

1. Architecture and Design ..... 2

1.1 Architecture ..... 3

1.2 System Architecture Diagram ..... 4

1.3 Prototype Design and Interactive Diagram ..... 5

1.4 Design Notebook ..... 9

1.5 Operational Concept Documents ..... 12

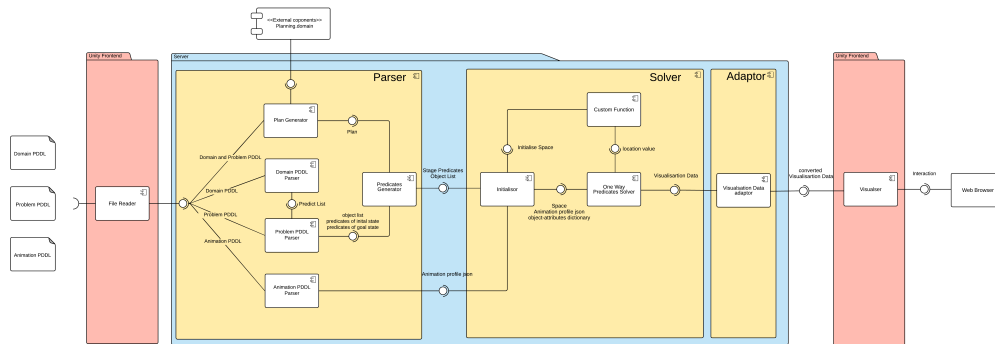
1.6 Information and Learning Resources ..... 14

# Architecture and Design

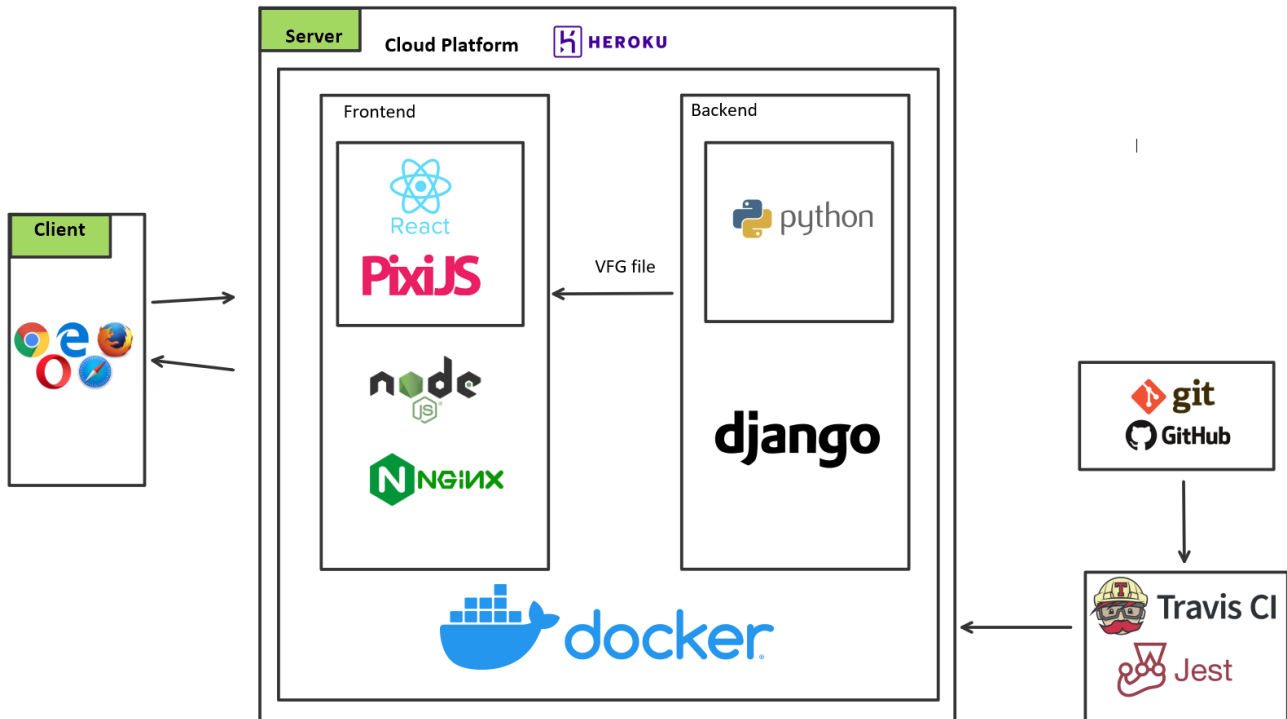
- [Architecture](#)
- [System Architecture Diagram](#)
- [Prototype Design and Interactive Diagram](#)
- [Design Notebook](#)
- [Operational Concept Documents](#)
- [Information and Learning Resources](#)

