1. Development Environment 2
1.1 Non-technical Environment 3
1.2 Technical Environment 4
1.3 Deployment 7
1.4 Development Environment Setup 9

Development Environment

This page contains:

- Non-technical Environment Includes the deployment and usage of non-technical tools.
- Technical Environment Includes the technical environment and tools planned to be used.
- Deployment Includes introduction and tutorial related to project deployment.
- Development Environment Setup Includes how to deploy the development environment.

Non-technical Environment

- Trello
  During the project development process, the team will use Trello for task assignment and tracking, including but not limited to technical tasks. Each team member should complete his/her tasks within the specified time and drag the completed ones into the corresponding board.

  Link to our Kanban: https://trello.com/b/yvYNsEfj/todos

- Confluence
  All documents will be updated on Confluence, including documents for project development and interface documents (if any).

- Slack
  Team members use Slack for informal instant communication.

  Link to our channel: click here

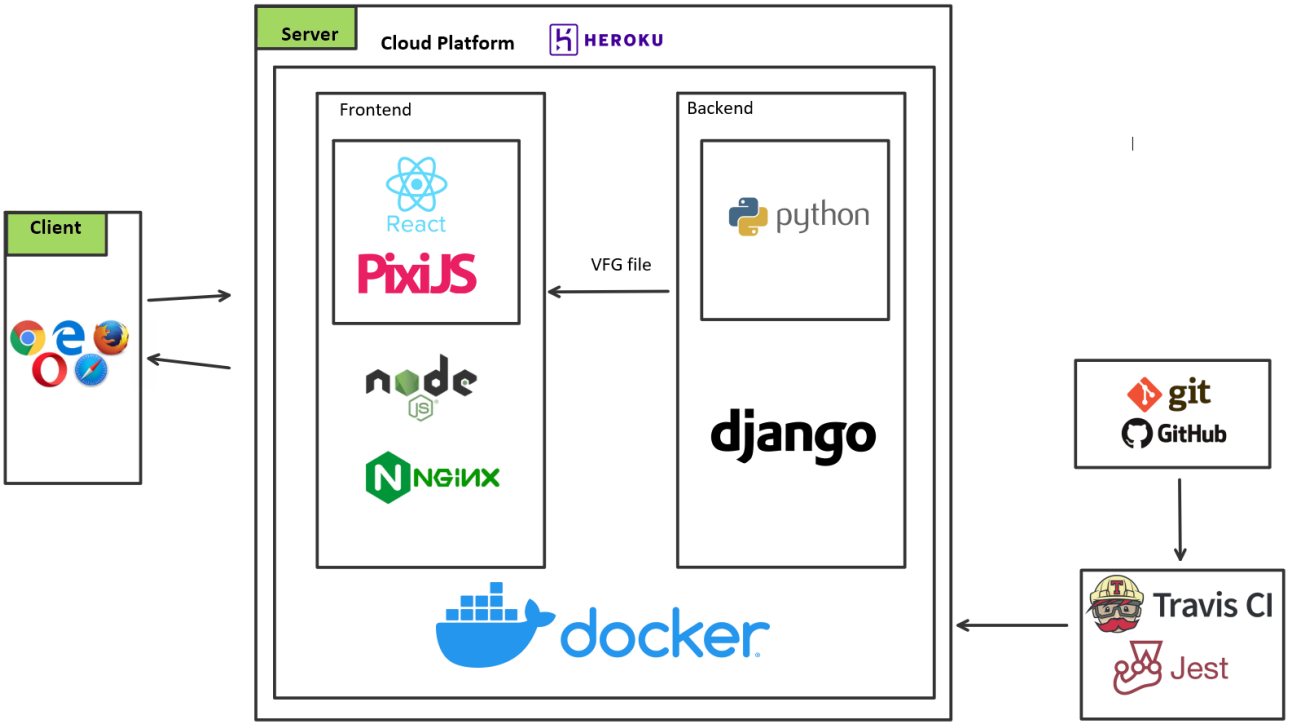
- Zoom
  Formal meetings and standups will be held on Zoom. The meeting link will be given on Slack before each meeting.

Technical Environment

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.



The planned tools of the project are listed below.

- **Technical Environment#GitHub**
- **Technical Environment#Visual Studio Code**
- **Technical Environment#Travis CI+Jest**
- **Technical Environment#Nginx**
- **Technical Environment#Docker**
- **Technical Environment#Visual Studio Code API**
- **Technical Environment#PixiJS**
- **Technical Environment#React**
- **Technical Environment Plan#MaterialUI**

- 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

- MateriaUI

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/

Deployment

Deployment can be divided into two parts: One is for backend and the other is for frontend.

As our project will not modify the backend, the deployment tutorial for backend please refers to the original documents: https://planimation.github.io/documentation/deployment_guide/

For a JS project, Nginx is used here replacing the former Apache Web Server. And you don't need to install Unity or its dependencies.

Here are the steps to deploy the frontend on the server.

For Ubuntu server, run the following commands to install Node environment to enable JavaScript running in the server:

```
sudo apt update
sudo apt install nodejs npm
```

Use the following command to validate its intallment:

```
nodejs --version
```

In the frontend root directory, build the project:

```
npm run build
```

Now, a directory called **build** is created. It stores all the compiled files.

