SOFTWARE REQUIREMENTS SPECIFICATION

For

To Do List

Prepared by:-
AHATHIYA S
NADHEEDHA S
KAVIENA SHARON C T
PRAVEENA B V

Academic Year: *2023-2024*

Department of Information Technology

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

React ->

Java SpringBoot ->

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

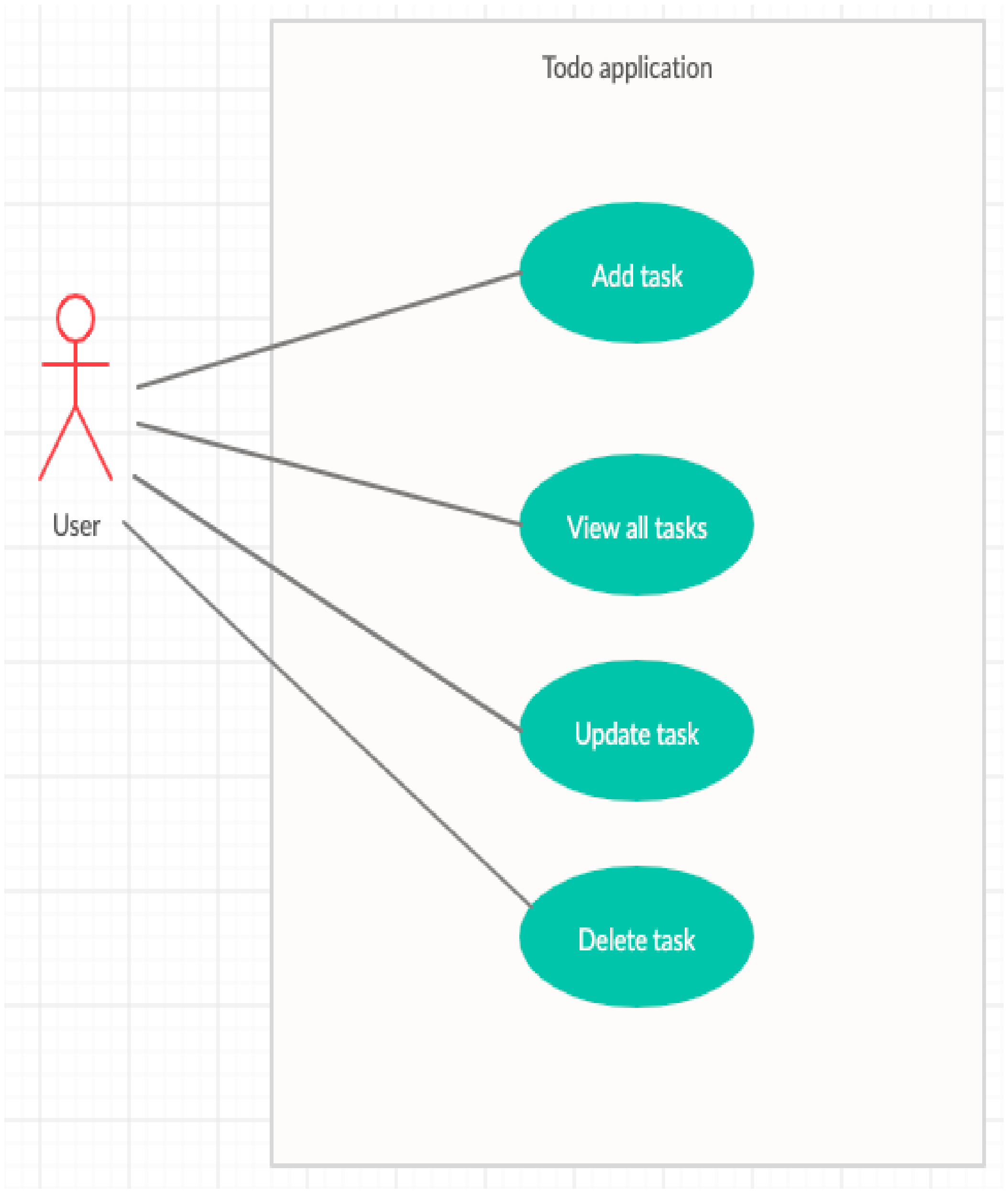
- ❖ Books

  "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.
- ❖ Websites

2. Overall Descriptions

2.1 Product Perspective

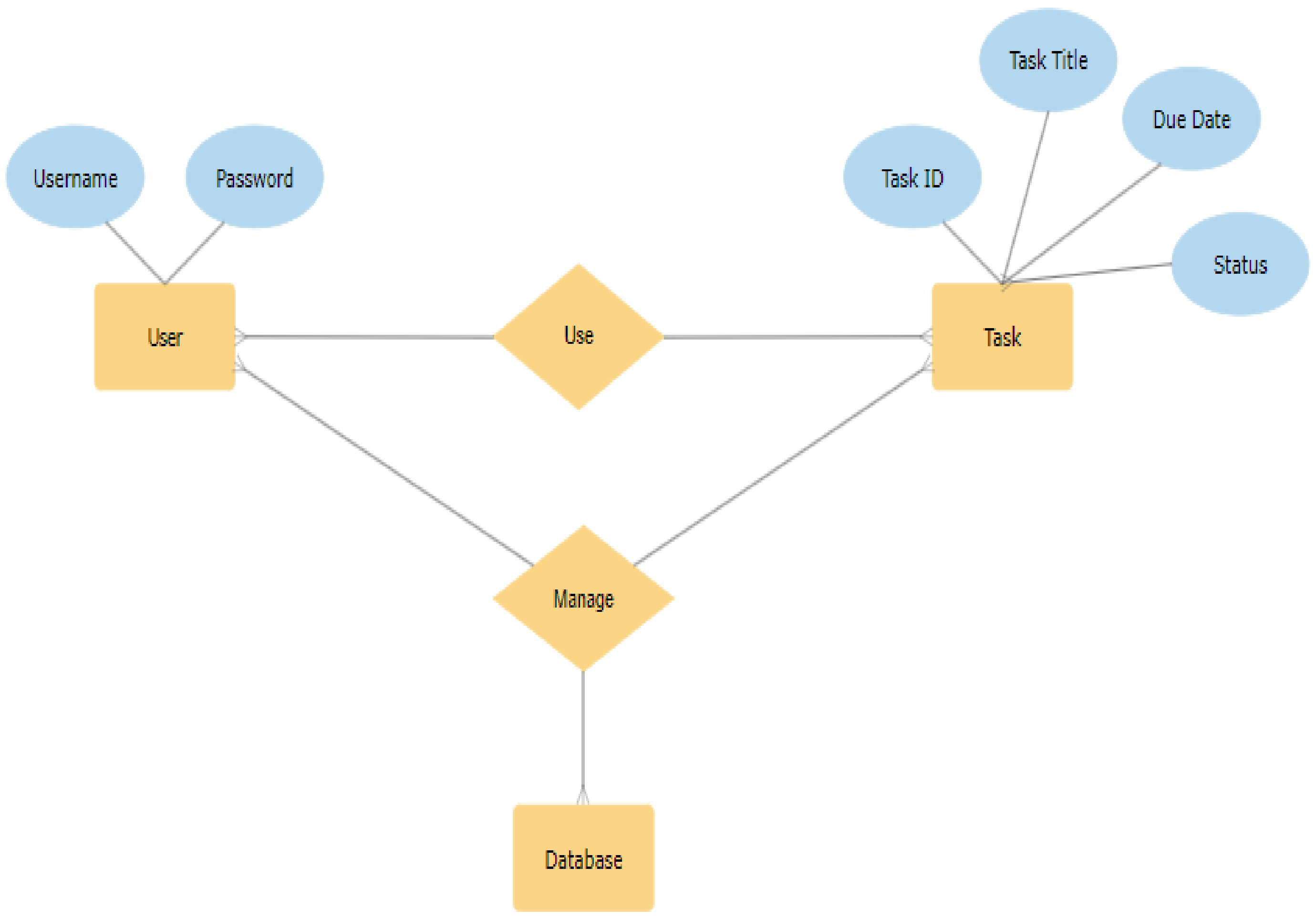
Use Case Diagram of To Do List



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.

2.2 Product Function

Entity Relationship Diagram of To Do List



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 and Database.The User entity encompasses attributes such as username and password. The Task entity includes attributes like task ID, task name, due date, and status. The Database entity manage the data.The relationships captured in the diagram include User-Task (one-to-many), Task-Database (many-to-many), and User-Database (many-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. Database :

- 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.

