Requirements

In this document

In child documents

1. In-scope

According to the requirements of our client, there are several requirements that need to be finished and in-scope for the product in this project.

- Integrate the Planimation Module as a VSCODE Plugin

  Planimation module should be integrated as a VSCODE plugin and comes preloaded with files to Planimation Blocks so that users could install the Planimation Module from the plugin menu of the online PDDL editor directly.

- PDDL editor Plugin

  Plugin for PDDL Editor to Launch Planimation application inside PDDL webpage.

- Load and Use the Planimation Module

  Users should be able to use the Planimation module in the online PDDL editor by installing the planimation plugin or they could access the stand-alone application directly by URL.

- Build Visualisation from Problem Files

  Users should be able to upload the planning problem related files, including the Domain PDDL file (for predicates and actions), the Problem PDDL file (for objects, initial state and goal) and the Animation profile (object is representation), to generate the visualisation of the plan (i.e. solution) of this problem.

- Build Visualisation from Solution VFG File

  Users should be able to upload the VFG file to generate visualisation directly.

- Planning Visualiser

  Users could access this planning visualiser after uploading the files and building the visualisation to observe the visualised sequential solutions of planning problems specified in PDDL step by step. Planning visualiser should have an animation player to display the animation of solutions, and users could press the ‘play/stop’ button, ‘next step’ button, ‘previous step’ button, ‘replay’ button and 'speed' bar to control the play of visualisation representation of planning problems’ sequential solutions. Planning visualiser should show the action and detailed step information of each step in a step panel. Users could also control the display of animation directly by selecting the specific step in the step panel. In addition, it should also show the sub-goal and final goal features of the planning problem.

- Export Animation Feature

  Users could export the animation of planning problem as a file and users could also select the format of the export file such as VFG file, MP4 file, GIF file and so on.

- User Manual

  Users could visit the user manual page to get a detailed guide and instructions for this planimation module.

- Demo Files

  The planimation module provides the domain file, problem file and animation profile in PDDL of several typical planning problems and users could get and use them directly.

2. Out-of-scope

There are some extensions and further requirements to enhance and improve this application, which is out-of-scope for this development stage.

- Modification of the Backend

  In this project, the main task is to develop a JavaScript frontend to substitute the existing Unity frontend using PixiJS. Hence, the team will mainly focus on the development of the frontend and will not modify the backend.

- Import Animation File in the Other Formats to Build Visualisation

  The team will not provide the functionality to upload the animation file in the other formats (like MP4 file, GIF file) than VFG file to generate the visualisation directly.

3. Constraints

This project is developed in the context of the capstone project in Software Project COMP90082 by a team of students of the Master in Information Technology at the University of Melbourne.

Constraint	Type	Description
Limited resources working on the project at any given time	Time constraint	It is expected for each member to work 20 hours weekly on the project.
Project delivery is bound to COMP90082	Time constraint	The project delivery date is not subject to change as it is mandated by COMP90082. The actual development time of this project is quite tight and just approximately two months from planning to the final product delivery. The development team needs to complete the development, test, and release the product in this very limited time, which might lead to imperfect testing and impact the robustness of the final product.
Project team to work using freely available technologies (Cost constraint)	Cost constraint	There is no budget set for the COMP90082 project, therefore all software functionality would have to utilize technologies that are available at no cost to the project team.
Fixed project team size	Cost constraint	This project is under the constraint that all teams are limited to five members. No additional resources can be hired to assist with the project deliverables.
A fixed set of requirements	Scope constraint	The requirements specified by the client are not negotiable, so changing the scope is not an option
Deployment environment requirements	Technical constraint	For deployment purposes, the development must be in docker containers and must be developed forking the current GitHub repository. Also, the main requirement of this project is that the development of a JavaScript frontend to substitute the existing Unity frontend, therefore, the team will mainly focus on the development of the frontend and continues to use the existing backend. The separate development of the frontend and backend might cause some challenges to the adaptation and integration of the frontend and backend.
Fixed working methodology is given by COMP90082	Organic constraint	Given by COMP90082 lecturers the SDLC will be Agile

Architectural and integration	Technical constraint	The backend and Online PDDL editor plugin cannot be change
Lack of development experience	Organic constraint	All members of the development team are students who might lack sufficient developing experience using the required developing tools (e.g. PixiJS and Django) although most of them are majoring in IT. The team may be unfamiliar with these tools at the beginning which might lead to low efficiency at the beginning stage.
Inconvenience communication among team members	Time constraint	Due to the restriction under the current global pandemic circumstance, it is difficult for the whole team to implement this project together geographically and communicate with each other in a face-to-face way frequently, which would affect the efficiency of communication and development work.

Background

In this document

1. Background

Planimation is an openSource framework to visualise sequential solutions of planning problems specified in PDDL. The framework was built by a team of University of Melbourne students in the context of the course SWEN90013 and under the lead of Professor Nir Lipovetzky and it has been continually supported by students and contributors. The project has important awards and recognitions.

The planimation goals are:

1. help to debug PDDL code for Online PDDL Editor users
2. increases user understanding of planning problems
3. showing planning solutions to non-technical audiences

The system uses a domain PDDL script, a problem PDDL script and a Animation profile script to produce visualisation.

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.

For a full image of the architecture refer to: Architecture

Key concepts:

Planning is a subtopic of the field of Artificial Intelligence, which is a model-based approach to autonomous behavior, based on finding an action sequence that produces desired state, a plan [1].

PDDL (Planning Domain Definition Language) is a computer language built to describe formal planning problems. In theory, a correctly described solvable problem can be solved efficiently by any solver. PDDL is the de-facto standard language for describing planning problems.[2]

---

2. Vision

Client:

The client is Dr. Nir Lipovetzky, Senior lecturer and researcher in Artificial Intelligence at University of Melbourne and Planimation curator (https://nirlipo.github.io/)

Users:

Planimation allows to user to:

1. Check errors in their PDDL code
2. Create general animation for every problem of an specific domain
3. Encode new domain animations

As identify by the client the main user profiles are:

1. Researchers in planning models
2. Students learning AI planning modelling
3. Industry partners looking for showcase solutions using planning modelling

The main goal is:

- Improve project efficiency, maintainability and extendability of Planimation software

The specific goals are:

- replace the current frontend programed in C# with the Unity engine, for a Javascript and PixiJS framework
- decrease the loading times produced by the Unity engine
- maintaining its current features
- integrate with current development tools (Online pddl Editor plugin)
- develop a new plugin for Visual Studio Code

High-level project scope:

Planimation frontend:

The main function of the frontend is to send information (PDDL scripts) to the backend, read VFG files, display and control visualisations and export them in multiple formats.

The Unity Engine is a cross-platform graphical framework that provides graphical outputs. Its main purpose is to design videogames. In the context of a planner visualiser the Unity is a heavy tool capable to manage many graphical models and shaders and translate them to webGL(http://benchung.com/unity3d-vs-three-js/). However, this procedure seems to be an overkill for the purpose of the visualiser, making it hard to maintain, change or integrate the code with other technologies. Also it generate certain inefficiencies as the application needs to run the engine first, process the files in C# and then translate it to JavaScript to run it with webGL.

Unity Engine Loading screen

On the other hand, PixiJS is a Javascript rendering openSource library that allow to create rich, interactive graphics, cross platform applications, and games without having to dive into the WebGL API or deal with browser and device compatibility (https://github.com/pixijs/pixijs). It focuses on efficiency, running complex graphics quite fast. It integrates seamlessly with HTML5 and other web technologies. Which seems to be more suitable for Planimation in the client's eyes, to achieve the project goals.

VSCode plugin

Visual Studio Code is an Integrated Development Environment (IDE) developed by Microsoft and one of the most popular developing tools [3] . The software integrate a large number of tools and can help to build code in numerous languages. It has the capacity to extender their functionalities by user coded plugins. The software has a an Extension API that allows to simplify the plugin creation.

Expected changes:

Stakeholder	Impact
Developing team	Gain experience in software development, good mark
Client	More control, maintainability and attractiviness
Supervisor	Accomplish course objectives
Users	Faster loading times, development from VSCode
Future contributors	Easy to extend and maintain the current code

3. Business Case

Business options

Alternatives include:

- Keep working in the current frontend, improving code transparency and documentation.
- Write frontend in plain JavaScript
- Write frontend with other JavaScript framework as threeJS

Expected benefits

Stakeholder	Financial	Non-financial
Developing team	Possibility to access better job positions	Gain experience, increase prestige as contributors
Client		Increase in Planimation prestige and increase in future contributors
Supervisor	Fulfill salary duties	Guide future software developers
Users	Save money building their own visualisation for their specific problems	Quickly develop projects visualisations
Future contributors		Easy to integrate, understand and manipulate Planimation

Timescale

The project needs to be finish by October 24. The development time is split in 2 sprints of 4 weeks each.

Cost

The main cost are the human resources and Infrastructure cost. No pay technology is budgeted.

People/hours

	Weekly hours	Number of weeks	Market reference	Total
5 Students	20	8	$38.46[4]	$30778
Supervisor	2	8	$64.10[5]	$1025
Total				$31803

Additional cost: power, hardware depreciation, internet.

Risk

Risk ID	Risk Type	Description	Probability (0-1)	Impact (1-5)	Justification
1	Project risk	Given the comunitary nature of the framework choose by the client, the framework may have poor support, or be abandoned in favor of other frameworks in future	0.1	3	Despite beign a stable framework with 5 major releases, open source proyect comunities could become impredictable or have quite uneven support
2	Business risk	Contributors cannot extend the code provided due to design problem or poor documentation	0.3	4	
3	Project risk	Design and implementation may be inefficient due to inexperience of the development team	0.2	5	It is a fact that team members have limited experience in software development and Javascript frameworks, which may lead to some errors or inefficient code/functionality. While the probability is relatively high, the impact is low because the learning curve for the proposed technologies is low.

