Home

Project Overview

Materials science is a complex field that focuses on the study of physical materials and their properties. MatMiner is an existing Python library that combines materials data with Machine Learning strategies and models to study material properties without the need for time-consuming physical experimentation. Our project, Hacking Materials ("HA"), aims to build an easy and intuitive user interface to the MatMiner library * to eliminate the need for its users to have substantial Python or machine learning knowledge or experience.

Materials engineering is a field in which physical materials (e.g. metals, ceramics, polymers, composites, etc.) are studied to understand their composition, characteristics and properties. In many industries, such as mining, manufacturing and others, finding the right material for each job is essential for success. Materials engineers and scientists follow many different methods to compare candidate materials and make recommendations based on how they each satisfy the specific use case requirements. Some of these methods include physical experimentation, which can in some cases take up to 20 years to complete.

To avoid this, computer simulations (based on existing databases of known material properties and machine learning algorithms) can be used to compare materials in a much more efficient way. One such tool for this approach is the Python library MatMiner, which allows easy access to ready-made datasets and integrates well with other machine learning Python libraries. However, to use this library, the user must have a substantial level of specialised knowledge in machine learning and programming, which most materials engineering do not have. To help them overcome this, the solution proposed by the client describes a simple and intuitive user interface that would act as a bridge between the user and the MatMiner library in the backend. The vision of this product is to make machine learning methodologies more accessible within the materials science and engineering industry as a whole, minimising the time and financial costs involved and leading to a more efficient industry.

Project Personnel

Dr Christian Brandl

Client

Team Members

Alastair Daivis
(Alastair Daivis)

Team Representative

Sanjeevani "Sanjee" Avasthi
(Sanjeevani Avasthi)

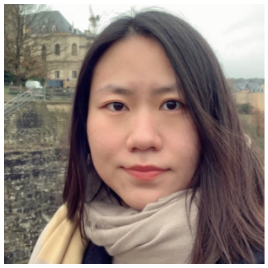
Developer

Useful Links

Trello	GitHub

Key Pages

- **| Specifications**
- **| Meetings**
- **| Assessments**
- **| Resources**

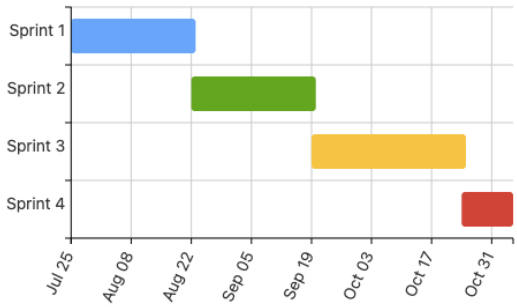
		
Mamta Lopes (Mamta Ritesh Lopes) Developer	**Ghina Yashar** (Ghina Yashar) Frontend Development Lead	**Chunbaixue "Caitlyn" Yang** (Chunbaixue Yang) Scrum Master

Supervisor: Mauro Mello Jr

Lecturer: Cristoph Treude

COMP90082 Software Project Subject Coordinator: Eduardo Araújo Oliveira

Project Timeline



Project Scope

At this stage in development, we are focusing on a minimal set of features applicable to all users of the product. We have worked with Dr. Brandl to identify a set of core features, and to cleanly separate these from "pro-user" and other nice-to-have features. These high-priority features are identified in the User Stories list.

For now, only the high-priority features are definitely in scope for development. Other nice-to-have features may be developed once the high priority features have been finished and a minimum viable product has been deployed.

This project is intended to be generally useful to a broad range of potential users, including:

- Educators, who may use it to demonstrate the Matminer library and its capabilities
- Students, who may use it to learn about these tools and experiment with them
- Materials engineers, who will need to be able to access more advanced features including downloading generated code and customising workflow components

Core technologies used

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.

