



## Hacking Materials User Interface (HA) Project

Client: Dr. Christian Brandl

Supervisor: Mauro Mello Jr

Authored by Team

BlueRing

BoxJelly

RedBack

# Table of Contents

<b>Introduction</b>	<b>3</b>
How to continue developing the product	4
<b>How to deploy the product</b>	<b>5</b>
Prerequisites	5
Deployment process	5
<b>How to test the product</b>	<b>6</b>
How to use PyTest	6
Where to find our manual test cases	6
Recommendations to future developers	7
Development practices	7
How to use common Trello boards between the 3 teams:	7
How to do code reviews	7
Test suggestions for both frontend and backend	7
Report bugs in GitHub	7
Submit early	7
Branching	7
Future work	8
Data preparation component	8
Add support for the application of multiple featurizers.	8
Add support for multiple column selection for featurization instead of just one.	8
Machine learning component	8
Plot data type consistency	8
Add support for more machine learning models	8
Use a database cluster	8
General improvements	8
Extendability:	8
Security:	8
Frontend code improvements	8
Automatically mapping step components to stages	8
Execution button consistency	8
Completing the remaining user stories	9
<b>Appendices</b>	<b>12</b>

# Introduction

The Hacking Materials User Interface (HA) project was conducted by three student teams (BlueRing, BoxJelly and RedBack) in the subject COMP90082 Software Project, 2022 semester 2, at the University of Melbourne.

The project is split into three main parts: frontend, backend and machine learning.

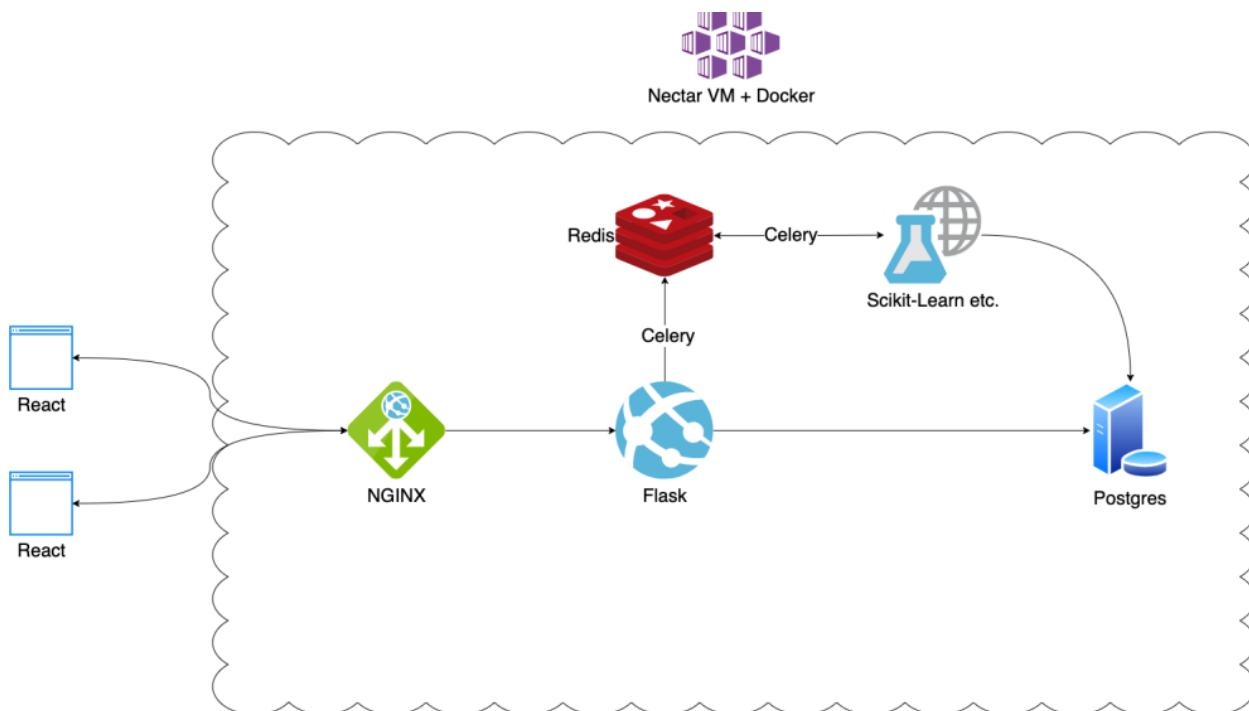
The front end uses React and TypeScript with a storybook for UI prototyping and presentation.

The backend uses Python and Flask, with the key packages being `sqlalchemy`, `matminer`, and `pytest`.

The database is Postgres.

The machine learning is in Python, with the key packages being `sklearn` and `matminer`.

The architecture diagram is as follows:



The workspace follows this storage structure:

Folder	Description
data-samples	Sample data to use in developing and testing the application

docs	Documentation of the project requirements and design
prototypes	Low-fidelity prototypes of the user interface design
src	Source code. Subdirectories are included for backend and frontend development, with machine learning code being considered part of the backend
tests	Documentation on testing the product. In practice, tests are included in the testing folders under <code>src/backend</code> and <code>src/frontend</code>
ui	Resources required by the user interface and prototypes

## How to continue developing the product

- First, you'll need to get access to the project's GitHub repository as described above. If you don't have access to that repository, ask The University of Melbourne COMP90082 Software Project staff for access.
- Once you have a copy of the repository, we strongly suggest that you start by reading the included `README.md` file.
- Getting Started guides for frontend and backend developers are included in the project's GitHub repository in the `DEVELOPERS.md` files under the `src/frontend` and `src/backend` directories. These files provide information about setting up your development environment, installing required tools and libraries, and building the project components.
- Additional documentation and guides are also available within the `docs/` directory under each of the `backend/` and `frontend/` directories.

At minimum, to develop this product you will need:

- A text editor (we recommend [Visual Studio Code](#))
- [Docker](#)
- [Python](#)
- [node.js](#)

# How to deploy the product

## Prerequisites

1. **Google** API key for SSO login - [Authenticate using API keys | Authentication | Google Cloud](#)
2. **Microsoft** API key for SSO login - [Enable authentication in a web API by using Azure Active Directory B2C | Microsoft Learn](#)
3. **Email Address** of assigned database administrator

### Worth a read:

- [Connect over SSH with Visual Studio Code](#)
- [Docker - Visual Studio Marketplace](#)

## Deployment process

There are three steps to deploying this product.

We have included screenshots of this deployment process in [Appendix 9. Deployment process screenshots](#).

1. Spin up a VM on Nectar ARDC, Google Cloud, AWS or Azure. Ensure you choose an **Ubuntu 22.04 image**.
  - a. Use your cloud provider's web interface to set up a virtual machine. For ARDC Nectar, this means:
    - i. Create a key pair
    - ii. Create a virtual machine, using that key pair
    - iii. Enable SSH and HTTPS access to the machine
  - b. Once the virtual machine has been deployed, you can access it via SSH using the IP address provided by your cloud provider.
2. Use SSH to connect to the machine.
3. Run `sudo apt update && sudo apt upgrade` to update the existing software on the machine. You may need to restart the machine if the kernel or other important software has been updated.
4. Clone the GitHub repository and run the deployment script using the commands below.

```
sudo apt install git
git clone https://github.com/COMP90082-2022-SM2/HA-2022-SM2.git
cd HA-2022-SM2/src/backend/
git checkout deployment
```

5. Ensure the deployment script is runnable, then run it:

```
sudo chmod +x ./deploy_on_ubuntu22_04.sh  
./deploy_on_ubuntu22_04.sh
```

The deployment script will:

- a. Initialize the directory structure
- b. Install and update Docker
- c. Start the deployment containers

At three stages, the script will ask for user input.

1. When creating a **self-signed certificate** for HTTPS support it will need information about. If you're deploying to the University of Melbourne's infrastructure, their IT department will likely be able to provide the correct information for you.
2. When initializing the database you will need to provide a **database administrator username and password**.
3. API keys for **Google and Microsoft Single Sign-on APIs**. This is discussed further in [Appendix 7. Set-up SSO authorisation guide](#), but it's also possible that The University of Melbourne IT department can assist with this.

With this done, the script will begin building the project and creating the required containers. At this point you may be able to access the website using your virtual machine's IP address, depending on your browser's security settings. However it's likely that this will not work because the domain of the HTTPS certificate will not match the domain name of the server.

6. Point a domain name to the IP address of the VM. We have used Cloudflare to do this, but again it is likely that The University of Melbourne IT department or your cloud provider can do this for you.

## How to test the product

### How to use PyTest

The guide can be found in `src/backend/src/tests/testing.md`. Additional resources are available in [Appendix 4. Learning Resources](#).

### Where to find our manual test cases

- Refer to [Appendix 6. Test Cases](#).
- High-level setup and preconditions for each test case are included, and the more detailed description can be found in [Appendix 6. Test Cases](#).

# Recommendations to future developers

## Development practices

### How to use common Trello boards between the 3 teams:

Since three teams were working on the same product, a shared epic board and separate user story board for each team were used. This made cross checking the progress of each team a bit difficult. Using one trello board for all teams would be recommended so that progress of tasks is visible for everyone.

### How to do code reviews

Quality assurance guidelines were used for code reviews and they can be found under [Appendix 2: Development Practices](#).

### Test suggestions for both frontend and backend

For frontend testing, we would recommend testing using Jest with React Testing Library and testing each component separately as well as each of the utils. We recommend these tools as they are common and effective, especially with the frontend technology stack used here. Our technical guides list also includes tutorials for these libraries.

### Report bugs in GitHub

Initially there was no way to report bugs for the application. Later GitHub Issues was adopted for bug reporting across the teams. This allowed team members to report problems when they encountered issues while using the application.

### Submit early

While working in a combined group of three teams, there can be varying versions of the submissions. Finishing the work earlier gives the teams a chance to review the consistency among the documents. Also, submitting well before the deadline can prevent the end-moment stress and the blunders that come with it.

### Branching

The deployment branch is separate to the main branch as the branch's configuration, found in `./src/frontend/nginx`, is designed for deployment. All other files should be kept synced with the main branch whose `./src/frontend/nginx` is designed for local development.

## Future work

### Data preparation component

#### **Add support for the application of multiple featurizers.**

A user story has been completed that allows users to apply a single featurizer to the data before proceeding to the next stage. However, this functionality can be extended by allowing the user to apply multiple featurizers simultaneously.

#### **Add support for multiple column selection for featurization instead of just one.**

The current application only allows users to select one column for featurization while the featurizers in matminer can support a list of columns in the dataset. In the future, it would be nice to have the option for users to select multiple columns from the application.

### Machine learning component

#### **Plot data type consistency**

Currently the plot data is a list of dictionaries in json format in the frontend. No type definition is given for the dictionary items. It may cause problems in further data manipulation and may not be easily extendable. Type definition can be provided in the future.

The identifier field in the plot data is not universal across different datasets. User selection can be added to let users decide what to include as the identifier.

#### **Add support for more machine learning models**

The decision tree model and random forest model are obviously not optimal under the current trend of machine learning. Neural network models have gradually become the best models.

In this project, the training of the machine learning model starts after the user selects the dataset and predicts the label. Then, in order to let the user get the reply as soon as possible, the neural network training time is very important.

For future work, more machine learning models can be added to the application to provide more training choices. Besides, the function for the users to select the split ratio between the training data and test data can also be provided to avoid overfitted results due to the same training data and test data. Finally, if the model's comparison can also be made into a visualization among different models, the application's performance can also be further improved.

#### **Use a database cluster**

In contrast to decision trees and random forest models, storing a neural network model consumes a lot of space. This can cause database or server crashes. Optimizing existing databases and servers to handle higher loads will help future versions of this product handle this scenario, for example by adding a load balancer that allocates tasks to multiple sub-servers to complete tasks together.

## General improvements

### Extendability:

The current design assumes that only 30 graduate students will use our product. As the number of users grows, new issues will emerge. Here are a few bottlenecks which could be helpful to future developers.

- There is only one container to handle requests. More servers and containers will be required if the number of users outgrows the server's capability that runs the container.
- There is only one container to store the database. Again, it's possible to run out of storage, or the responses become slow because of hardware limitations.
- Database design can use some optimization. We dump almost everything into one column as a JSON object for flexibility. This approach can slow down some DB operations if new functionalities are added to the product. Another point is that we use user emails as primary keys, which slows down data retrieval operations and could lead to personal information leaks. One solution is to use generated unique tokens as the primary keys.
- The task management component also needs to rework to support multi-server scenarios.

### Security:

We didn't consider cyber security, as none of our team members has studied this topic. All messages ( some of them have user emails and the configurations of their work ) are sent in plain text, which leads to information leaks. Any hacker could easily steal our user's work. Future developers should at least encrypt these messages or find better solutions to minimize the security risks.

## Frontend code improvements

### Automatically mapping step components to stages

Add the step components in a new field for each step in the STEPS\_DATA object in the `src/frontend/src/steps/data.tsx` file. Then, the step components can be removed from the `src/frontend/src/sections/appBody/inputPanel/workflowStages/` files. Instead, the stages can be adjusted to automatically render all the steps in STEPS\_DATA that have the stage number as the first digit in their number. Steps 4 & 5 from the "Adding a new workflow step" process under the "Managing Workflow Steps & Stages" guide in `src/frontend/docs/manage-workflow.md` can be removed.

### Execution button consistency

Currently, the execution button does not accomplish anything in Stage 3 as the plots are generated when the Stage 3 step buttons are clicked.

It is suggested that either the Execution button is removed for Stage 3, or the Stage 3 steps be modified to only collect the user's preferences (e.g. Should a scatter plot be shown? If yes, which columns to use for each axis?) and then the execution button is what triggers the rendering for the plots.

## Completing the remaining user stories

There are a lot more features that had been initially identified but marked as out of scope due to time constraints. A list of all user stories that have not been completed are included in [Appendix 1](#) for reference.

# Appendices

<b>Appendix 1. User stories that were not completed</b>	<b>12</b>
<b>Appendix 2. Development guidelines</b>	<b>14</b>
<b>Appendix 3. Core technologies</b>	<b>18</b>
<b>Appendix 4. Learning resources</b>	<b>19</b>
<b>Appendix 5. Acceptance criteria</b>	<b>23</b>
<b>Appendix 6. Test cases</b>	<b>26</b>
<b>Appendix 7. Set-up SSO authorisation guide</b>	<b>56</b>
<b>Appendix 8. Frontend workspace structure</b>	<b>67</b>
<b>Appendix 9. Deployment process screenshots</b>	<b>71</b>

# Appendix 1. User stories that were not completed

Completed user stories are not included here, but note that some of them may still be improved as suggested above.