2.4 Operating Environment

The Application will be operating in a windows environment. To do List Application 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 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 application would be the internet connection. The hardware configuration include Hard Disk: 256 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 Requirements:

- Minimum Supported Operating Systems:
    - -Windows: Windows 10 or later (64-bit)
    - -macOS: macOS 10.15 (Catalina) or later
    - -iOS: iOS 12 or later
    - -Android: Android 8.0 (Oreo) or later
- Web Browser Compatibility:
    - Chrome, Firefox, Safari, and Edge, with the latest stable versions.
- Software Dependencies:
    - Node.js for server-side JavaScript (recommended version: LTS).
    - A modern web framework like React or Angular for web-based interfaces.
- Server-Side Software Stack:
    - Database Management System: PostgreSQL 13 or MySQL 8.
    - Web Server: Nginx or Apache.
    - Programming Languages: Node.js (Express.js), Python (for data processing), PHP (if needed).

Hardware Configuration Requirements:

- Client Devices:
    - Minimum RAM: 2 GB (for responsive performance)
    - Minimum Storage: 32 GB of available storage (for caching and app data)
    - Display: HD resolution or higher
- Server Hardware:
    - CPU: Quad-core Intel Xeon or equivalent
    - RAM: 16 GB (for small to medium-scale applications)
    - Storage: 256 GB SSD (for OS and application)
    - Network: Gigabit Ethernet
    - Redundancy: Consider RAID configurations and backup systems for data resilience.

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

Designing a graphic interface for a system involves creating a user-friendly, visually appealing, and functional interface that allows users to access and interact with news content effectively. Here are some key components and design considerations for the graphic interface of a news media system:

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 Requirements:

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:

The appendix section will contain additional information and supplementary materials related to the News Media Application, including:

- Use case diagrams
- Entity-relationship diagrams
- Mockups and wireframes of the user interface
- Flowcharts illustrating system processes
- Sample data sets for testing and development
- Technical documentation for APIs and data sources
- Any other relevant materials that provide a deeper understanding of the system's architecture and functionality.

6.3 Glossary:

The glossary will provide definitions for technical terms, acronyms, and domain-specific terminology used throughout the software requirements specification document. It will serve as a reference for stakeholders to ensure clear communication and understanding of the document.

- API (Application Programming Interface): A set of rules and protocols that allow different software applications to communicate and interact with each other.

- Data Encryption: The process of encoding data to protect it from unauthorized access or tampering, ensuring data confidentiality.

- Database Management System (DBMS): Software used to manage, store, and retrieve data from a database.

- JSON (JavaScript Object Notation): A lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate.

- GUI (Graphical User Interface): The visual interface that allows users to interact with software applications using graphical elements, such as buttons and menus.

- HTML (Hypertext Markup Language):The standard markup language for creating web pages and web applications.

- HTTPS (Hypertext Transfer Protocol Secure): A secure version of HTTP that encrypts data exchanged between a user's browser and a website.

- MySQL: An open-source relational database management system.

- Node.js: An open-source JavaScript runtime environment used for server-side programming.

- Notification: A message or alert sent to a user's device to inform them of events, such as breaking news or updates.

- Nginx: A web server and reverse proxy server used to serve web content efficiently.

- PHP: A server-side scripting language commonly used for web development.

- PostgreSQL: An open-source relational database management system.

- SQL (Structured Query Language): A domain-specific language used for managing and querying relational databases.

- SSL/TLS (Secure Sockets Layer/Transport Layer Security): Protocols used to secure data transmission over the internet.

- SRS (Software Requirement Specification): A document that outlines the requirements and specifications of a software project.

- UI (User Interface): The graphical layout and elements that users interact with in a software application.

- URL (Uniform Resource Locator): A web address that specifies the location of a resource on the internet.

- User Authentication: The process of verifying the identity of a user, typically through a username and password.

- UX (User Experience): The overall experience and satisfaction of users when interacting with a software application.

- Web Browser Compatibility: Ensuring that a website or web application functions correctly and looks consistent across different web browsers.

- Web Server: Software that serves web pages and content to users' web browsers.