# Architecture

The architecture of the system is composed of a backend server and a frontend user interface. The backend is an Apache/Python Django and the frontend is a graphic interface built with Unity graphic engine for WebGL. The frontend has the task to provide user interface, pass PDDL script to the backend, display and control (interaction) visualisation and export files. Likewise the backend can parse PDDL scripts to solve the problem and produce a series of steps (or plan) in the format of a Visualisation File (VFG) that is later read by the frontend to generate the visualisation.



# System Architecture Diagram



The system consists of the frontend and the backend. Note that in our project, we only do the frontend development.

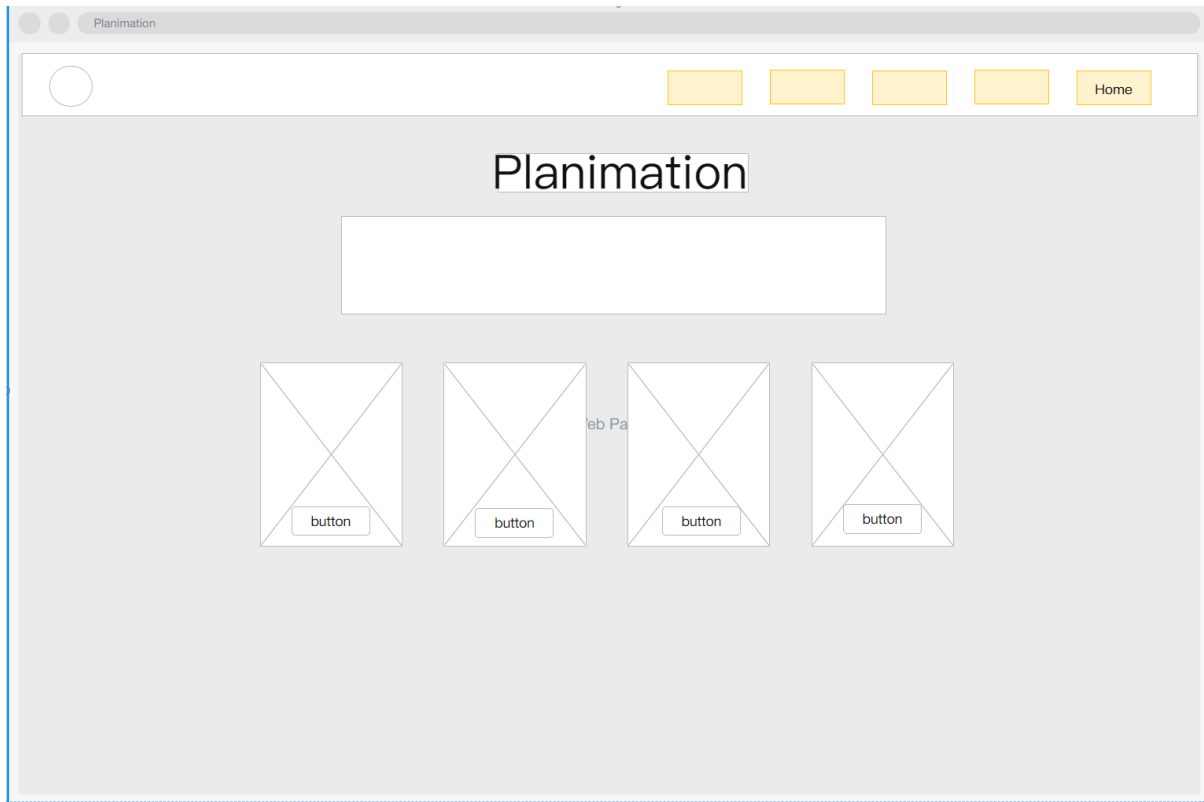
As the frontend framework is changed from Unity to JavaScript, the system architecture deployment related to the frontend has also been adjusted accordingly. In order to improve the frontend rendering performance on browsers, our frontend project plans to use React framework. The web server has also been changed from Apache to lightweight Nginx.

The planned system architecture is shown below. Since this project does not involve any changes to the backend and database, this page will only introduce the tools that will be used in this frontend project.