ID	User Story	Size (days)	Priority
1	As a student, I want to clean and tune data input Input Data, so that I have less noise on visualizations.	5	3 - Could have
2	As a general user, I want to attach comments to workflow objects Jupyter Notebook, so that I can document my work	5	3 - Could have
3	As a Pro user, I want to be able to add new datasets in the future External Data, so that if there's a new dataset that can be used on a new project, it can be added instantly	5	3 - Could have
4	As a Pro user, I want to edit python code on the interface Jupyter Notebook, so that I can have control how the ML algorithms works	5	3 - Could have
5	As a Pro user, I want to upload my own script (in python) if possible Jupyter Notebook, so that I can extend the tool to support custom models and featurizers	5	3 - Could have
6	As a pro user, I want to have access to more processing power Administration, so that I can run more complex operations or use more data	5	3 - Could have
7	As a pro user, I want to combine multiple models together Machine Learning, so that I can model more complex data manipulations	5	3 - Could have
11	As a student, I want to analyze the relationship between different features Machine Learning, so that I can identify which features I need to select for my analysis	5	2 - Should have
12	As a student, I want to export my work to a Jupyter Notebook Jupyter Notebook, so that I can extend my work beyond the capability of the application	5	2 - Should have
13	As a Pro user, I want to add new features Input Data, so that they can be reused in the future	5	2 - Should have
14	As a pro user, I want to easily find and read documentation on the pro features Administration, so that I can use them with ease	5	2 - Should have
15	As a pro user, I want to be able use additional ML models Machine Learning, so that I can improve accuracy	5	3 - Could have
16	As a general user, I want to add specific materials to the workflow for	3	3 - Could have

ID	User Story	Size (days)	Priority
	analysis Machine Learning, so that compare the performance of the specific material my client or I choose with other material		
17	As a pro user, I want to be kept informed about job status Administration, so that I can avoid polling my workspace to check for results	3	3 - Could have
18	As a pro user, I want to be able to apply new featurizers Input Data, so that I can create new features	3	3 - Could have
19	As a student, I want to quickly browse the Materials available in the database for retrieval and simulations Input Data, so that I can quickly perform queries.	3	2 - Should have
24	As a student, I want to receive provided hints and guidance for new users Administration, so that I can quickly learn how to use software	3	2 - Should have
25	As a general user, I want to select specific features from a dataset Input Data, so that I can improve the precision of my model	3	2 - Should have
26	As a pro user, I want to have the option to change the hyperparameters used in the machine learning model Machine Learning, so that I can fine tune my test results.	3	2 - Should have
27	As a pro user, I want to be able to access new databases External Data, so that I can access additional data	3	2 - Should have
28	As a general user, I want to be able to reference / view citations for original data sources Input Data, so that I can retrieve data.	1	3 - Could have
37	As a general user, I want to be able to preview the output of each execution Input Data, so that I could explore the data	5	2 - Should have
38	As a pro user, I want to have my pro user settings persist on each visit Administration, so that I don't have to reconfigure settings to use the features I need	1	2 - Should have
39	As a user, I want to be able to select split ratio of data Machine Learning, so that to train and test the model	1	2 - Should have

## Appendix 2. Development guidelines

### Quality assurance

#### Work quality guidelines

1. Code should be able to run. No syntax or compile-time errors.
2. If unit or integration tests are included in a pull request, they should pass
3. All preexisting tests should still pass with the new changes in the pull request
4. A branch should not be put up for a pull request if it has merge conflicts with main. All conflicts should be resolved **before** requesting review of a pull request.
5. Code should be understandable and contain documentation

#### Code review guidelines

1. Each PR must be reviewed by one member from each of the other two student teams.
2. Pull requests should:
  1. Be assigned reviewers within **24 hours** of being submitted
  2. Be given initial feedback by reviewers within **48 hours** of being submitted
3. Commit messages should describe the work you've done and the steps you took to verify the correctness of your work
4. Pull request descriptions should summarize the commit messages
5. Pull request changes should be reviewed by using the Review Changes button under the Files Changed tab, to encourage the use of **Accept** or **Review Changes** messages.
6. Pull requests should only be merged by the creator of the pull request
7. Branches that have been successfully merged to main should be deleted by the creator of the pull request
8. Reviewers should check that pull requests follow the work quality guideline above

#### Acceptance criteria definition guidelines

1. Acceptance criteria should be defined from the user's point of view
2. Acceptance criteria should contain a list of steps to test the desired functionality

#### Definition of "done" for a user story

1. Acceptance criteria should be defined for the user story, and should pass
2. All related code has **passed code review and merged to main**

## Development processes

### Sprint lifecycle

#### Sprint kickoff

##### Attendees:

Everyone

##### Tasks:

- Pull user stories into the sprint from the product backlog
- Ensure user stories have complete definitions:
  - Do they have acceptance criteria?
  - Do they have test cases?
  - Are they ready for development? What are their dependencies?
- Re-estimate user stories with T-shirt size and priority
- Schedule kickoff meetings for user stories that are ready for development
- Review action items from the previous sprint's retrospective meeting

The aim of the sprint kickoff meeting is to define the high-level objectives for the sprint and consolidate learnings from previous sprints.

#### Feature kickoff

##### Attendees:

The product owner, plus a subject matter expert for each domain the user story will involve. As an example, this might include:

- The product owner
- A frontend developer
- A backend developer
- A user experience (UX) designer
- Someone who worked on a related feature who may be able to give useful information

##### Tasks:

- Conduct a design discussion for the user story
- Break the user story down into multiple smaller tasks
- For each task, define:
  - Dependencies
  - Size (using **magic estimation**)
  - Relevant test cases and acceptance criteria

- Assigned developer
- Create tasks in Trello

## Development

### Branching

- Use the format feature/t-<ticket> as a feature branch template, where <ticket> is the Trello card number
- Where branches have other purposes, prefixes other than "feature" may be used. For example:
  - "spike", for experimental work that shouldn't be merged
  - "fix", for fixes to existing features
- In general, follow [Git Flow](#) as a guide to using branches for development (we probably only need main, release, scratch, and feature branches)

### Style

- Use auto-formatters and linters to maintain consistent code style.
  - For Python, we use black and pylint
  - For TypeScript, we use Prettier

### Tests

Where tests are available, **remember to run them before submitting a pull request(PR)**. Reviewers should check that tests pass before approving PRs for merge.

### Code reviews

Reviewers should:

- Verify that the pull request description includes:
  - A description of **what work was done**, and **why it was needed**. Usually this just means a link or reference to the relevant user story or Trello card.
  - A description of **how the developer knows their work is correct**. The reviewer can then use this to cross-check the code.
- Verify that the tests pass. At minimum, this means checking that the automated tests run by GitHub Actions passed. For integration and manual tests, see [How to test the product](#), [Appendix 5. Acceptance criteria](#), and [Appendix 6. Test cases](#).
- Verify that all test cases and acceptance criteria identified in the relevant feature kickoff have been satisfied.

At least one developer from each of the other two teams **must** approve the pull request before it can be merged.

## Deployment

See [How to deploy the product](#)

## Sprint retrospectives

Each sprint should end with a sprint retrospective.

If multiple teams are involved in the project, the team representatives should also conduct separate retrospectives to discuss team interactions.

## Releases

### Versioning

Most of this is defined by the university. In general though, we use [semantic versioning](#), and release by tagging releases on GitHub.

## Appendix 3. Core technologies

### Front-end

This project is primarily accessed via a website, which we're building with ReactJS and Styled Components. The development language is Typescript.

We are using Storybooks to automate documentation and previewing for developed UI components.

### Back-end

The front-end is supported by a Flask application, which interacts with Matminer on the user's behalf and surfaces required information in the UI.

The development language is Python. We're also using Pytest and Sphinx for running unit tests and generating documentation.

Key libraries we're using include Matminer and SQLAlchemy.

Our database is Postgres, and all back-end services are built as Docker containers. We have not yet selected a deployment technology, and are discussing options with the Melbourne University IT department.

## Appendix 4. Learning resources

### Backend

- Pytest: <https://docs.pytest.org/en/7.1.x/getting-started.html>
- SQLAlchemy: <https://docs.sqlalchemy.org/en/14/dialects/postgresql.html>
- Flask :
  - <https://flask.palletsprojects.com/en/2.2.x/>
  - <https://www.tutorialspoint.com/flask/index.htm>
  - [https://youtu.be/Z1RJmh\\_OqeA](https://youtu.be/Z1RJmh_OqeA)(Quick guide from setup, creating virtual env similar to the Developer.md , connecting to SQLAlchemy and the frontend)
  - <https://github.com/realpython/discover-flask>

### Accessing workflow related data

#### Workflow configurations data

WorkflowConfigurations class in workflow\_configurations.py is a data class in the backend containing all the configurations needed for executing each stage in the whole workflow. This class is created and saved to the database when a new user is created.

The choices made by users from the frontend are captured in this class and then saved in the Users table in the database.

When new choices are added in the frontend, the WorkflowConfigurations class should be extended as well to accommodate the changes, so choices made by the user can be consistent throughout the whole application.

save\_workflow and load\_workflow functions in workflow\_data\_handler.py can be used in the backend to save and load the WorkflowConfigurations object to and from the database.

#### Dataset for machine learning and plotting

In the first stage of the whole workflow, featurized data will be created after the stage execution. This data is then also saved in the Users table in the database. Using save\_featurized\_data in workflow\_data\_handler.py will allow you to add the dataframe to the database if there's any change to it.

To access the data for machine learning, or plotting, load\_featurized\_data in workflow\_data\_handler.py should be used to return the data.

## Frontend

- React: <https://reactjs.org/tutorial/tutorial.html>
- Style components: <https://www.robinwieruch.de/react-styled-components/>
- Testing:
  - [RTL Cheat sheet](#)
  - [RTL + Jest tutorial](#)
  - [Jest cheat sheet](#)
- TypeScript:
  - [TS Cheat sheet](#)
  - [TS tutorial](#)
  - [TS + React tutorial](#)
- Storybooks:
  - [Tutorial](#) (start from step 2)
  - [Official “how to write stories” guide](#)

## Docker

Docker is a tool for managing isolated runtime environments (called "containers"). You can install Docker by following the Docker guide (linked below).

You can think of it as a combination of:

- A way to run virtual machines
  - But it's much more efficient
  - And makes it easier to access host machine resources, for example network ports and filesystems
- A way to build custom virtual machines, just like make can be used to build binaries or applications
  - The build instructions are described by the Dockerfile file

Bear in mind though that **containers are not virtual machines** - that analogy is useful for describing an overview of what Docker does, but it is not accurate.

The two most important things you can do with Docker are:

- Build container **images**, for example by running: `docker image build -t flask-docker .`
  - A container *image* is like a compiled program: it does nothing unless it is run
  - Unlike a program though, it doesn't exist in an obvious way on the filesystem. The Docker runtime is responsible for managing images.

- `-t flask-docker` tells Docker to "tag" the image with the string "flask-docker". You can refer to Docker objects (including images) by their hash, but naming them makes it easier.
- The `.` tells Docker to use the Docker file in the current directory. You could provide a path to another directory instead.
- Run a container image, for example by running `docker run -d -p 80:5000 flask-docker`
  - Running an image produces an object called a "container" - this is like a process, just as an image is like a compiled program.
  - The `-d` argument tells Docker to "detach" the container - to run it in the background and not show its output.
  - The `-p 80:5000` argument tells Docker to map port 80 on the host machine to port 5000 inside the container. If you're building a Flask application (which by default listens for requests on port 5000), this means that you would access the application by navigating to port 80 on the host machine.
  - `flask-docker` is the tag of the image we want to run.

Once a container is running, it is given a tag by the Docker runtime. This tag is separate from the tag of the image.

You can view running containers by running `docker ps`.

When you're finished with a container, you can kill it by running `docker kill <container-tag>`

For more information, you can go through the [Docker getting started guide](#), or refer to their [reference documentation](#).

## Docker-compose

An alternative to Docker approach detailed above.

Docker-compose can automate the process of setting up & managing multiple containers. See [Install Docker Compose](#)

Before running the command below create a .env file in this directory and add the following lines

```
POSTGRES_USER="db_user"  
POSTGRES_PASS="somehardpassword"  
PGA_USER="user@gmail.com"  
PGA_PASS="anotherhardpassword"
```

Replace the text inside the inverted commas with appropriate email/password/usernames

Then create a `.pgpassfile` in the postgres-init folder and add the following line

```
pg_db:5432:ha_db:username:password
```

Replacing username and password with the postgres username and password chosen above

For security, ensure `.env` and `.pgpassfile` are added to `.gitignore`

Running docker-compose up -d in this folder will set up 3 containers with persistent storage

- Flask app as defined in **Dockerfile**, available at 127.0.0.1:5000.
- postgres database available at 127.0.0.1:5432.
- PGAdmin, a UI to easily manage the database 127.0.0.1:5050. [pgAdmin 4 Docs](#)

## Using a docker development container

While it's a little unusual, you can also use Docker to set up a **development container**. This is a container that you would use to create an isolated environment for software development - for example, if your main machine is a Windows machine, but you want to use a Linux development environment.

Visual Studio Code includes some tools for this:

<https://code.visualstudio.com/docs/remote/containers>

In this kind of setup, you would create a **separate container** for your development environment, and connect to it from Visual Studio Code.

From the VSCode terminal, you would be able to access the container via a shell just as you could on a normal Linux machine.

Inside the container you could install Docker and develop as normal. However, there could be some issues with accessing web services hosted inside the development container. It may be simpler to use the **Windows Subsystem for Linux** instead:

<https://code.visualstudio.com/docs/remote/wsl>

## Appendix 5. Acceptance criteria

STORY ID	USER STORY	GIVEN	WHEN	THEN
21	Select datasets from existing databases	I'm at the dataset selection step and I can see a list of datasets	I select one of the datasets and click on save	I should receive some feedback on whether the action is successful or not
23	Control job execution	I have a task in progress	I click in the start, view and cancel buttons	The running tasks can be started, viewed and canceled
32	Browse and select built-in featurizers	I'm at the featurizer selection step	I click on the dropdown menu	I see a list of all built-in featurizers with readable names
		I'm at the featurizer selection step and I can see a list of featurizers	I select one of the featurizers and click on save	I should receive some feedback on whether the action is successful or not
34	Browse built-in datasets	I'm at the dataset selection step	I click on the dropdown menu	I see a list of all built-in datasets with readable names
35	Export model selections, parameters, and data flows	I am logged into my account	I click on the export button	I can download a file that has all the data needed to recreate my current workflow state
41	View the columns of the saved dataset and select the column I want to featurize	I've selected a dataset and saved my selection	I click on the dropdown menu	I see a list of column names in the dropdown menu
		I'm at the data columns selection step and I see a list of column names	I select one of the data columns and clicked on save	I should receive some feedback on whether the action is successful or not
20	Create an account using single-sign on, restricted to the *.unimelb.edu.au domain	I have a registered account for use of this product	I update any relevant personal information	That updated information persists in the database and is linked to my account
			I make changes to my workflow	That workflow data can be saved in the database and linked to my account. Only I can access it

STORY ID	USER STORY	GIVEN	WHEN	THEN
			I delete my account	All linked data is deleted from the database. Others' data won't be affected
		I own a Unimelb account and would like to access the web application features	I click on the login button	I can sign in using my <a href="mailto:unimelb.edu.au">unimelb.edu.au</a> email as a form of authentication
		I am a first-time user and don't own a Unimelb account	I click on the login button	If I try to sign in using a non-Unimelb email I receive a notification that only <a href="mailto:unimelb.edu.au">unimelb.edu.au</a> emails are allowed
29	Save project specific data/checkpoints	I am logged into my account	I make changes to my workflow	That data can be saved to the database under my user account
35	Export model selections, parameters, and data flows	I am logged into my account	I click on the export button	I can download a file that has all the data needed to recreate my current workflow state
36	Import exported model selections, parameters, and data flows	I am logged into my account	I click on the import button	I can upload a previously exported workflow state
		I have uploaded a previously exported workflow state	The file passes all data integrity checks	My UI will match the state represented in the imported file
10	opt in to pro-user features	I am a pro user and would like to access my pro-user features	I click on the opt button	I jump into a new window with all pro-user features accessible.
23	Control job execution	I have a task in progress	I click on the start, view and cancel buttons	The running tasks can be started, viewed and canceled
33	Browse Machine Learning models	I have selected the target feature and I'm at the model selection step	I click on the dropdown menu	I should see a list of Machine Learning models with readable names
31	Select a Machine Learning model	I'm at the model selection step and I can see a list of Machine Learning models	I select one of the Machine Learning models and click on save	I should receive some feedback on whether the action is successful or not

STORY ID	USER STORY	GIVEN	WHEN	THEN
39	be able to select split ratio of data	I've selected a dataset and saved my selection	I type in a ratio of train and test data and click the continue button	I should receive some feedback on whether the action is successful or not
26	Browse and be able to change the hyperparameters used in the machine learning model	I successfully select one of the Machine Learning models	I receive the feedback that my selection is succeed	I should see some adjustable functionality for the hyperparameters of the model I selected
		I see some adjustable functionality for the hyperparameters of the model I selected	I adjust the values of these hyperparameters and click the continue button	I should receive some feedback on whether the action is successful or not
15	be able use additional ML models	I have selected the target feature and I'm at the model selection step	I click on the dropdown menu	I see enough Machine Learning models with readable names
7	Combine multiple models together	I have selected the target feature and I'm at the model selection step	I receive the successful feedback from feature selection	I should see multiple dropdown menus that allow me to select multiple Machine Learning models to combine
		I see multiple dropdown menus that allow me to select multiple Machine Learning models to combine	I select several different models	I should receive some feedback on whether the action is successful or not
40	Select features used for x and y axis	I've selected a dataset and saved my selection	I click on the dropdown menu	I see a list of column names in the dropdown menu
		I am at the x and y axis selection step and I see a list of column names	I select two of the data columns as x and y axis and clicked on execution	I should receive some feedback on whether the action is successful or not
9	View and plot the results of the model	I successfully select x and y axis used for plotting	I click the plot button	I can see plots about the selected columns

STORY ID	USER STORY	GIVEN	WHEN	THEN
8	Browse different regression models	I successfully select x and y axis used for plotting	I click on the dropdown menu of selecting regression models	I see several types of regression models
8	See different results of plotting graphs	I see several types of regression models	I choose one of the regression models and click on the plot button	I see a correct regression plot of my selection

## Appendix 6. Test cases

### Definitions

#### Standard setup

This is the minimum set of requirements for a user to log in and run ML jobs.