* Ward, L., Dunn, A., Faghaninia, A., Zimmermann, N. E. R., Bajaj, S., Wang, Q., Montoya, J. H., Chen, J., Bystrom, K., Dylla, M., Chard, K., Asta, M., Persson, K., Snyder, G. J., Foster, I., Jain, A., Matminer: An open source toolkit for materials data mining. Comput. Mater. Sci. 152, 60-69 (2018).

| Specifications

| Project Overview

The Problem

Materials engineering is a field in which physical materials (e.g. metals, ceramics, polymers, composites, etc.) are studied to understand their composition, characteristics and properties. In many industries, such as mining, manufacturing and others, finding the right material for each job is essential for success. Materials engineers and scientists follow many different methods to compare candidate materials and make recommendations based on how they each satisfy the specific use case requirements. Some of these methods include physical experimentation, which can in some cases take up to 20 years to complete.

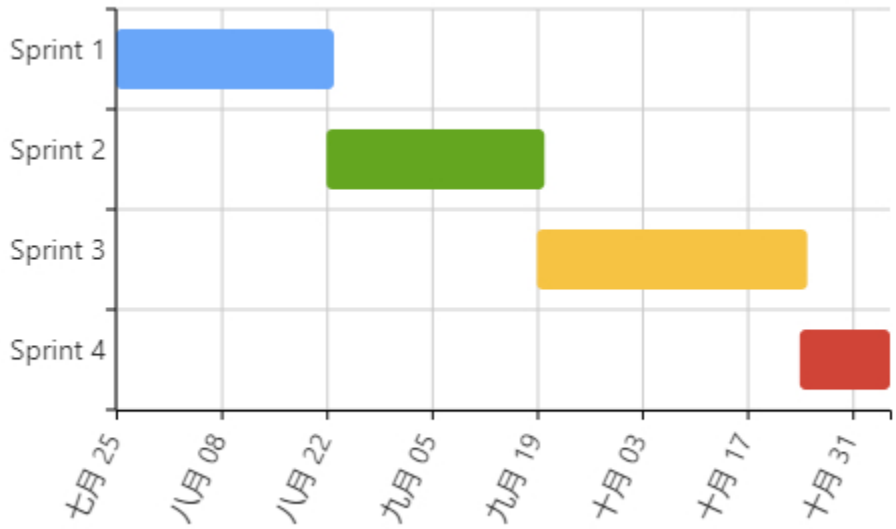
To avoid this, computer simulations (based on existing databases of known material properties and machine learning algorithms) can be used to compare materials in a much more efficient way. One such tool for this approach is the Python library MatMiner, which allows easy access to ready-made datasets and integrates well with other machine learning Python libraries. However, to use this library, the user must have a substantial level of specialised knowledge in machine learning and programming, which most materials engineering do not have. To help them overcome this, the solution proposed by the client describes a simple and intuitive user interface that would act as a bridge between the user and the MatMiner library in the backend. The vision of this product is to make machine learning methodologies more accessible within the materials science and engineering industry as a whole, minimising the time and financial costs involved and leading to a more efficient industry.

The Client

The client of this project is **Dr Christian Brandl**, who is a senior Mechanical Engineering lecturer at the University of Melbourne and has a Ph.D. in Materials Science & Engineering. Aside from his scholarly work, Dr Brandl also acts as a Materials Consultant to clients who come to him seeking recommendations as to which materials may fit their specific needs, or with questions regarding the properties and characteristics of different materials. Dr Brandl hopes to use the product of this project to demonstrate to his clients the capabilities of machine learning and the possibilities it creates in the materials science field.

Schedule

The project will run over four sprints, with the dates of each outlined below.



| Personas

Pro User - Prepared by Team RedBack

Alex

age: 45
residence: Melbourne
education: Masters Degree in Physics
occupation: Materials Engineer
marital status: Divorced without kids

"There has got to be a better way to do this."

Motivation : As an experienced Materials Engineer, Alex's job requires him to narrow down candidate materials by performing physical experiments to choose a material which can takes years to do. He needs a tool that can speed up the process by narrowing down candidate materials for experimentation using Machine Learning and simulations.