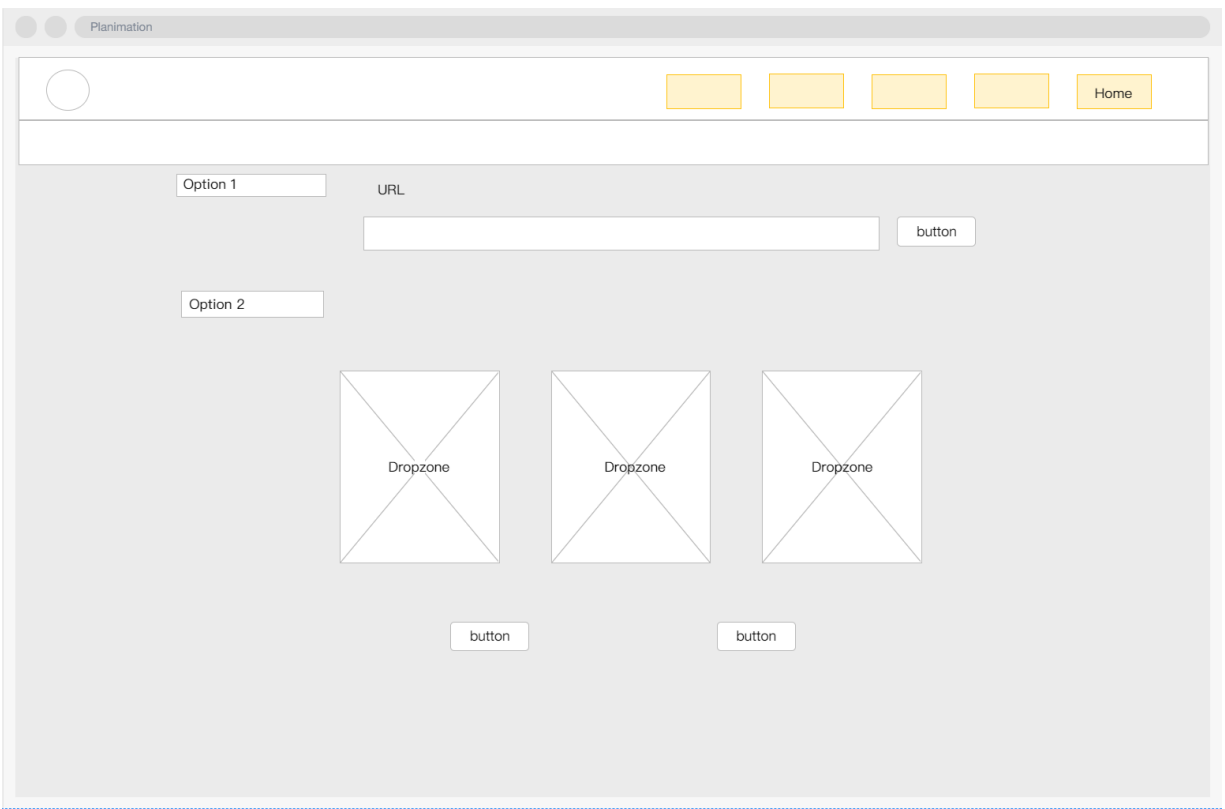
# Prototype Design and Interactive Diagram

- This paper prototype design just follows the old project which is based on unity, what we have decided is to re-arrange the elements on each page to make it looks better with JS frontend components.
- In the reason of the easier usage of components and design compared with unity. With react and material-UI, we could make it looks what we exactly want in a comfortable way.