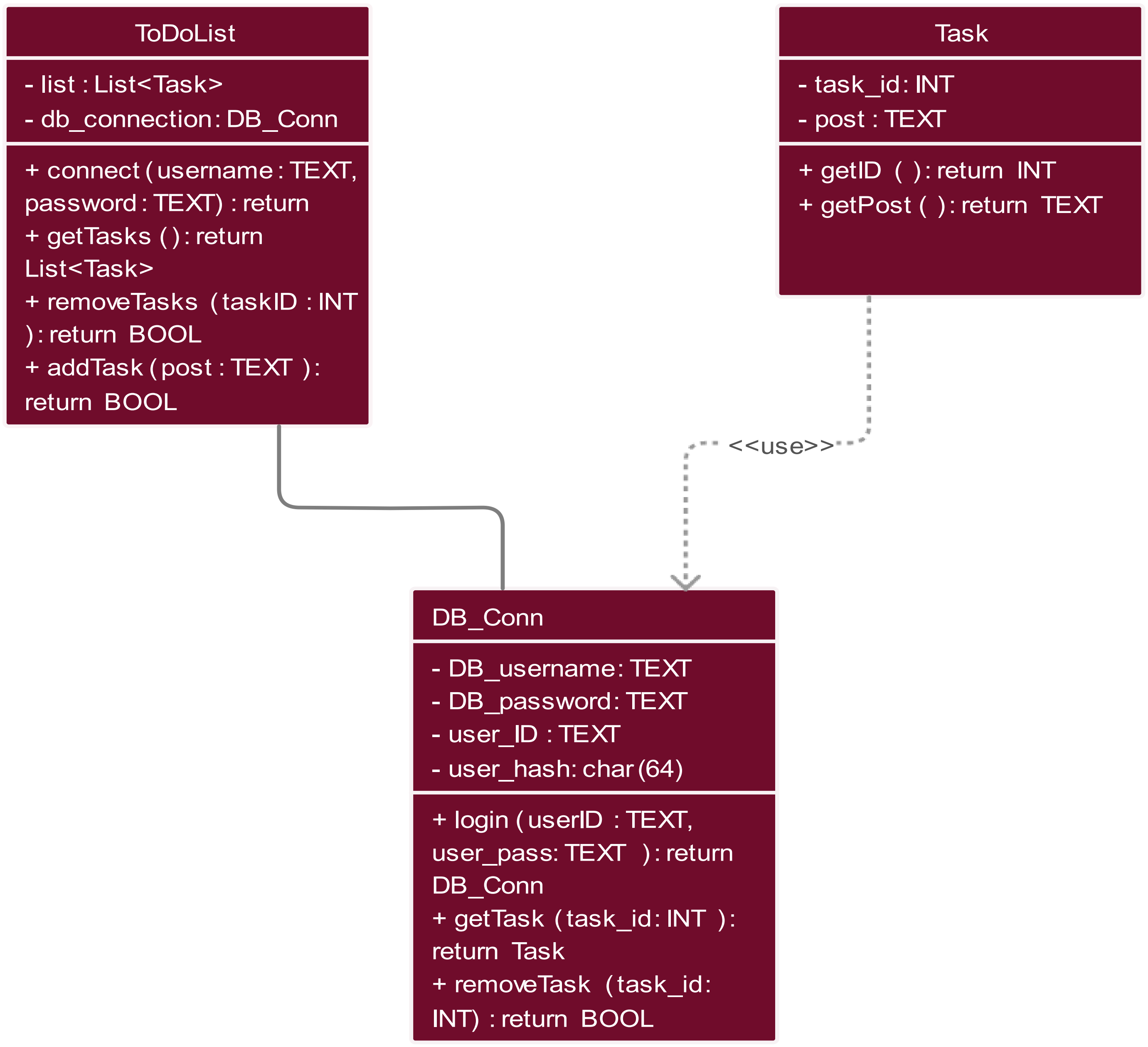
- Wireframe: A visual representation of a web page or application's layout and structure, often used in the design phase.

This glossary should help clarify the terminology used in the News Media Application Software Requirements Specification.

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 Database, serving as core components with interdependencies on other classes, thereby establishing a comprehensive framework for the to-do list website.


ToDoList
- list : List<Task>
- db_connection: DB_Conn
+ connect (username : TEXT, password : TEXT) : return
+ getTasks () : return List<Task>
+ removeTasks (taskID : INT) : return BOOL
+ addTask (post : TEXT) : return BOOL
Task
- task_id : INT
- post : TEXT
+ getID ( ) : return INT
+ getPost ( ) : return TEXT
<<use>>
DB_Conn
- DB_username: TEXT
- DB_password: TEXT
- user_ID : TEXT
- user_hash: char (64)
+ login (userID : TEXT, user_pass: TEXT) : return DB_Conn
+ getTask (task_id: INT) : return Task
+ removeTask (task_id: INT) : return BOOL