Comfort With Technology

PROGRAMMING WITH PYTHON

MACHINE LEARNING

CLOUD BASED STORAGE

MATERIAL SCIENCE

Criteria For Success:

Alex can find the right materials efficiently, with accurate results and that matches the client's requirements.

Needs

- Products to accelerate his workflow
- Access to wide variety of related tools and resources

Wants

- Suitable models and featurizers for different use cases
- Demonstrate reproducible results to his clients
- Share resources with others
- Refining generated workflow to reuse

Values

- Extensibility
- Accuracy
- Reliability
- Responsiveness
- Scalability
- Transparency

Fears

- Tool is too inflexible
- Losing access to progress on his work
- Not being able to verify his results
- Not having support with the tool

Student - Prepared by team BoxJelly

Assol Anahita

age: 22
residence: Melbourne
education: Material Engineering
occupation: Student
marital status: Single

"It's SO time consuming to do material researches and get decent results through just a semester."

Motivation: As a material engineering graduate student, Assol gets frustrated and demotivated when she can't make sense of the data she has because she doesn't have a tool or sufficient programming/machine learning skills to process the material data. She is also frustrated that she can't use machine learning algorithms to help her engineer new materials even though she is told by her supervisor that this idea works in theory.

Comfort With Technology

PROGRAMMING WITH PYTHON

MACHINE LEARNING

CLOUD BASED STORAGE

MATERIAL SCIENCE

Criteria For Success:

Assol can perform materials data requests/retrievals and accurate materials property predictions supported by Machine Learning technology with easy to follow steps button clicks user interface.

Needs

- Easy-to-use interface Material science data processing and retrieval application
- A tool to predict property of a material with assistance of Machine Learning technology without prior knowledge of Python and Machine Learning programming

Wants

- A data mining application that helps her researches
- A better understanding on how Machine Learning can help her to learn more about a material
- Ability to use ML algorithms as a black box
- Freedom to select features on her own terms
- A tool to accelerate research progress

Values

- Convenience
- Quickness
- Safety
- Understandable

Fears

- Spends hours without getting anything done because she neither has an adequate tool to do data mining, nor the programming skill to analyse the data herself
- Have to conduct countless experiments to figure out the properties of the materials
- Hard to choose suitable ML algorithms

Industry User - Prepared by team BlueRing

Gray Zhou

age: 28
residence: Ningde, Fujian, China
education: Master of Material Engineering
occupation: R & D Engineer of Polymer
marital status: Single

"It is fantastic to apply a multi-function online tool with ML methods if it is efficient and reliable. Nobody will refuse a tool that can save his time"

Motivation: Gray Zhou is a R & D Engineer of polymer in a battery factory. His work is searching for better materials for battery production. Gray spends lots of time testing different materials, but some of tests are waste of time because of the poor performance observed. He needs a system that can predict some useful properties of materials so that he can remove samples with low predicted performance and boost the research. His company provides some solutions, but they are awkward and only have limited functions.

Comfort With Technology

PROGRAMMING WITH PYTHON

MACHINE LEARNING

CLOUD BASED STORAGE

MATERIAL SCIENCE

Criteria For Success:

Provide a website or online-tool with quick, visual interface which can help him in daily development of new materials.
A successful product should help him save noticeable time on data processing and provide reliable prediction of properties.

Needs

- Retrieve and extract required data, process the data with ML methods to get some properties
- Provide graphs which can be modified with interface about predicted properties
- Help finding the material with best predicted properties

Wants

- Ability to interact with the graph to further compare serval materials in detail
- Upload data from his lab for predicting
- Explain what ML method the system applied and how it helps the prediction
- Continue his work on mobile devices without gaps of interaction

Values

- Easy to get started on both desktop and mobile
- Efficient back-end process
- Abilities to select functions and filter results
- Well organized visualization of interface and graphs