- Backend, Frontend, Databases, Job Queue, and ML Service are available
- Google API key has been registered
- Databases have been created

### Test cases

#### US30

<b>Test Type</b>	Functional
<b>Execution Type</b>	Manual
<b>Objective</b>	Users are able to view citations, authors, and help text for featurizers
<b>Setup</b>	Standard setup
<b>Execution Steps</b>	<ol style="list-style-type: none"><li>1. Log in using a Melbourne Uni student account</li><li>2. Select a featurizer from the featurizers dropdown menu</li><li>3. Click the "Featurizer Details" button</li><li>4. <b>Verify:</b><ol style="list-style-type: none"><li>1. <b>Citations</b> are shown for the selected featurizer</li><li>2. <b>Implementors</b> are shown for the selected featurizer</li><li>3. <b>Help text</b> is shown for the selected featurizer</li><li>4. A list of <b>generated labels</b> is shown for the featurizer</li></ol></li></ol>

<b>Notes</b>	A good featurizer for testing this is "Miedema"  There's no guarantee that every featurizer has all four of the items to be verified, but that one has all of them.  Remember to check the browser and backend logs for errors.
<b>Time constraint</b>	1m per tested featurizer

## US32

<b>Test Type</b>	Functional
<b>Execution Type</b>	Manual
<b>Objective</b>	Users are able to select a featurizer from a dropdown list
<b>Setup</b>	Standard setup; Use either PGAdmin or a psql connection to the database to verify changes
<b>Execution Steps</b>	<ol style="list-style-type: none"> <li>1. Log in using a Melbourne Uni student account</li> <li>2. <b>Verify:</b> The featurizers dropdown is displayed in step 1.3, and has content</li> <li>3. Select a featurizer, click "Save"</li> <li>4. <b>Verify:</b> <ol style="list-style-type: none"> <li>1. A "Your selection was successfully saved" message appears</li> <li>2. The featurizer selection is communicated to the backend and the database is updated</li> </ol> </li> </ol>
<b>Notes</b>	
<b>Time constraint</b>	2m per tested featurizer

**US34 and US21**

<b>Test Type</b>	Functional
<b>Execution Type</b>	Manual
<b>Objective</b>	The user can browse and select built-in datasets
<b>Setup</b>	Standard setup; Use either PGAdmin or a psql connection to the database to verify changes
<b>Execution Steps</b>	<ol style="list-style-type: none"> <li>1. Log in using a Melbourne Uni student account</li> <li>2. <b>Verify:</b> The datasets dropdown is displayed in step 1.1, and has content</li> <li>3. Select a dataset, click "Save"</li> <li>4. <b>Verify:</b> <ol style="list-style-type: none"> <li>1. A "Your selection was successfully saved" message appears</li> <li>2. The dataset selection is communicated to the backend and the database is updated</li> </ol> </li> </ol>
<b>Notes</b>	
<b>Time constraint</b>	2m per tested dataset

**US41**

<b>Test Type</b>	Functional
<b>Execution Type</b>	Manual
<b>Objective</b>	Users can select a column to featurize from the columns available in the dataset they have already selected

<b>Setup</b>	Standard setup, plus: <ul style="list-style-type: none"> <li>• Use either PGAdmin or a psql connection to the database to verify changes</li> <li>• Use a Python notebook to verify the list of columns that should be listed for a given dataset</li> </ul>
<b>Execution Steps</b>	<ol style="list-style-type: none"> <li>1. Log in using a Melbourne Uni student account</li> <li>2. Select a dataset from the step 1.1 dropdown, and click "Save"</li> <li>3. <b>Verify:</b> <ol style="list-style-type: none"> <li>1. A "Your selection was successfully saved" message appears</li> <li>2. The dataset columns dropdown is displayed in step 1.2, and has content</li> <li>3. The list of columns matches those listed by Matminer</li> </ol> </li> <li>4. Select a dataset column</li> <li>5. <b>Verify:</b> The dataset column selection is communicated to the backend and the database is updated</li> </ol>
<b>Notes</b>	
<b>Time constraint</b>	5m per tested dataset (to verify the columns match)

## Foundation

### Foundation TC01

<b>Test Type</b>	Functional
<b>Execution Type</b>	Manual
<b>Objective</b>	Users can step through the different stages of the workflow, and saved inputs are preserved.
<b>Setup</b>	Standard setup

<b>Execution Steps</b>	<ol style="list-style-type: none"> <li>1. Log in using a Melbourne Uni student account</li> <li>2. In step 1.1, select a dataset and click "Save"</li> <li>3. <b>Verify:</b> a "Your selection was successfully saved" message appears</li> <li>4. Step through to stage 2</li> <li>5. <b>Verify:</b> the output from step 3 is still visible</li> <li>6. Step through to stage 3</li> <li>7. <b>Verify:</b> the output from step 3 is still visible</li> <li>8. Step back to stage 1</li> <li>9. <b>Verify:</b> the dataset previously selected is still selected</li> </ol>
<b>Notes</b>	The dataset selection should only be preserved if the user clicked "Save"; if the selection was not saved it does not need to be preserved.
<b>Time constraint</b>	2m

### Foundation TC02

<b>Test Type</b>	Functional
<b>Execution Type</b>	Manual
<b>Objective</b>	Output messages are not overwritten by new messages
<b>Setup</b>	Standard setup
<b>Execution Steps</b>	<ol style="list-style-type: none"> <li>1. Log in using a Melbourne Uni student account</li> <li>2. In step 1.1, select a dataset and click "Save"</li> <li>3. <b>Verify:</b> a "Your selection was successfully saved" message appears</li> <li>4. Bring down the back-end systems</li> <li>5. Select a dataset column from the step 1.2 dropdown menu</li> <li>6. <b>Verify:</b> <ol style="list-style-type: none"> <li>1. An error message appears in the output pane</li> <li>2. The previous "Your selection was successfully saved" message is still shown and is not replaced.</li> </ol> </li> </ol>
<b>Notes</b>	The dataset selection should only be preserved if the user clicked "Save"; if the selection was not saved it does not need to be preserved.
<b>Time constraint</b>	2m

### US23

<b>Test Type</b>	Functional
------------------	------------

<b>Execution Type</b>	Manual
<b>Objective</b>	Users can execute a featurizer
<b>Setup</b>	Standard setup
<b>Execution Steps</b>	<ol style="list-style-type: none"> <li>1. Log in using a Melbourne Uni student account</li> <li>2. Select a dataset from the step 1.1 dropdown, and click "Save"</li> <li>3. <b>Verify:</b> <ol style="list-style-type: none"> <li>1. A "Your selection was successfully saved" message appears</li> <li>2. The dataset columns dropdown is displayed in step 1.2, and has content</li> <li>3. The list of columns matches those listed by Matminer</li> </ol> </li> <li>4. Select a dataset column from the step 1.2 dropdown, and click "Save"</li> <li>5. <b>Verify:</b> The dataset column selection is communicated to the backend and the database is updated</li> <li>6. Select a featurizer column from the step 1.3 dropdown, and click "Save"</li> <li>7. <b>Verify:</b> The featurizer selection is communicated to the backend and the database is updated</li> <li>8. Click "Run Stage 1"</li> <li>9. <b>Verify:</b> <ol style="list-style-type: none"> <li>1. The "Run Stage 1" button becomes disabled</li> <li>2. Job execution begins (Use Docker logs or Flower to verify this)</li> <li>3. Job execution ends successfully</li> <li>4. The "Run Stage 1" button becomes enabled again</li> <li>5. A success message appears in the output pane</li> </ol> </li> </ol>
<b>Notes</b>	<p>Depending on the combination of dataset, formula, and featurizer, this might not work because they're not all compatible. A combination that should work is:</p> <ul style="list-style-type: none"> <li>• Dataset: Elastic Tensor 2015</li> <li>• Column: Formula</li> <li>• Featurizer: StrToComposition</li> </ul>
<b>Time constraint</b>	12m

## US10: Login with the pro user(successful)

<b>Test Type:</b> Functional	<b>Execution type:</b> Manual
<b>Objective:</b>  Pro users are able to log in with a pro-user feature	
<b>Setup</b>  The databases have been created, and a google API key has been registered for the project	
<b>Pre-condition</b>  1. A registered user has a pro-user access	
<b>Notes:</b>  [1]: Click on Start exploring on the home page  *application pop-up a google SSO  [2]: When a google pop-up reveals, tick the "i am pro user".  *application shows ticked marked on "i am pro user"  [3]: Click sign in with google	
<b>Time constraint:</b>  Minimum: 10 seconds  Maximum: 1 minute	

## US 20

### US 20, TC01: create a new account (successful)

<b>Test Type:</b> Functional	<b>Execution type:</b> Manual
<b>Objective:</b>  Users are able to create an account using an SSO with his/her student account.	
<b>Setup</b>  The databases have been created, and a google API key has been registered for the project	
<b>Pre-condition</b>  1. The user does not have an account on the databases.	
<b>Notes:</b>  [1]: Click on Start exploring on the home page  *application pop-up a google SSO  [2]: When a google pop-up reveals, click on login using google, then it will create an account using the student email account.  *application redirects to the /workflow	
<b>Time constraint:</b>  Minimum: 10 seconds  Maximum: 1 minute	

**US 20, TC02: existing user login (successful)**

<b>Test Type:</b> Functional	<b>Execution type:</b> Manual
<b>Objective:</b>  Users are able to log on to an account using SSO with his/her student account.	
<b>Setup</b>  The databases have been created, and a google API key has been connected to the system	
<b>Pre-condition</b>  1. The user has an account on the databases.	
<b>Notes</b>  [1]: Click on Start exploring on the home page  *application pop-up a google SSO  [2]: When a google pop-up reveals, click on login using google, then log in using the student email account.  *application redirects to the /workflow	
<b>Time constraint:</b>  Minimum: 10 seconds  Maximum: 1 minute	

## US 29

### US 29, TC01: Save selected dataset (successful)

<b>Test Type:</b> Functional	<b>Execution type:</b> Manual
<b>Objective:</b>  The dataset user selected is saved successfully.	
<b>Setup</b>  The database has been established in the back end.	
<b>Pre-condition</b>  1. The user has logged in.  2. The user selected a dataset at step 1.1	
<b>Notes:</b>  [1]: Click on the save button at step 1.1  * Application shows “Step 1.1 output:”  * Application shows a green box where a notice saying, “Your selection was successfully saved”	
<b>Time constraint:</b>  Minimum: 1 second  Maximum: 10 seconds	

### US 29, TC02: Save selected dataset column (successful)

<b>Test Type:</b> Functional	<b>Execution type:</b> Manual
<b>Objective:</b>  The dataset column user selected is saved successfully.	

**Setup**

The database has been established in the back end.

**Pre-condition**

1. The user saved the dataset successfully.
2. The user selected a dataset column at step 1.2

**Notes**

[1]: Click on the save button at step 1.2

\* Application shows “Step 1.2 output:”

\* Application shows a green box where a notice saying, “Your selection was successfully saved”

**Time constraint:**

Minimum: 1 second

Maximum: 10 seconds

**US 29, TC03: Save selected featurizer (successful)**

<b>Test Type:</b> Functional	<b>Execution type:</b> Manual
<b>Objective:</b>  The featurizer user selected is saved successfully.	
<b>Setup</b>  The database has been established in the back end.	
<b>Pre-condition</b>	
<ol style="list-style-type: none"><li>1. The user saved the dataset column successfully.</li><li>2. The user selected a featurizer at step 1.3</li></ol>	

**Notes**

[1]: Click on the save button at step 1.3

\* Application shows “Step 1.3 output:”

\* Application shows a green box where a notice saying, “Your selection was successfully saved”

**Time constraint:**

Minimum: 1 second

Maximum: 10 seconds

**US 29, TC04: Save selected machine learning model (successful)**

<b>Test Type:</b> Functional	<b>Execution type:</b> Manual
<b>Objective:</b>  The machine learning model user selected is saved successfully.	
<b>Setup</b>  The database has been established in the back end.	
<b>Pre-condition</b>  1. The user saved the featurizer successfully.  2. The user selected a machine learning model at step 2.1	
<b>Notes</b>  [1]: Click on the save button at step 2.1  * Application shows “Step 2.1 output:”  * Application shows a green box where a notice saying, “Your selection was successfully saved”	

**Time constraint:**

Minimum: 1 second

Maximum: 10 seconds

**US 29, TC05: Save selected property (successful)**

<b>Test Type:</b> Functional	<b>Execution type:</b> Manual
<b>Objective:</b>  The property user selected is saved successfully.	
<b>Setup</b>  The database has been established in the back end.	
<b>Pre-condition</b>  1. The user saved the machine learning model successfully.  2. The user selected property at step 2.2	
<b>Notes</b>  [1]: Click on the save button at step 2.2  * Application shows “Step 2.2 output.”  * Application shows a green box where a notice saying, “Your selection was successfully saved”	
<b>Time constraint:</b>  Minimum: 1 second  Maximum: 10 seconds	

**US 29, TC06: Save selected x-axis feature (successful)**

<b>Test Type:</b> Functional	<b>Execution type:</b> Manual
<b>Objective:</b>  The x-axis feature user selected is saved successfully.	
<b>Setup</b>  The database has been established in the back end.	
<b>Pre-condition</b>  1. The user saved the property successfully.  2. The user selected an x-axis feature at step 3.1	
<b>Notes</b>  [1]: Click on the save button at step 3.1  * Application shows “Step 3.1 output:”  * Application shows a green box where a notice saying, “Your selection was successfully saved”	
<b>Time constraint:</b>  Minimum: 1 second  Maximum: 10 seconds	

**US 29, TC07: Save selected y-axis feature (successful)**

<b>Test Type:</b> Functional	<b>Execution type:</b> Manual
<b>Objective:</b>  The y-axis feature user selected is saved successfully.	

<b>Setup</b>  The database has been established in the back end.
<b>Pre-condition</b>  1. The user saved the x-axis feature successfully.  2. The user selected a y-axis feature at step 3.2
<b>Notes</b>  [1]: Click on the save button at step 3.2  * Application shows “Step 3.2 output:”  * Application shows a green box where a notice saying, “Your selection was successfully saved”
<b>Time constraint:</b>  Minimum: 1 second  Maximum: 10 seconds

### US 29, TC08: Save selected regression function (successful)

<b>Test Type:</b>  Functional	<b>Execution type:</b>  Manual
<b>Objective:</b>  The regression function user selected is saved successfully.	
<b>Setup</b>  The database has been established in the back end.	
<b>Pre-condition</b>  1. The user saved the y-axis feature successfully.  2. The user selected a regression function at step 3.4	

**Notes**

[1]: Click on the save button at step 3.4

\* Application shows “Step 3.4 output:”

\* Application shows a green box where a notice saying, “Your selection was successfully saved”

**Time constraint:**

Minimum: 1 second

Maximum: 10 seconds

## **US35, TC01: Export workflow (Successful)**

<b>Test Type:</b> Functional	<b>Execution type:</b> Manual
<b>Objective:</b>  Exporting current workflow into .json state	
<b>Setup</b>  The databases have been created, a google API key has been registered for the project, and the user has logged on.	
<b>Pre-condition</b>  1. A registered and logged-in user 2. The user has selected workflow step 1. Prepare Data and 2. Machine learning step that he want to save as a json file	
<b>Notes:</b>  [1]: Click the export workflow  *Application downloads the JSON step of the workflow into the download folder user desired	

**Time constraint:**

Minimum: 10 seconds

Maximum: 1 minute

## **US36, TC01: Import workflow (successful)**

<b>Test Type:</b> Functional	<b>Execution type:</b> Manual
<b>Objective:</b>  Importing a .json file into the workflow	
<b>Setup</b>  The databases have been created, a google API key has been registered for the project, and the user has logged on.	
<b>Pre-condition</b>  1. A registered and logged-in user	
<b>Notes:</b>  [1]: Click import workflow  *Application open a windows folder for user to upload their saved workflow in .json format  [2]: Click the desired .json files  *Application will have all the .json	
<b>Time constraint:</b>  Minimum: 10 seconds  Maximum: 1 minute	

## **US 31: As a general user, I want to be able to select a Machine Learning model.**

### **US31, TC 01: Select a model (successful)**

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b>  Verify if a machine learning model is correctly selected.	
<b>Setup:</b>  The IO stream has been established between the front end and the back end.	
<b>Pre-Condition:</b>  1. The user has inputted the data or selected the data.  2. The user has selected the target feature.	
<b>Notes:</b>  [1] Select the machine learning model in the interface.  Must select a choice in the given place.  Do not select the models that are out of scope in the given choices (Linear regression model and Random forest model).  [2] Submit the information.  * Application starts machine learning using the given input dataset and selected feature.	
<b>Time constraint:</b>  Minimum: 10 min  Maximum: 30 min	

**US31, TC 02: Select a model (unsuccessful)**

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b>	
Verify if a machine learning model is correctly selected.	
<b>Setup:</b>	
The IO stream has been established between the front end and the back end.	
<b>Pre-Condition:</b>	
<ol style="list-style-type: none"><li>1. The user has inputted the data or selected the data.</li><li>2. The user has selected the target feature.</li></ol>	
<b>Notes:</b>  [1] Try not to select the machine learning model in the interface.  [2] Submit the information.  * Application cannot learn without a selected machine learning model.	
<b>Time constraint:</b>  Minimum: 10 min  Maximum: 30 min	

**US 40: As a general user, I want to select features used for the x-axis and y-axis.**

**US40, TC 01: Select an x-axis feature (successful)**

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b>  Verify if a feature for the x-axis is correctly selected.	
<b>Setup:</b>  The IO stream has been established between the front end and the back end.	
<b>Pre-Condition:</b>  1. The user has inputted the data or selected the data.  2. The user has executed the machine learning model.	
<b>Notes:</b>  [1] Select the feature for the x-axis in the interface.  Must select a choice in the given place.  Do not select the features that are out of scope in the given choices (The features are displayed in the interface).  [2] Click the Execute button.  * Application receives the x-axis feature value.	
<b>Time constraint:</b>  Minimum: 1 min  Maximum: 5 min	

### US40, TC 02: Select an x-axis feature (unsuccessful)

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b>	
Verify if a feature for the x-axis is correctly selected.	
<b>Setup:</b>	
The IO stream has been established between the front end and the back end.	
<b>Pre-Condition:</b>	
1. The user has inputted the data or selected the data.  2. The user has executed the machine learning model.	
<b>Notes:</b>  [1] Try not to select a feature for the x-axis in the interface.  [2] Click the Execute button.  * Application cannot learn without a selected x-axis feature value.	
<b>Time constraint:</b>  Minimum: 1 min  Maximum: 5 min	

### US40, TC 03: Select a y-axis feature (successful)

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
---------------------------------	----------------------------------

**Objective:**

Verify if a feature for the y-axis is correctly selected.