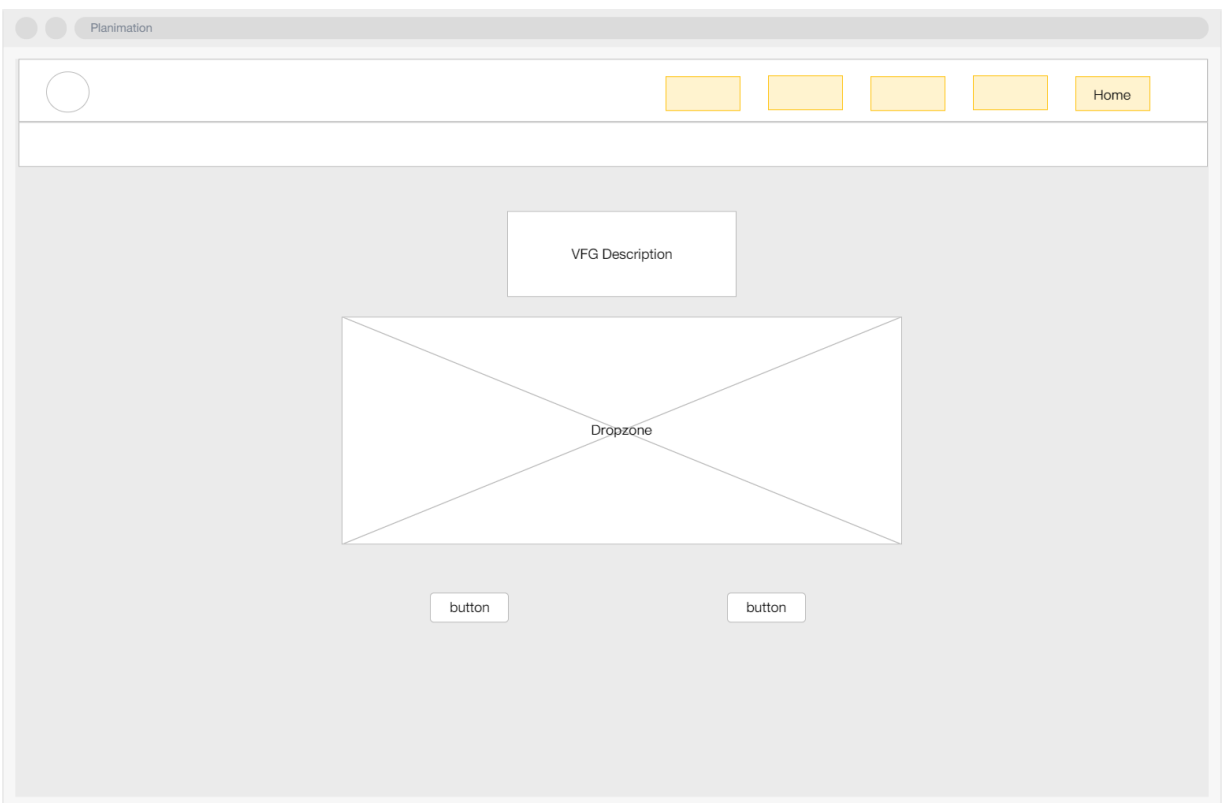
## HomePage as the cover page for the Application



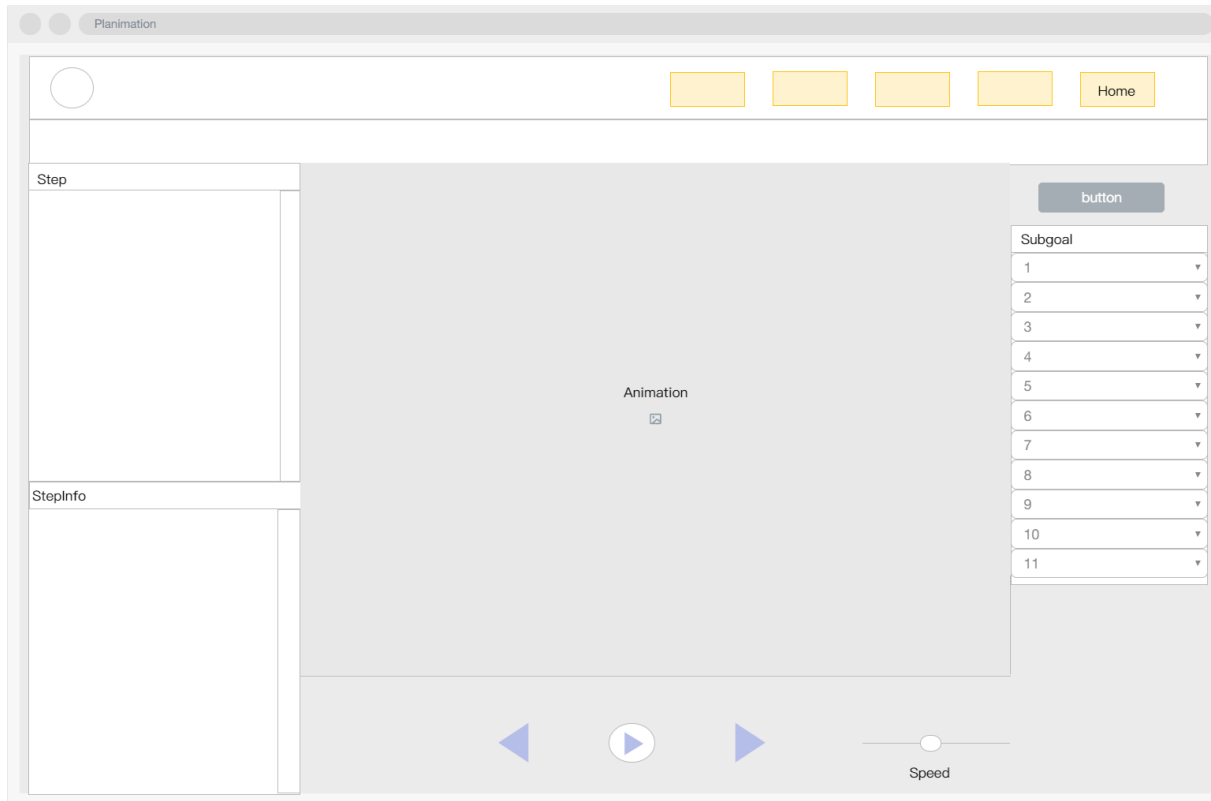
## Page for generating visualization from uploading PDDL files