Fears

- Not enough guidance in the web or tool so him may feel confused to find functions he wants.
- Lackingunderstandofwhatthesystemdone,then reducing the confidence level of his report
- Frequently unavailability

| Motivational Model

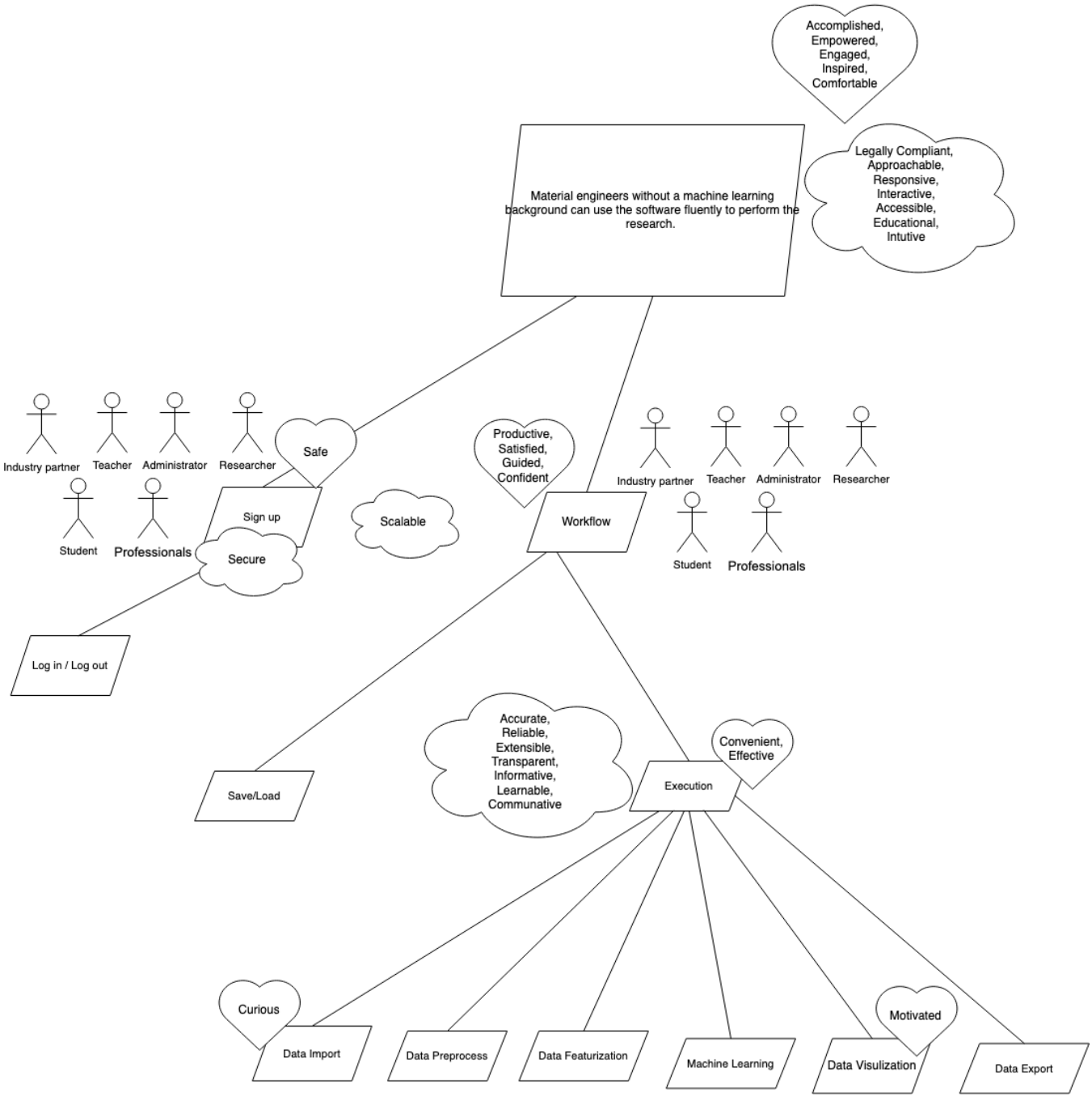
Do/Be/Feel lists

Version 1.1 - Sprint 1 Final (cross-team effort)