4	Product risk	Proposed framework PixiJS cannot deliver the desired features	0.2	5	Choosing the wrong framework might end up in some important requirements not being delivered. This has a high impact because if the risk occurs, the team might have to change the framework which would cause a big delay and effort loss
5	Business risk	The backend or OPE plugin cannot integrate to the new frontend.	0.2	5	If the code does not integrate with the current project it maybe necessary to change the code of other parts, increasing complexity
6	Project risk	The team may experience delays in the original schedule due to unexpected issues	0.8	3	The development team has no experience in project management nor in Javascript development. So it is very likely that original estimations of the schedule will not be met due to unexpected technical issues and optimistic schedules

References:

1 Lipovetzky, N. (2020). Lecture 1. Short Introduction to Automated (AI) Planning [Lecture notes]. Retrieved from https://canvas.lms.unimelb.edu.au/courses/89089/pages/lecture-3?module_item_id=2215058

2. Lipovetzky, N. (2020). Lecture 3. Introduction to Planning How to Describe Arbitrary Search Problems.[Lecture notes]. Retrieved from https://canvas.lms.unimelb.edu.au/courses/89089/pages/lecture-3?module_item_id=2215058

3. https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-integrated-development-environment

4. https://au.talent.com/salary?job=junior+developer

5. https://au.talent.com/salary?job=Senior+Developer

Motivational model

1. DO/BE/FEEL TABLE

Version 2.0

Who	Do	Be	Feel	Concerns
Development team	▪ Excecute project	▪ Quality deliver	▪ Achievement	▪ Inexperience ▪ time/ cost constrains
Client	Sets goals: ▪ Improve Planimation maintanability ▪ Improve Planimation extendability ▪ Improve Planimation efficiency	Sets acceptance criterion: ▪ Involved between sprints (potencial change) ▪ Quality assesment ▪ The system need to be deployed within the current client infrastructure (Heroku)	▪ Proud ▪ Satisfied	▪ High expectation ▪ Change his mind (requirements)
Maintainer (User)	▪ Access information through Documentation ▪ Deploy changes	▪ Easy to understanding ▪ Clean code	▪ Intriged ▪ motivated, ▪ Understanding	▪ No interested ▪ Speding too much time in simple changes
Student (User)	▪ Load and use the Planimation module directly ▪ Use Planimation from VScode Plugin ▪ Use Planimation from PDDL Editor ▪ Build visualisation from Problem files ▪ Build visualisation from solution VFG file	▪ Increase undertading in PDDL ▪ Increase in Planning problems ▪ Understandable, accesible, conviniences (easy to use) ▪ Data must be displayed correctly and unambiguously ▪ will check their pddl code implementations for errors using the tool.	▪ Curious ▪ Understanding	▪ No interested
Industry Partner	▪ Same as previous user ▪ Export animation feature	▪ Find visual solution ▪ Comunicate industry solution to other formats ▪ minimum loading times while maintaining current performance.	▪ Productive ▪ Proud	▪ HIgh expectations
Researcher (User)	▪ Same as previous ▪ Change solver URL ▪ Change code	▪ Can try any new planning problem described in PDDL ▪ Can change code to fit new reseach problems ▪ the output should be consistent, same visualisation given an input data.	▪ Productive ▪ Motivated	▪ Inmpredictive research areas

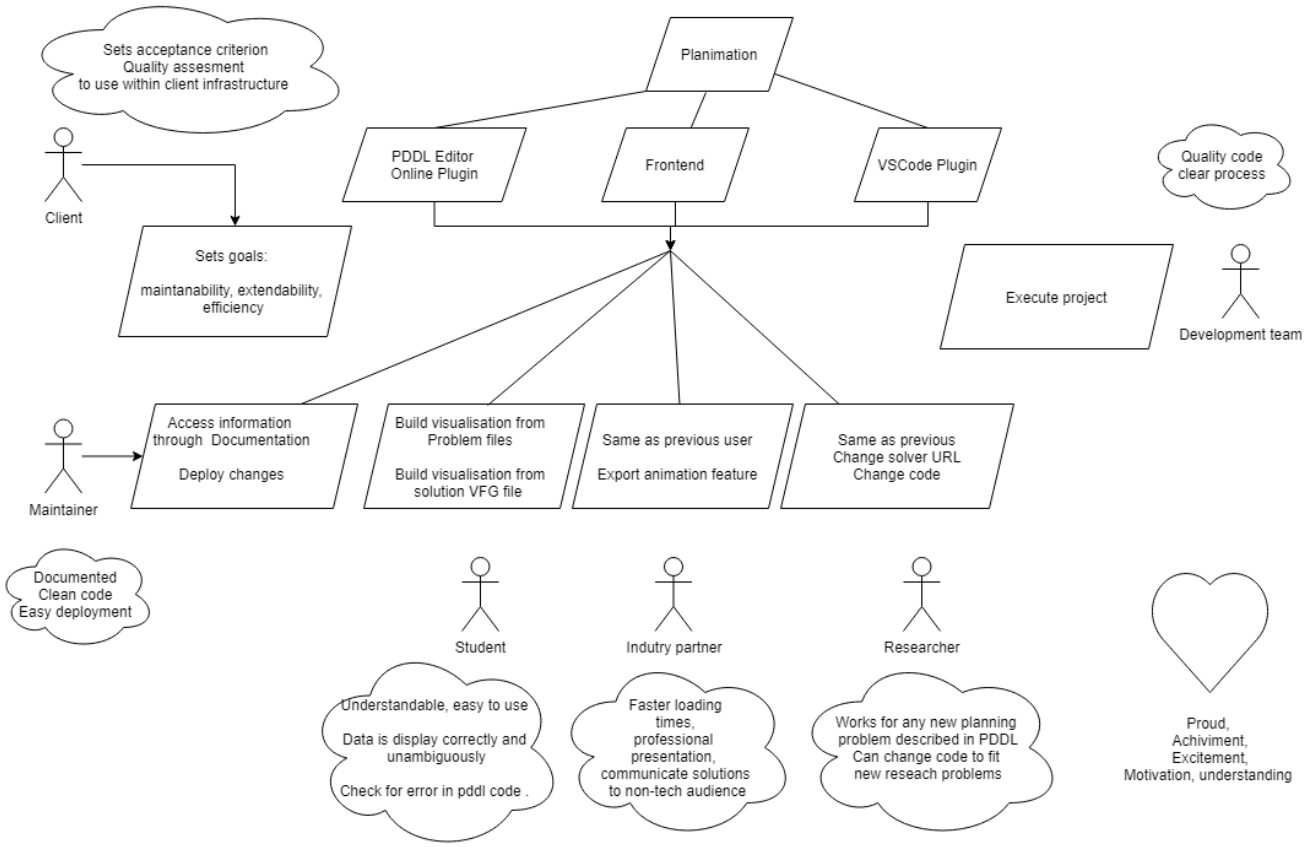
Version 1.0

Who	Do	Be	Feel	Concerns
Development team		Quality deliver	Achievement	Inexperienced
Client	Planimation Module Plugin, VS Code Plugin, Use PixiJS framework	Trackless	Proud	High expectations
Supervisor		Transparent process and quality of the delivered product		
Users	Load and use the Planimation module, Build visualisation from Problem files, Build visualisation from solution VFG file ,Export animation feature	Fast loading times for visualisation, professional presentation	Excited	

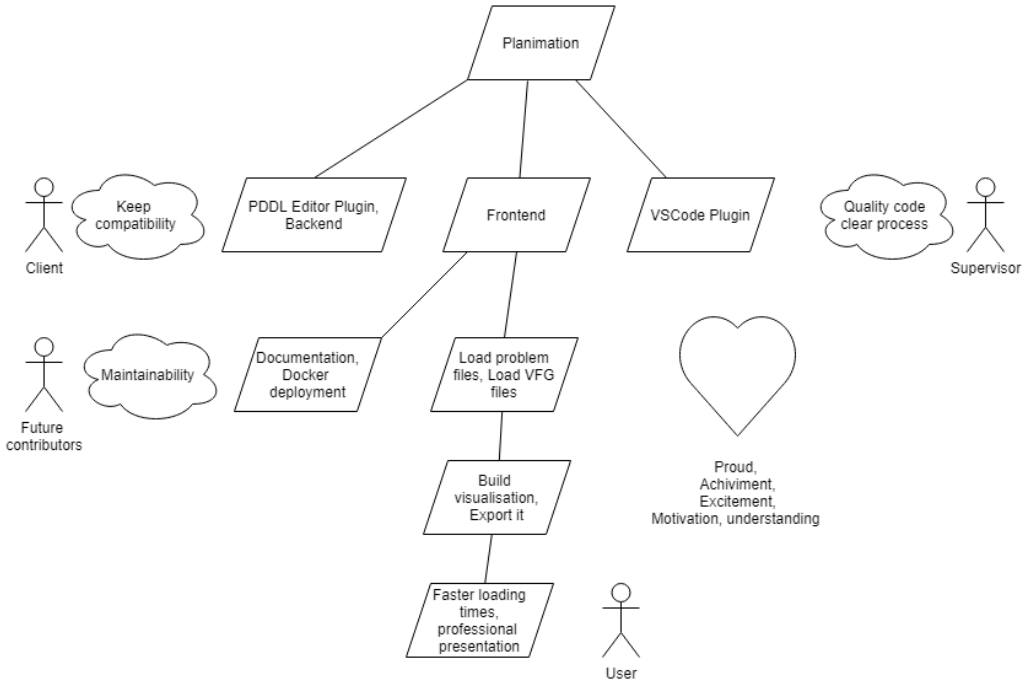
Furture developers	Documentation, Dockerised deployment	Interesing, easy read, inviting	Intriged, motivated, Understanding	

2. Model Chart

version 2.0



version 1.0


Planimation
Client
Keep compatibility
PDDL Editor Plugin, Backend
Frontend
VSCode Plugin
Quality code clear process
Supervisor
Future contributors
Maintainability
Documentation, Docker deployment
Load problem files, Load VFG files
Proud, Achivment, Excitement, Motivation, understanding
Build visualisation, Export it
Faster loading times, professional presentation
User

Users Personas

Users:

Planimation allows to user to:

1. Check errors in their PDDL code
2. Create general animation for every problem of an specific domain
3. Encode new domain animations