**Setup:**

The IO stream has been established between the front end and the back end.

**Pre-Condition:**

1. The user has inputted the data or selected the data.
2. The user has executed the machine learning model.

**Notes:**

[1] Select the feature for the y-axis in the interface.

Must select a choice in the given place.

Do not select the features that are out of scope in the given choices (The features are displayed in the interface).

[2] Click the Execute button.

\* Application receives the y-axis feature value.

**Time constraint:**

Minimum: 1 min

Maximum: 5 min

**US40, TC 04: Select a y-axis feature (unsuccessful)**

<b>Test Type:</b>	<b>Execution Type:</b>
Functional	Manual

**Objective:**

Verify if a feature for the y-axis is correctly selected.

**Setup:**

The IO stream has been established between the front end and the back end.

**Pre-Condition:**

1. The user has inputted the data or selected the data.
2. The user has executed the machine learning model.

**Notes:**

[1] Try not to select a feature for the y-axis in the interface.

[2] Click the Execute button.

\* Application cannot learn without a selected y-axis feature value.

**Time constraint:**

Minimum: 1 min

Maximum: 5 min

**US 09: As a general user, I want to be able to view and plot the results of the model.**

**US09, TC 01: Plot the diagram (successful)**

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b>  Verify if a diagram can be displayed correctly.	
<b>Setup:</b>  The IO stream has been established between the front end and the back end.	
<b>Pre-Condition:</b>  1. The user has inputted the data or selected the data.  2. The user has executed the machine learning model.  3. The user has selected features for the x-axis and the y-axis.	
<b>Notes:</b>  [1] Click the Plot button.  Must select an x-axis and the y-axis feature value before plotting the diagram.  * Application receives the requests and plots the diagram.	
<b>Time constraint:</b>  Minimum: 1 min  Maximum: 10 min	

### US09, TC 02: Plot the diagram (unsuccessful)

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b>  Verify if a diagram can be displayed correctly.	
<b>Setup:</b>  The IO stream has been established between the front end and the back end.	
<b>Pre-Condition:</b>  1. The user has inputted the data or selected the data.  2. The user has executed the machine learning model.	
<b>Notes:</b>  [1] The required choices are not selected:  [1.1] Not select the x-axis feature.  [1.2] Not select the y-axis feature.  [2] Click the Plot button.  * Application cannot plot the diagram without the x-axis and y-axis feature values.	
<b>Time constraint:</b>  Minimum: 1 min  Maximum: 10 min	

### US 08: As a student, I want to use different types of plotting graphs

### US08, TC 01: Select a type of regression model (successful)

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b>  Verify if a regression model is correctly selected.	
<b>Setup:</b>  The IO stream has been established between the front end and the back end.	
<b>Pre-Condition:</b>  1. The user has inputted the data or selected the data.  2. The user has executed the machine learning model.	
<b>Notes:</b>  [1] Select the regression model in the interface.  Must select a choice in the given place.  Do not select the regression models that are out of scope in the given choices (The regression model choices are displayed in the interface).  [2] Click the Execute button.  * Application receives the regression model.	
<b>Time constraint:</b>  Minimum: 1 min  Maximum: 5 min	

### US08, TC 02: Select a type of regression model (unsuccessful)

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
<b>Objective:</b>  Verify if a regression model is correctly selected.	
<b>Setup:</b>  The IO stream has been established between the front end and the back end.	
<b>Pre-Condition:</b>  1. The user has inputted the data or selected the data.  2. The user has executed the machine learning model.	
<b>Notes:</b>  [1] Try not to select a regression model in the interface.  [2] Click the Execute button.  * Application cannot execute without a selected regression model value.	
<b>Time constraint:</b>  Minimum: 1 min  Maximum: 5 min	

### US08, TC 03: Create the regression plot (successful)

<b>Test Type:</b> Functional	<b>Execution Type:</b> Manual
---------------------------------	----------------------------------

**Objective:**

Verify if a regression plot can be displayed correctly.

**Setup:**

The IO stream has been established between the front end and the back end.

**Pre-Condition:**

1. The user has inputted the data or selected the data.
2. The user has executed the machine learning model.
3. The user has selected features for the x-axis and the y-axis.
4. The user has selected the regression model.

**Notes:**

[1] Click the Plot button.

Must select an x-axis, the y-axis feature value, and a regression model before plotting the diagram.

\* Application receives the requests and plots the diagram.

**Time constraint:**

Minimum: 1 min

Maximum: 10 min

**US08, TC 04: Create the regression plot (unsuccessful)**

Test Type:	Execution Type:
Functional	Manual

**Objective:**

Verify if a regression plot can be displayed correctly.

**Setup:**

The IO stream has been established between the front end and the back end.

**Pre-Condition:**

1. The user has inputted the data or selected the data.
2. The user has executed the machine learning model.

**Notes:**

[1] The required choices are not selected:

- [1.1] Not select the x-axis feature.
- [1.2] Not select the y-axis feature.
- [1.3] Not select the regression model.

[2] Click the Plot button.

\* Application cannot plot the diagram without filling in the required choices.

**Time constraint:**

Minimum: 1 min

Maximum: 10 min

# Appendix 7. Set-up SSO authorisation guide

## 1. Background information

The first version of this application is set-up to be only accessible via SSO (Single SignOn) login restricted unimelb.edu.au email accounts.

Emails with unimelb.edu.au domain are either provided by:

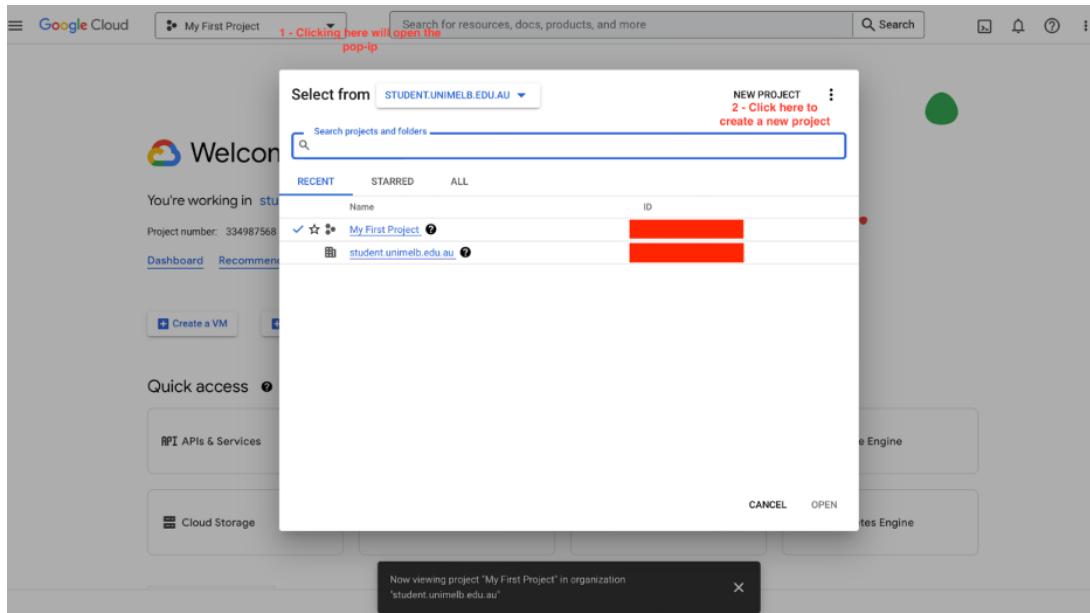
- Students accounts: Gmail
- Staff accounts: Microsoft

In order to be able to use Gmail and Microsoft SSO authentication the following steps must be done:

1. Create a client ID in Google Cloud Console and Microsoft Portal Azure and associate it with Hacking Materials Applications.
2. Client ID is a unique identifier associated with an application that assists with client and server [OAuth 2.0 authentication](#).
3. Once the client ID is created plug it in specific places in the source code **before deploying the application**.

## 2. Create a client id in the Google Cloud Console

To get a client ID from Google, create a google account, go over to your [Google Cloud Console](#) and create a new project.



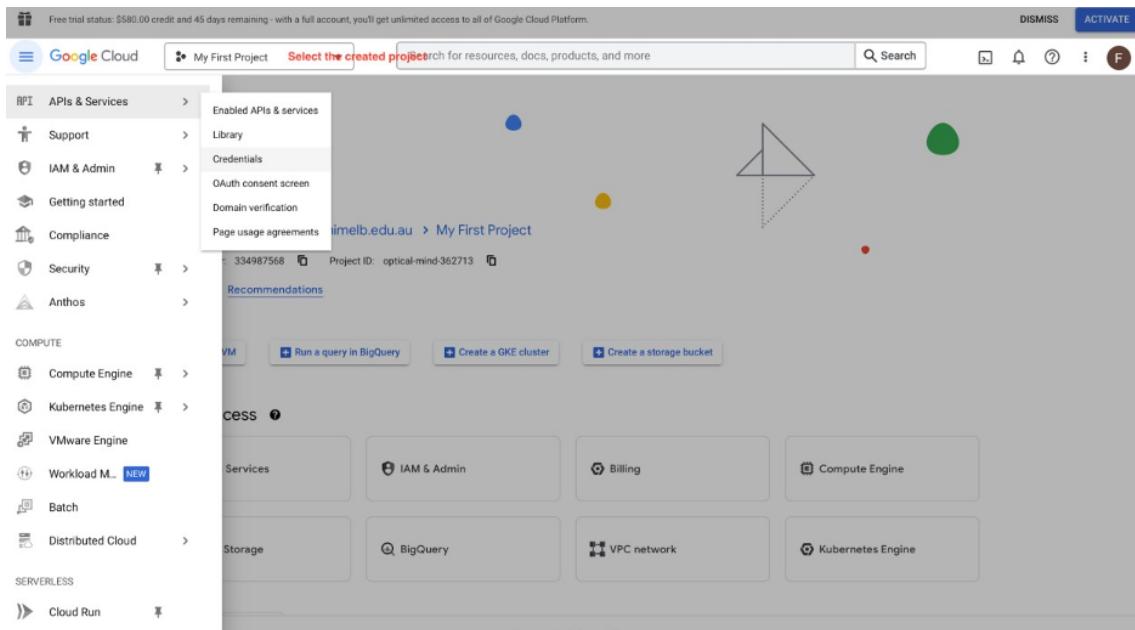
The screenshot shows the 'New Project' page in the Google Cloud console. At the top left is the Google Cloud logo. To its right is a search bar with the placeholder 'Search for resources, docs, products, a...'. Below the search bar, the text 'New Project' is displayed. A callout box contains a warning message: '⚠ You have 24 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)' and a 'MANAGE QUOTAS' link. The main form fields include:

- Project name \***: A text input field containing 'My Project 7933' with a question mark icon in the top right corner.
- Organization \***: A dropdown menu showing 'student.unimelb.edu.au' with a question mark icon in the top right corner. Below it, a note says 'Select an organization to attach it to a project. This selection can't be changed later.'
- Location \***: A text input field showing 'student.unimelb.edu.au' with a 'BROWSE' button to its right. Below it, a note says 'Parent organization or folder'.

At the bottom of the form are two buttons: a blue 'CREATE' button on the left and a 'CANCEL' button on the right.

After creating the project. Go back to the console, select the project in the dropdown list, from the left side menu select APIs & Services > Credentials

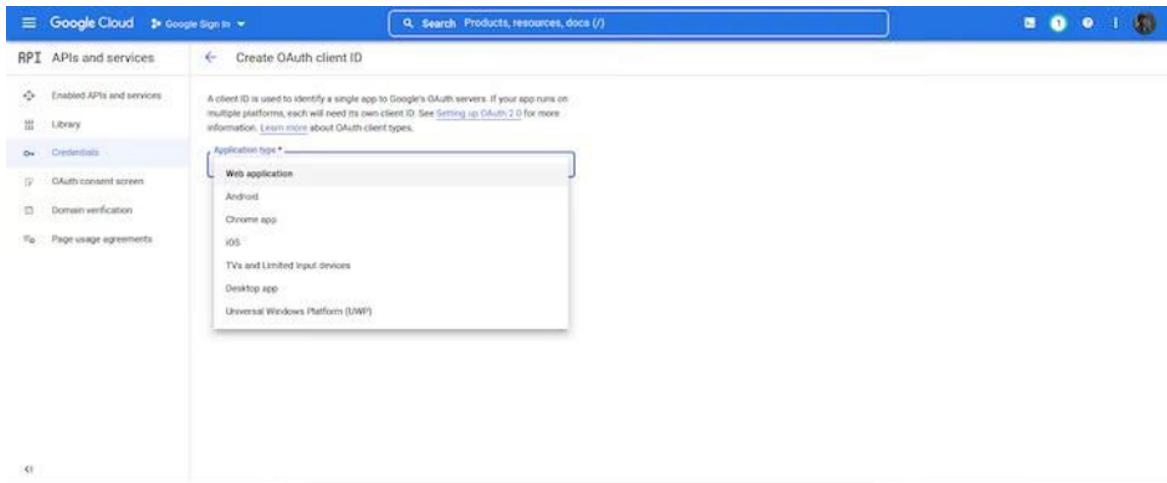
## COMP90082 2022 SM2 - Hacking Materials User Interface (HA) Project



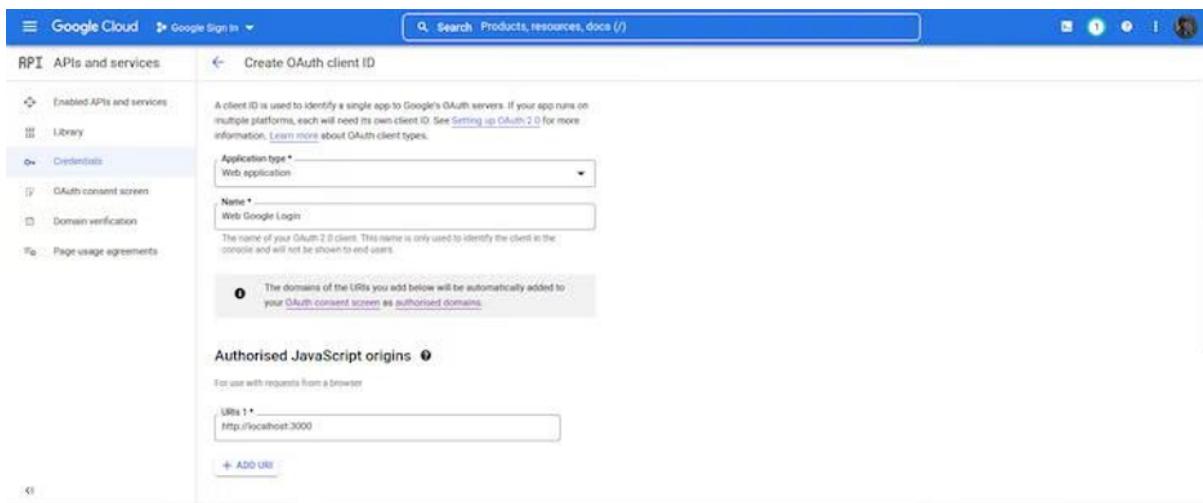
Click + create credentials > Oauth client ID

A screenshot of the "Credentials" section under "API &amp; Services". The left sidebar has "Enabled APIs and services", "Library", "Credentials" (which is selected), "OAuth consent screen", "Domain verification", and "Page usage agreements". The main content area shows "Create credentials to access your project". It lists "API key", "OAuth client ID", and "Service account". Under "OAuth 2.0 Client I", it says "Help me choose" and "Service Accounts". A table lists "Client ID" and "Actions". The URL in the address bar is "https://console.cloud.google.com/apis/credentials/oauthclient/list?project=optical-mind-362713&amp;documentId=355295".