Who (users)	Do (functional goals)	Be (qualitative goals)	Feel (emotional goals)
Students	Add more database, machine learning method and plot types	Accessible	Accomplished
Administrators	Compare data using tables & plots	Accurate	Comfortable
Professionals	Data Pre-processing: Calculate descriptive statistics	Approachable	Confident
Industry Partners	Data Pre-processing: Consider anonymized data	Communicative	Convenient
Teachers	Data Pre-processing: Overview of the current import data	Educational	Curious
Researcher	Data Pre-processing: Reduces noise and eliminates ambiguity	Extensible	Effective
Code maintainers	Data Pre-processing: Standardizing data to bring it into the formatting range	Informative	Empowered
	Data Visualization: Data processing: Tabular data & Plotted Graph	Interactive	Engaged
	Edit python code directly in the interface	Intuitive	Guided
	Export input data	Learnable	Inspired
	Export jupyter notebook file	Legally Compliant	Motivated
	Export output data tables and figures	Reliable	Productive
	Featurization data: Add multiple composition-based features	Responsive	Safe
	Featurization data: Add multiple simple density features	Scalable	Satisfied
	Import Data: Creat working spaces when importing	Secure	
	Import Data: Drag and drop import of files	Transparent (progress, error messages, notebook export...)	
	Import Data: Import data files (CSV, XES, Parquet) from local system		
	Log in/Log out		
	Machine Learning: Define input data and output data: Splitting data into training, test, and validation sets		
	Machine Learning: Determining model features and training the model: Configure and adjust hyperparameters for optimum performance		
	Machine Learning: Evaluate model performance and establish benchmarks: Continuous measurement and monitoring of model performance		
	Machine Learning: Evaluate model performance and establish benchmarks: Evaluate models using validation methods and validation datasets		
	Machine Learning: Get model results: The most important features of the current ML model		
	Machine Learning: Select the machine learning model to be used		
	Maintain software		
	save/load workflows		
	Sign up		

Goal Model

Version 1.1 - Sprint 1 Final (cross-team effort)



| User Stories

Version 1.3 - Sprint 1 Final (cross-team effort)

Epics & Owning team allocation

	Epic	Total Size	Highest Priority within Epic	Assigned Team
1	**Input Data**	27	1 - Must Have	RedBack
2	**Administration**	31	1 - Must Have	BoxJelly
3	**Machine Learning**	16	1 - Must Have	BlueRing
4	**Data Visualisation**	13	1 - Must Have	BlueRing
5	**Jupyter Notebook**	20	2 - Should Have	Unassigned - stretch goal
6	**External Data**	8	2 - Should Have	Unassigned - stretch goal

