Design document: The Ultimate To-Do List

Introduction

The purpose of this document is to specify the design decisions and their explanation, and serve as a guideline for people to see how this personal project was built from the ground-up. It includes content that ranges from a simple description of the project and its purpose, to its software architecture structure. Its target audience is whoever was looking for this GitHub repository, hoping that it is comprehensible even without a background in web development.

Although there are several "To-do list" projects for web development beginners on the internet (too much, to be precise), I want this to be unique, and to make the user feel like he can actually use it on his daily basis.

Overview

1. Executive summary

The main goal of this project is to show the community (and potential employers) what can I do with all the knowledge that I've gained throughout the last year. Don't be fooled by the 'To-Do List' part of the title: This project packs some interesting features, ranging from the simple functionality of a to-do list to API-fetching and the ability to customize its appearance like never before on these kinds of projects.

Like any other project, this will also help to improve my web development skills and help me gain context on the field (which is massive).

2. System overview

2.1. Product description

This is a simple (yet powerful) to-do list offline web application that has some interesting customization features.

2.2. Key features

- Tasks and lists management (CRUD) and storage, like a regular to-do list application.
- Background customization with images selected from the user's disk or from the Unsplash API. These images can be animated or not (depending on the user preferences).
- Support for different themes ("light" or "dark") depending on the user preferences.
- Support for storage of tasks, lists, and user preferences in a local database.

2.3. Main user activities

The user can:

- Create, Read, Update and Delete tasks and lists.
- Set different background images from his device or select them as a specified gallery of random and high-quality images from the Unsplash API.
- Choose between "ON" and "OFF" background animations.
- Choose between "Light" or "Dark" themes.

3. Design specification

3.1. Business requirements

The user expects the application to:

- Let him create, read, update, and delete tasks and lists.
- Store efficiently his tasks and lists and retrieve them as quickly as possible.
- Be compatible with his PC, tablet, or phone.
- Let him customize the application's background and themes.

- Save all his tasks, lists, and preferences so that they remain consistent over time.

3.2. User interface design

It will be analyzed for each functionality of the application:

3.2.1. To-do list

If the user is new, he will be welcomed with the layout shown in *Fig.1*. If the user has inserted some tasks in the current list, the layout will be the shown in *Fig.2*. If the user wants to see the lists he has created, the layout shown in *Fig.3* will be displayed.

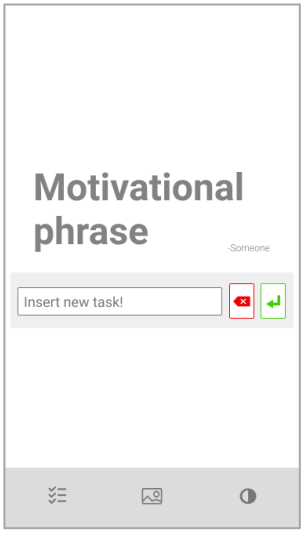

Fig. 1: Welcoming page

Fig. 2: Main page

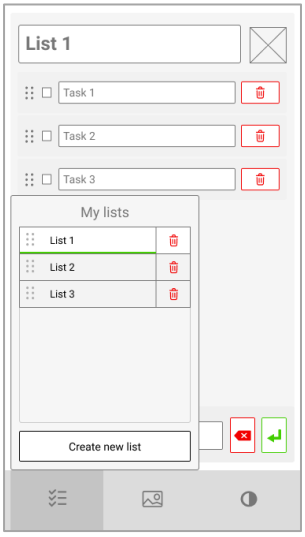

Fig. 3: User's list

3.2.2. Custom preferences: Background images and animations

The user can change the background image and its animation mode by selecting the corresponding option as shown in *Fig.4*.

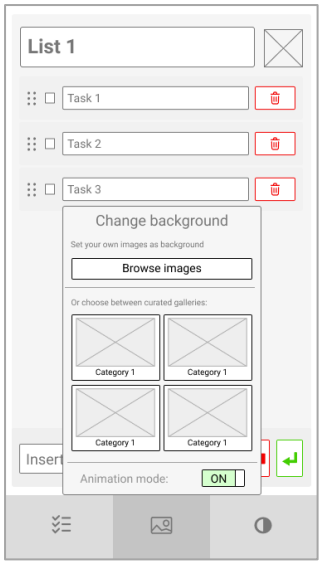

Fig. 4: Background selection

3.2.3. Custom preferences: Toggle themes

The user can toggle between "Light" and "Dark" themes as shown in *Fig.5*.