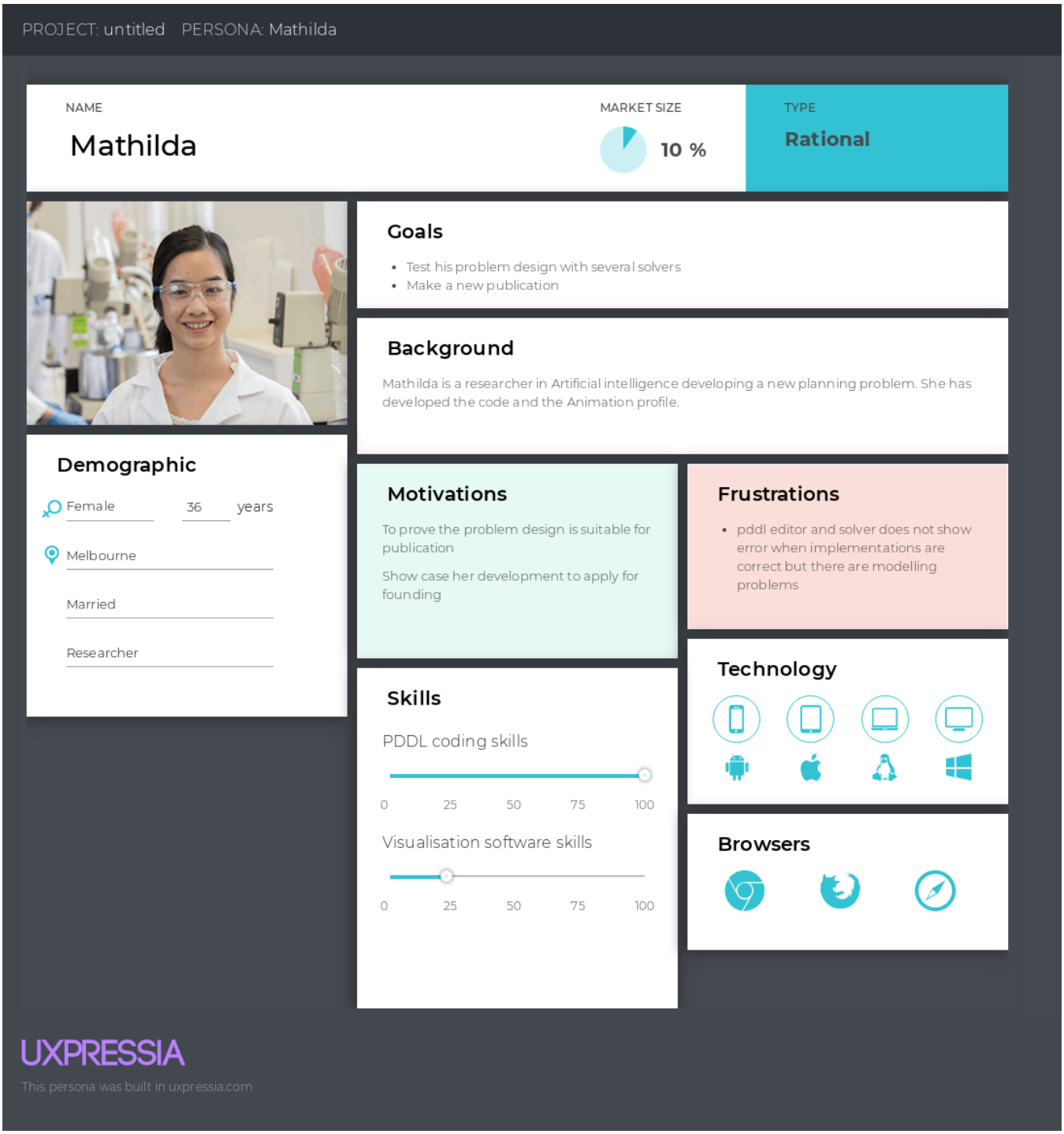
As identify by the client the main user profiles are:

1. Researchers in planning models
2. Students learning AI planning modelling
3. Industry partners looking for showcase solutions using planning modelling

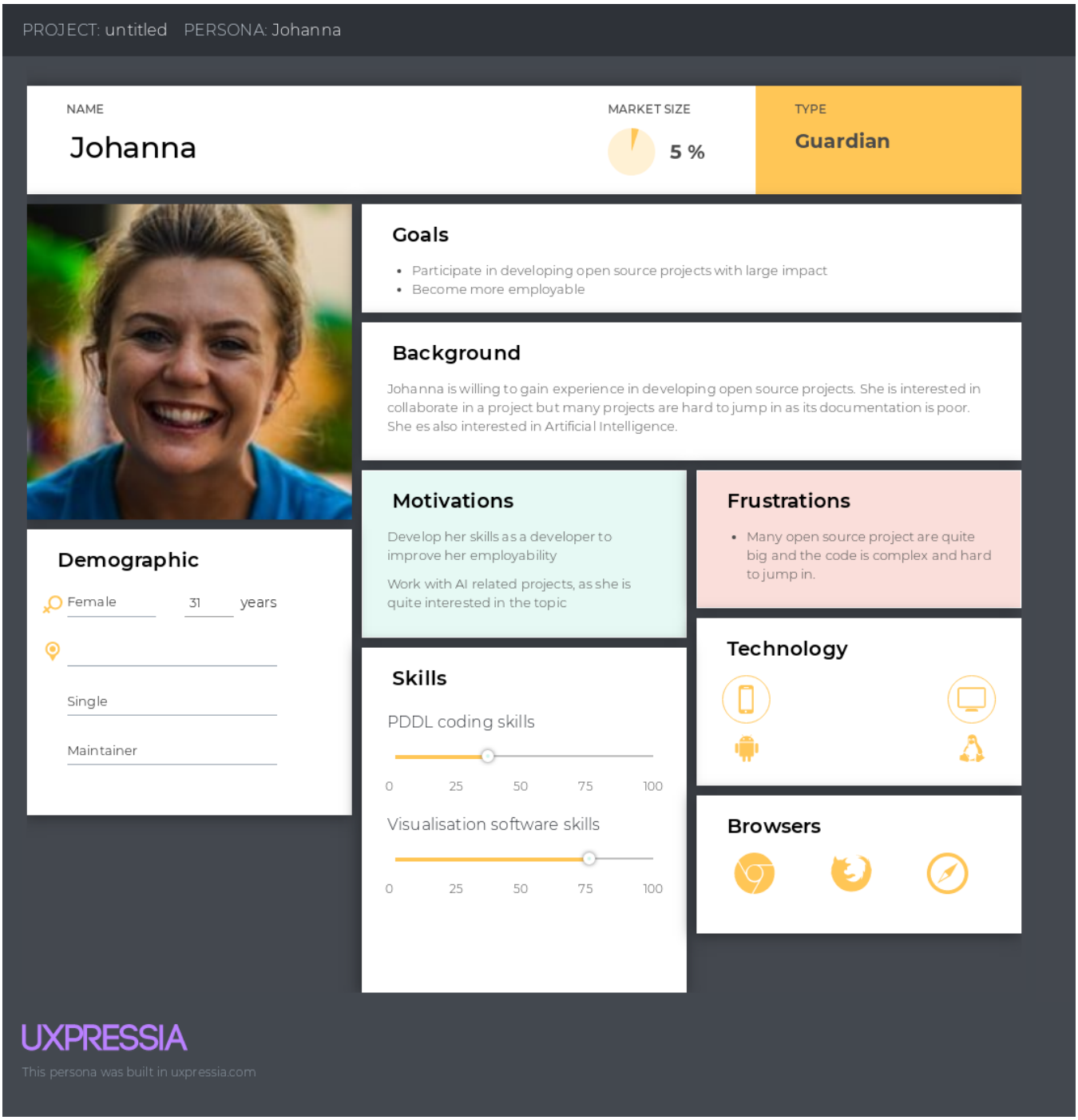
Type	Bio	Goals	Problem/ frustration
Student	John is a Master Student from Information Technologies coursing COMP90054 Artificial Intelligence Planning for Autonomy course. He has a special interest for the subject and he want to take the mot of it.	• Learn to code in pddl • Check if his implementation are correct • Show to his friend and family what is he learning	• pddl editor and solver does not show error when implementations are correct but there are modelling problems
Researcher	Mathilda is a researcher in Artificial intelligence developing a new planning problem. She has developed the code and the Animation profile.	• Test his problem design with several solvers • Make a new publication	• pddl editor and solver does not show error when implementations are correct but there are modelling problems
Maintainer	Johanna is willing to gain experience in developing open source projects. She is interested in collaborate in a project but many projects are hard to jump in as its documentation is poor. She es also interested in Artificial Intelligence.	• Participate in developing open source projects with large impact • Become more employable	• Many open source project are quite big and the code is complex and hard to jump in.
Industry partner	Mickey works in the port industry. He preparing a project to use AI to automaticly move the load out of large cargo ships. He needs to showcase how AI planning algorithms can solve this problem to investers and his company bosses.	• To show to a non-technical audience how a planning algorithm can solve the industry problem	• Does not have skills or software knowledge to make visualisation

1.Student

2.Researcher



3.Maintainer



4.Industry partner

PROJECT: untitled PERSONA: Mickey

NAME

Mickey

MARKET SIZE

15 %

TYPE

Idealist

Goals

- To show to a non-technical audience how a planning algorithm can solve the industry problem

Background

Mickey works in the port industry. He preparing a project to use AI to automaticly move the load out of large cargo ships. He needs to showcase how AI planning algorithms can solve this problem to investers and his company bosses.