User Stories

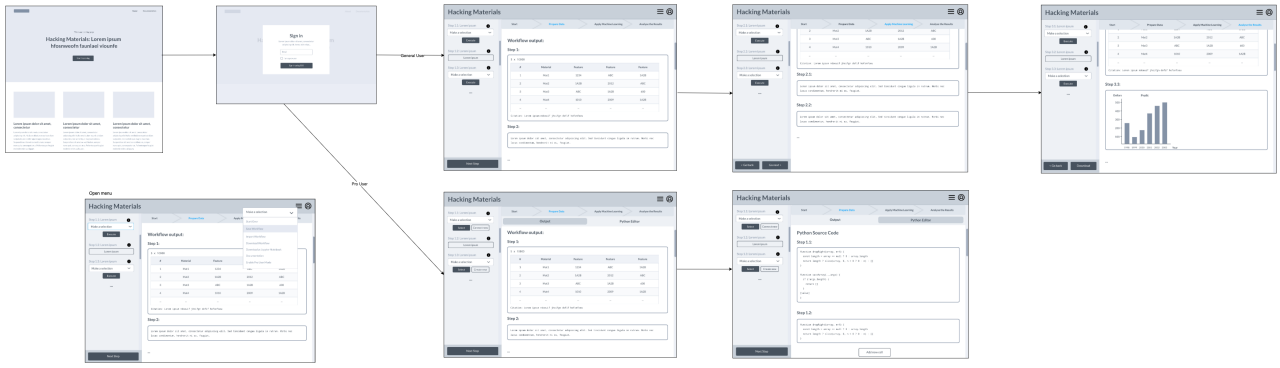
ID		Role		Action		Goal	Epic	Size (days)	Priority	Acceptance Criteria (Added when kick-off)
30	**As a**	general user	**I want to**	be able to view the citations for used featurizers	**so that**	I could be know more about the source of the featurizer (legally compliant)	Input Data	1	1 - Must have	
32	**As a**	general user	**I want to**	browse and select built-in featurizers	**so that**	I can discover ways of manipulating my data		1	1 - Must have	Given I'm at the featurizer selection step. When I click on the dropdown menu. Then I see a list of all built-in featurizers with readable names. Given I'm at the featurizer selection step and I can see a list of featurizers. When I select one of the featurizers and click on submit. Then I should receive some feedback on whether the action is successful or not.
34	**As a**	general user	**I want to**	browse built-in datasets	**so that**	I can discover data to experiment with		1	1 - Must have	Given I'm at the dataset selection step. When I click on the dropdown menu. Then I see a list of all built-in datasets with readable names.
19	**As a**	student	**I want to**	quickly browse the Materials available in the database for retrieval and simulations	**so that**	I can quickly perform queries.		3	1 - Must have	

21	**As a**	general user	**I want to**	be able to select datasets from exisiting databases	**so that**	I do not have to worry about how the data is loaded		3	1 - Must have	Given I'm at the dataset selection step and I can see a list of datasets. When I select one of the datasets and click on submit. Then I should receive some feedback on whether the action is successful or not.
37	**As a**	general user	**I want to**	be able to preview the input data	**so that**	I could explore the data		1	2 - Should have	
25	**As a**	general user	**I want to**	Select specific features from a dataset	**so that**	I can improve the precision of my model		3	2 - Should have	
13	**As a**	Pro user	**I want to**	add new features	**so that**	they can be reused in the future		5	2 - Should have	
28	**As a**	general user	**I want to**	be able to reference / view citation for original data sources	**so that**	I can retrieve data.		1	3 - Could have	
18	**As a**	pro user	**I want to**	be able to apply new featurizers	**so that**	I can create new features		3	3 - Could have	
1	**As a**	student	**I want to**	clean and tune data input	**so that**	I have less noise on visualizations.		5	3 - Could have	
29	**As a**	student	**I want to**	save project specific data/checkpoints	**so that**	I can pick up where I left off for specific projects	Administration	1	1 - Must have	
35	**As a**	pro user	**I want to**	export model selections, parameters, and data flows	**so that**	I can save my work and share it with others		1	1 - Must have	
36	**As a**	pro user	**I want to**	import exported model selections, parameters, and data flows	**so that**	I can continue work I had previously saved		1	1 - Must have	
20	**As a**	student	**I want to**	Create an account using single-sign on, restricted to the *.unimelb.edu.au domain	**so that**	my research remains private		3	1 - Must have	
23	**As a**	pro user	**I want to**	Control job execution	**so that**	I can start, view progress of, and cancel jobs related to my project		3	1 - Must have	
10	**As a**	pro user	**I want to**	be able to opt in to pro-user features	**so that**	I can access pro user features		5	1 - Must have	
38	**As a**	pro user	**I want to**	have my pro user settings persist on each visit	**so that**	I don't have to reconfigure settings to use the features I need		1	2 - Should have	
24	**As a**	student	**I want to**	receive provided hints and guidance for new users	**so that**	I can quickly learn how to use software		3	2 - Should have	
14	**As a**	pro user	**I want to**	easily find and read documentation on the pro features	**so that**	I can use them with ease		5	2 - Should have	
17	**As a**	pro user	**I want to**	Be kept informed about job status	**so that**	I can avoid polling my workspace to check for results		3	3 - Could have	
6	**As a**	pro user	**I want to**	have access to more processing power	**so that**	I can run more complex operations or use more data		5	3 - Could have	
31	**As a**	general user	**I want to**	able to select a Machine Learning model	**so that**	I could use it to train and run the data	Machine Learning	1	1 - Must have	
33	**As a**	general user	**I want to**	browse built-in ML models	**so that**	I can discover ways of manipulating my data		1	1 - Must have	
39	**As a**	user	**I want to**	be able to select split ratio of data	**so that**	to train and test the model		1	2 - Should have	
26	**As a**	pro user	**I want to**	have the option to change the hyperparameters used in the machine learning model	**so that**	I can fine tune my test results.		3	2 - Should have	
15	**As a**	pro user	**I want to**	be able use additional ML models	**so that**	I can improve accuracy		5	2 - Should have	

