SRS FOR TO-DO PLANNER

1.INTRODUCTION

The To-Do Planner Project aims to develop a comprehensive task management application to help users organize their daily tasks, track progress, set reminders, and enhance productivity. The application will be available as a web-based platform and as mobile applications for iOS and Android devices.

2.PURPOSE

The purpose of this document is to outline the functional and non-functional requirements of the To-Do Planner Project, serving as a foundation for design, development, testing, and validation.

3.SOFTWARE REQUIREMENTS

1.Operating system:

You'll need an operating system that matches your target platform (e.g., Windows, macOS, Linux, Android, iOS, or web-based).

2.Development Environment:

An Integrated Development Environment (IDE) like Visual Studio Code, Xcode, or Android Studio for coding and debugging tasks.

3.Programming languages:

i) Web based:JavaScript (for frontend) and a backend language like Python or Node.js.

ii)Mobile: Java/Kotlin (Android) or Swift/Objective-C (iOS).

iii)Desktop: Languages such as Python, Java, or C#.

4.Database:

Select a database management system (DBMS) like SQLite, MySQL, or PostgreSQL to store user data and task information.

5.Frontend development:

i)HTML, CSS for designing the user interface.

ii)JavaScript and a frontend framework (e.g., React, Angular, or Vue.js) for dynamic interactions.

6.Backend development:

Use a web framework (e.g., Django, Ruby on Rails, Express.js) and a backend language (e.g., Python, Ruby, Node.js) to create server-side components and APIs.

7.Web server:

Employ a web server like Apache or Nginx to host and serve your web-based application.

8.**Mobile Development Tools:**

i)For Android, use Android Studio with Java/Kotlin.

ii)For iOS, use Xcode with Swift/Objective-C.

iii)Consider cross-platform frameworks like React Native or Flutter for building apps on both Android and iOS.

9.APIs and Libraries:

Depending on your features, integrate third-party APIs (e.g., Google Calendar API) and libraries to enhance functionality.

10.Security Measures:

Implement security features including data encryption, user authentication, and authorization to safeguard user data and privacy.

4.HARDWARE REQUIREMENTS

1.Development Machine:

i)A capable computer with a modern processor (e.g., Intel Core i5 or equivalent).

ii)Sufficient RAM (8GB or more) for efficient development and testing.

iii)Adequate storage space (256GB SSD or more) for your development tools and project files.

2.Operating System:

Choose an appropriate operating system for development, such as Windows, macOS, or Linux, depending on your development stack.

3.Database Server:

If your planner involves storing user data and task lists, set up a database server (e.g., MySQL, PostgreSQL, MongoDB) or use cloud-based databases.

4.Internet Connection:

A reliable internet connection is essential for accessing development resources, downloading packages, and testing your application on various devices.

5.Backup and Version Control:

Implement robust backup procedures for your code and project files. Use version control software like Git for code management and collaboration.

6.Testing Devices:

If you plan to develop a mobile app, acquire physical devices for testing on different platforms (iOS and Android) and various screen sizes.

7.Security Measures:

Implement security features to protect user data and ensure secure communication (e.g., SSL certificates for web applications).

8.Scalability (For Production):

If you anticipate a large user base, plan for scalability requirements, such as load balancing, autoscaling, and possibly distributed databases.

5.Functional Requirements

5.1 User Registration and Authentication:

i)Users can create accounts with unique usernames and passwords.
ii)Users can log in securely using their credentials.
iii)Password reset and recovery functionality should be available.

5.2 Task Management:

i)Users can create new tasks with titles and descriptions.
ii)Tasks can be categorized by creating and assigning labels or tags.
iii)Users can prioritize tasks (e.g., high, medium, low priority).
iv)Users can set due dates for tasks.
v)Tasks can be marked as complete, in progress, or not started.
vi)Users can edit and update task details.
vii)Users can delete tasks they no longer need.

5.3 Reminders and Notifications:

i)Users can set reminders for specific tasks with customizable notification times.

ii)Users should receive notifications (email, in-app, or push notifications) for upcoming task deadlines and reminders.

5.4 User interface:

i)The user interface should be intuitive and responsive on both web and mobile platforms.

ii)Users should be able to navigate through their tasks, categories, and settings effortlessly.

iii)Visual cues should be used to distinguish task statuses and priorities.

5.5 Synchronization:

i)User data, including tasks, categories, and settings, should be synchronized across different devices.

ii)Synchronization should be seamless and real-time to ensure consistent user experience.

6.Non-Functional Requirements

6.1 Performance:

i)The system should support a large number of users concurrently.

ii)Response times for task creation, updates, and retrieval should be within a few seconds.

6.2 Security:

i)User passwords should be securely hashed and stored.

ii)Secure socket layer (SSL) should be implemented to ensure data encryption during transmission.

iii)User sessions should have proper timeout mechanisms.

6.3 Usability:

i)The user interface should be visually appealing and user-friendly.

ii)Users of varying technical expertise should be able to navigate and use the application without extensive training.

6.4 Availability:

i)The system should have high availability, aiming for 99.9% uptime.

ii)Regular maintenance and updates should be scheduled during off-peak hours.

6.5 Compatibility:

i)The web application should be compatible with major modern browsers (e.g., Chrome, Firefox, Safari, Edge).

ii)The mobile applications should be compatible with recent versions of iOS and Android.