Demographic

Male 35 years

Port Melbourne

Divorced

Industry partner

Motivations

He need to justify the project he has benn working for a couple of months now. He needs to make a couple of presentation to motivate its team, encourage his boss and capturate some invesment

Frustrations

- Does not have skills or software knowledge to make visualisation

Technology

Skills

PDDL coding skills

0 25 50 75 100

Visualisation software skills

0 25 50 75 100

Browsers

UXPRESSIA

This persona was built in uxpressia.com

Functional requirements

Requirement	Breakdown requirements
1. Load and Use the Planimation Module (from plugins or stand alone app)	Build a main interface for access to four sub-modules (including generating the visualisation from problem files, generating the visualisation from VFG file, accessing the user manual and accessing the demo).
2. Build Visualisation from Problem Files (pddl)	Upload domain, problem, and animation PDDL files for generating the visualisation of the plan (i.e. solution) of this planning problem.
3. Build Visualisation from Solution VFG File	Upload a VFG file for generating the visualisation directly.
	View the animation of the visualisation of a particular planning problem on the visualizer page after uploading the files.
4. Planning Visualiser with PixiJS	Show each step of a plan, the status of any step in the animation by selecting a particular step, and the detailed step information of the selected step on the visualizer page.
	Show subgoals of each step and all the steps corresponding to a certain subgoal.
	Show the visualization of the final goal state button
	Show visualization status of the previous or next step buttons
	Build control the display of the animation, including play, pause, and reset.
	Build a display speed of the animation.
5. Export Animation Feature	Export the animation file in VFG file, MP4 file or GIF
6. Keep compatibility with PDDL editor Plugin	Load this planimation platform from the PDDL online editor as a plugin
7. Integrate the Planimation Module as a VSCODE Plugin	Load Planimation from Visual Studio Code
8. Demo Files should be updated	Demo video or doc demonstration to learn how to operate planimation
9. User Manual should be updated	User manual to help users to operate planimation.
10. Documentation for other code contributors	Write documentation for advance users or project maintainer

Non-functional requirements

1. Quality of Service (QoS)

Aspect	subaspect	Description
1.1 Safety	-	Planimation data must be displayed correctly and unambiguously, as the user will check their pddl code implementations for errors using the tool.
1.2 Security	-	Although no personal information is shared, the system need to prevent malicious use of the platform by using basic security good practices in deployment.
1.3 Reliability	-	The frontend should be consistent, the output the same data and visualisation given an input data.
1.4 Performance	**1.4.1 Time**	The frontend should get rid of loading times produced by Unity engine, while maintaining current performance.
	1.4.2 Cost	The system need to be deployed within the current client infrastructure (Heroku)
	1.4.3 Throughput	Maintain the current standards of communication with the backend.
1.5 Interface	**1.5.1 User Interaction**	Mantain and inprove the current Understandable, accesible, conviniences (easy to use) of the frontend.
	1.5.2 Device Interaction	Work in different screen sizes
	1.5.3 Software/Service Interoperability	Maintain connection with current PDDL Editor plugin

2. Compliance Requirements (Standards)

Planimation prescribes software effects on the environment to conform to national laws, international regulations, social norms, cultural or political constraints, and standards.

- The activities and recommendations provided by the system should comply to human safety limits.

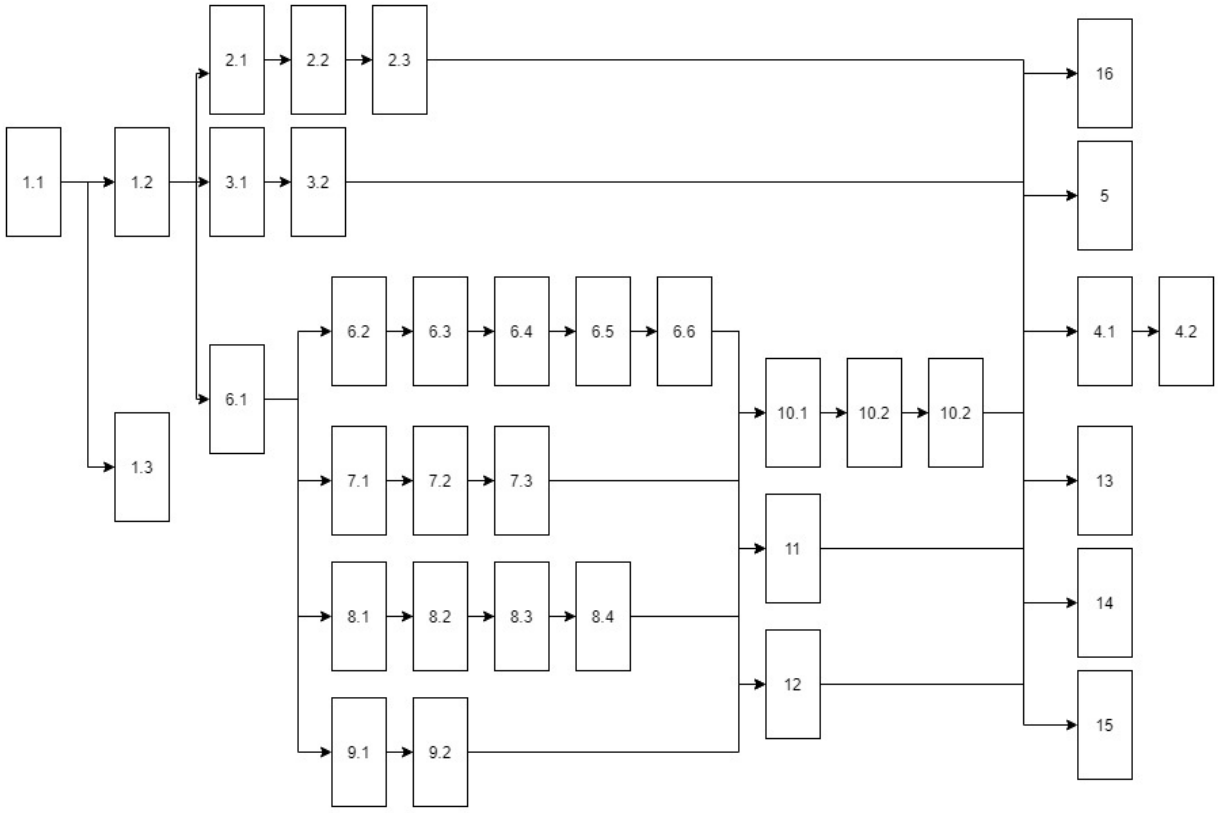
3. Development Constraint (Process)

Acpect	sub-aspect	Description
4.1 Deadline	-	The proyect is contrained within the context of COMP90018 subject. The end of the project need to meet the the final delivery by the 24th October.
4.2 Variability	-	Features can change during the deployment phrase. As the client wants to teams to work in parallel. The visualisation should be able to represent custom Animation profiles as long it follows this regulations: https://planimation.github.io/documentation/ap_guide/
4.3 Maintainability	**4.3.1 Changeability**	This is not only a process requirement but directed related to the main goal of the project. The source code should have enough documentation to facilitate ist understanding.
	4.3.2 Analysability	The code need to ensure to use best practices in software development to invite new developers to easily customize or contribute to the project. Documentation, architecture design, comments and variable names needs to be clear and easy to read.

References

Van Lamsweerde, A. (2019). Axel Van Lamsweerde for Non-Functional Requirements on Software Services. Retrieved from https://www.researchgate.net/figure/Axel-Van-Lamsweerde-for-Non-Functional-Requirements-on-Software-Services-adopted-from_fig3_268290531

Task dependency diagram



Story ID	User Story	Breakdown Tasks	Components
1	As a user, I could access the main interface for access to four sub-modules (including generating the visualisation from problem files, generating the visualisation from VFG file, accessing the user manual and accessing the demo).	1.1 Build the whole framework for the application	UI
		1.2 Design and build the interface page	UI
		1.3 Build a navigation bar applying to the entire application	UI
2	As a user, I could upload domain, problem, and animation PDDL files for generating the visualisation of the plan (i.e. solution) of this planning problem.	2.1 Build the interface page including the drag&drop zones for uploading problem files	UI
		2.2 Upload the problem files including the domain file, problem file and animation file by dragging files into the specific ‘drag and drop' zones	visFromProblem
		2.3 Connect problem files and fetch them to the backend and receive the corresponding VFG file	visFromProblem
3	As a user, I could upload a VFG file for generating the visualisation directly.	3.1 Build the interface page including the drag&drop zone for uploading the VFG file	visFromVFG
		3.2 Upload the VFG file by dragging the VFG file into the ‘drag and drop’ zone	visFromVFG
4	As a user, I could find a demo video or doc demonstration to learn how to operate this animation.	4.1 Create a demo page(or Github page) and connect it with other pages	demo page
		4.2 Show the demo video and doc demonstration to help users learn how to operate this animation.	demo page
5	As a user, I could find a user manual to help me use this web-based application.	5. Create a user manual page and connect it with other pages	manual page
6	As a user, I could view the animation of the visualisation of a particular planning problem on the visualizer page after uploading the files.	6.1 Build the interface page for the visualizer page	UI
		6.2 Parses the VFG file and convert the content of the VFG file to a JSON object	visScreen

		6.4 Get the rendered objects from the JSON object and visualize them on the screen	visScreen
		6.5 Extract the position data of all rendered objects from the JSON object	visScreen
		6.6 Realize the animation of the motion of the rendered objects	visScreen
7	As a user, I could check each step of the plan, the status of any step in the animation by selecting a particular step, and the detailed step information of the selected step on the visualizer page.	7.1 Build a step panel to show all the steps of the plan	visSteps
		7.2 Select a step in the step panel and then display the visualization of this step	visSteps
		7.3 Build a step information panel to show the detailed information of the selected step	visSteps
8	As a user, I could check the subgoals of each step and all the steps corresponding to a certain subgoal.	8.1 Show all the subgoals of the plan	visSubGoals
		8.2 Highlight all subgoals in the corresponding step status	visSubGoals
		8.3 Click a subgoal, then list all the steps that achieve this subgoal	visSubGoals
		8.4 Click a step in the step list of the subgoal and then display the visualization of this step	visSubGoals
9	As a user, I could view the visualization of the final goal state.	9.1 Add the 'show the goal' button	visSubGoals
		9.2 Show the visualization of the final goal	visSubGoals
10	As a user, I could check the visualization status of the previous or next step.	10.1 Add the 'previous step' button and the 'next step' button	visControl
		10.2 Click the 'previous step' button and show the visualization of the previous step	visControl
		10.3 Click the 'next step' button and show the visualization of the next step	visControl
11	As a user, I could control the display of the animation, including play, pause, and reset.	11. Create component that could control the play of animation including play and pause	visControl
12	As a user, I could control the display speed of the animation.	12. Create component that changes tick speed	visControl
13	As a user, I could export the animation file.	13. The animation file could be saved as a video file like mp4 etc.	visScreen
14	As a user, I could load this planimation platform from the PDDL online editor as a plugin	14. Ensure and test that current plugin is still functional	
15	As a user, I could load Planimation from Visual Studio Code	15. Create a plugin for VSCODE to show planning animations	VSC plugin
16	As an advance user or project maintainer, I want to quickly understand the code to make updates or modifications	16. Create or update documentation	Documentation

User cases

Revision History

Date	Version	Description	Author
19 Aug 2021	01.00	Initial draft	Felipe Ramos
22 Aug 2021	01.10		Felipe Ramos
23 Oct 2021	01.20	Add diagram	Xiaoyu Zhang

Contents

1. Introduction

1.1 Proposal

This document specifies the COMP90018 project Planimation for team Boxjelly (ex Visual Heuristics) use cases, describing the flow of events, inputs and outputs of each use case to be implemented. It describes the cases in formal sense, ignoring the specifics of the type of planning problems the users could pass to the system, such does problems are vitually an infinite number of cases. However, it is expected that this user cases can work for all of them.

