To-Do Application Requirement Specification

Author: Nkwi Cyril

@TechChantier

1. Introduction

Todo application is a software application that is used to manage the day-to-day tasks of its users.

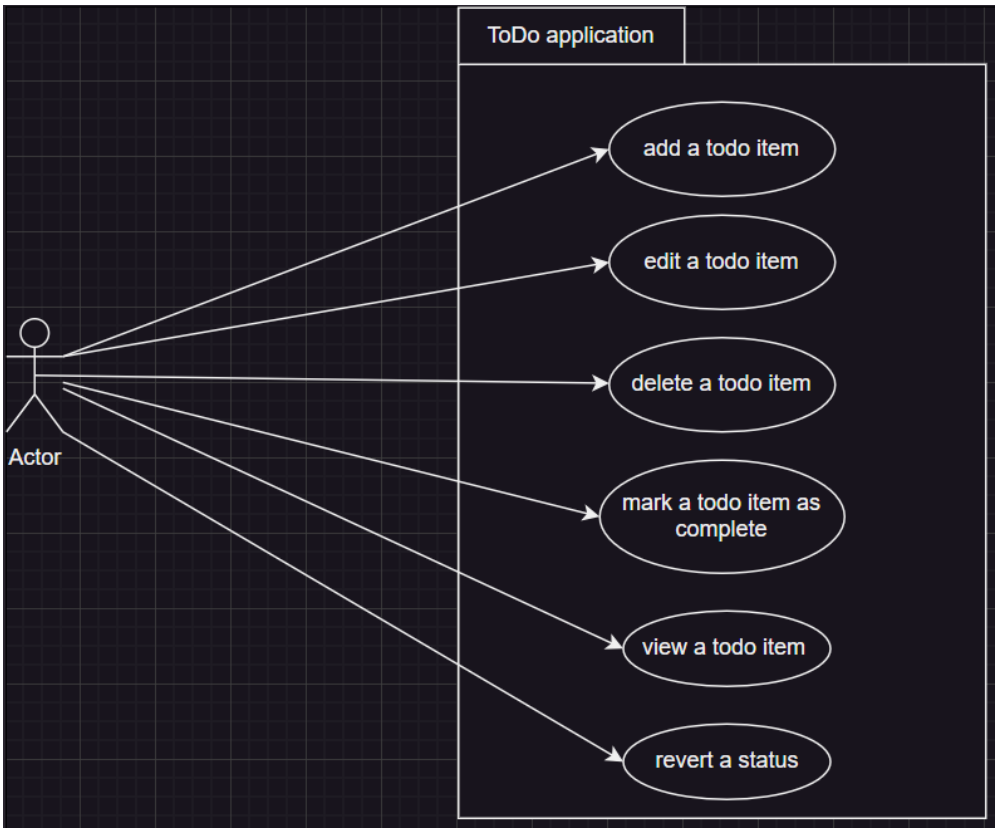
2. Overall Perspective

2.1. Application Description

This is an application which will give its users the super-power of customizing their task according to the status of each task. From creating a task and marking the task as complete to uploading images that suit the task description. Below has enlisted design and other features needed in order to build such an application.

2.2. Use Case Scenario

Below, with the help of statements and a use case diagram shows how this to-do application will be used.



Description:

- A user should be able to **add or create** a to-do item.
- A user should be able to **edit or update** a to-do item.
- A user should be able to **delete** a to-do item.
- A user should be able to **mark a to-do item as complete.**
- A user should be able to **view** a to-do item.
- A user should be able to **revert** a status change on a to-do item. That is from **Not Done to Done** and from **Done to Not-Done.**

2.3. To-do item

To facilitate database design and implementation, each to-do item needs to have the following attributes:

1. **Id**: This will be used to pull from or make changes to a particular to-do item in the database.
2. **Title:** Issued by a user.
3. **Description:** Issued by a user and gives a brief description of the task
4. **Image link:** Link to the image uploaded by the user.
5. **Time completed:** Time when the to-do item was marked as done. This is not set when the to-do item is incomplete.

The diagram below shows the attributes of a to-do item.

3. Functional Requirements

Here are the various functionalities the to-do application is meant to incorporate.

A.

- **User story:** A user enters the application to the **View** page and wants to add a task to do.
- **Functional requirement:**
  - The application must allow users to add tasks alongside upload images for them with descriptions and titles for each task.
  - This is to be made possible with the help of an **Add +** button which when clicked, renders the **New** page with a form to add tasks through.

B.

- **User story:** A user made a mistake while creating the task and wants to edit a task. Maybe it was a spelling error.
- **Functional requirement:**
  - The application must allow users to edit tasks by altering the previous content of that task item.
  - This will be done on the **Edit** page with a submit button to confirm the update.

C.

- **User story:** A user finds a task unnecessary at the moment and wants to delete a task.
- **Functional requirement:**
  - The application must allow users to delete tasks in the **View** page by the click of the delete button.
  - A popup notification should be available in order to confirm the delete.

D.

- **User story:** A user wants to view the full content of a set task.
- **Functional requirement:**
  - The application must allow users to view the complete content and details of a task.
  - A **View button** should be made available for this to happen in the **View** page which will then render the content in the **Show** page.

E.

- **User story:** In order to make a task convey information about it, a user wants to add an image to the task being created.
- **Functional requirement:**
  - The application must allow users to upload an image from their local storage to visualize the set task.

F.

- **User story:** A user has completed the set task and wants to mark it as done.
- **Functional requirement:**
  - The application must allow users to make a to-do item as done.
  - **Done** button should be provided for this and on click, add the item to the **done section.**

G.

- **User story:** A user marked a task done by mistake.
- **Functional requirement:**
  - The application must allow users to revert their option to mark a task done.
  - **Revert** button should be provided on each done task below to revert change.

4. Pages and Routes Design

This application requires 4 pages for proper functionality. These include:

A. View Page

- Route to be rendered: **http://localhost/**
- Displays all the tasks created (those to be done and those which are done).
- Render all tasks in the database according to status.

B. Show Page:

- Route to be rendered: **http://localhost/show/<item_id>**
- Display elements and content of a specific task.
- Makes use of the item id as parameter.

C. Edit page:

- Route to be rendered: **http://localhost/edit/<item_id>**
- Update or edit the content of specific task.
- Makes use of the item id as parameter.

D. Create page:

- Route to be rendered: **http://localhost/create.**
- Create or add a new task and display it in the View page.

5. Technological Requirements
   A. Client-Side:
   - HTML
   - CSS
   - jQuery

   B. Server-Side:
   - PHP

   C. Database:
   - SQL using MySQL DBMS
   - XAMPP Server