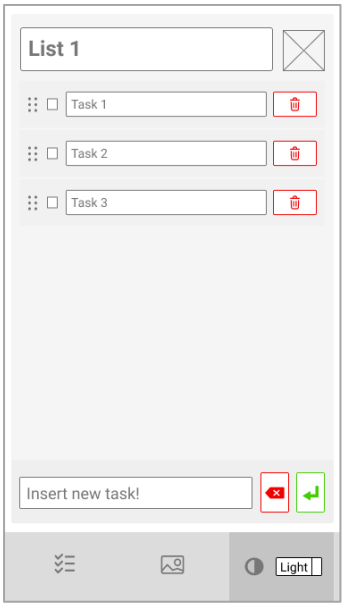

Fig.5: Theme selection

4. Design guidelines and considerations

4.1. Operational environment

The application will run on any major mobile/PC browser (Chrome, Edge, Safari, Mozilla, Opera, Samsung Internet, Internet Explorer 11).

4.2. Dependencies

For the background image customization functionality, the system depends on a third-party API from *Unsplash*. Visit Unsplash API for more information. Also, the browser must be compatible with IndexedDB (all modern browsers support it).

5. Software architecture

5.1. Requirements

5.1.1. Functional requirements

User story: As a new user, I want to be able to get started quickly on the application.

- The system will allow users to create tasks from the first second, no authentication or mid-steps.
- The system will allow users to customize the application from the first second.

User story: As a user, I want to manage my tasks and lists as easy as possible.

- The system will allow users to create, read, update, and delete tasks and lists from the UI by setting an intuitive and easy-to-use layout and page elements.

User story: As a user, I want to customize the application to my will.

- The system will allow users to change the background image from two sources: 1) The user's local images. 2) Curated images fetched from the Unsplash website.
- The system will allow users to toggle animations for the background images.
- The system will allow users to change the theme of the application between "Light" and "Dark".

User story: As a user I want to leave the page, and if I come back later, I expect to find everything as I left it.

- The system will save changes on runtime, notifying about the state of the changes with an indicator in the UI.
- The system will return the application to the state it had when the user left the application.

User story: As a user, I want to use the application offline.

- The system will be able to run offline.
- The system will notify the user if the application is online or offline using an indicator, and the only functionality that won't work offline will be the image fetching from the Unsplash API.

5.1.2. Non-functional requirements

- Maintainability: The application needs to be open for the inclusion of new potential features or be able to merge to a bigger application in the future.
- Scalability: The application needs to serve millions of English-speaking users.

- Reliability and safety: The application needs to store user data and keep it indefinitely and securely.
- Usability: The application needs to be user-friendly by having a simple and easy-to-understand UI.
- Efficiency: The application needs have a responsive UI and be able to load and save user information quickly.
- Portability: The application needs to run in PCs, tablets, and mobile devices properly.

5.1.3. Restrictions and constraints

- Time to market: The application needs to be deployed by Monday, July 26th.
- Cost: The application has a limited budget. All of it will keep the developer alive (food, water, shelter from the elements, electricity, and internet).

5.2. Software architecture diagram

As seen in *Fig.6*, The Ultimate To-Do List is divided on three key modules (see the *documentation page* for more information about each module and its components):

Storage
Service that deals with the interaction with the database to save and retrieve app information.
It has three different modules:
- Database (object)
- Database management module
- Information management module

Customization
Service that deals with the user preferences regarding the customization of the UI (background images, animation mode, and theme).

It follows the *microservice architecture pattern*, in which each microservice (background and theme) is completely independent of the other.

To-Do
Service that deals with all the to-do list functionality (tasks and lists management).

It follows the *n-tier architecture pattern*, which is composed of three layers:

- Presentation: Layer that manages all user-triggered events and rendering of the UI.