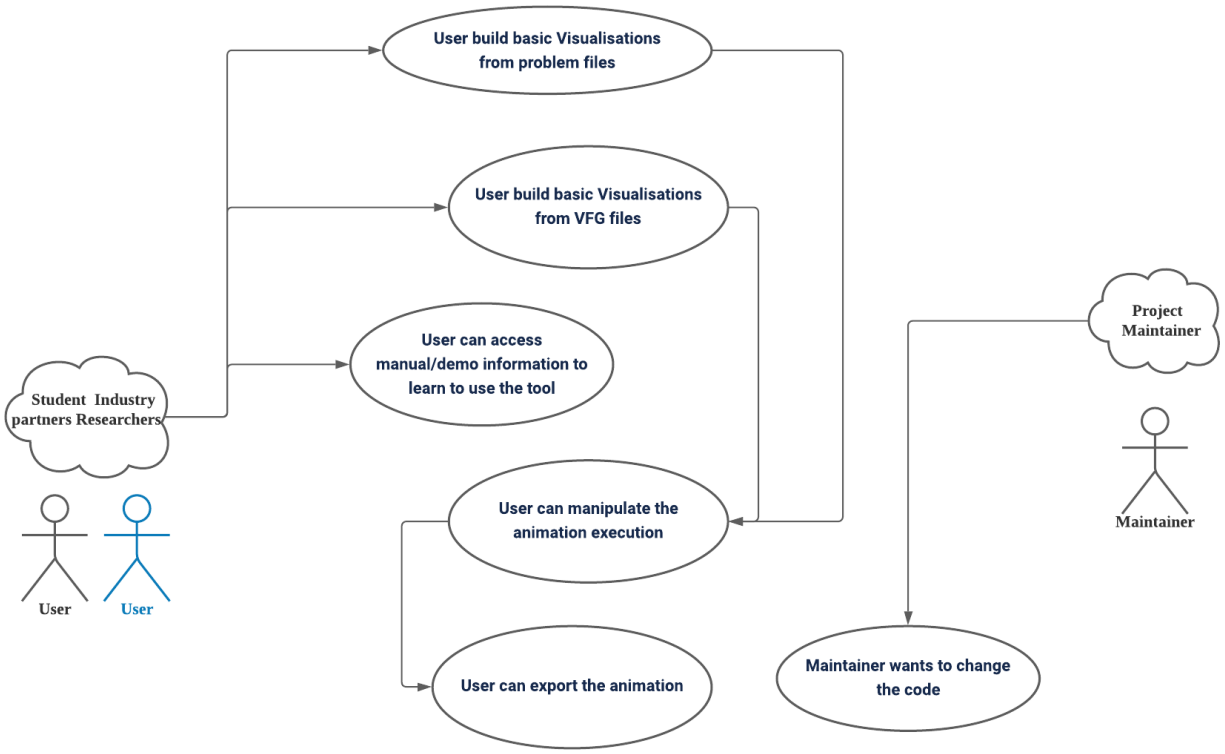
2. Actors

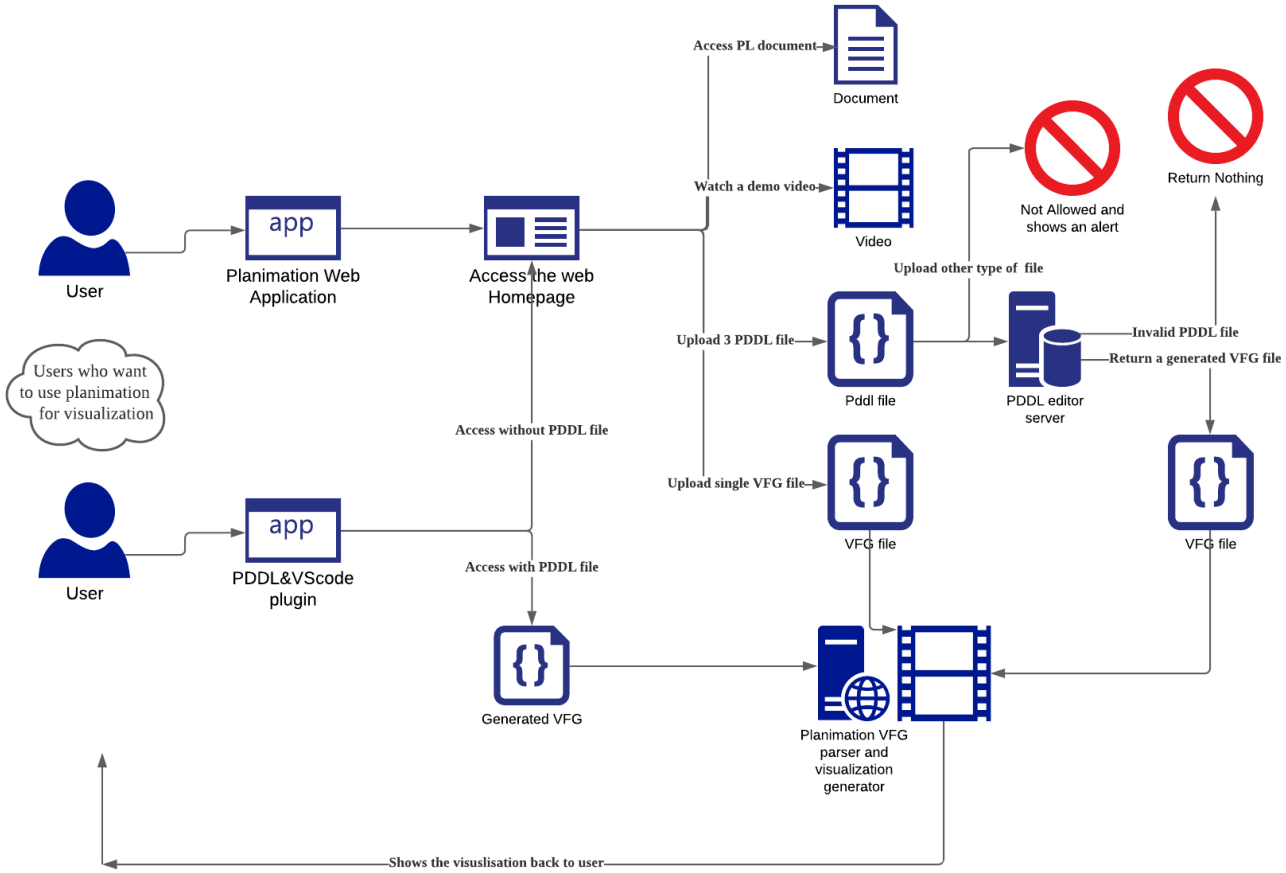
The project needs to provide to two types of actors: The planimation users and code maintainers.

Actor	Type	Description
Researchers in planning models	User	Researchers looking to describe new palnning problem or test solvers. They also may want to modify the current code to fit to their necessities or build their own Animation profiles.
AI planning modelling Students	User	They seek to learn to describe planning problems through pddl, look for errors in their implementations and gain better understanding of the solutions built.
Industry partners	User	Looking for showcase solutions using planning modelling. They may require to present their planning solutions in a graphical manner to audicences that are non-familiar artificial intelligence planning
Planimation Contributor	Maintainer	Any of the previous categories or someone else may want to contribute to the source code to expanded or maintain it
Planimation administrator	Maintainer/client	Wish to keep it running, maintain and expand planimation functionality.

3. Use Cases

3.0.1 User cases Diagram


User build basic Visualisations from problem files
User build basic Visualisations from VFG files
User can access manual/demo information to learn to use the tool
User can manipulate the animation execution
User can export the animation
Maintainer wants to change the code
Student Industry partners Researchers
User
User
Project Maintainer
Maintainer



3.1 User build basic Visualisations from problem files

[UC001]

Pre-conditions

The user has a set of pddl code files corresponding to a domain, a problem and an animation profile.

Main Events Flow

1. The user enter the main menu
2. The user click on "visualisation from problem in pddl"
3. The user drag and drop respective files in the corresponding blocks.
4. The system validates the fields – [AF02] [AF04]
5. The system displays the name of the files in the corresponding blocks
6. The system loads and format the data ready to be sent to the backend.
7. The user clicks on upload button and the system send the data to tha backend.
8. The system recibes a VFG files and pass it to the visualisation panel

Alternative Flows

[AF01] Cancel the operation

1. The user wants to cancel the operation.
2. The execution flow goes back to step 1. Any files loaded to the platform are discarted.

[AF02] One of the files are missing

1. The system displays a message informing that one or more files are missing

[AF03] Invalid file format

1. The system identifies a file with a extension different to ".pddl" and displays a message informing it

[AF04] Invalid content type

1. The system passes the file to the backend. The backend detects problems with the content and return an error code. The system display an error page and passes to the screen the error message from the backend.

Exception Flow

[EF01] Unknown Exception

1. The system logs the runtime exception that has happened during the application execution however not recognized

[EF02] No solution Exception

1. The system logs the exception that has happened during the application execution. Go to error page.

[EF03] No connection Exception

1. The system logs the exception that has happened during the application execution. Go to error page.

Post-conditions

The system loads the visualisation and present it to the user.

3.2 User build basic Visualisations from VFG files

[UC002]

Pre-conditions

The user has a VFG files compatible with Planimation.

Main Events Flow

1. The user enter the main menu
2. The user click on "visualisation from VGF file"
3. The user drag and drop respective file in the load block.
4. The system validates the fields – [AF02] [AF03]
5. The system displays the name of the file in the corresponding blocks
6. The user clicks on upload button and the system pass it to the visualisation panel

Alternative Flows

[AF01] Cancel the operation

1. The user wants to cancel the operation.
2. The execution flow goes back to step 1. Any files loaded to the platform are discarted.

[AF02] The files ar missing

1. User click the upload button without uploading a file. The system displays a message informing that the files is missing

[AF03] Invalid file format

1. The system identifies a file with a extension different to ".vfg" and displays a message informing it

Exception Flow

[EF01] Unknown Exception

1. The system logs the runtime exception that has happened during the application execution however not recognized

[EF02] Wrong content Exception

1. The system logs the exception that has happened during the application execution. Go to error page.

Post-conditions

The system loads the visualisation and present it to the user.

