**System Requirements Specification**

**(SRS)**

Ticktrak app

**Version:** 1.5

**Last Updated:** June 18, 2025

**Author: C**arlo Merin

1. **Introduction**
   1. Purpose

Many businesses and individuals struggle to efficiently track tasks and subtasks. The primary goal of the TickTrak app is to help users create, view, update, delete, and mark tasks as complete, while providing a clear view of their overall progress.

TickTrak offers a simple and organized platform for task tracking and team collaboration. It allows users to manage tasks effectively, monitor progress, and work together as a team. Users can form teams for collaborative task management, with team owners having the ability to manage team memberships.

* 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)
    - Progress tracking chart (percentage of completed sub tasks)
    - Collaboration teams / groups
  1. Tech stack

This application uses:

* + 1. Php
       - Laravel v.10 php
    2. Database:
       - Mysql
    3. Javascript
       - Node js, alpine js, rest api / fetch
    4. Css
       - Tailwind css, flowbite
    5. Icons
       - Heroicons, untitledui
    6. 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:
      1. Manage own profile.
      2. Create, view, update and delete tasks and sub tasks.
   2. Admin:
      1. Manage all the users.
      2. Perform administrative tasks.
   3. Teams owner:
      1. Manage team and members
      2. Accept / reject request to join
      3. Change visibility setting
   4. Team members:
      1. View team tasks and collaborate
      2. See all members
      3. Leave a team
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
   5. Progress tracking
      1. Calculate the percentage of completed and total sub tasks
         * Percentage = (number of completed sub tasks / total number of sub tasks) \* 100
      2. Add progress bar and chart for representation
      3. Add confetti for interaction
   6. Collaborative team
      1. Create a team will generate group code
      2. User who creates a team will be an owner
      3. User can join based on visibility setting
      4. Team owner manage settings
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 | delete: cascade
      8. Team\_id – foreign key | exists: teams, id | null | delete: set null
   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 | delete: cascade
   4. Table: teams
      1. Id – primary key
      2. Team\_Name – required | string | max:255
      3. Team\_Code – required | string | unique | max: 10
      4. Visibility – required | string | enum [‘private’, ‘public’, ‘open’] | default: private
      5. Owner\_id – foreign key | exists: users, id | delete: cascade
   5. Table: team members
      1. Id – primary key
      2. Role – required | string | enum [‘owner’, ‘editor’, ‘member’] | default: member
      3. User\_id – foreign key | exists: users, id | delete: cascade
      4. Team\_id – foreign key | exists: teams, id | delete: cascade
   6. Table: team join requests
      1. Id – Primary Key
      2. Status – required | string | enum [‘pending’, ‘approved’, ‘reject’] | default: pending
      3. User\_id – foreign key | exists: users, id | delete: cascade
      4. Team\_id – foreign key |exists: teams, id | delete: cascade
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:sanctum

|  |  |  |  |
| --- | --- | --- | --- |
| **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. Team api endpoints (api/teamapicontroller)
     1. Middleware – auth:sunctum

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Method** | **Url** | **Description** |
| Index | Get | /api/teams | List of teams belongs to |
| Show | Get | /api/teams/{team\_name} | View specific team |
| Store | Post | /api/teams | Create new teams |
| Destroy | Delete | /api/teams/{teamId} | Delete Team |

* 1. Team member api endpoints (api/teammemberapicontroller)
     1. Middleware – auth:sunctum
     2. Authentication – is Owner

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Method** | **Url** | **Description** |
| Index | Get | /api/teams/{team\_name} | View member list |
| Change role | Patch | /api/teams/{teamId}/members/{userId)/promote | Change member role |
| Destroy | Delete | /api/teams/{teamid}/members/{userid} | Remove a team member |

* 1. Team join requests api endpoints (api/teamjoinrequestapicontroller)
     1. Middleware – auth:sunctum
     2. Authentication – is Owner

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Method** | **Url** | **Description** |
| Index | Get | /api/teams/{teamId} | View member list |
| Join | Post | /api/teams/{teamId}/join | Join team via team code |
| Status | Post | /api/teams/{teamid}/request/{userId}/status | update user request to status |
| Destroy | Delete | /api/teams/{teamid}/members/{userid} | Remove team member |

1. **Models**
   1. User
      1. Fillable attributes
         * Name | profile | email | password
      2. Relationship
         * Hasmany(task)
         * BelongstoMany(team) -> withPivot(‘roles’) -> withTimestamp()
   2. Task
      1. Fillable attributes
         * Title | details | priority | status | due\_date | user\_id | team\_id
      2. Relationship
         * Belongsto(user)
         * Hasmany(subtasks)
         * BelongstoMany(team)
   3. Subtask
      1. Fillable attributes
         * Description | due\_date| is\_complete | task\_id
      2. Relationship
         * Belongsto(task)
   4. Team
      1. Fillable attributes
         * Team\_name | team\_code | visibility | owner\_id
      2. Relationship
         * Belongsto(user) | name: owner | owner\_id
         * BelongstoMany(user) -> withPivot(‘roles’) -> withTimestamps() | name: members | team\_ members
         * Hasmany(teammember) | name: teamMembers
         * Hasmany(teamjoinrequest) | name: joinrequests
   5. Team member
      1. Fillable attributes
         * Role | user\_id | team\_id
      2. Relationship
         * Belongsto(user)
         * Belongsto(team)
   6. Team join requests
      1. Fillable attributes
         * Status | user\_id | team\_id
      2. Relationship
         * BelongsTo(user)
         * BelongsTo(team)
2. **User interface**
   1. Top navbar
      1. Search function
      2. Dark/light mode
      3. User 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. Teams / groups (placeholder)
         * Button: create team (modal)
           1. Title: create team
           2. Description: creating this group will enable team collaboration and task sharing.
           3. Input field: name
           4. Close button: close
           5. Submit button: create team
      4. Others (placeholder)
         * Setting
           1. Update Profile
           2. Change Password
           3. Delete Account
         * Logout
   3. Task list UI (crud) Page
      1. Header:
         * Label: task list
         * Button: add tasks (modal)
           1. Title: add task
           2. Input fields: title, priority, due date, details
           3. Close button: close
           4. Submit button: add task
      2. Main:
         * Search box
         * Table
           1. # | title | progress | priority | due date | action

*Note: remove the* ***description column in a table to be replace by progress bar*** *for initial representation of the sub tasks completed percentage.*

* + - * 1. 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) Page
     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 the sub tasks as complete

Action: delete

* 1. Team UI Page
     1. Team sidebar
        + Team dashboard
        + Team tasks
          1. Sub tasks
        + Settings (Owner view)

**Change logs:**

1. **V.1.1 ( june 14, 2025)**

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 ( june 15, 2025)**

Implementing crud api for tasks.

* 1. Add models, controllers and relationship
  2. Add routes api / rest api

1. **V.1.3 ( june 16, 2025)**

Implementing crud api for sub tasks.

* 1. Add models, controllers and relationship
  2. Add routes api / rest api

1. **V.1.4 ( june 17, 2025)**

Implementing progress percentage calculation for sub tasks.

* 1. Show the percentage change when mark as complete in chart.
  2. Shows confetti when all sub tasks completed for interaction to the users.
  3. Shows alert message when all sub tasks complete.

1. **V.1.5 ( june 18, 2025)**

Implementing team / collaborations with join logic and visibility feature

1. Add team / group placeholder menu in sidebar
2. Add modal button beside placeholder menu for creating new team
3. Add tables, modes, controllers and relationship
4. Enable user to request join via team code
5. Allow team owner to approve and reject users join requests
6. Add visibility setting