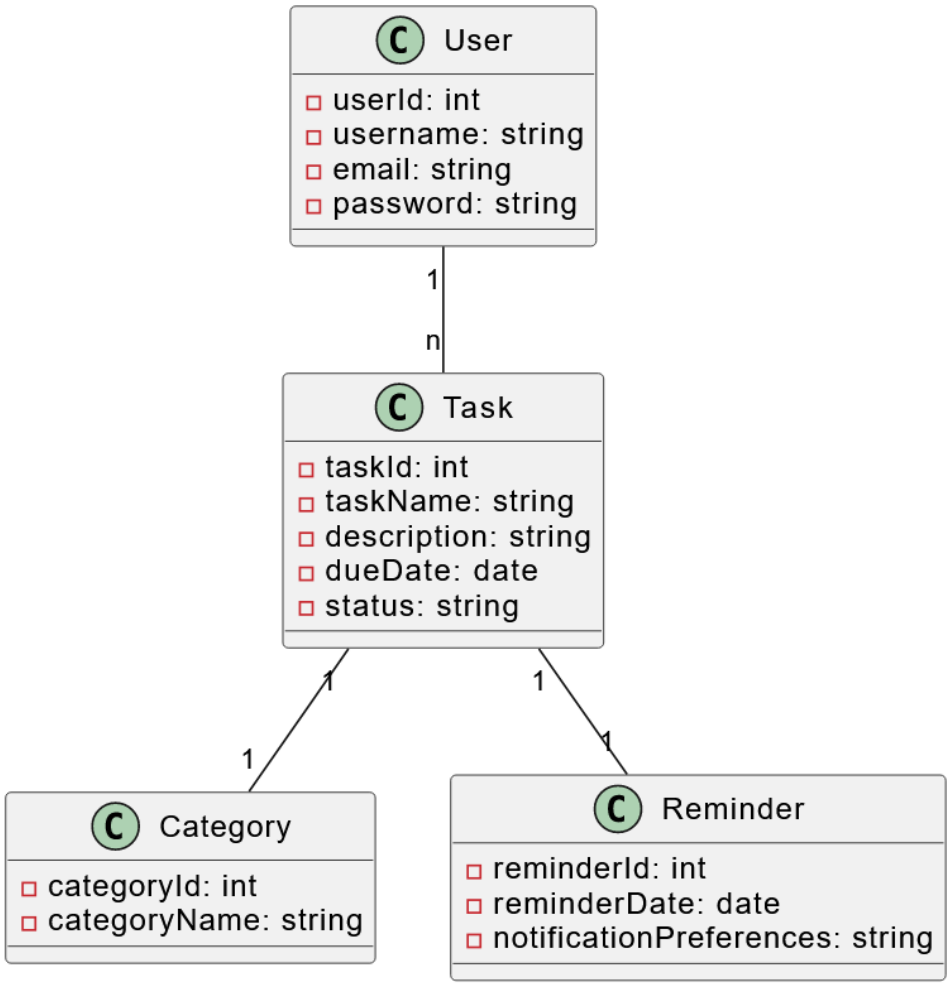
- ➢ Administrator: A login ID representing a user with administrative privileges to the software.
- ➢ User: A general login ID assigned to most users.
- ➢ Client: The intended users of the software.
- ➢ SQL: Structured Query Language; used for retrieving information from a database.
- ➢ SQL Server: A server used to store data in an organized format.
- ➢ Layer: Represents a section of the project.
- ➢ User Interface Layer: The project section referring to what the user interacts with directly.
- ➢ Application Logic Layer: The project section referring to the web server, where all computations are completed.
- ➢ Data Storage Layer: The project section referring to where all data is recorded.
- ➢ Use Case: A broad-level diagram of the project showing a basic overview.
- ➢ Class Diagram: A type of static structure diagram that describes the system's structure by showing the system's cases, their attributes, and the relationships between the classes.
- ➢ Interface: Something used to communicate across different mediums.
- ➢ Unique Key: Used to differentiate entries in a database.

6.4 Class Diagram

A class serves as an abstract, user-defined description of a data type, detailing its attributes and the operations applicable to instances (i.e., objects) of that data type. Each class possesses a name, a set of attributes defining its properties, and a set of operations that can be executed on instances of that class. The structure of these classes and their interrelationships, captured at a specific point in time, constitutes the static model.

Within this project, several primary classes exist, each interconnected with other classes essential for their functionality. Diverse relationships link these classes, encompassing normal associations, aggregations, and generalizations. These relationships are illustrated using role names and multiplicity indicators.

Prominent among these classes are 'User', 'Task', and 'Categories', serving as core components with interdependencies on other classes, thereby establishing a comprehensive framework for the to-do list website.


User
userId: int
username: string
email: string
password: string
1
n
Task
taskId: int
taskName: string
description: string
dueDate: date
status: string
1
1
1
1
Category
categoryId: int
categoryName: string
Reminder
reminderId: int
reminderDate: date
notificationPreferences: string