For Application Type select Web application.



Next, we will choose a name for our client ID. This name is simply used to note or define the particular ID we are creating. So for example, if we are creating web, iOS, and Android IDs, we can include “Web ID,” “Android ID,” “iOS ID,” and so on in their names to differentiate them.



Next, we will also be adding two types of URLs: authorized JavaScript origins and authorized redirect URLs.

The authorized JavaScript origins URL is the URL from which your application is originating the login. In this case we originate the login from the home page. Therefore if the hosting address is `http://hackingmaterials.live`, this should be the value to be filled here.

The authorized redirect URL is the link that Google will redirect the user to after the login is successful. In this case we configured it to take to the original link.

The screenshot shows the 'Client ID for Web application' configuration screen. On the left sidebar, 'Credentials' is selected under 'APIs & Services'. The main area contains two sections: 'Authorized JavaScript origins' and 'Authorized redirect URIs'. Both sections have three input fields each, showing 'http://localhost', 'http://localhost:3000', and 'https://hackingmaterials.live'. Below these sections is a note: 'Note: It may take 5 minutes to a few hours for settings to take effect'. At the bottom are 'SAVE' and 'CANCEL' buttons.

Finally, save, click on the “Create” button to create your web client ID. You will be taken back to the homepage, where you will see your newly created credential. Click on the copy icon to copy your new web client ID.

The screenshot shows the 'Credentials' page in Google Cloud. Under 'OAuth 2.0 Client IDs', there is one entry named 'Web client 1' with a creation date of 'Sep 17, 2022' and type 'Web application'. The 'Client ID' field, which contains the value '334987568-1kkvpu4...', is highlighted with a red box. Other sections like 'API Keys' and 'Service Accounts' are also visible.

### 3. “PLUG” the Google client ID in the frontend source code.

In the FRONTEND folder go to .env file and replace `copy_google_client_id_here` with the client ID created in the previous step.

```

REACT_APP_GOOGLE_CLIENT_ID=copy_google_client_id_here
REACT_APP_MSAL_CLIENT_ID=copy_microsoft_client_id_here
REACT_APP_BACKEND_API_URI=''

```

#### 4. “PLUG” the Google client ID in the backend source code

In the BACKEND folder go to .env file and replace `copy_google_client_id_here` with the same client ID.

```

POSTGRES_USER="db_user"
POSTGRES_PASS="passpass"
PGA_USER="hidarae@mail.com"
PGA_PASS="passpass"
REDIS_PASSWORD="passpass"
FLASK_SECRET=608dc6121deb65d6ee22c8b10c005d378022bcc1df9b52db439ab87fc121ba1
REACT_APP_GOOGLE_CLIENT_ID=copy_google_client_id_here
POSTGRES_URL="postgresql://${POSTGRES_USER}:${POSTGRES_PASS}@localhost:5432/ha_db"
CELERY_BROKER_URL="redis://:${REDIS_PASSWORD}@localhost:6379/0"
CELERY_RESULT_BACKEND="redis://:${REDIS_PASSWORD}@localhost:6379/0"
STR_CONN="postgresql://${POSTGRES_USER}:${POSTGRES_PASS}@localhost:5432/ha_db"
REACT_APP_MSAL_CLIENT_ID=copy_microsoft_client_id_here

```

#### 5. Create a client id from Microsoft Azure’s Cloud Portal

After creating a microsoft email account, after signed-in got to <https://portal.azure.com/#home>, select App registrations.

The screenshot shows the Microsoft Azure homepage. At the top, there's a search bar and a user profile icon. Below the header, a "Welcome to Azure!" message is displayed, followed by a note about free trials and student benefits. There are three main service tiles: "Start with an Azure free trial" (blue key icon), "Manage Azure Active Directory" (shield and server icon), and "Access student benefits" (pen and paper icon). Below these are sections for "Azure services" (with icons for Create a resource, App registrations, App Services, Quickstart Center, Virtual machines, Storage accounts, SQL databases, Azure Cosmos DB, Kubernetes services, and More services) and "Resources" (Recent and Favorite tabs, Name, Type, Last Viewed columns).

1. Enter a **Name** for your application. Users of your app might see this name, and you can change it later.
2. Choose the **Supported account types (Accounts in any organizational directory to restrict to org accounts such as unimelb.edu.au)** for the application. Choose Single-page application (SPA) and register a Redirect URI.
3. Select **Register** to create the app registration.

Home > App registrations >

### Register an application

**⚠️** This application will not be associated with any directory and will be subject to limitations. You should not create production apps outside of a directory.

**\* Name**  
The user-facing display name for this application (this can be changed later).

**Supported account types**

Who can use this application or access this API?

Accounts in any organizational directory (Any Azure AD directory - Multitenant)

Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

Personal Microsoft accounts only

[Help me choose...](#)

**Redirect URI (optional)**  
We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Single-page application (SPA)

By proceeding, you agree to the Microsoft Platform Policies [»](#)

**Register**

Once registered, select App Registrations and select the application created.

The screenshot shows the 'App registrations' section of the Azure portal. A search bar at the top contains the text 'ha-materials-react'. Below it, a table lists the application details:

Display name	Application (client) ID	Created on	Certificates & secrets
ha-materials-react	c0bb6b7e-be8b-4466-81e7-55f00ba4c345	10/23/2022	-

On the application dashboard you can find the Application (client) ID.

The screenshot shows the 'ha-materials-react' application dashboard under the 'Manage' section. The 'Overview' tab is selected. In the 'Essentials' section, the 'Application (client) ID' field is highlighted with a red box. Other fields shown include 'Display name' (ha-materials-react), 'Object ID' (9aa5be53-bac7-43d1-a2a2-b73e75923221), and 'Directory (tenant) ID' (fbcdcf31-a31e-4b4a-93e4-5f571e91255a). The 'Client credentials' and 'Redirect URIs' sections are also visible.

(optional) In case you need to change the redirect URI, it can be done under the **Authentication** tab.

Home > App registrations > ha-materials-react

ha-materials-react | Authentication

Search Got feedback?

Overview Quickstart Integration assistant

Manage

Branding & properties

Authentication

Certificates & secrets

Token configuration

API permissions

Expose an API

Owners

Manifest

Support + Troubleshooting

Troubleshooting

New support request

Single-page application

Redirect URIs

The URLs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating or signing out users. The redirect URI you send in the request to the login server should match one listed here. Also referred to as reply URLs. [Learn more about Redirect URLs and their restrictions](#)

https://localhost http://localhost

Add URI

Grant types

Your Redirect URI is eligible for the Authorization Code Flow with PKCE.

Front-channel logout URL

This is where we send a request to have the application clear the user's session data. This is required for single sign-out to work correctly.

https://localhost

Implicit grant and hybrid flows

Request a token directly from the authorization endpoint. If the application has a single-page architecture (SPA) and doesn't use the authorization code flow, or if it invokes a web API via JavaScript, select both access tokens and ID tokens. For ASP.NET Core web apps and other web apps that use hybrid authentication, select only ID tokens. [Learn more about tokens](#).

Select the tokens you would like to be issued by the authorization endpoint:

Access tokens (used for implicit flows)

Save Discard

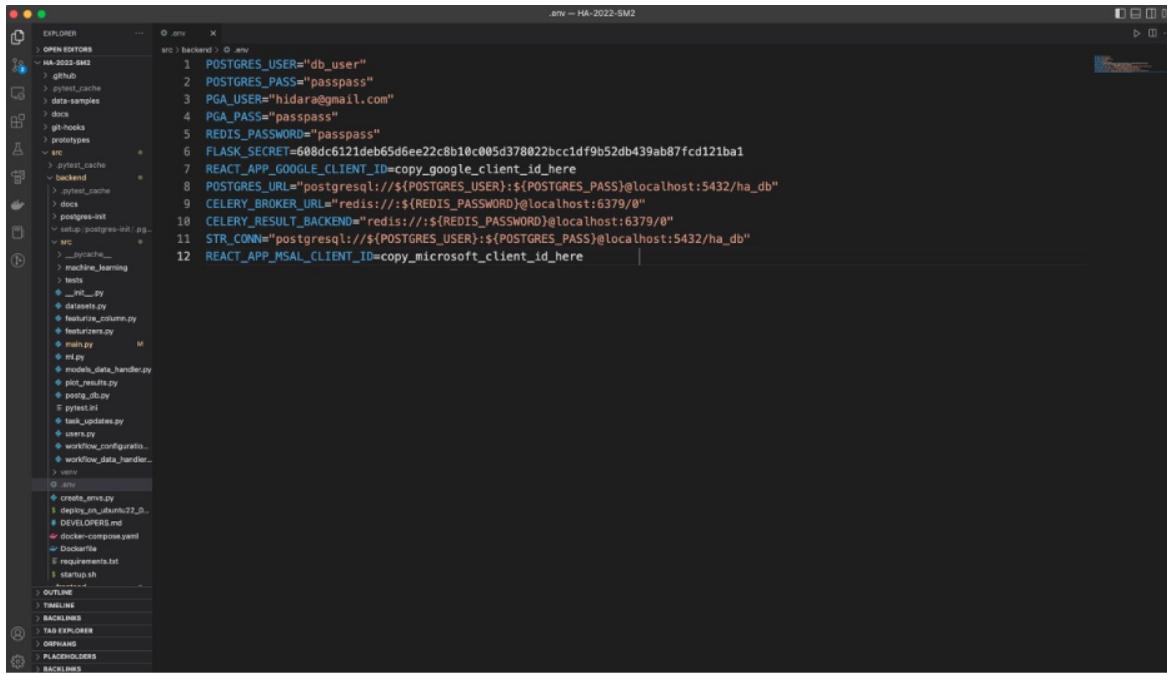
## 6. “PLUG” the Microsoft client ID in the frontend source code.

In the FRONTEND folder go to the `.env` file and replace `copy_microsoft_client_id_here` with the client ID created in the previous step.

```
REACT_APP_GOOGLE_CLIENT_ID=copy_google_client_id_here
REACT_APP_MSAL_CLIENT_ID=copy_microsoft_client_id_here
REACT_APP_BACKEND_API_URI=''
```

## 7. “PLUG” the Microsoft client ID in the backend source code

In the BACKEND folder go to .env file and replace **copy\_microsoft\_client\_id\_here** with the same client ID.



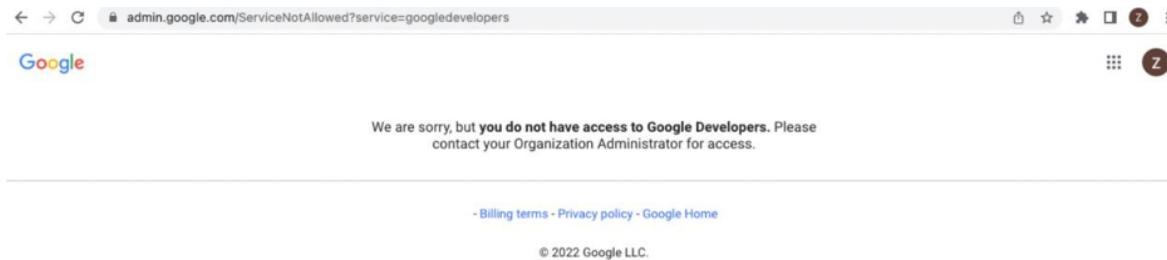
```

.env -- HA-2022-SM2
src > backend > .env
1 POSTGRES_USER="db_user"
2 POSTGRES_PASS="passpass"
3 PG_A_USER="hidara@gmail.com"
4 PG_A_PASS="passpass"
5 REDIS_PASSWORD="passpass"
6 FLASK_SECRET=608dc6121deb65d6ee22c8b10c005d378022bcc1df9b52db439ab87fc121ba1
7 REACT_APP_GOOGLE_CLIENT_ID=copy_google_client_id_here
8 POSTGRES_URL="postgresql://${POSTGRES_USER}:${POSTGRES_PASS}@localhost:5432/ha_db"
9 CELERY_BROKER_URL="redis://:${REDIS_PASSWORD}@localhost:6379/0"
10 CELERY_RESULT_BACKEND="redis://:${REDIS_PASSWORD}@localhost:6379/0"
11 STR_CONN="postgresql://${POSTGRES_USER}:${POSTGRES_PASS}@localhost:5432/ha_db"
12 REACT_APP_MSAL_CLIENT_ID=copy_microsoft_client_id_here

```

## 8. If you get the following error pages when creating Google Client Id or Microsoft Id

If you get one of the below pages while creating either the Google client ID or Microsoft client ID while using an organizational account, you might need to contact the administrator to set the permissions accordingly.



Microsoft Azure Search resources, services, and docs (G+) zhaqiw@student.unim... THE UNIVERSITY OF MELBOURNE

Home > You do not have access X



You do not have access

Your administrator has disabled the App registrations experience in the Azure portal. You can still register or manage applications using PowerShell or another client such as Visual Studio.

[Learn more about restricted access](#)

Summary	
Session ID 62bb33bc37b341f5a8fc43ca51fd3fe	Resource ID Not available
Extension Microsoft_AAD_RegisteredApps	Content ApplicationsListBlade
Error code 403	

# Appendix 8. Frontend workspace structure

## PROPOSAL - 1<sup>st</sup> September 2022

The proposed structure would follow this rough directory tree:

```
|_assets  
|_src  
  |_components  
    |_exampleComponent  
      |_examples.tsx  
      |_index.tsx  
      |_test.tsx  
      |_styled.tsx  
    |_dropdownSelectStepType  
      |_examples.tsx  
      |_index.tsx  
      |_test.tsx  
      |_styled.tsx  
    ...  
  |_steps  
    |_datasetSelection // (e.g.)  
      |_index.tsx  
      |_test.tsx  
      |_HelpModal  
        |_index.tsx  
        |_styled.tsx  
    ...  
  ...  
  ...  
  |_sections  
    |_appHeader  
      |_index.tsx  
    ...  
    |_appBody  
      |_InputPanel  
        |_index.tsx  
      ...  
      |_ViewingWinow  
        |_index.tsx  
      ...  
      |_index.tsx  
    |_appFooter // amendment suggested by Felipe  
    ...  
  |_App.tsx  
  ...  
 |_package.json
```

\_README.md

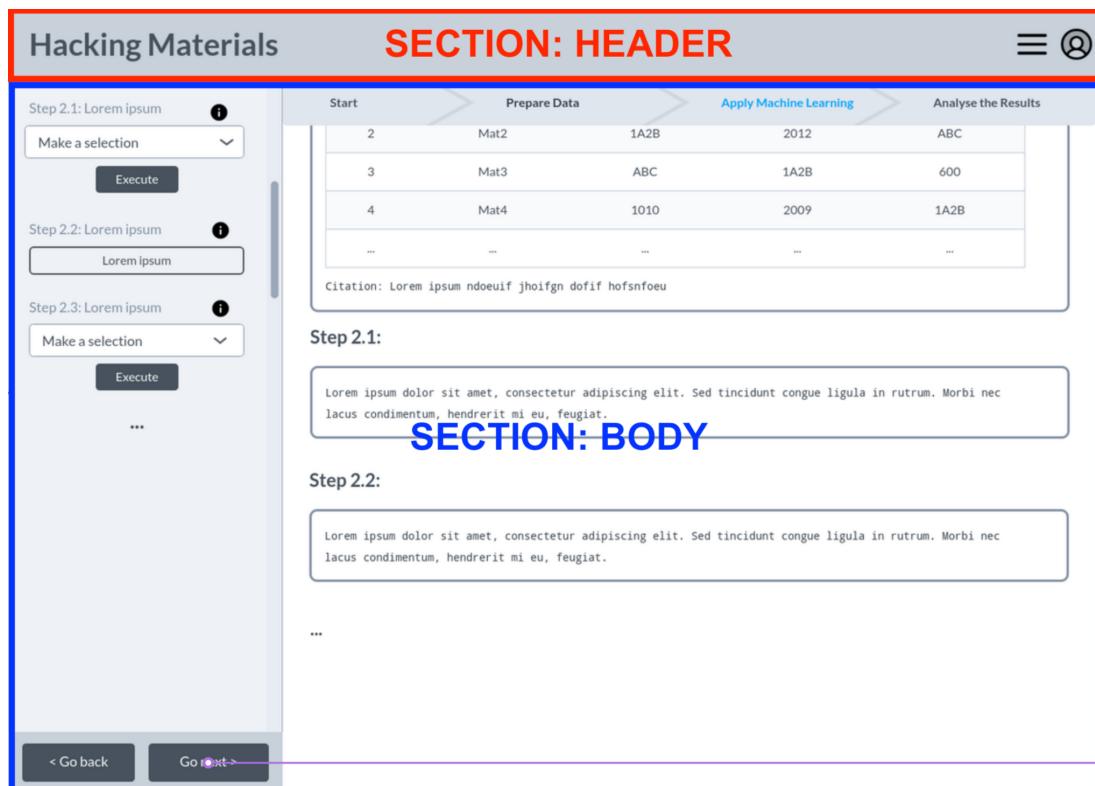
...