7	**As a**	pro user	**I want to**	combine multiple models together	**so that**	I can model more complex data manipulations		5	3 - Could have	
22	**As a**	general user	**I want to**	see clear annotation or explanation of data points and features	**so that**	I can understand the results of the analysis	Data Visualisation	3	1 - Must have	
8	**As a**	student	**I want to**	use different type of plotting graphs	**so that**	I have flexibility to visualize data according to my needs.		5	1 - Must have	
9	**As a**	general user	**I want to**	able to view and plot the results of the model	**so that**	I could analysis and visualise the effects of the model		5	1 - Must have	
12	**As a**	student	**I want to**	export my work to a Jupyter Notebook	**so that**	I can extend my work beyond the capability of the application	Jupyter Notebook	5	2 - Should have	
2	**As a**	general user	**I want to**	attach comments to workflow objects	**so that**	I can document my work		5	3 - Could have	
4	**As a**	Pro user	**I want to**	edit python code on the interface	**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	**so that**	I can extend the tool to support custom models and featurizers		5	3 - Could have	
27	**As a**	pro user	**I want to**	be able to access new databases	**so that**	I can access addional data	External Data	3	2 - Should have	
3	**As a**	Pro user	**I want to**	be able to add new datasets in the future	**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	
11	**As a**	student	**I want to**	analyze the relationship between different features	**so that**	I can identify which features I need to select for my analysis	?	5	2 - Should have	
16	**As a**	general user	**I want to**	add specific materials to the workflow for analysis	**so that**	compare the performance of the specific material my client or I choose with other material	?	3	3 - Could have	

| Low-fidelity Prototype

The prototype was created in Marvel and can be viewed here.



Descriptive Notes

- Landing page:
  - Static page with information about the app and project
  - Link to access the app
    - On click, it opens a login modal
    - Once user is logged in, they're redirected to the app
- Single page app:
  - Top bar:
    - User profile button at the top opens a menu to give the user the option to log out
    - Menu button at the top has options to import or save a workflow, download it in different formats, start over, a link to the documentation and a toggle to enable pro view.
  - General user:
    - The workflow is divided into major and minor steps. Each major step would have its own page. User can go back and forth between the major steps as needed.
    - Left panel:
      - All the minor steps are numbers and named to guide the user
      - Inputs can be of different types
      - Each step has a tooltip button that would open a modal with guidance information about the step
      - The steps and options in the left panel should always be the same no matter what selections the user made in previous steps. Any step that requires customised inputs would open in a modal.
        - Example 1: Step 3.1 might be "Selecting a plot type". As there is a known, limited list of different plot types, this step may be a drop-down menu that is displayed directly in the left panel.
        - Example 2: Step 3.2 might be customising the selected plot's configuration options. As different plot types may need different configuration options, these options will not be displayed in the panel directly. Instead, the panel will include only a button that says "Configure plot", which would open a modal with the specific options applicable to the selected plot type.
      - Pinned buttons at the bottom of the panel: navigate between the different major steps. Last step page may also have a button to download the full workflow.
    - Viewing window:
      - At the top of the viewing window, the user can see the progression of major steps with the current step highlighted.
      - The output of each minor step is labelled with the step number and contained inside a box. The output inside the box is the same output produced by running the python code, simply copied over for transparency.
      - The outputs from the previous pages are also always displayed, so it's not just the outputs of the current page.
      - Where a resource with citations is used, the citations will be automatically printed after the output of the step where the resource was selected.
  - Pro user:
    - Left panel: has all the same options as a general user, plus additional buttons to configure their own settings as needed
    - Viewing window: the window has 2 tabs:
      - Output: same the as the viewing window of the general user
      - Python source code:
        - An editable view of all the code generated by their selections, looks similar to a Jupyter notebook.
        - User can add new cells as desired
        - Brings up the following question: what happens if the user edits the code generated by one of the steps? This may lead to inconsistencies between what is shown in the step's input field and what the code now actually does. This is an implementation decision so is not a major concern right now, but one option that we decided to show in the prototype is that the step's input in the left panel would change to say "Custom" or something similar, indicating that the configuration was changed.

