Software Requirements Specification (SRS)

Teddy's To-do list

Prepared by: Digital hub/Toqa Ezzatly

Contents

1 Introduction

1.1 Purpose

The aim of the project is to develop a user-friendly to-do list that is visually appealing, designed to attract young girls and help them learn how to schedule and prioritize their tasks and new adventures.

1.2 Scope

The project, called Teddy Pear To-Do, is developed using the MERN stack and boasts a user-friendly design with a clear layout. It allows users to prioritize their tasks as high, medium, or low and to categorize them by activity type, including personal, work, shopping, other, or all categories. Furthermore, it provides the option to schedule tasks for future dates, going beyond mere bulleted lists. Additionally, users can modify scheduled tasks, eliminating the need to delete and rewrite them from the beginning.

1.3 References

- **MongoDB Documentation**: Comprehensive guide and reference for MongoDB, the database component of the MERN stack. Available at `https://docs.mongodb.com/`.
- **Express.js Documentation**: Official documentation for Express.js, the web application framework for Node.js. Available at `https://expressjs.com/en/4x/api.html`.
- **React Documentation**: Official documentation for React, the JavaScript library for building user interfaces. Available at `https://reactjs.org/docs/getting-started.html`.
- **Node.js Documentation**: Official documentation for Node.js, the JavaScript runtime built on Chrome's V8 JavaScript engine. Available at `https://nodejs.org/en/docs/`.
- **Mongoose Documentation**: Documentation for Mongoose, an ODM (Object Data Modeling) library for MongoDB and Node.js. Available at `https://mongoosejs.com/docs/`.
- **GitHub Repository**: The project's source code and version history. Available at `https://github.com/your-repository`.
- **NPM Packages**: Various npm packages used in the project, such as `axios`, `redux`, and `jsonwebtoken`. Refer to the `package.json` file for a complete list.
- **Online Tutorials and Courses**: Various online resources and tutorials that were helpful during the development process, such as those on `https://www.udemy.com/` and `https://www.freecodecamp.org/`.

1.4 Overview

This section provides a summary of the structure of the SRS document. The document is organized into the following sections:

- **Introduction**: This section outlines the purpose, scope, and references of the SRS document. It provides an overview of the product perspective, product functions, user characteristics, constraints, assumptions, and dependencies.

- **Specific Requirements**: This section details the functional and non-functional requirements of the system. It includes descriptions of the external interface requirements and any other specific requirements necessary for the system's operation.
- **Other Requirements**: This section includes any additional requirements that do not fit into the previous sections. It may contain appendices with supplementary information, diagrams, or data.

The SRS document aims to provide a comprehensive and detailed description of the software system, ensuring that all stakeholders have a clear understanding of the system's requirements and functionalities.

1.5 Product Perspective

This MERN stack project is designed to seamlessly integrate into the current environment by leveraging modern web technologies. The product fits into the existing system as follows:

- **MongoDB**: The database layer, MongoDB, provides a flexible and scalable NoSQL database solution that stores application data in a JSON-like format. It integrates with existing data sources and supports efficient querying and indexing.
- **Express.js**: The server-side framework, Express.js, handles HTTP requests and routes, providing a robust and scalable backend. It integrates with existing APIs and middleware, ensuring smooth communication between the client and server.
- **React**: The frontend library, React, offers a dynamic and responsive user interface. It integrates with existing frontend frameworks and libraries, enhancing the user experience with its component-based architecture.
- **Node.js**: The runtime environment, Node.js, executes JavaScript code on the server side. It integrates with existing server-side technologies and tools, providing a high-performance and scalable solution for handling concurrent requests.
- **Deployment**: The product can be deployed on various cloud platforms, such as AWS, Azure, or Google Cloud, ensuring compatibility with existing infrastructure and services.
- **Security**: The product incorporates security best practices, including authentication and authorization mechanisms, to protect sensitive data and ensure secure interactions with other systems.
- **Scalability**: The product is designed to scale horizontally and vertically, accommodating increasing user demands and integrating with load balancers and caching mechanisms to optimize performance.

Overall, this MERN stack project enhances the current environment by providing a modern, scalable, and secure web application solution that integrates seamlessly with existing systems and technologies.