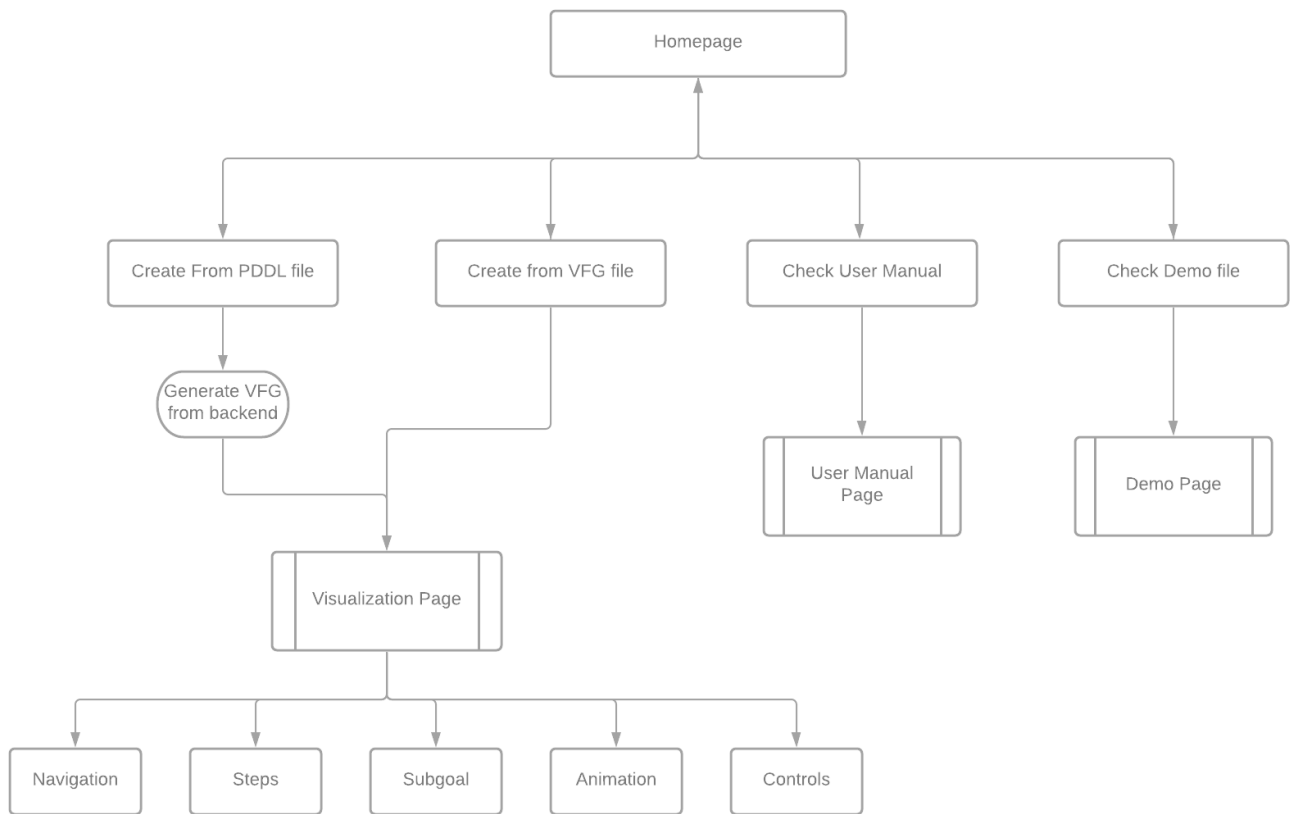
### Page for generating visualization from VFG file