3.3 User can access manual/demo information to learn to use the tool

[UC003]

Pre-conditions

The user has access Planimation webpage.

Main Events Flow

1. The user enter the main menu
2. The user click on "manual" or "demo" button
3. The system displays the corresponding page with information to operate the system

Exception Flow

[EF01] Unknown Exception

1. The system logs the runtime exception that has happened during the application execution however not recognized

[EF02] No server found Exception

1. The system logs the exception that has happened during the application execution. Go to error page.

Post-conditions

The system loads the information to the user.

3.4 User can manipulate the animation execution

[UC004]

Pre-conditions

The system has loaded a correct visualisation from a VGF file.

Main Events Flow

1. The sytem load a visualisation from a VFG file
2. The user click on "play" button, "next/previous" step button, selects a step from the steps component, select a subgoal from the subgoal component or click the "show goal" button.
3. The system displays transitions from objects sprites.

Alternative Flows

[AF01] Modify the reproduction speed

1. User click the speed slider button. The system change the transition speed accodingly.

[AF02] First and last step

1. When the first or last step is reached, the next or previous button becomes desactivated accordingly.

Exception Flow

[EF01] Unknown Exception

1. The system logs the runtime exception that has happened during the application execution however not recognized

Post-conditions

The system generate transitions as requested by the user.

3.5 User can export the animation

[UC005]

Pre-conditions

The system has loaded a correct visualisation from a VGF file.

Main Events Flow

1. The sytem load a visualisation from a VFG file
2. The user click on "export animation" button and select a format option.
3. The system transform the animation into the desired format and send it to the user's machine.

Alternative Flows

[AF01] Cancel the operation

1. The user wants to cancel the operation.
2. The execution flow goes back to step 1. Any files are discarted.

[AF02] Long loading time

1. If a particular transformation excedes 200 miliseconds, the system display a loading bar to communicate to the user the systems is still working

Exception Flow

[EF01] Unknown Exception

1. The system logs the runtime exception that has happened during the application execution however not recognized

[EF02] Animation cannot be processed Exception

1. The system logs the exception that has happened during the application execution. Go to error page and cancel any download

Post-conditions

A animation file is downloaded by the user.

3.6 Maintainer wants to change the code

[UC006]

Pre-conditions

The maintainer want to change the source code.

Main Events Flow

1. The user can access the documentation page
2. The maintainer can access the specific information of the section that needs to be changed

Alternative Flows

[AF01] An specific information is not in the documentation

1. The maintainer can navigate through the folders and read the comments and code directly.

Post-conditions

A maintainer can easily implement the change.

User stories

version 2.1

Story ID	User Story	Acceptance Criteria	Breakdown Tasks	Components	Priority	User /function points
1	As a user, I could access the main interface for access to four sub-modules (including generating the visualisation from problem files, generating the visualisation from VFG file, accessing the user manual and accessing the demo).	• When the user enters the main interface, the user could easily find the buttons to the four sub-pages (including generating the visualisation from problem files, generating the visualisation from VFG file, accessing the user manual and accessing the demo) on the page. • When the user clicks the corresponding button, he can successfully jump to the corresponding page.	1.1 Build the whole framework for the application	UI	Must have	5
			1.2 Design and build the interface page	UI	Must have	3
			1.3 Build a navigation bar applying to the entire application	UI	Must have	2
2	As a user, I could upload domain, problem, and animation PDDL files for generating the visualisation of the plan (i.e. solution) of this planning problem.	• When the user enters the page for generating visualization from PDDL problem files, the user could see three ‘drag and drop’ zones for uploading the required domain, problem, and animation files, each uploading zone has a corresponding description of the file to be uploaded. • User could drag the PDDL files to the corresponding zones, if the files are not a .ppdl file, the system will give an alert message. • User could click ‘submit‘ button to send the uploaded problem files and jump to the visualization page successfully if the upload files could get a planning problem solution.	2.1 Build the interface page including the drag&drop zones for uploading problem files	UI	Must have	1
			2.2 Upload the problem files including the domain file, problem file and animation file by dragging files into the specific ‘drag and drop’ zones	visFromProblem	Must have	2
			2.3 Connect problem files and fetch them to the backend and receive the corresponding VFG file	visFromProblem	Must have	3
3	As a user, I could upload a VFG file for generating the visualisation directly.	• When the user enters the page for generating visualization from VFG file, user could see one ‘drag and drop’ zone for uploading the VFG file. • User could drag the VGF file to the zones, if the file is not a .vfg file, the system will give an alert message. • User could click ‘submit’ button to send the uploaded VFG file and jump to the visualization page successfully if the upload file could be parsed.	3.1 Build the interface page including the drag&drop zone for uploading the VFG file	visFromVFG	Must have	1
			3.2 Upload the VFG file by dragging the VFG file into the ‘drag and drop’ zone	visFromVFG	Must have	2
4	As a user, I could find a demo video or doc demonstration to learn how to operate this animation.	• When the user enters the demo page, user could see a demo video that includes all functional operations to help users learn how to operate this animation.	4.1 Create a demo page(or Github page) and connect it with other pages	demo page	could have	5
			4.2 Show the demo video to help users learn how to operate this animation.	demo page	could have	5

5	As a user, I could find a user manual to help me use this web-based application.	• When the user enters the user manual page, user could browse a detailed instructions for this platform, including the environment deployment, functions of each interface and how to operate these functions, and so on.	5. Create a user manual page and connect it with other pages	manual page	could have	5
6	As a user, I could view the animation of the visualisation of a particular planning problem on the visualizer page after uploading the files.	• When the user enters the visualization page, user could see an animation player for the solved planning problem. • User could see the solution of the uploaded problem in an animated way. • There should be an animation between steps • All kinds of planning problems should be able to be visualized	6.1 Build the interface page for the visualizer page	UI	Must have	2
			6.2 Parses the VFG file and convert the content of the VFG file to a JSON object	visScreen	Must have	2
			6.4 Get the rendered objects from the JSON object and visualize them on the screen	visScreen	Must have	4
			6.5 Extract the position data of all rendered objects from the JSON object	visScreen	Must have	2
			6.6 Realize the animation of the motion of the rendered objects	visScreen	Must have	5
			6.7 Add transition animation between steps	visScreen	Must have	7
			6.8 Generalize the visualization and realize the visualization of all kinds of planning problems	visScreen	Must have	10
7	As a user, I could check each step of the plan, the status of any step in the animation by selecting a particular step, and the detailed step information of the selected step on the visualizer page.	• The user can see the detailed information when selecting a certain step, and the corresponding subgoal information should also be shown. • The corresponding visualization should be displayed. • All the steps should be displayed and clickable.	7.1 Build a step panel to show all the steps of the plan	visSteps	Must have	2
			7.2 Select a step in the step panel and then display the visualization of this step	visSteps	Must have	3
			7.3 Build a step information panel to show the detailed information of the selected step	visSteps	Must have	2
8	As a user, I could check the subgoals of each step and all the steps corresponding to a certain subgoal.	• The user can see the detailed information when selecting certain subgoals, and the corresponding step information should also be shown. • The corresponding visualization should be displayed. • All the subgoals should be displayed and clickable.	8.1 Show all the subgoals of the plan	visSubGoals	Must have	2
			8.2 Highlight all subgoals in the corresponding step status	visSubGoals	Must have	3
			8.3 Click a subgoal, then list all the steps that achieve this subgoal	visSubGoals	Must have	4
			8.4 Click a step in the step list of the subgoal and then display the visualization of this step	visSubGoals	Must have	3
			8.5 Add an arrow beside the subgoal items to indicate there is a drop-down list	visSubGoals	Must have	1
9	As a user, I could view the visualization of the final goal state.	• The animation should show the last state when the user clicks 'show the goal' button.	9.1 Add the 'show the goal' button	visSubGoals	Must have	1
			9.2 Show the visualization of the final goal	visSubGoals	Must have	2
10	As a user, I could check the visualization status of the previous or next step.	• When users click 'next step', the animation should pause at the next step, and show the next steps and subgoals information. • When users click 'previous step', the animation should pause at the previous step, and show the previous steps and subgoals information.	10.1 Add the 'previous step' button and the 'next step' button	visControl	Must have	1
			10.2 Click the 'previous step' button and show the visualization of the previous step	visControl	Must have	2
			10.3 Click the 'next step' button and show the visualization of the next step	visControl	Must have	2
11	As a user, I could control the display of the animation, including play, pause, and reset.	• When the animation has not begun, the user can play the animation with a button. • When the animation is playing, the user can press a button to pause. • The user can reset the animation to the beginning.	11.1 Add the 'Play' button, 'Pause' button and 'Reset' button.	visControl	Must have	2
			11.2 Click the "Pause" button, the "Pause" button will turn grey, and vice versa.	visControl	Must have	3
			11.3 Click the "Play" button, and the animation will play the visualisation step by step in order.	visControl	Must have	6
			11.4 Click the "Pause" button, and the animation will stop playing.	visControl	Must have	3
			11.5 Click the "Reset" button, the canvas will be reset to the initial stage of the animation.	visControl	Must have	3
12	As a user, I could control the display speed of the animation.	• The user can drag the slider to control the animation speed when playing. • The fastest speed is 5x, the slowest speed is 1x. • If the animation is not currently playing, the change of speed should come into effect as soon as it plays.	12.1 Add a slider component to control the speed of animation.	visControl	Must have	2
			12.2 Slide the slider component and it would show the relative speed level.	visControl	Must have	2
			12.3 Drag the slider point to the right, the animation play speed will increase, and vice versa.	visControl	Must have	7
13	As a user, I could export the animation file.		13.1 Add an 'Export' button.	visControl	Must have	2

