**System Requirements Specification**

**(SRS)**

Ticktrak App

**Version:** 1.3

**Last Updated:** June 17, 2025

**Author:** Carlo Merin

1. **Introduction**
   1. Purpose

The Ticktrak App is a task-tracking platform designed to help users manage their tasks and track progress. The main objective of this is to enable the users to create, view, update, delete and complete tasks while retain the view of their progress.

* 1. Scope

This Application includes:

* + - Welcome/Landing Page
    - User Authentication (Login/Register)
    - Dashboard View
    - Side and Top Navigation Menu
    - User Profile/Setting
    - Task List (Table with search and pagination controls)
    - Sub Task List (each task have multiple sub tasks)
  1. Tech Stack

This Application uses:

* + 1. PHP
       - Laravel v.10
       - Composer
    2. JavaScript
       - Node JS
       - Alpine JS
       - Rest API / Fetch
    3. CSS
       - Tailwind CSS
       - Flowbite
    4. Icons
       - Heroicons
       - Untitledui
    5. Graphs
       - Apexcharts

1. **Definition, Acronyms and Abbreviation**

|  |  |
| --- | --- |
| **Terms** | **Definition** |
| SRS | Software Requirement Specification |
| API | Application Programming Interface |
| CRUD | Create, Retrieve, Update, Delete |

1. **User Role and Permissions**
   1. User – Create, View, Update and Delete Tasks and Sub Tasks.
   2. Admin – Manage All the users.
2. **Functional Requirements**
   1. User Authentication
      1. Register and Login
   2. User Profile/Setting
      1. Add Profile Picture
      2. Update Information
      3. Change Password
      4. Delete Account
   3. Task Management
      1. Create new Tasks
      2. View List of Tasks
      3. Update / Delete Tasks
   4. Sub Task Management
      1. Create new Sub Tasks
      2. View List of Sub Tasks
      3. Update / Delete Sub Tasks
3. **Non-Functional Requirements**
   1. Security: Password hashing
   2. Performance: Response should be 500ms minimum
   3. Validation: Proper input sanitation and error message
4. **Database Schema (Tables)**
   1. Table: Users
      1. id – Primary Key
      2. name – String | max:255
      3. profile – String | Null
      4. email – String | Email | Unique
      5. password – String | Hash
   2. Table: Tasks
      1. id – Primary Key
      2. title – Required | String | max:255
      3. details – Null | text
      4. priority – Required | String | enum [‘low’, ‘medium’, ‘high’] | default = low
      5. status – Required | String | enum [‘pending’, ‘completed’] | default = pending
      6. due\_date – Null | date
      7. user\_id – Foreign Key | exists: users, id
   3. Table: Sub Tasks
      1. id – Primary Key
      2. description – Required | text
      3. is\_complete – Boolean | String | default = false
      4. due\_date – Null | date
      5. task\_id – Foreign Key | exists: tasks, id
5. **Routes (API)**
   1. Task API Endpoints (API/TaskApiController)
      1. middleware – auth:sunctum

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Method** | **Url** | **description** |
| index | GET | /api/tasks | View task list |
| store | POST | /api/tasks | Create new tasks |
| update | UPDATE | /api/tasks/{id} | Update selected tasks |
| destroy | DELETE | /api/tasks/{id} | Delete selected tasks |

* 1. Sub Task API Endpoints (API/SubTaskApiController)
     1. middleware – auth:sunctum

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Method** | **Url** | **description** |
| index | GET | /api/tasks/{taskId} | View subtask list |
| store | POST | /api/tasks/{taskId} | Create new subtasks |
| Update Mark | PATCH | /api/tasks/{taskId}/subtasks/{subtaskId} | Update selected subtasks to complete/done |
| Destroy | DELETE | /api/tasks/{taskId}/subtasks/{subtaskId} | Delete selected subtasks |

1. **Models**
   1. User
      1. Fillable Attributes
         * name | profile | email | password
      2. Relationship
         * hasMany(Task)
   2. Task
      1. Fillable Attributes
         * title | details | priority | status | due\_date | user\_id
      2. Relationship
         * belongsTo(User)
         * hasMany(SubTasks)
   3. SubTask
      1. Fillable Attributes
         * description | due\_date| is\_complete | task\_id
      2. Relationship
         * belongsTo(Task)
2. **User Interface**
   1. Top Navbar
      1. Search Function
      2. Dark/Light Mode
      3. Profile (Name and Email)
   2. Side Navbar
      * Default View: Shows only Icons (Menu and Others)
      * Toggled View: Shows both Icons and label for each items in the side panel
      1. Application Logo
      2. Menu (placeholder)
         * Dashboard
         * My Task
      3. Others (placeholder)
         * Setting
         * Logout
   3. Task List UI (CRUD)
      1. Header:
         * Label: Task List
         * Button: Add Tasks (Modal)
           + Title: Add Task
           + Input Fields: Title, Priority, Due Date, Details
           + Close Button: Close
           + Submit Button: Add Task
      2. Main:
         * Search Box
         * Table
           1. # | Title | Description | Priority | Due Date | Action
           2. Action:

Anchor Tag Button: Jump Into Sub Task page with Selected Task

Button: Update Task (Modal)

Title: Update Task

Input Fields: Title, Priority, Due Date, Details

Close Button: Close

Submit Button: Update Task

Button: Delete Task (Modal)

Title: Delete Task

Description: Are you sure you want to delete this task?

Close Button: Close

Submit Button: Delete Task

* + - * Pagination Controls
        1. Limit per page (10, 25, 50, etc.)
  1. Sub Task UI (CRUD)
     1. Header:
        + breadcrumb: Go back to the task page
     2. Main:
        + Grid
          1. Left Side

Chart: Milestones Progress

Chart

Priority

Deadline

Form: Add Milestones

Input Field: Description, Due Date

Submit Button: Add Milestones

* + - * 1. Right Side

Title: Title of the selected tasks

Details: Details of the selected tasks

Notes: Please Check the box after completing the sub tasks

Table:

Checkbox | Description | Deadline | Action

Checkbox: to mark as complete

Action: Delete

**Change Logs:**

1. v.1.1

Add Collapsible sidebar to maximize the space of the content of the main page.

* 1. Default: showing sidebar icons only.
  2. Toggled: shows both icons and label.

1. v.1.2

Implementing CRUD API for Tasks.

* 1. Add Models, Controllers and Relationship
  2. Add Routes API / Rest API

1. v.1.3

Implementing CRUD API for SubTasks.

* 1. Add Models, Controllers and Relationship
  2. Add Routes API / Rest API