## Page for visualization of the planning problem



## Interactive Diagram





# Design Notebook

## Design Notebook

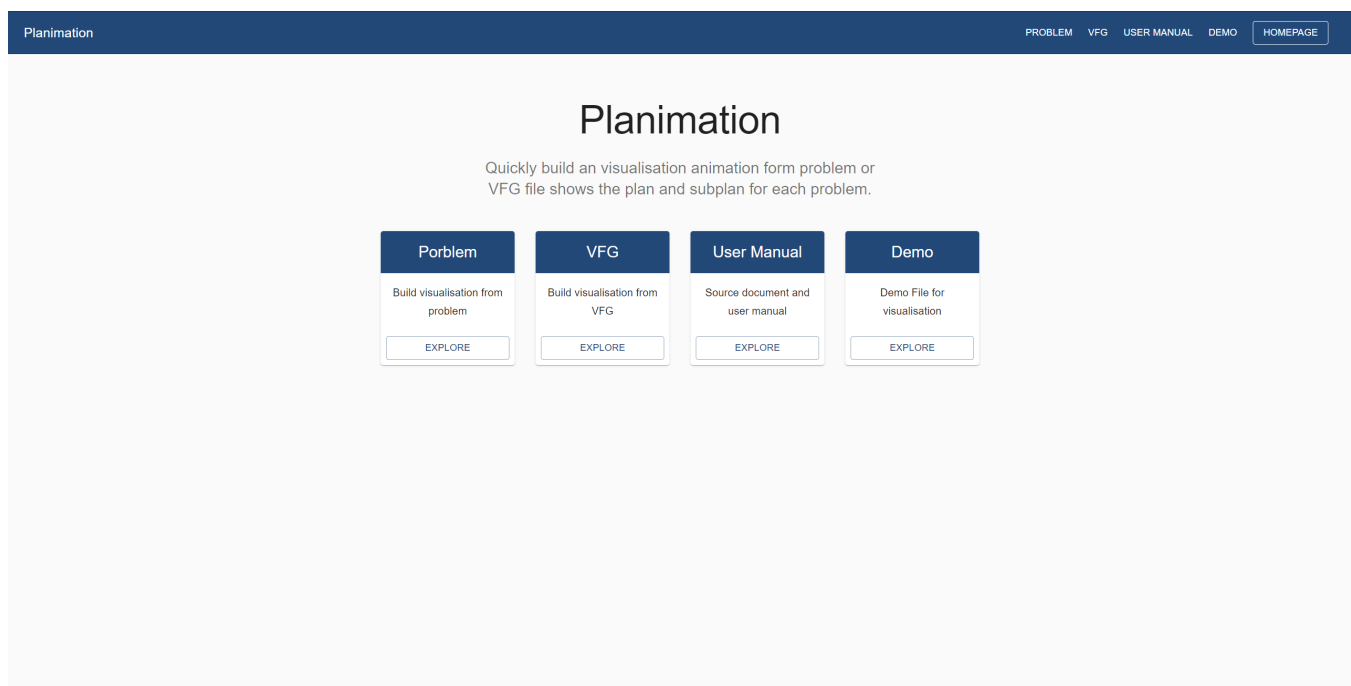
### User Interface

To fast set up and develop the functionality of the project, we needed to improve the visuals on the front-end. Hence, we had to decide on a proper front-end UI framework to work with. After a quick discussion on choices like React Bootstrap, Semantic UI, and Ant Design, we decided to go with Material UI. This is because of its compatibility with React as well as its offering of components is low-level enough to promote customization and extension. Other features such as dropzone support and file transfer function will be helpful for the better arrangement of page elements. Installation was straightforward, although a little work had to be done to get icons working correctly.

API design is hard because you can make it seem simple but it's actually deceptively complex, or make it actually simple but seem complex.

### HomePage Design with Material-UI

In coordinator homepage: Shows the four sections of the project.






### PDDL file upload Page

On this page, you need to upload 3 PDDL files to the PDDL editor server, and it will lead you to the visualization page with a generated vfg file.

