This application is a to-do manager that keeps track of tasks that you need to do, supporting basic CRUD (Create, Read, Update and Delete), organisation and search functionalities. This document contains a non-exhaustive list of expected use cases and a summary of its execution plan.

For all use cases below, the *System* is the application and the *Actor* is the user.

Use case: UC01 – Create Task

MSS:

1. Actor chooses to create a task.
2. System requests for details of the task.
3. Actor enters the requested details.
4. System adds the task to the manager and displays its details.

Use case ends.

Extensions:

3a. System detects an error in the entered data.

3a1. System requests for correct details.

3a2. Actor enters new details.

Steps 3a1 and 3a2 are repeated until the data entered is correct.

Use case resumes from step 4.

*a. At any time, Actor chooses to cancel task creation process.

*a1. System requests to confirm the cancellation.

*a2. Actor confirms the cancellation.

Use case ends.

Use case: UC02 – Search Task

MSS:

1. Actor chooses to search for a task.
2. System requests for query details.
3. Actor enters the requested details.
4. System displays all tasks that matches the query.

Use case ends.

Extensions:

3a. System detects an error in the entered data.

3a1. System requests for correct details.

3a2. Actor enters new details.

Steps 3a1 and 3a2 are repeated until the data entered is correct.

Use case resumes from step 4.

Use case: UC03 – Read Task

Preconditions: System contains one or more tasks.

MSS:

1. Actor chooses a task to read.
2. System displays details of the task.

Use case ends.

Use case: UC04 – Update Task

Preconditions: System contains one or more tasks.

MSS:

1. Actor reads a task (UC03).
2. Actor chooses to update the selected task.
3. System requests for new details.
4. Actor enters the requested details.
5. System updates the details of the task.

Use case ends.

Extensions:

4a. System detects an error in the entered data.

4a1. System requests for correct details.

4a2. Actor enters new details.

Steps 4a1 and 4a2 are repeated until the data entered is correct.

Use case resumes from step 5.

*a. At any time after step 2, Actor chooses to cancel task updating process.

*a1. System requests to confirm the cancellation.

*a2. Actor confirms the cancellation.

Use case ends.

Use case: UC05 – Delete Task

Preconditions: System contains one or more tasks.

MSS:

1. Actor chooses to delete a task.
2. System requests for the task(s) to be deleted.
3. Actor chooses task(s) to be deleted.
4. System requests to confirm the deletion.
5. Actor confirms the deletion.

Use case ends.

Extensions:

*a. At any time, Actor chooses to cancel task deletion process.

*a1. System requests to confirm the cancellation.

*a2. Actor confirms the cancellation.

Use case ends.

This application follows the *model-view-controller* (MVC) design pattern, in which the logic of the application is separated into distinctly demarcated components. The details and implementation of each component are as follows:

- View
  - The *view* component comprises the front-end of the application. This is the component that the user sees in the form of the user interface.
  - The view component for this project is currently planned to show an overall view of all the tasks, with limited details shown only. Users can select specific tasks to show comprehensive details of the tasks.
  - This component will be written in React and styled with CSS and its libraries (such as Bootstrap).
- Controller
  - The *controller* component handles the execution of user events, such as clicks or key presses – its functionalities integrated into the view component, so users will not directly see this component.
  - The controller component for this project will listen to activities from the user, and update both the view and the model components to display the relevant information to the user.
  - This component will also be written in React.
- Model
  - The model handles the back-end of the application. This component interacts with the database, which stores the tasks given by the user. It listens for the controller and sends the relevant data to the view component.
  - The model component for this project interacts with a PostgreSQL database, and pulls data from it.
  - This component will be written in Ruby on Rails.