1.6 Product Functions

The major functions of the software include:

- **User Authentication and Authorization**: Provides secure login and registration functionalities, including password encryption and role-based access control.
- **CRUD Operations**: Allows users to create, read, update, and delete data entries in the database, ensuring data integrity and consistency.
- **Data Visualization**: Displays data in a user-friendly manner using charts, graphs, and tables to help users understand and analyze information.
- **Responsive Design**: Ensures the application is accessible and usable on various devices, including desktops, tablets, and mobile phones.
- **Real-time Updates**: Implements real-time data updates using WebSockets or similar technologies to provide users with the latest information without needing to refresh the page.
- **Search and Filtering**: Enables users to search and filter data based on various criteria, improving the user experience and making it easier to find relevant information.
- **Notifications**: Sends notifications to users about important events or updates, keeping them informed and engaged.
- **API Integration**: Integrates with external APIs to fetch and display data from other services, enhancing the functionality of the application.
- **Error Handling and Logging**: Provides robust error handling and logging mechanisms to ensure the application runs smoothly and issues can be diagnosed and resolved quickly.
- **User Profile Management**: Allows users to manage their profiles, including updating personal information and preferences.

1.7 User Characteristics

The primary goal of the project is to create a delightful to-do list for girls, assisting them in developing the habit of organizing their days and prioritizing their tasks and plans. The intended user base includes:

- **Young Girls and Teenagers**: Users who are in school or college and need to manage their academic and extracurricular activities.
- **Working Women**: Users who need to balance their professional responsibilities with personal tasks and goals.
- **Mothers and Homemakers**: Users who manage household chores, family schedules, and personal projects.
- **Tech-Savvy Users**: Users who are comfortable with digital tools and seek efficient ways to organize their tasks.

The application aims to be user-friendly, visually appealing, and motivating to encourage regular use and habit formation.

1.8 Constraints

List any design, regulatory, or environmental constraints. The constraints for this project include:

- **Design Constraints**:
  - The application must be responsive and work seamlessly on various devices, including smartphones, tablets, and desktops.
  - The user interface should be intuitive and easy to navigate, with a focus on accessibility and usability.
  - The design should be visually appealing to the target audience, with appropriate color schemes and graphics.
- **Regulatory Constraints**:
  - The application must comply with data privacy regulations, such as GDPR and CCPA, to protect user information.
  - Any user data collected must be securely stored and handled according to industry best practices.
- **Environmental Constraints**:
  - The application should be optimized for performance to ensure smooth operation even in areas with limited internet connectivity.
  - The application should minimize battery consumption on mobile devices to enhance user experience.

1.9 Assumptions and Dependencies

- **Assumptions**
  - Stable internet connection for users to access the application.
  - Users should have a basic knowledge of using web application and online tools.
  - The application will be hosted on a reliable cloud platform to access tools and resources, including IDEs, version control systems and testing environment.
  - The project will adhere to the planned timeline and budget constraints.
- **Dependencies**
  - The application depends on the availability and performance of the chosen cloud hosting provider.
  - The application uses various npm packages and libraries, which need to be maintained and updated regularly to ensure compatiblity and security.
  - The project is dependent on the timely delivery of design assests and requirements from stakeholders.
  - The application requires integeration with the existing systems or databases, which must be accessible and compatible with the new system.

2 Specific Requirements

2.1 Functional Requirements

List the functional requirements using detailed points. The functional requirements for this project include:

- **User Registration and Login**: The system shall allow users to register and log in using their email and password.
- **Task Management**: The system shall allow users to create, read, update, and delete tasks.
- **Task Prioritization**: The system shall allow users to set priorities for their tasks (e.g., high, medium, low).
- **Due Dates and Reminders**: The system shall allow users to set due dates for tasks and receive reminders.
- **Task Categories**: The system shall allow users to categorize tasks (e.g., work, personal, school).
- **Search and Filter**: The system shall allow users to search and filter tasks based on various criteria.
- **User Profile Management**: The system shall allow users to manage their profiles, including updating personal information.
- **Notifications**: The system shall send notifications to users about important events or updates.

2.2 Non-functional Requirements

Outline performance, reliability, security, and other non-functional requirements. The non-functional requirements for this project include:

- **Performance**: The system shall respond to user actions within 2 seconds.
- **Reliability**: The system shall have an uptime of 99.9
- **Scalability**: The system shall be able to handle up to 10,000 concurrent users.
- **Security**: The system shall use encryption for data storage and transmission. User passwords shall be hashed and salted.
- **Usability**: The system shall be user-friendly and intuitive, with a focus on accessibility.
- **Compatibility**: The system shall be compatible with major web browsers and mobile devices.
- **Maintainability**: The system shall be easy to maintain and update, with clear documentation.