Step 1 - Change planner URL (Optional)

PASTE

Step 2 - Upload Problem, Domain and Animation Profile Files

| Domain File<br>for predicates and actions   | Problem File<br>for objects, initial state and goal                               | Animation File<br>object representations  |
|---|---|---|
| Drag and drop domain.pddl   | Drag and drop problem.pddl  | Drag and drop animation.pddl  |
|  |  |  |

CANCEL


UPLOAD FILES

## VFG upload page

On this page, you need to upload a single vfg file for animation generating. It will lead you to the visualization page once your .vfg file is accepted.

Select VFG file to generate  
visualisation directly.

Drag and drop a file here or click



CANCEL

CONTINUE

## Visualization Page

This is the main functionality page for the project. It will show the animation generated from the VFG file on pages 1 and page 2. On this page, you could de the following operation.

Steps

0. Initial Stage

1. (unstack c e)

2. (stack c f)

3. (unstack e j)

4. (put-down e)

5. (unstack j b)

6. (stack j e)

7. (unstack b g)

8. (put-down b)

9. (unstack g h)

10. (put-down g)

11. (unstack h a)

Step Info

No Step Information

SHOW THE GOAL

EXPORT

Subgoal

0/9

(on c f) ✓

(on j e) ✓

(on h b) ✓

(on d c) ✓

(on g i) ✓

(on a g) ✓

(on b a) ✓

(on e h) ✓

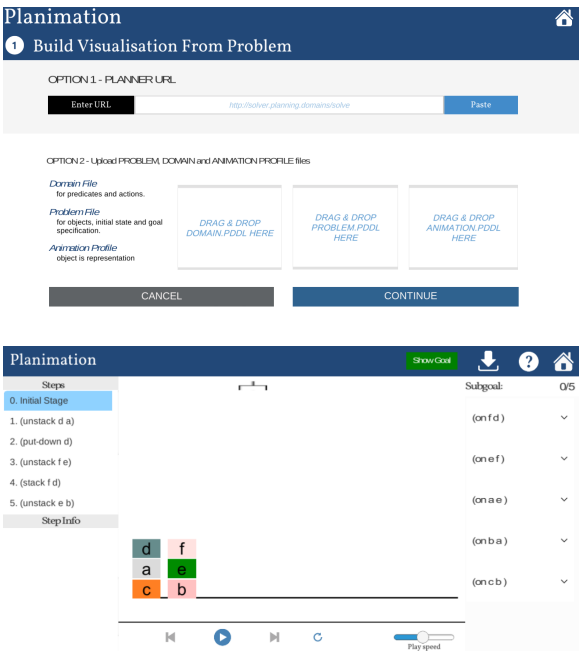
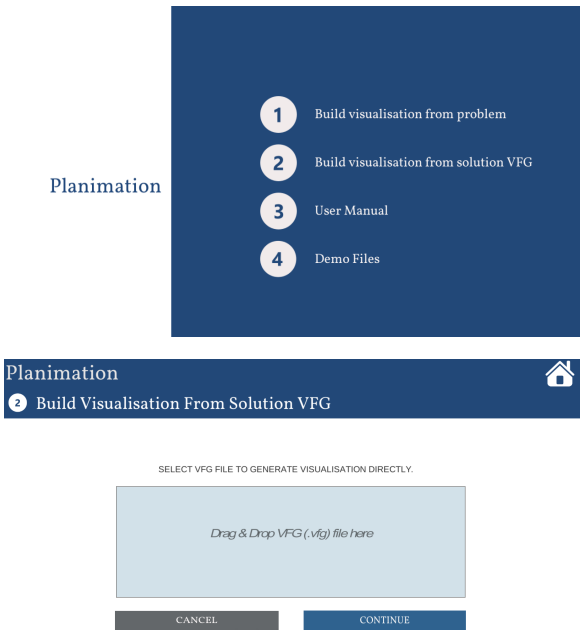
(on f j) ✓