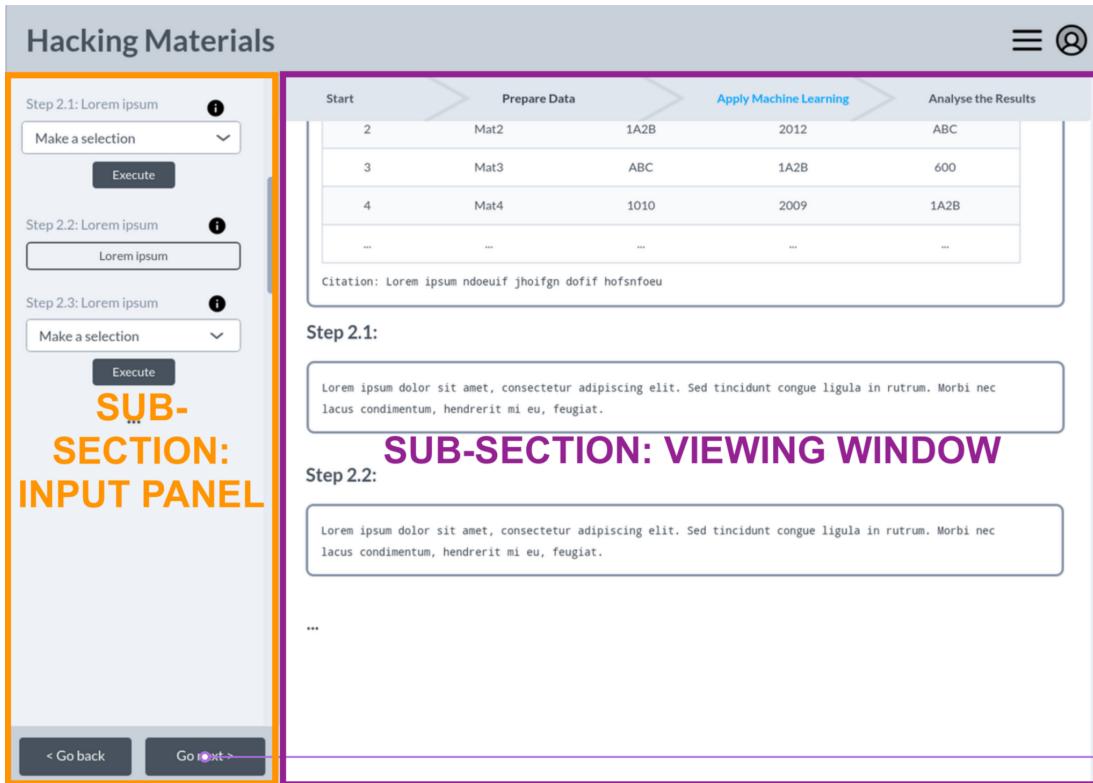
The main ideas of this are as follows:

## Sections

By referring to the [low-fidelity prototype](#) created earlier in the project, we divide the main application page into 2 main sections:

- Header: the top bar, which does not need to have context of what stage the user is up to and what's happening at any given point.
- Body: Includes 2 subsections that both need to know which stage the user is at (e.g. "Pre-process data" or "Apply machine learning"):
  - Left-side panel: named in the structure as InputPanel. Example code for this panel and how it shows the workflow steps is included in the sample code section below.
  - Main window on the right: named in the structure as ViewingWindow

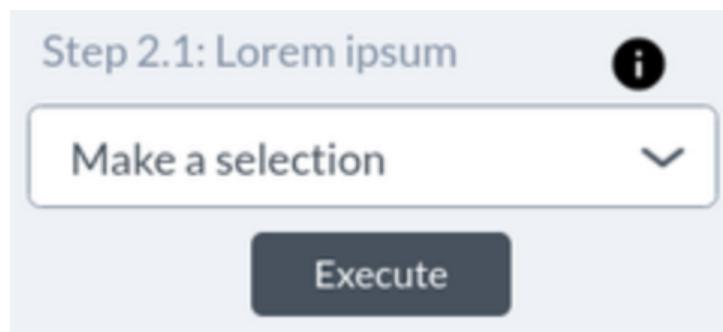




## Components

This folder will mainly contain all reusable components, e.g. Button, Tooltip, Modal, etc.

Notably, some of these reusable components would be "step types", e.g. `DropdownSelectStepType`. This example step type refers to the entire object shown below, including a step number, title, tooltip, dropdown list, button and whatever else may be needed. We would create this as a reusable component because many steps have similar requirements, e.g. selecting a dataset and selecting a featurizer should both be dropdown list type steps.



## Steps

The word "steps" in this section refers specifically to the workflow steps that would be shown in the input panel, e.g. Dataset Selection step, Featurizer Selection step, etc.

A separate folder is created for these so that there would be a clear pattern that is easy to follow whenever more steps need to be added. Each step would use a step type component that is imported from the /components folder. E.g. the DatasetSelectionStep would use the DropdownSelectStepType, as shown in the sample code snippet below.

## Sample code snippets

```
import DropdownSelectStepType from '../../components/dropdownSelectStepType';
import HelpModal from './HelpModal';
...
const DatasetSelectionStep = (props) => {
  ...
  const STEP_KEY = "dataset_selection"

  const options = api_call_here() // calls backend API to get the dataset options

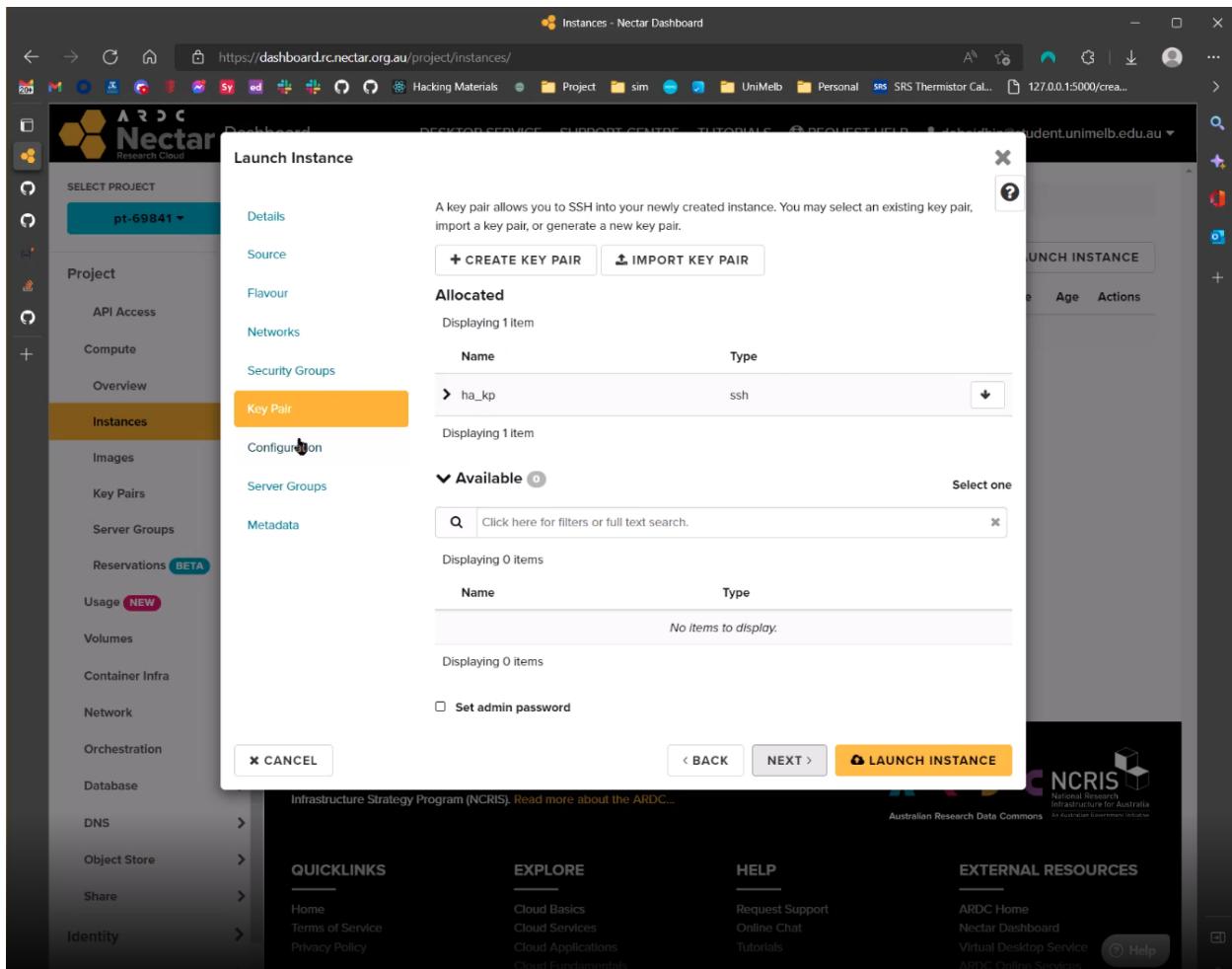
  const onSubmit = selected_value => send_to_backend() // send to backend using api

  return (
    <DropdownSelectStepType
      stepNumber={props.stepNumber}
      title="Select Dataset"
      description="bla bla"
      tooltipContent={HelpModal}
      options={options}
      onSubmit={onSubmit}
    />
  );
};
```

## AMENDMENT - 2<sup>nd</sup> September 2022

It was suggested that an appFooter is potentially added as a section as well. No objections to this so far.

## Appendix 9. Deployment process screenshots



*Create a key pair to use when accessing the virtual machine*

The screenshot shows a web browser window for the Nectar Dashboard at <https://dashboard.rc.nectar.org.au/project/instances/>. The left sidebar is titled 'pt-69841' and contains a tree view of project resources under 'Project'. The 'Instances' node is selected and highlighted in yellow. A modal dialog box titled 'Launch Instance' is open in the center of the screen. The dialog has several tabs: 'Details', 'Source', 'Flavour', 'Networks', 'Security Groups', 'Key Pair', 'Configuration' (which is active and highlighted in yellow), 'Server Groups', and 'Metadata'. Under 'Configuration', there is a section for 'Customization Script' with a 'Choose File' button and a note stating 'Content size: 0 bytes of 16.00 KB'. Below this is a 'Disk Partition' dropdown set to 'Automatic'. There is also a checkbox for 'Configuration Drive'. At the bottom of the dialog are 'CANCEL', 'BACK', 'NEXT >', and a large orange 'LAUNCH INSTANCE' button. The background of the dashboard shows a list of instances with columns for 'Age' and 'Actions'. The footer of the page includes the ARDC and NCRIS logos, quick links, explore sections, help options, and external resources.

*Launch the virtual machine*

The screenshot shows the Nectar Dashboard interface for launching a new instance. The left sidebar is a navigation menu with various project and service links. The main area is titled 'Launch Instance' and is currently on the 'Security Groups' step. It displays two sections: 'Allocated' and 'Available'. The 'Allocated' section contains three items: 'default' (Default security group), 'ssh' (Allow SSH), and 'http' (Allow HTTP/S). The 'Available' section contains three items: 'icmp' (Allow ICMP (e.g. ping)), 'flask\_temp' (Temp allow flask port 5000), and 'react'. A search bar is available for filtering. At the bottom of the form are 'NEXT >' and 'LAUNCH INSTANCE' buttons.

Select the security groups to launch the instance in.

**Allocated** 3

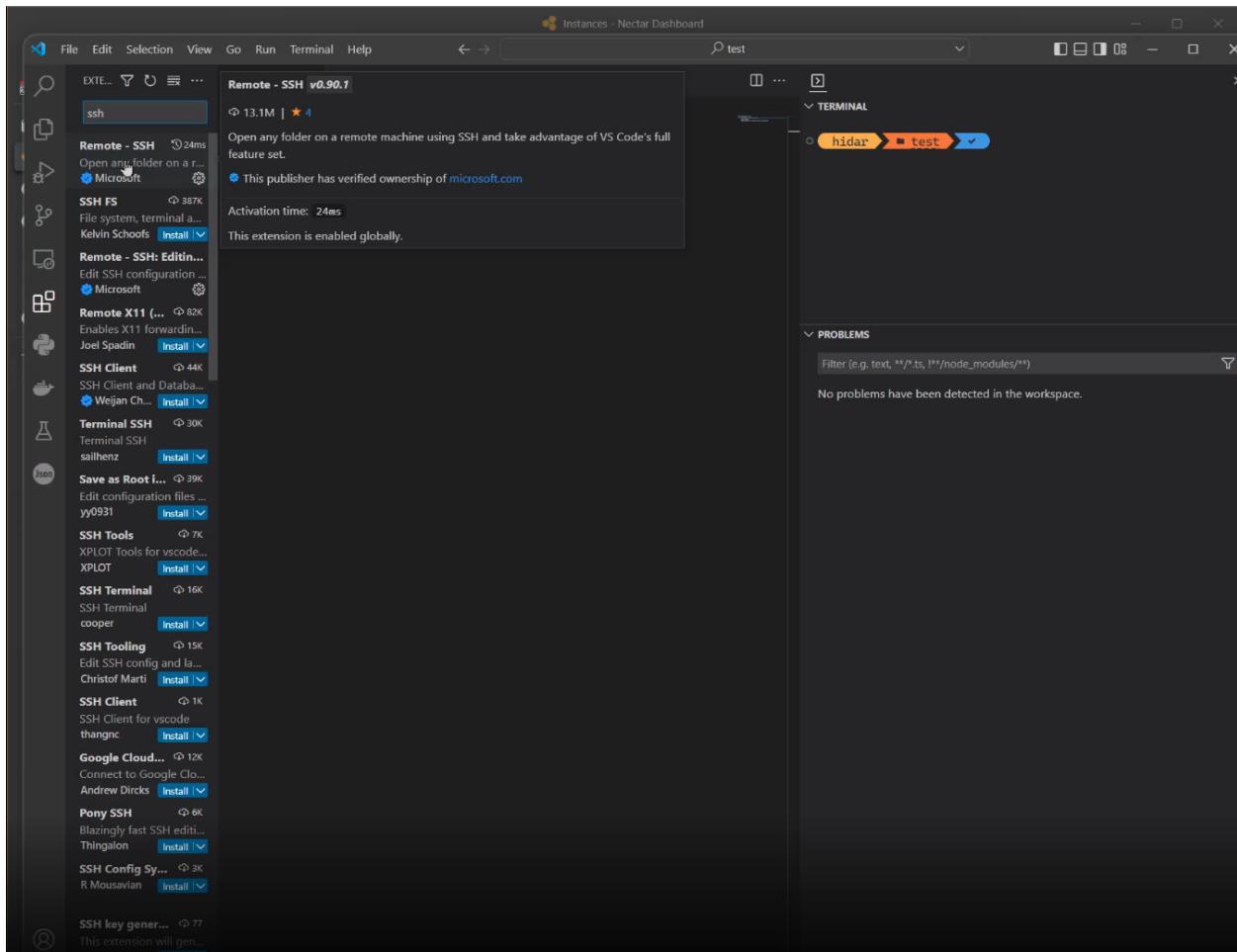
Name	Description
default	Default security group
ssh	Allow SSH
http	Allow HTTP/S

**Available** 3 Select one or more

Name	Description
icmp	Allow ICMP (e.g. ping)
flask_temp	Temp allow flask port 5000
react	

