MVP

Automatic Project Detection And Tooling For Devs

1 User stories

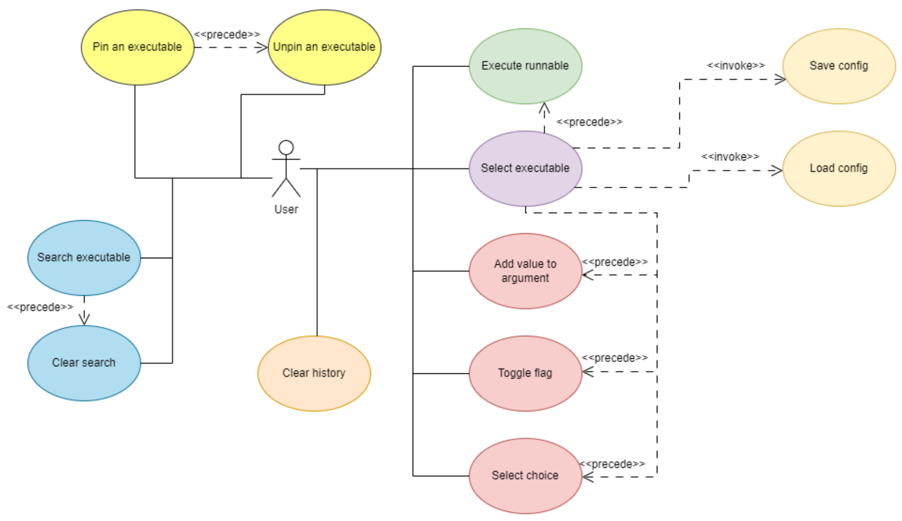

Figure 1: Use case diagram

1.1 Select executable

- **As a** user **I want** to select an executable or a script from the list of programs that the app found, **so that** I can configure its arguments.

1.2 Add value to argument

- **As a** user **I want** to give a value to an argument **so that** I can add it to the configuration.

1.3 Toggle flag

- **As a** user **I want** set a flag **so that** I can chose that the flag stores True or False.

1.4 Select choices

- **As a** user **I want** to choose a value from the list of possible choices for the argument **so that** I can set it.

1.5 Execute programs

- **As a** user **I want** to execute a configured program **so that** I can press a button on the screen and the tool runs it with the given arguments.

1.6 Save config

- **As a** user **I want** to save the configuration that I gave before **so that** the program saves it for me.

1.7 Load config

- **As a** user **I want** to load the previously saved configuration **so that** the program loads it for me.

1.8 Clear history

- **As a** user **I want** to clear my configuration history **so that** the program deletes every configuration.

1.9 Pin an executable

- **As a** user **I want** to pin an executable or script **so that** the program highlights it.

1.10 Unpin an executable

- **As a** user **I want** unpin an executable **so that** the program removes the highlight.

1.11 Search executable

- **As a** user **I want** find a specified program **so that** I can search it from the list.

1.12 Clear search

- **As a** user **I want** to clear the search history **so that** the program doesn't show in the list of previously searched items.

2 Prioritized user stories

Prioritizing with the help of MoSCoW method.

- M - Must have
- S - Should have
- C - Could have
- W - Won't have

2.1 Must have user stories:

- Select executable:
  - Filters all executables from the working direcoty.
  - Collects all details of these, such as description, name, and arguments. It also collects all information about the arguments.
  - Full path to the program, that will be needed during execution.
- Execute program:
  - Executing the programs with using its full path and the appropriate compiler or interpreter.
  - If information available of the values of the arguments, These must be used during execution.
- Add argument
  - This user story ensures that the user can provide a value for an argument as input.

2.2 Should have user stories

- Toggle flag
  - If the argument is a flag, user can switch whether it should store a true or false.
- Select choices
  - If there are choices within the argument, the user should select one of them.
- Save config
  - Users are able to store their configurations.
- Load config

- Users are able to load their previously saved configurations.

- Clear history
  - Users are able to delete their configurations.

2.3 Could have user stories

- Search an executable
  - Users are able to search for specified scripts.
- Clear search
  - After searching users can delete their search history.

2.4 Won't have user stories

- Pin an executable
  - Users are able to pin programs as favourite, and those will be highlighted.
- Unpin an executable
  - Users are able to unpin a pinned program.

3 The content of the MVP

MVP should contain these user stories, which have been prioritized.

- Select executable
- Add values to arguments
- Toggle flag
- Select choices
- Execute program
- Save config
- Load config
- Clear history
- Search executable
- Clear search

4 MVP complete

The complete MVP should contain these user stories below.

- Select executable
- Add values to arguments
- Toggle flag
- Select choices
- Execute executable
- Save config
- Load config
- Clear history

5 MVP minimal

None of this user stories can be taken out to avoid losing the point of MVP.

- Select executable
- Add values to arguments
- Execute Python programs

6 MVP real

The user stories below have the potential to be implemented realistically by the end of the project.

- Select executable
- Add argument value
- Toggle flag
- Select choices
- Execute program

7 KPI: Implementation of Core Features and Functionalities

7.1 KPI Definition

- **Description**: The KPI measures the completion rate of the core features and functionalities planned for the project.
- **Correlation**: There is a strong correlation between the number of implemented features and the project's progress. Each feature's completion represents a milestone towards achieving the overall project goal.
- **Mathematical Precision**: The KPI is calculated as the percentage of core features implemented out of the total planned features. For example, if 8 out of 10 planned features are fully implemented by the deadline, the KPI would be 80%.

7.2 KPI Goal

- **Target**: Achieve the implementation of 100% of the planned core features (such as "Execute Program," "Save Config," "Load Config," "Toggle Flag," etc.) by the end of the semester.
- **Significance**: Completing all core features ensures that the minimum viable product (MVP) is fully functional, enabling potential future iterations or enhancements.
- **Time-bound**: The goal is to achieve full implementation by the end of the semester, providing a clear deadline for measuring progress.