# Operational Concept Documents

- Current layout
- Initial react component design

## Current layout



## Error.

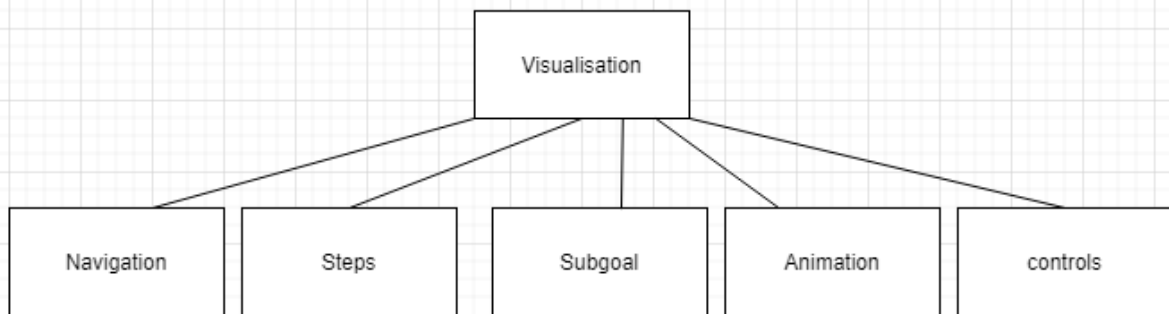
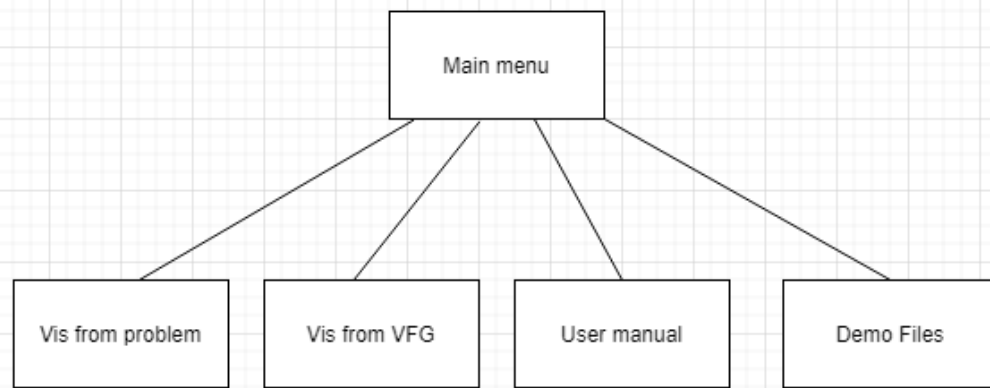
The process ends with an exception

Failed to get the plan/solution -- Suspected timeout.

ff: goal can be simplified to FALSE. No plan will solve it



## Initial react component design



# Information and Learning Resources

- **GitHub**

The project uses GitHub as a tool for project iteration, collaboration within team and version control. To increase the efficiency of developing, it's recommended to install Git on the development environment.

- **Visual Studio Code**

Visual Studio Code is a great code editor for many programming languages. It has Git commands built-in and many extensions to improve developer's productivity.

- **Travis CI+Jest**

Travis CI is a useful tool to test and can easily synchronize the repository on GitHub. It was also adopted by the original project Planimation. Here, we continue to use its service to continuous integration.

Jest is a test framework designed for JavaScript.

- **Nginx**

Nginx is a fast and dynamic web server. Compared to Apache, 4 times more concurrent connections are handled [Technical Environment#\[1\]](#). We use Nginx to replace Apache Web Server when deploying.

- **Docker**

The original project uses Docker to run the application on the cloud platform. Therefore, we continue to use it in this project, although several modifications need to be done during deployment.

- **Visual Studio Code API**

Use Visual Studio Code API to build a plugin(extension) for VS Code.

- **PixiJS**

PixiJS is a HTML5 Creation Engine which is suggested to be used in the project. To gain better performance, our project plans to use React+PixiJS.

- **React**

An open-source declarative frontend framework based on JavaScript. React encourages the developer to break the UI interface into components which makes code is easier to understand and maintain. React offers a better performance on rendering UI in the browsers, so it's very efficient for a frontend project requiring lots of interactions and animation

- **MaterialUI**

React components for faster and easier web development. Build your own design system, or start with Material Design. which provides a robust, customizable, and accessible library of foundational and advanced components, enabling you to build your own design system and develop React applications faster.

## References:

[1] <https://blog.coolicehost.com/ten-great-advantages-of-nginx/>

## Resources:

- Travis CI: <https://docs.travis-ci.com/>
- Jest: <https://jestjs.io/docs/getting-started>
- Docker: <https://docs.docker.com/>
- Visual Studio Code API: <https://code.visualstudio.com/api>
- Pixi JS: <https://www.pixijs.com/>
- Pixi JS Tutorial : <https://github.com/kittykatattack/learningPixi>
- Pixi JS Demos : <https://pixijs.io/examples/#/demos-basic/container.js>
- React: <https://reactjs.org/docs/getting-started.html>
- ReactPIXI: <https://reactpixi.org/>
- MaterialUI: <https://mui.com/>