**LAUNCH INSTANCE**

Create a security group in Nectar to enable HTTPS and SSH access



*Connect to the machine using SSH. In this example, we have used a Visual Studio Code remote connection*

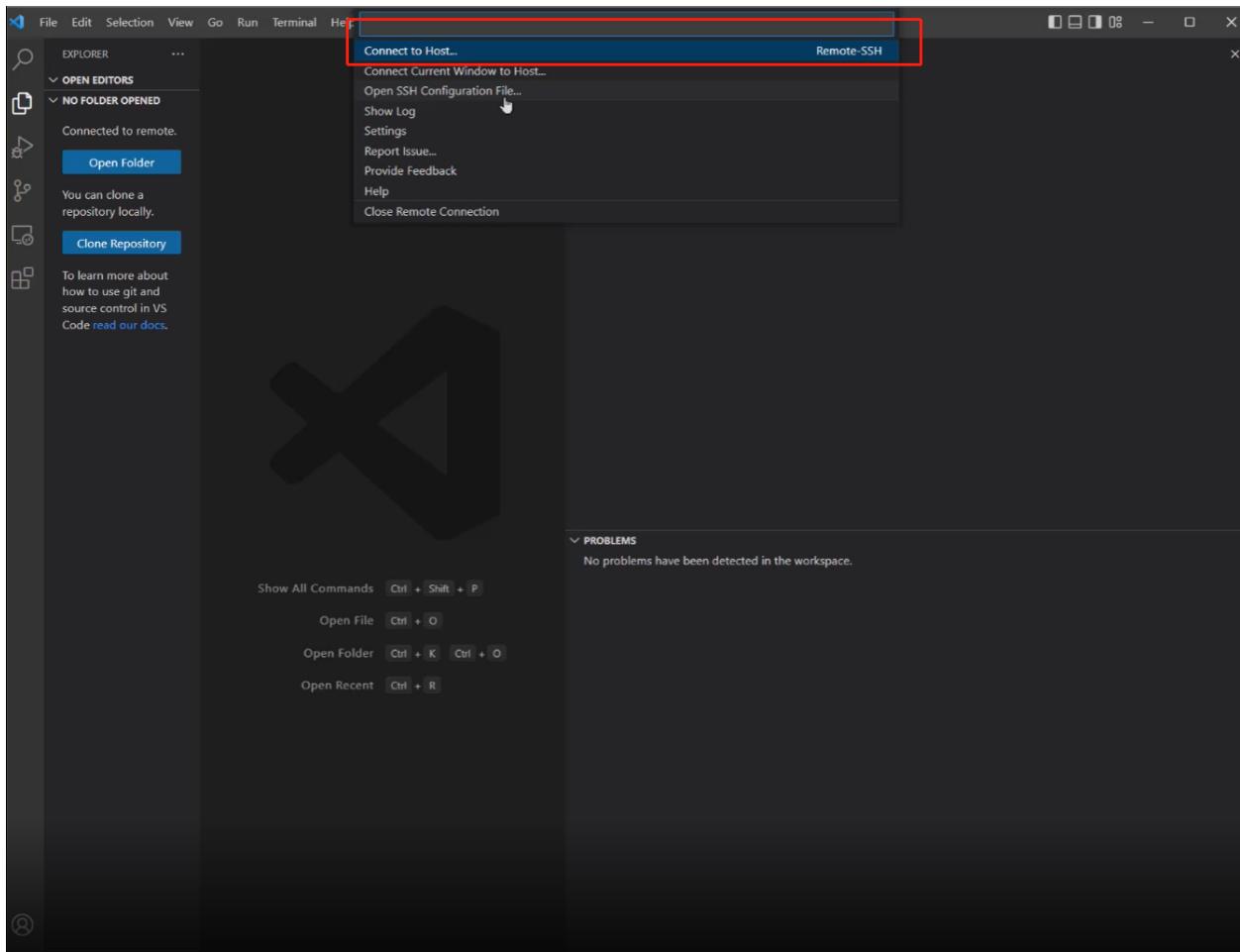
The screenshot shows the Nectar Dashboard interface. On the left, a sidebar menu is open under the 'Compute' section, with 'Instances' selected. The main content area displays a table of instances. One instance, 'HA\_test', is highlighted. A red arrow points from the 'HostName' entry in the SSH configuration file below to the IP address '203.101.230.254' in the table.

Instance Name	Image Name	IP Address	Flavour	Key Pair	Status	Availability Zone	Task	Power State	Age	Actions
NeCTA R Ubuntu 22.04 LTS (Jammy)	qld-data	10.255.138.215	m3.small	ha_kp	Active	qrisccloud	None	Running	0 minutes	<button>CREATE SNAPSHOT</button>
HA_test	qld	203.101.228.7								

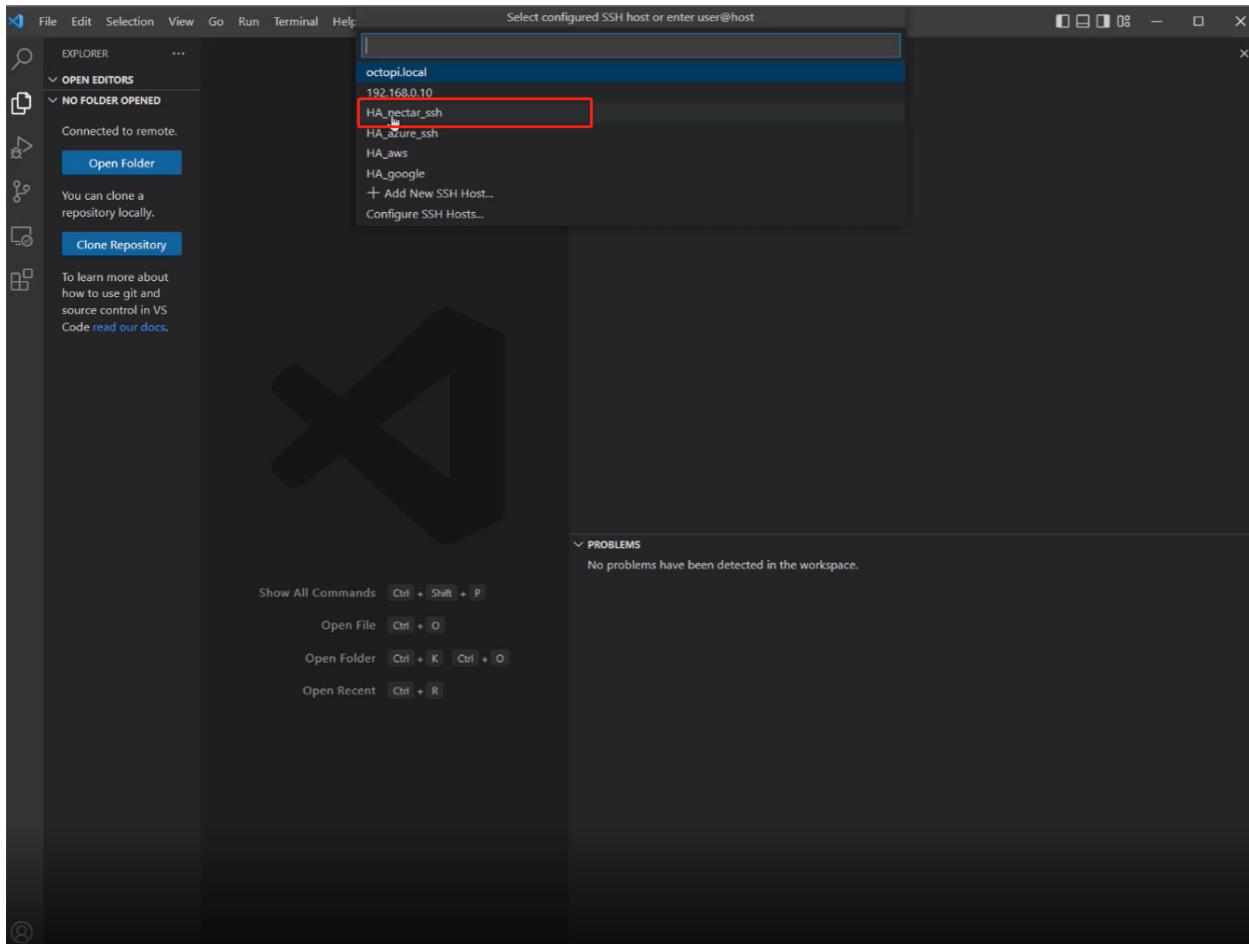
Below the table, an SSH configuration file is shown:

```
Host HA_nectar_ssh
  HostName 203.101.230.254
  User ubuntu
  IdentityFile C:\Users\xxxx\.ssh\nectar
```

Set up an SSH configuration file entry for the virtual machine

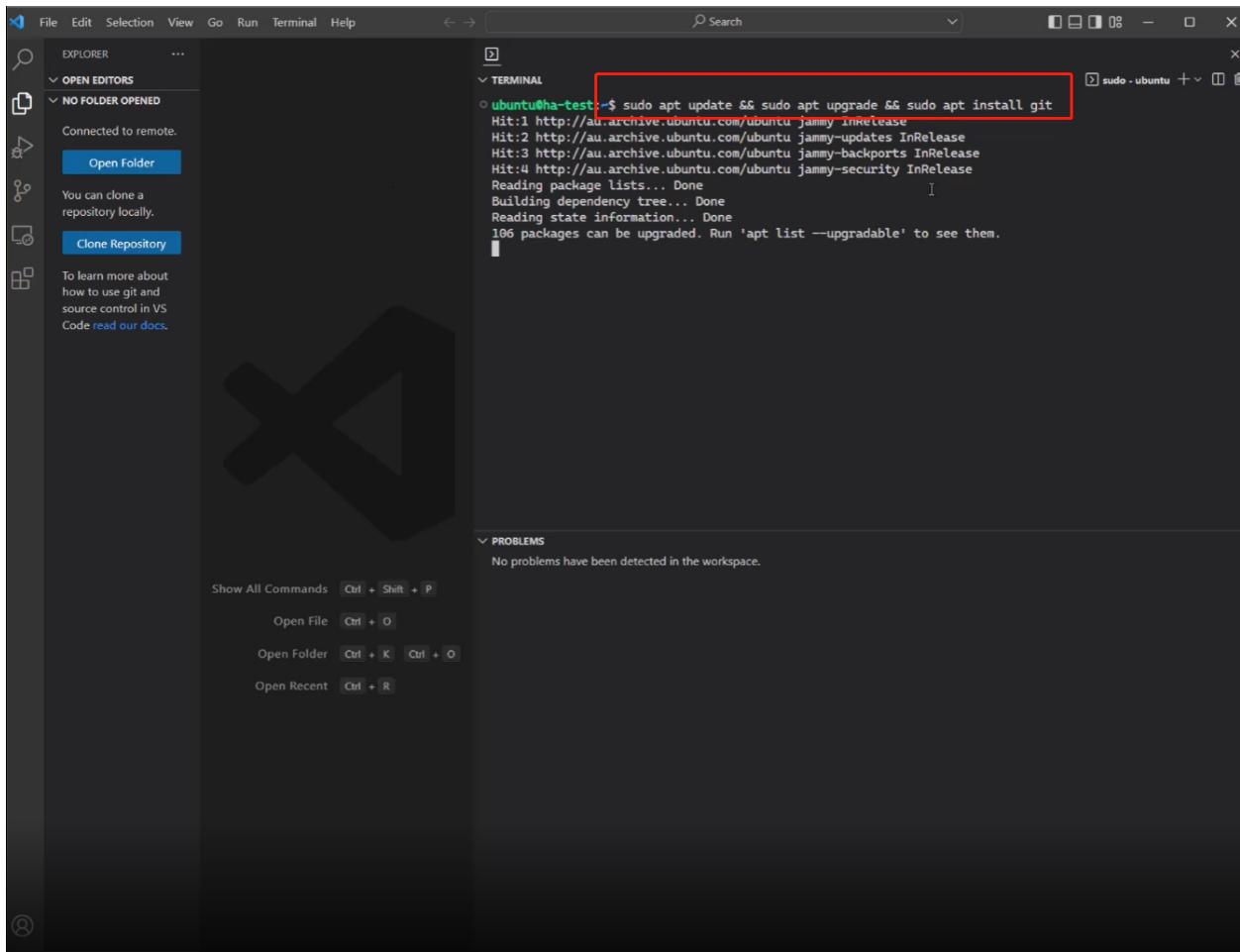


*Connect to the virtual machine using SSH*



*Choose the right machine to connect to*

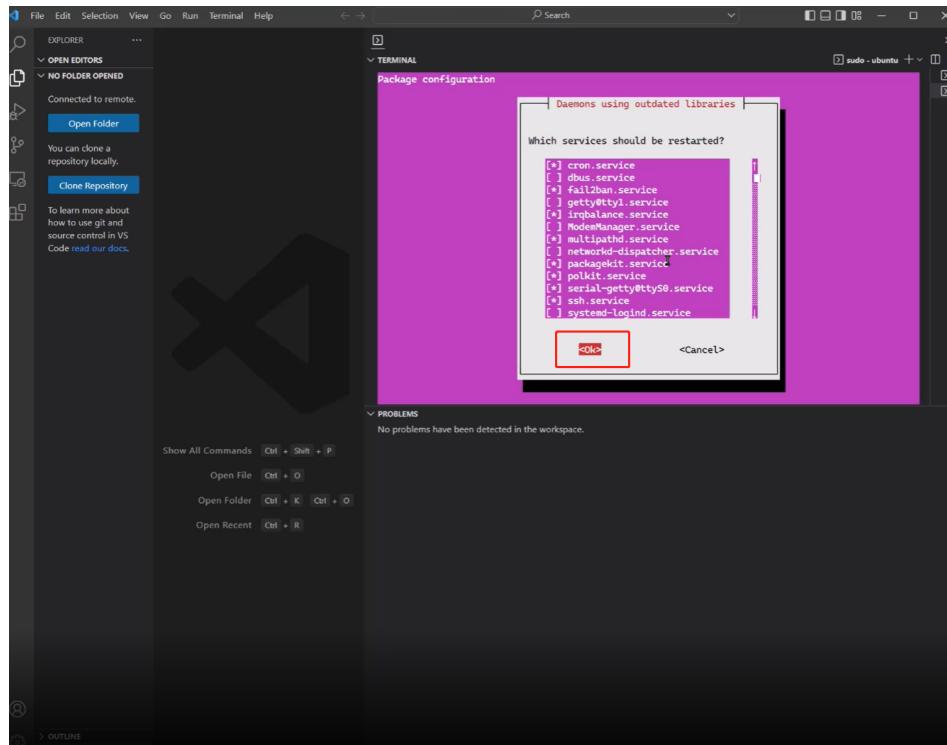
# COMP90082 2022 SM2 - Hacking Materials User Interface (HA) Project