2.3 External Interface Requirements

Describe the interfaces to external systems or devices. The external interface requirements for this project include:

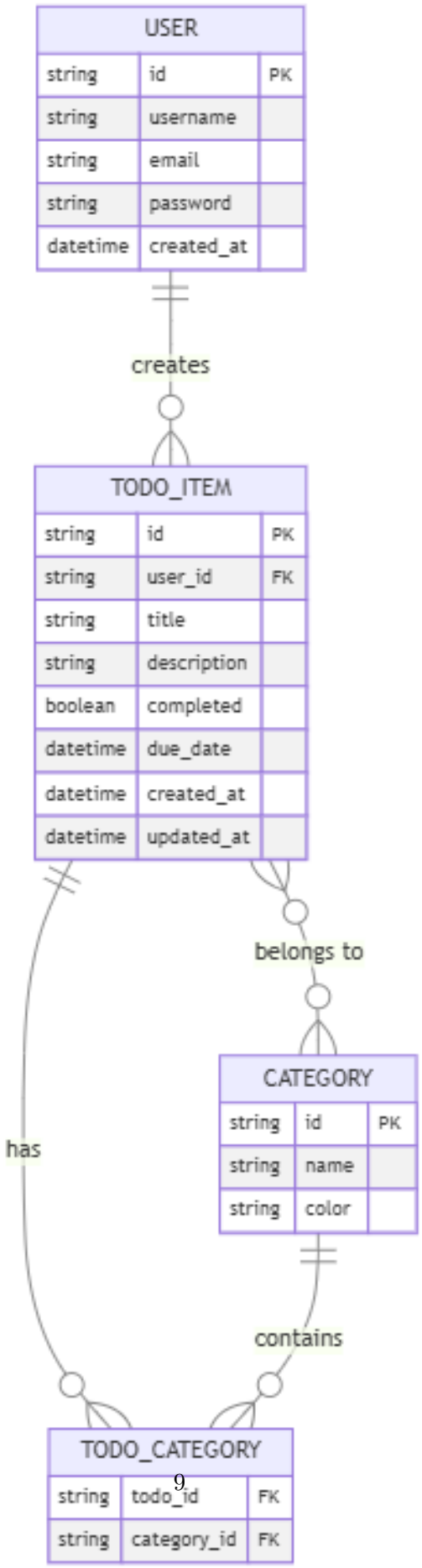
- **API Integration**: The system shall integrate with external APIs for authentication, notifications, and data fetching.
- **Database Integration**: The system shall connect to a MongoDB database for data storage.
- **Third-Party Services**: The system shall integrate with third-party services for email notifications and analytics.

3 Other Requirements

3.1 Appendices

Include supplementary information, diagrams, or data. The appendices for this project may include:

- **ER Diagrams**: Entity-Relationship diagrams to illustrate the database schema.
- **Flowcharts**: Flowcharts to depict the workflow of the application.
- **Wireframes**: Wireframes to show the layout and design of the user interface.


USER
string id PK
string username
string email
string password
datetime created_at
creates
TODO_ITEM
string id PK
string user_id FK
string title
string description
boolean completed
datetime due_date
datetime created_at
datetime updated_at
belongs to
CATEGORY
string id PK
string name
string color
has
contains
TODO_CATEGORY
string todo_id FK
string category_id FK

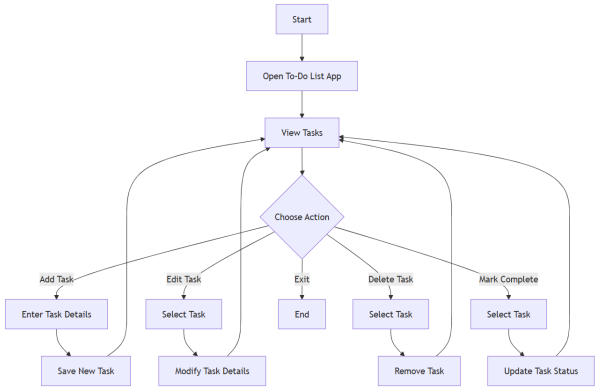


Figure 2: Workflow of the app

Revision History

Version	Date	Author(s)	Description
1.0	2024-12-1	Toqa Ezzatly	A Very basic and proto-type
1.1	2024-12-1	Toqa Ezzatly	Proto-type with some edit option
1.2	2024-12-1	Toqa Ezzatly	UI design
2.0	2024-12-3	Toqa Ezzatly	First released version
2.1	2024-12-5	Toqa Ezzatly	Categories and priority options

Figure 3: wireframe