- Logic: Layer that manages all logic regarding tasks and lists inside the application and connection to the *Storage service* (database) to save and retrieve the information.

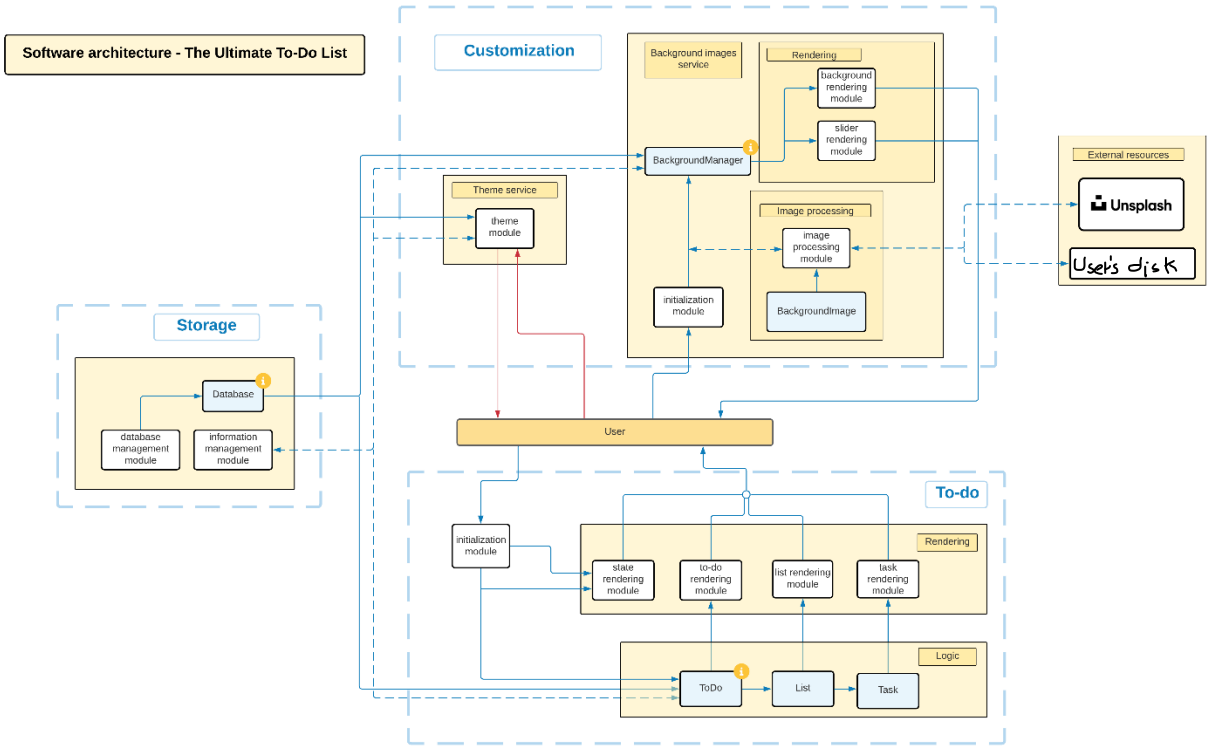

Fig. 6: Software Architecture diagram

5.3. Hardware

5.3.1. Hardware architecture

As the application will be serverless, all the processing will be done on the client-side.

5.3.2. Hardware requirements

PC: Minimum: 2GB of RAM, and CPU-integrated graphics.
Mobile devices: Minimum: 1GB of RAM.

All devices must have a modern browser installed, have at least 5kB of available storage, and average internet connection.

5.4. Data management

5.4.1. Database type

The application will use *IndexedDB*, a JavaScript API supported by most browsers. It is a lightweight non-relational database that stores information indefinitely in the user's machine.

The application will store and retrieve information about tasks, lists and background images as the user interacts with it, as the following section will explain.

5.4.2. Database model

The IndexedDB model is a NoSQL database that uses *document data stores* as its way to store data.

It has three main object stores:

- Lists: Stores the different lists that the user has inside the application.
- To-do information: Stores information about the current list being rendered in the UI and the next index of the list to be created.
- Custom preferences: Stores the user's custom preferences, like background images.