A screenshot of the Visual Studio Code (VS Code) interface. The window title is "sudo - ubuntu". The terminal tab is active, displaying the following command and its execution:

```
ubuntu@ha-test:~$ sudo apt update && sudo apt upgrade && sudo apt install git
Hit:1 http://au.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://au.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://au.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://au.archive.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
106 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

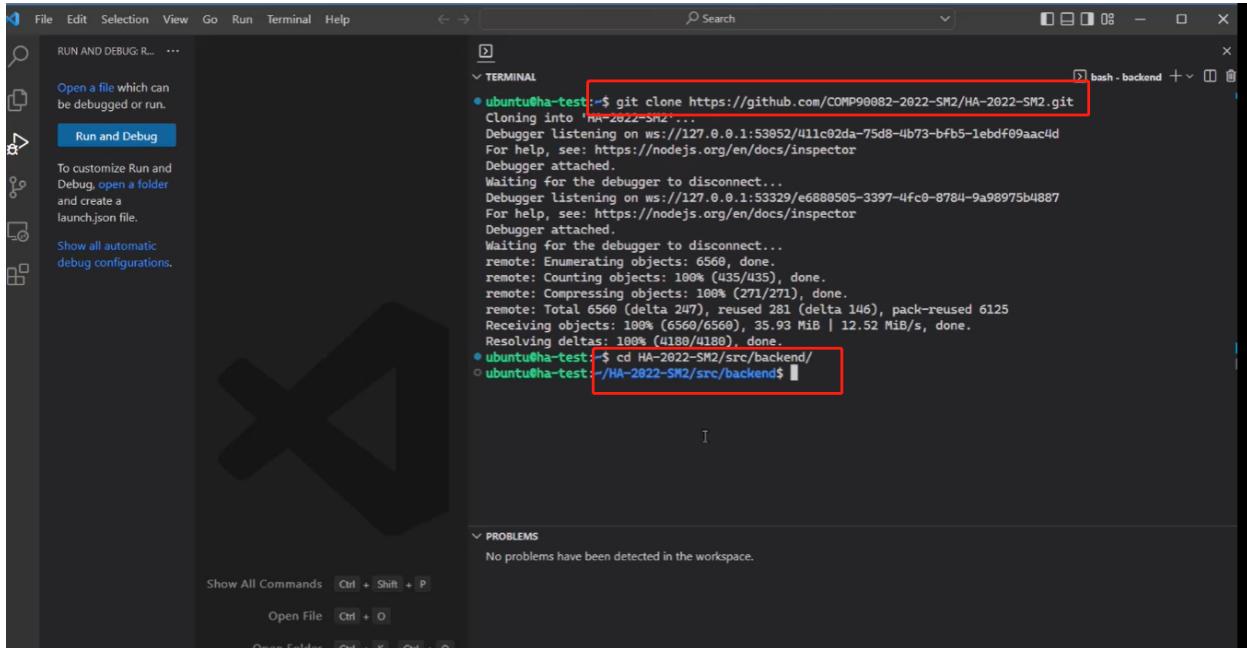
The terminal output is highlighted with a red rectangle. The rest of the interface shows the Explorer sidebar with "No Folder Opened" and the Problems panel which states "No problems have been detected in the workspace".



*Update the existing software on the machine*

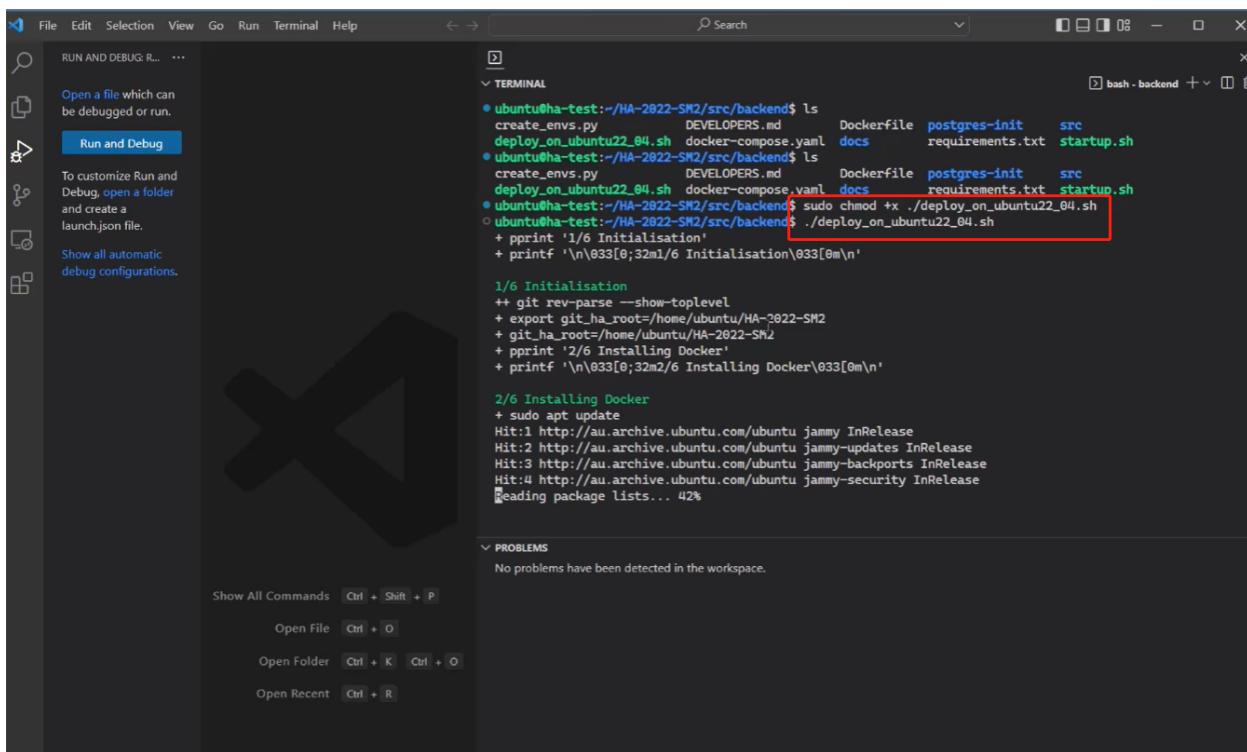
A screenshot of the Visual Studio Code interface. The terminal window shows a series of commands being run to update packages. The output includes: 'Restarting services...', 'systemctl restart cron.service fail2ban.service irqbalance.service multipathd.service packagekit.service polkit.service serial-getty@ttyS0.service ssh.service udisks2.service', 'Service restarts being deferred:', 'systemctl restart ModemManager.service /etc/needrestart/restart.d/dbus.service', 'systemctl restart getty@tty1.service', 'systemctl restart networkd-dispatcher.service', 'systemctl restart systemd-logind.service', 'systemctl restart unattended-upgrades.service', 'systemctl restart user@1000.service'. Below this, it says 'No containers need to be restarted.' and 'No user sessions are running outdated binaries.' Further down, it shows 'No VM guests are running outdated hypervisor (qemu) binaries on this host.', 'Reading package lists... Done', 'Building dependency tree... Done', 'Reading state information... Done', 'git is already the newest version (1:2.34.1-1ubuntu1.5).', 'The following packages were automatically installed and are no longer required: libflashrom1 libftdi1-2', 'Use 'sudo apt autoremove' to remove them.', '0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.', and finally 'ubuntu@ha-test:~\$ sudo reboot now'. A red box highlights the command 'ubuntu@ha-test:~\$ sudo reboot now'.

*Reboot the machine*



The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the command `git clone https://github.com/COMP90082-2022-SM2/HA-2022-SM2.git` being run, followed by the output of the cloning process. The output includes messages about cloning into the directory, debugger listening on port 53329, and various progress and completion messages. The entire terminal session is highlighted with a red box.

*Clone the GitHub repository*



The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the command `./deploy_on_ubuntu22_04.sh` being run, followed by the output of the deployment script. The output includes messages about changing permissions, executing the script, and installing Docker. The entire terminal session is highlighted with a red box.

*Run the deployment script*

COMP90082 2022 SM2 - Hacking Materials User Interface (HA) Project

The screenshot shows the Visual Studio Code interface with the following details:

- File Edit Selection View Go Run Terminal Help** - The top menu bar.
- Search** - The search bar at the top right.
- RUN AND DEBUG** - A button on the left sidebar.
- Open file which can be debugged or run.** - A tooltip for the RUN AND DEBUG button.
- Run and Debug** - A button on the left sidebar.
- To customize Run and Debug, open a folder and create a launch.json file.** - A tooltip for the Run and Debug button.
- Show all automatic debug configurations.** - A tooltip for the Run and Debug button.
- TERMINAL** - A terminal window showing a large grid of characters (mostly '+', '.', and ',') representing a certificate request.
- You are about to be asked to enter information that will be incorporated into your certificate request.** - A message indicating the user is about to be prompted for certificate information.
- What you are about to enter is what is called a Distinguished Name or a DN.** - A message explaining what the fields represent.
- There are quite a few fields but you can leave some blank** - A message telling the user they can leave fields empty.
- For some fields there will be a default value,** - A message about default values for fields.
- If you enter '.', the field will be left blank.** - A message about the '.' character as a placeholder.
- Country Name (2 letter code) [AU]:** - A prompt for the country code.
- State or Province Name (full name) [Some-State]:Victoria** - A response for the state/province.
- Locality Name (eg, city) []:Melbourne** - A response for the locality.
- Organization Name (eg, company) [Internet Widgits Pty Ltd]:Melbourne University** - A response for the organization.
- Organizational Unit Name (eg, section) []:** - A prompt for the organizational unit.
- Common Name (e.g. server FQDN or YOUR name) []:** - A prompt for the common name.
- Email Address []:** - A prompt for the email address.
- PROBLEMS** - A section titled PROBLEMS.
- No problems have been detected in the workspace.** - A message indicating no problems were found.

### *Enter Self-Signed Certificate data*

File Edit Selection View Go Run Terminal Help

Search

RUN AND DEBUG R... ...

Open a file which can be debugged or run.

Run and Debug

To customize Run and Debug, open a folder and create a launch.json file.

Show all automatic debug configurations.

TERMINAL

```
+ printf '\n\033[0;32m4/6 Updating Hacking Materials code\033[0m\n'
4/6 Updating Hacking Materials code
+ sudo cp /etc/ssl/private/nginx-selfsigned.key /home/ubuntu/HA-2022-SM2/src/frontend/nginx/nginx-selfsigned.key
+ sudo cp /etc/ssl/certs/nginx-selfsigned.crt /home/ubuntu/HA-2022-SM2/src/frontend/nginx/nginx-selfsigned.crt
+ sudo chown ubuntu:ubuntu /home/ubuntu/HA-2022-SM2/src/frontend/nginx/nginx-selfsigned.key
+ sudo chown ubuntu:ubuntu /home/ubuntu/HA-2022-SM2/src/frontend/nginx/nginx-selfsigned.crt
+ chmod 664 /home/ubuntu/HA-2022-SM2/src/frontend/nginx/nginx-selfsigned.key
+ chmod 664 /home/ubuntu/HA-2022-SM2/src/frontend/nginx/nginx-selfsigned.crt
+ pprint '5/6 Creating environment variables'
+ printf '\n\033[0;32m5/6 Creating environment variables\033[0m\n'

5/6 Creating environment variables
+ pprint '|t|t|t|t|t ENTER DOCKER ENVIRONMENT VALUES\n'
+ printf '\n\033[0;32m|t|t|t|t|t ENTER DOCKER ENVIRONMENT VALUES\033[0m\n'

ENTER DOCKER ENVIRONMENT VALUES

+ sleep 3
+ python3 /home/ubuntu/HA-2022-SM2/src/backend/create_envs.py /home/ubuntu/HA-2022-SM2
Please enter an email address for the database administrator: test@test.com

Google Api keys are in this format 111111111-111abc123abc123abc123abc.apps.googleusercontent.com
Please enter/paste Google Api client key: 290585236837-sm7bnbjh3jqv5ca77t9273c3q7fo7d06.apps.googleusercontent.com
```

PROBLEMS

No problems have been detected in the workspace.

Show All Commands Ctrl + Shift + P

Open File Ctrl + O

Open Folder Ctrl + K Ctrl + O

Open Recent Ctrl + R

*Enter API keys for Google and Microsoft Single-Sign On APIs*

```

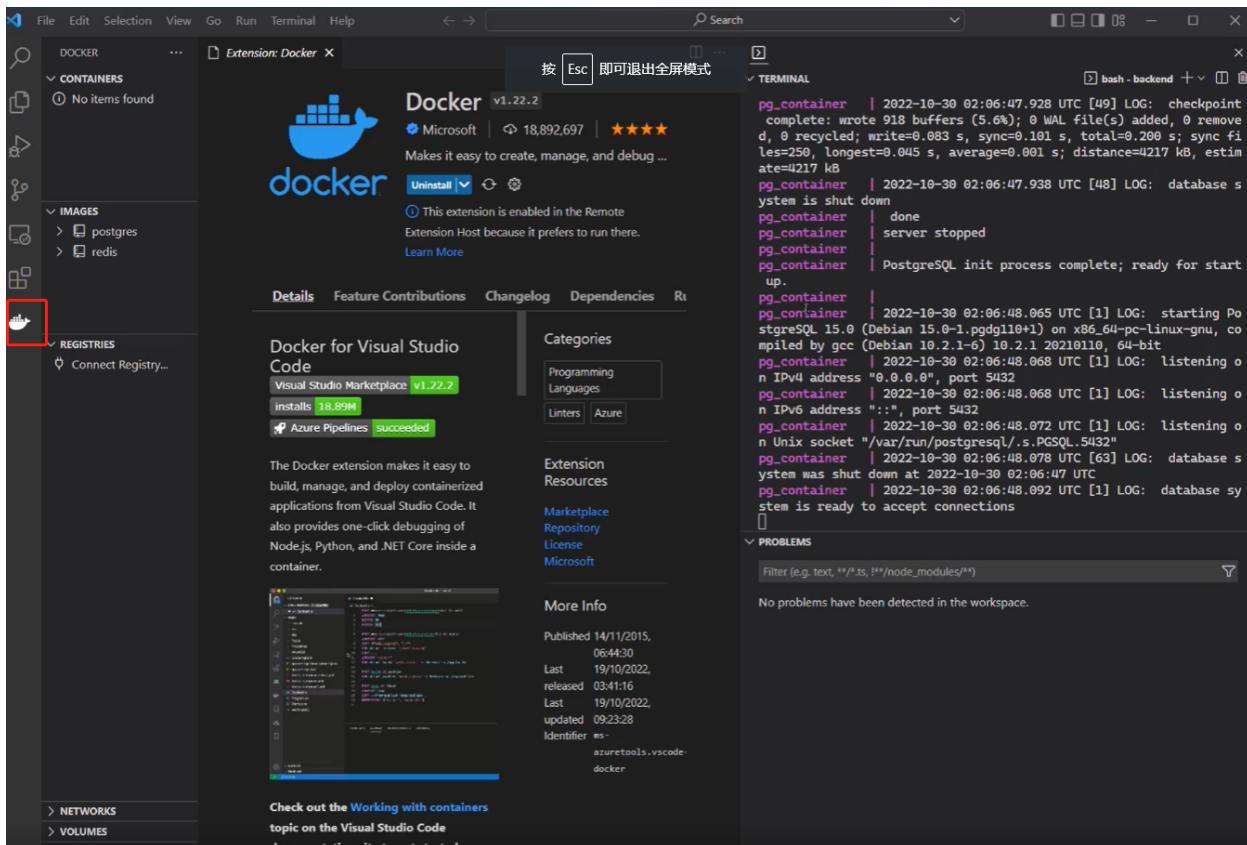
REACT_APP_GOOGLE_CLIENT_ID=290505236837-sm7bnbjh3jqv5ca77t9273c3q7fo7d06.apps.googleusercontent.com
POSTGRES_URL="postgresql://${POSTGRES_USER}:${POSTGRES_PASS}@localhost:5432/ha_db"
CELERY_BROKER_URL="redis://${REDIS_PASSWORD}@localhost:6379/0"
CELERY_RESULT_BACKEND="redis://${REDIS_PASSWORD}@localhost:6379/0"
STR_CONN="postgresql://${POSTGRES_USER}:${POSTGRES_PASS}@localhost:5432/ha_db"

Frontend environment variables:
REACT_APP_GOOGLE_CLIENT_ID=290505236837-sm7bnbjh3jqv5ca77t9273c3q7fo7d06.apps.googleusercontent.com
REACT_APP_BACKEND_API_URI=""

enter yes to accept [yes]
yes
Frontend & Backend .env files created
+ pprint '6/6 Starting containers'
+ pprint '3/3m6/6 Starting containers\033[0m\033[0m\033[0m\033[0m\033[0m\033[0m'

6/6 Starting containers
+ cd /home/ubuntu/HA-2022-SM2/src/backend/
+ docker compose --profile backend --profile frontend up
[+] Running 0/2
  .: redis Pulling
  .: pg_db Pulling
  
```

Accept the generated environment variables



The application containers are launched

The screenshot shows the Cloudflare dashboard for the domain `hackingmaterials.live`. The left sidebar lists various services: Overview, Analytics, DNS (selected), Email, SSL/TLS, Security, Access, Speed, Caching, Workers Routes, Rules, Network, Traffic, Custom Pages, and Apps. The main content area is titled "DNS" and contains instructions to "Manage your Domain Name System (DNS) settings". A note says "A few more steps are required to complete your setup." It lists adding an MX record for the root domain and setting up restrictive SPF, DKIM, and DMARC records to prevent email spoofing. Below this is a table titled "DNS management for `hackingmaterials.live`". The table has columns: Type, Name, Content, Proxy status, TTL, and Actions. It shows three entries: an A record for the root domain with content `203.101.224.252`, an A record for `hackingmaterials.live` with content `103.101.224.252` (which is highlighted with a red box), and an A record for `www` with content `203.101.224.252`. The "Proxy status" column indicates "Proxied" for all entries. The "TTL" column shows "Auto". The "Actions" column includes "Edit" links for each row. At the bottom, there's a section titled "Cloudflare Nameservers" with instructions to ensure authoritative DNS servers are changed to Cloudflare's nameservers.

Type	Name	Content	Proxy status	TTL	Actions
A	*	203.101.224.252	Proxied	Auto	Edit
A	<code>hackingmaterials.live</code>	<code>103.101.224.252</code>	Proxied	Auto	Edit
A	www	203.101.224.252	Proxied	Auto	Edit

*Add a DNS entry for the virtual machine in Cloudflare*