		• The user can download the animation in the format they want. • The downloaded file can play the animation.	13.2 Export the VFG file of this planning problem.	visControl	Must have	4
14	As a user, I could load this planimation platform from the PDDL online editor as a plugin	• The user can choose to load this planimation plugin. • The user can expect the same functions in the editor as from the platform.	14. Create a plugin that could run the planimation platform on the PDDL online editor by adding the plugin to the PDDL online editor platform.	plugin	could have	20
15	As a user, I could load Planimation from Visual Studio Code	• The user can find Planimation in VS Code extension panel. • The user can download and config the Planimation extension.	15. Create a plugin for VSCODE to show planning animations	VSC plugin	could have	10

version 2

Story ID	User Story	Acceptance Criteria	Breakdown Tasks	Components	Priority	User /function points
1	As a user, I could access the main interface for access to four sub-modules (including generating the visualisation from problem files, generating the visualisation from VFG file, accessing the user manual and accessing the demo).	• When the user enters the main interface, the user could easily find the buttons to the four sub-pages (including generating the visualisation from problem files, generating the visualisation from VFG file, accessing the user manual and accessing the demo) on the page. • When the user clicks the corresponding button, he can successfully jump to the corresponding page.	1.1 Build the whole framework for the application	UI	Must have	5
			1.2 Design and build the interface page	UI	Must have	3
			1.3 Build a navigation bar applying to the entire application	UI	Must have	2
2	As a user, I could upload domain, problem, and animation PDDL files for generating the visualisation of the plan (i. e. solution) of this planning problem.	• When the user enters the page for generating visualization from PDDL problem files, the user could see three ‘drag and drop’ zones for uploading the required domain, problem, and animation files, each uploading zone has a corresponding description of the file to be uploaded. • User could drag the PDDL files to the corresponding zones, if the files are not a .ppdl file, the system will give an alert message. • User could click ‘submit‘ button to send the uploaded problem files and jump to the visualization page successfully if the upload files could get a planning problem solution.	2.1 Build the interface page including the drag&drop zones for uploading problem files	UI	Must have	1
			2.2 Upload the problem files including the domain file, problem file and animation file by dragging files into the specific ‘drag and drop' zones	visFromProblem	Must have	2
			2.3 Connect problem files and fetch them to the backend and receive the corresponding VFG file	visFromProblem	Must have	3
3	As a user, I could upload a VFG file for generating the visualisation directly.	• When the user enters the page for generating visualization from VFG file, user could see one ‘drag and drop’ zone for uploading the VFG file. • User could drag the VGF file to the zones, if the file is not a .vfg file, the system will give an alert message. • User could click ‘submit’ button to send the uploaded VFG file and jump to the visualization page successfully if the upload file could be parsed.	3.1 Build the interface page including the drag&drop zone for uploading the VFG file	visFromVFG	Must have	1
			3.2 Upload the VFG file by dragging the VFG file into the ‘drag and drop’ zone	visFromVFG	Must have	2
4	As a user, I could find a demo video or doc demonstration to learn how to operate this animation.	• When the user enters the demo page, user could see a demo video that includes all functional operations to help users learn how to operate this animation.	4.1 Create a demo page(or Github page) and connect it with other pages	demo page	could have	5
			4.2 Show the demo video to help users learn how to operate this animation.	demo page	could have	5
5	As a user, I could find a user manual to help me use this web-based application.	• When the user enters the user manual page, user could browse a detailed instructions for this platform, including the environment deployment, functions of each interface and how to operate these functions, and so on.	5. Create a user manual page and connect it with other pages	manual page	could have	5
6	As a user, I could view the animation of the visualisation of a particular planning		6.1 Build the interface page for the visualizer page	UI	Must have	2

	problem on the visualizer page after uploading the files.	• When the user enters the visualization page, user could see an animation player for the solved planning problem. • User could see the solution of the uploaded problem in an animated way.	6.2 Parses the VFG file and convert the content of the VFG file to a JSON object	visScreen	Must have	2
			6.4 Get the rendered objects from the JSON object and visualize them on the screen	visScreen	Must have	4
			6.5 Extract the position data of all rendered objects from the JSON object	visScreen	Must have	2
			6.6 Realize the animation of the motion of the rendered objects	visScreen	Must have	5
7	As a user, I could check each step of the plan, the status of any step in the animation by selecting a particular step, and the detailed step information of the selected step on the visualizer page.	• The user can see the detailed information when selecting a certain step, and the corresponding subgoal information should also be shown. • The corresponding visualization should be displayed. • All the steps should be displayed and clickable.	7.1 Build a step panel to show all the steps of the plan	visSteps	Must have	2
			7.2 Select a step in the step panel and then display the visualization of this step	visSteps	Must have	3
			7.3 Build a step information panel to show the detailed information of the selected step	visSteps	Must have	2
8	As a user, I could check the subgoals of each step and all the steps corresponding to a certain subgoal.	• The user can see the detailed information when selecting a certain subgoals, and the corresponding step information should also be shown. • The corresponding visualization should be displayed. • All the subgoals should be displayed and clickable.	8.1 Show all the subgoals of the plan	visSubGoals	Must have	2
			8.2 Highlight all subgoals in the corresponding step status	visSubGoals	Must have	3
			8.3 Click a subgoal, then list all the steps that achieve this subgoal	visSubGoals	Must have	4
			8.4 Click a step in the step list of the subgoal and then display the visualization of this step	visSubGoals	Must have	3
9	As a user, I could view the visualization of the final goal state.	• The animation should show the last state when the user clicks 'show the goal' button.	9.1 Add the 'show the goal' button	visSubGoals	Must have	1
			9.2 Show the visualization of the final goal	visSubGoals	Must have	2
10	As a user, I could check the visualization status of the previous or next step.	• When user click 'next step', the animation should pause at the next step, and show the next steps and subgoals information. • When user click 'previous step', the animation should pause at the previous step, and show the previous steps and subgoals information.	10.1 Add the 'previous step' button and the 'next step' button	visControl	Must have	1
			10.2 Click the 'previous step' button and show the visualization of the previous step	visControl	Must have	2
			10.3 Click the 'next step' button and show the visualization of the next step	visControl	Must have	2
11	As a user, I could control the display of the animation, including play, pause, and reset.	• When the animation has not began, the user can play the animation with a button. • When the animation is playing, the user can press a button to pause. • The user can reset the animation to the beginning.	11. Create component that could control the play of animation including play and pause	visControl	Must have	4
12	As a user, I could control the display speed of the animation.	• The user can drag the slider to control the animation speed when playing. • The fastest speed is 5x, the slowest speed is 1x. • If the animation is not currently playing, the change of speed should come into effect as soon as it plays.	12. Create component that changes tick speed	visControl	Must have	5
13	As a user, I could export the animation file.	• The user can download the video with the format they want. • The downloaded file can play the animation.	13. The animation file could be saved as a video file like mp4 etc.	visScreen	Must have	10
14	As a user, I could load this planimation platform from the PDDL online editor as a plugin	• The user can choose to load this planimation plugin. • The user can expect the same functions in the editor as from the platform.	14. Ensure and test that current plugin is still functional		Must have	4
15	As a user, I could load Planimation from Visual Studio Code	• The user can find Planimation in VS Code extension panel. • The user can download and config the Planimation extension.	15. Create a plugin for VSCODE to show planning animations	VSC plugin	could have	20

16	As an advance user or project maintainer, I want to quickly understand the code to make updates or modifications	• The user needs to login and verified to be a maintainer to edit the documents. • A verified user can create or update documents. • A visitor can only read the documents.	16. Create or update documentation	Documentation	could have	5

version 1.1

Breaking downtask

User Story Table

ID	User	Component	Subtask	Story	Priority
1.1	Students /Researchers /Industry Partners	Vis from problem	- Connect pddl files and fetch them to backend and receive VFG file	As a user, I could upload domain, problem, animation PDDL for generating the animation as an option.	Must have
1.2		Vis from problem	- Built the interface page including the drag&drop zones for uploading problem files.		
1.3		Vis from problem	- Once the VFG file is loaded, the user will be taken to the Visualiser screen		
2.1	Students /Researchers /Industry Partners	Vis from Solution (VFG)	- Built the interface page including the drag&drop zone for uploading vfg files.	As a user, I could choose to upload VFG file for visualization.	Must have
2.2		Vis from Solution (VFG)	- Once the correct files are uploaded, the user will be taken to the Visualiser screen		
2.3		Vis from Solution (VFG)	-Parse VFG files		
3	Students /Researchers /Industry Partners	Main menu	-Create a user manual page and connect it with other pages	As a user, I could find a user manual to help me use this web-based application.	Could have
4.1	Students /Researchers /Industry Partners	Main menu	-Create a demo page(or Github page) and connect it with other pages	As a user, I could find a demo video or doc demonstration to learn how to operate this animation.	Could have
4.2		Main menu	-Show the demo video and doc demonstration to help users learn how to operate this animation.		
5.1	Students /Researchers /Industry Partners	Steps	-Create a component with a plan list on the animation page that sh ows all the steps in the solution. Steps can be clicked to navigate the animation to that step.	After uploading the file, I could check each step of the plan and the plan status on the page.	Must have
5.2		Steps	-Create a step information panel to show detailed information about each step including the actions.		
6	Students /Researchers /Industry Partners	Controls	Create component that could control the play of animation including play and pause	Also, I could choose to play or pause the animation anytime in the plan.	Must have
7	Students /Researchers /Industry Partners	Controls	Create component that changes tick speed	If I want, I could play it with a higher speed or lower speed.	Should have
8	Students /Researchers /Industry Partners	Steps	-make plan list clickable and interactable with the animation from the VFG file	I could check the status of any step in the animation by selecting it from the step bar.	Should have
9	Students /Researchers /Industry Partners	Navigation	create a save button component to download the Visualisation file.	The animation file could be saved as a video file like mp4 etc.	Could have
10	Students /Researchers /Industry Partners	Subgoal/Steps	one component includes two button and a progress bar to control the display	Any time during the animation. I could easily move to the previous or next step.	Should have
11.1	Students /Researchers /Industry Partners	UI	deploy the environment with a UI framework (Material-UI)	User can identify every element through a clear UI design	Could have

11.2		UI	set colors, fonts, component sizes, icons		
12	Students /Researchers /Industry Partners	VSCODE Plugin	Create a plugin for VSCODE to show planning animations	An user can use VSCODE to show planning animations	Could have
13.1	Students /Researchers /Industry Partners	Visualisation screen	Create component to Render object sprites from VFG files	Create screen showing the animation objects (Sprites)	Must have
13.2	Students /Researchers /Industry Partners	Visualisation screen	Animate Sprites according to VFG file	Create animation (ticks) between objects.	Must have