Before moving to the next step, please ensure that docker is installed and running in the server. How to install Docker: https://docs.docker.com/engine/install/ubuntu/

Download nginx image:

```
docker pull nginx
```

In the frontend root directory, run:

```
docker run -d --name=nginx -p 8080:80 -v $PWD/dist:/usr/share/nginx/html nginx
```

Check the running container:

```
docker ps
```

Now the project is successfully deployed on the server!

Deployment for production in Heroku

Prerequisites

- Make sure this files are present in the project root directory:
  - `.dockerignore`, `Dockerfile`, `nginx.conf` and `static.json`
- Git is installed
- Docker is intalled

1- Install Heroku CLI

Create an a Heroku account or link an existing one. Download and install Heroku CLI using the following instructions https://devcenter.heroku.com/articles/heroku-cli

Link the count using the following command

```
heroku container:login
```

2- Create a new application

```
heroku create
```

Then add to git heroku the URL of you app

```
git remote add docker https://git.heroku.com/<your-heroku-app-name>.git
```

3- Build and push the docker container to heroku and release it

```
heroku container:push web --remote docker
```

Realese the app into production

```
heroku container:release web --remote docker
```

4- Docker container updates

To update the docker image repeat the steps in point 3 with the new files.

Development Environment Setup

Before building your development environment, you'll need:

A code editor

VS Code is suggested. Download link

Recommended extensions: Jest; GitLens;tslint;

WebStorm is an alternative JavaScript IDE for frontend development.

Git

Check if you installed Git:

```
git --version
```

Please refer to https://docs.github.com/en/get-started/quickstart/set-up-git to see the installment and basic usage of Git.

Node

Please follow the instructions and download Node.js to your system. https://nodejs.org/en/

Recommended version: 14.17.5. Npm is the package manager for the Node JavaScript It is already installed along with Node.

Now you can check the versions:

```
node -v
v14.17.5
npm -v
6.14.14
```

Install **yarn**:

```
npm install -g yarn
```

Building development environment:

Clone the repository from GitHub:

```
git clone https://github.com/visual-heuristics/frontend-js
```

It's good if you could setup your username and email address for the repository, so the other team members will have a clear idea the contributors of branches and codes on GitHub.

In the current working directory to the local repository:

```
git config user.name "Mona Lisa"

git config user.email "email@example.com"
```

Initial preparations for the development:

NB: All These steps have already been done in this project.

You can jump to the next section as this part is only recorded for the documentary.

- Setting up an organization account and invites all the team members on GitHub.

- Fork the original repository https://github.com/planimation/Frontend-JS

All the project iterations will be conducted in this repository: https://github.com/visual-heuristics/Frontend-JS

The pull request will open to the maintainer of the original repository by the end of the development process. Therefore, the modifications on this repository would not make any changes on the upstream.

- Install React boilerplate:

```
npm install -g create-react-app
```

Outside the working directory:

```
create-react-app frontend-js
```

In the **frontend-js** directory:

```
npm run eject // expose webpack.config.js
npm install --save-dev less less-loader
npm install --save react-router-dom // For page router
npm install –-save @inlet/react-pixi // for react pixi
npm install --save-dev typescript
```

- Modify **webpack.config.js**:

add:

```
const lessRegex = /\.less$/;
const lessModuleRegex = /\.module\.less$/;
```

add the following code in **module.rules**:

```
{
              test: lessRegex,
              exclude: sassModuleRegex,
              use: getStyleLoaders(
                {
                  importLoaders: 2,
                  sourceMap: isEnvProduction && shouldUseSourceMap,
                },
                'less-loader'
              ),
              sideEffects: true,
            },
            {
              test: lessModuleRegex,
              use: getStyleLoaders(
                {
                  importLoaders: 2,
                  sourceMap: isEnvProduction && shouldUseSourceMap,
                  modules: true,
                  getLocalIdent: getCSSModuleLocalIdent,
                },
                'less-loader'
              ),
            },
```

- Sync the repository to Travis CI:

Tutorial https://docs.travis-ci.com/user/tutorial/

install **jest**

```
npm install --save-dev jest
```

add the following codes in **package.json** :

```
"scripts": {
  "test": "jest"
}
```

Create a file .travis.yml in the **frontend-js** folder:

```
language: node_js
node_js:
- 8
branchs:
only:
- master
cache:
directories:
- node_modules
install:
- yarn install
scripts:
- yarn test
- yarn build
```

Install frontend development environment:

Open the working directory with VS code, in the terminal:

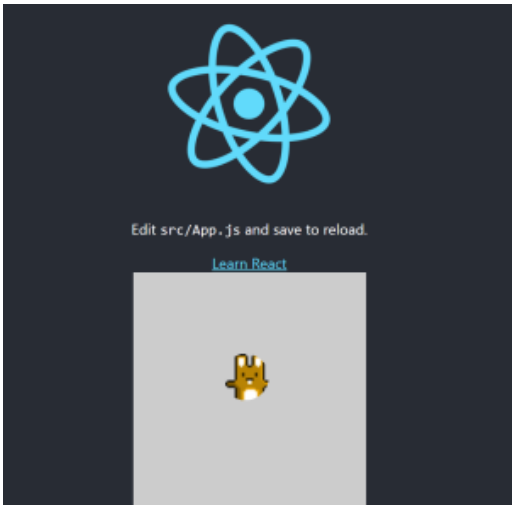
```
yarn //or “npm install”
```

This command will automatically install all the packages you need in this project.

Run the demo:

```
npm run start
```

This will open a page on your browser, and you should see a rabbit like this:



Now, the development environment is all set up! Start coding

Test your code:

```
yarn test  //or “npm test”
```