Version 1

User Story Table

Story ID	User	Components	Story	Priority
1	Students/Researchers/Industry Partners	Vis from problem	As a user, I could upload domain, problem, animation PDDL for generating the animation as an option.	Must have
2	Students/Researchers/Industry Partners	Vis from Solution (VFG)	As a user, I could choose to upload VFG file for visualization.	Must have
3	Students/Researchers/Industry Partners	Main menu	As a user, I could find a user manual to help me use this web-based application.	Should have
4	Students/Researchers/Industry Partners	Main menu	As a user, I could find a demo video or doc demonstration to learn how to operate this animation.	Should have
5	Students/Researchers/Industry Partners	Steps	After uploading the file, I could check each step of the plan and the plan status on the page.	Must have
6	Students/Researchers/Industry Partners	Controls	Also, I could choose to play or pause the animation anytime in the plan.	Must have
7	Students/Researchers/Industry Partners	Controls	If I want, I could play it with a higher speed or lower speed.	Should have
8	Students/Researchers/Industry Partners	Steps	I could check the status of any step in the animation by selecting it from the step bar.	Should have
9	Students/Researchers/Industry Partners	Navigation	The animation file could be saved as a video file like mp4 etc.	Could have
10	Students/Researchers/Industry Partners	Subgoal/Steps	Any time during the animation. I could easily move to the previous or next step.	Should have
11	Students/Researchers/Industry Partners	UI	Clear design of UI	Could have

Product Backlog

Version control

Date	Version	Description	Author
22 Aug 2021	01.00	Initial draft	Felipe Ramos
19 Sep 2021	01.10		Felipe Ramos
27 Sep 2021	02.00		Ziqi Meng

Product Backlog Version 2.0

Story ID	User Story	Breakdown Tasks	Components	Priority
1	As a user, I could access the main interface for access to four sub-modules (including generating the visualisation from problem files, generating the visualisation from VFG file, accessing the user manual and accessing the demo).	1.1 Build the whole framework for the application	UI	Must have
		1.2 Design and build the interface page	UI	Must have
		1.3 Build a navigation bar applying to the entire application	UI	Must have
2	As a user, I could upload domain, problem, and animation PDDL files for generating the visualisation of the plan (i.e. solution) of this planning problem.	2.1 Build the interface page including the drag&drop zones for uploading problem files	UI	Must have
		2.2 Upload the problem files including the domain file, problem file and animation file by dragging files into the specific ‘drag and drop' zones	visFromProblem	Must have
		2.3 Connect problem files and fetch them to the backend and receive the corresponding VFG file	visFromProblem	Must have
3	As a user, I could upload a VFG file for generating the visualisation directly.	3.1 Build the interface page including the drag&drop zone for uploading the VFG file	visFromVFG	Must have
		3.2 Upload the VFG file by dragging the VFG file into the ‘drag and drop’ zone	visFromVFG	Must have
4	As a user, I could find a demo video or doc demonstration to learn how to operate this animation.	4.1 Create a demo page(or Github page) and connect it with other pages	demo page	could have
		4.2 Show the demo video to help users learn how to operate this animation.	demo page	could have
5	As a user, I could find a user manual to help me use this web-based application.	5. Create a user manual page and connect it with other pages	manual page	could have
6	As a user, I could view the animation of the visualisation of a particular planning problem on the visualizer page after uploading the files.	6.1 Build the interface page for the visualizer page	UI	Must have
		6.2 Parses the VFG file and convert the content of the VFG file to a JSON object	visScreen	Must have
		6.4 Get the rendered objects from the JSON object and visualize them on the screen	visScreen	Must have
		6.5 Extract the position data of all rendered objects from the JSON object	visScreen	Must have
		6.6 Realize the animation of the motion of the rendered objects	visScreen	Must have
		6.7 Add transition animation between each step	visScreen	Must have
		6.8 Generalize the visualization and realize the visualization of all kinds of planning problems	visScreen	Must have
7	As a user, I could check each step of the plan, the status of any step in the animation by selecting a particular step, and the detailed step information of the selected step on the visualizer page.	7.1 Build a step panel to show all the steps of the plan	visSteps	Must have
		7.2 Select a step in the step panel and then display the visualization of this step	visSteps	Must have
		7.3 Build a step information panel to show the detailed information of the selected step	visSteps	Must have
8	As a user, I could check the subgoals of each step and all the steps corresponding to a certain subgoal.	8.1 Show all the subgoals of the plan	visSubGoals	Must have
		8.2 Highlight all subgoals in the corresponding step status	visSubGoals	Must have

		8.3 Click a subgoal, then list all the steps that achieve this subgoal	visSubGoals	Must have
		8.4 Click a step in the step list of the subgoal and then display the visualization of this step	visSubGoals	Must have
		8.5 Add an arrow beside the subgoal items to indicate there is a drop-down list	visSubGoals	Must have
9	As a user, I could view the visualization of the final goal state.	9.1 Add the 'show the goal' button	visSubGoals	Must have
		9.2 Show the visualization of the final goal	visSubGoals	Must have
10	As a user, I could check the visualization status of the previous or next step.	10.1 Add the 'previous step' button and the 'next step' button	visControl	Must have
		10.2 Click the 'previous step' button and show the visualization of the previous step	visControl	Must have
		10.3 Click the 'next step' button and show the visualization of the next step	visControl	Must have
11	As a user, I could control the display of the animation, including play, pause, and reset.	11.1 Add the 'Play' button, 'Pause' button and 'Reset' button.	visControl	Must have
		11.2 Click the "Pause" button, the "Pause" button will turn grey, and vice versa.	visControl	Must have
		11.3 Click the "Play" button, and the animation will play the visualisation step by step in order.	visControl	Must have
		11.4 Click the "Pause" button, and the animation will stop playing.	visControl	Must have
		11.5 Click the "Reset" button, the canvas will be reset to the initial stage of the animation.	visControl	Must have
12	As a user, I could control the display speed of the animation.	12.1 Add a slider component to control the speed of animation.	visContro	Must have
		12.2 Slide the slider component and it would show the relative speed level.	visContro	Must have
		12.3 Drag the slider point to the right, the animation play speed will increase, and vice versa.	visContro	Must have
13	As a user, I could export the animation file.	13.1 Add an 'Export' button.	visScreen	Must have
		13.2 Export the VFG file of this planning problem.	visScreen	Must have
14	As a user, I could load this planimation platform from the PDDL online editor as a plugin	14. Create a plugin that could run the planimation platform on the PDDL online editor by adding the plugin to the PDDL online editor platform.	plugin	could have
15	As a user, I could load Planimation from Visual Studio Code	15. Create a plugin for VSCODE to show planning animations	VSC plugin	could have

Product Backlog Version 1.1

Epic	Story ID	User Story	Breakdown Tasks	Components	Priority
	1	As a user, I could access the main interface for access to four sub-modules (including generating the visualisation from problem files, generating the visualisation from VFG file, accessing the user manual and accessing the demo).	1.1 Build the whole framework for the application	UI	Must have
			1.2 Design and build the interface page	UI	Must have
			1.3 Build a navigation bar applying to the entire application	UI	Must have
	2	As a user, I could upload domain, problem, and animation PDDL files for generating the visualisation of the plan (i.e. solution) of this planning problem.	2.1 Build the interface page including the drag&drop zones for uploading problem files	UI	Must have
			2.2 Upload the problem files including the domain file, problem file and animation file by dragging files into the specific ‘drag and drop' zones	visFromProblem	Must have
			2.3 Connect problem files and fetch them to the backend and receive the corresponding VFG file	visFromProblem	Must have
	3	As a user, I could upload a VFG file for generating the visualisation directly.	3.1 Build the interface page including the drag&drop zone for uploading the VFG file	visFromVFG	Must have

			3.2 Upload the VFG file by dragging the VFG file into the ‘drag and drop’ zone	visFromVFG	Must have
	4	As a user, I could find a demo video or doc demonstration to learn how to operate this animation.	4.1 Create a demo page(or Github page) and connect it with other pages	demo page	could have
			4.2 Show the demo video to help users learn how to operate this animation.	demo page	could have
	5	As a user, I could find a user manual to help me use this web-based application.	5. Create a user manual page and connect it with other pages	manual page	could have
	6	As a user, I could view the animation of the visualisation of a particular planning problem on the visualizer page after uploading the files.	6.1 Build the interface page for the visualizer page	UI	Must have
			6.2 Parses the VFG file and convert the content of the VFG file to a JSON object	visScreen	Must have
			6.4 Get the rendered objects from the JSON object and visualize them on the screen	visScreen	Must have
			6.5 Extract the position data of all rendered objects from the JSON object	visScreen	Must have
			6.6 Realize the animation of the motion of the rendered objects	visScreen	Must have
	7	As a user, I could check each step of the plan, the status of any step in the animation by selecting a particular step, and the detailed step information of the selected step on the visualizer page.	7.1 Build a step panel to show all the steps of the plan	visSteps	Must have
			7.2 Select a step in the step panel and then display the visualization of this step	visSteps	Must have
			7.3 Build a step information panel to show the detailed information of the selected step	visSteps	Must have
	8	As a user, I could check the subgoals of each step and all the steps corresponding to a certain subgoal.	8.1 Show all the subgoals of the plan	visSubGoals	Must have
			8.2 Highlight all subgoals in the corresponding step status	visSubGoals	Must have
			8.3 Click a subgoal, then list all the steps that achieve this subgoal	visSubGoals	Must have
			8.4 Click a step in the step list of the subgoal and then display the visualization of this step	visSubGoals	Must have
	9	As a user, I could view the visualization of the final goal state.	9.1 Add the 'show the goal' button	visSubGoals	Must have
			9.2 Show the visualization of the final goal	visSubGoals	Must have
	10	As a user, I could check the visualization status of the previous or next step.	10.1 Add the 'previous step' button and the 'next step' button	visControl	Must have
			10.2 Click the 'previous step' button and show the visualization of the previous step	visControl	Must have
			10.3 Click the 'next step' button and show the visualization of the next step	visControl	Must have
	11	As a user, I could control the display of the animation, including play, pause, and reset.	11. Create component that could control the play of animation including play and pause	visControl	Must have
	12	As a user, I could control the display speed of the animation.	12. Create component that changes tick speed	visControl	Must have
	13	As a user, I could export the animation file.	13. The animation file could be saved as a video file like mp4 etc.	visScreen	Must have
	14	As a user, I could load this planimation platform from the PDDL online editor as a plugin	14. Ensure and test that current plugin is still functional		Must have
	15	As a user, I could load Planimation from Visual Studio Code	15. Create a plugin for VSCODE to show planning animations	VSC plugin	could have
	16	As an advance user or project maintainer, I want to quickly understand the code to make updates or modifications	16. Create or update documentation	Documentation	could have