| Notes on provided material

> **Note on below content**
>
> The bullet points below were copied directly from the notes provided by the client and are not edited at all for transparency (except for the additions of headings to divide the groups).

Organised Wishlist

As described in the provided notebook and the rough prioritisation described in the kickoff meeting.

High priority features:

- Save, load and modify workflow within interface
- Web interface with user logins
- Server based online application
- Documentation **- Assumed**

Low priority features:

- Create new features based on existing features (PRO user?)
- Interface to interact with python code directly (PRO interface)
- **Extendable** to include more databases, machine learning methods and plot types (e.g., by PRO user)
- Hints on sensible/possible next steps and interaction with UI
- List all required citation </aside>

Unknown priority features:

- Define a set of user defined features
- Generates corresponding python code (or jupyther file with markdown)
- Download and upload files to/from server (jupyther file?)
- Interaction with visualisation, e.g.,click on data point in plot and open corresponding data entry </aside>

General characteristic:

- Should be usable by engineer or postgrad engineering students without programming experience and just WIKIPEDIA knowledge of machine learning

MatMiner example workflow Requirements

Key
Not described by the client, just a suggestion from within the team
Requirement suggested in the relevant step within the Jupyter notebook
Requirement included in the wishlist

#	Workflow step	Requirement
1.0 A	Select dataset from built in datasets with *matminer.dataset*	UI to select dataset from available options
1.0 B	Select a dataset from an online resource with *matminer. data_retrieval*	UI to set up the connection to the online databases
1.1	Preview DataFrame to explore or clean it	“Preview data” button prints same result?
1.2	Remove unneeded columns	Allow user to remove columns from list?
1.3	Review descriptive statistics of the data with *describe()*	“Describe” button prints same result?
2.1, 2.2, 2.3	Featurization: convert inputs into numbers that meaningfully represent the underlying physical quantity using the descriptors library	Select from possible features available
2.1	Printing citation for each featurizer using *citations()*	List required citation (. *citation()* result) for all used featurizers
3.1	Define input and output data of the ML model	Select output column from list of columns? All remaining numerical featurized data assumed as input?
3.2.1	Pick, train and run the ML model to generate predictions	Select from list of possible models?
3.2.2	Display the fitting score and cross-validation score	Automatically print results after predictions?
3.2.3	Plot results using plotting library	Select plot type? Select plotting library? Interaction with visualisation (e.g.,click on data point in plot and open corresponding data entry)
3.3.1	Repeat 3.2.1 with different models	Save previous results and allow to run more models?
3.3.2	Plot training error instead of test error (optionally)	Pick which data to plot?
3.3.3	See which features were most important in the model	“Analyse” option?

| Development

| Quality Assurance Guidelines

Work quality guidelines
- Code should be able to run, there's no syntax error
- If test(s) are included in the pull request, they should pass
- All previous tests should still pass with the new changes in the pull request
- A branch should not be put up for a pull request if it has merge conflicts with main, all conflicts should be resolved before that
- Code should be understandable and contain documentation
- Code should not include global state

Code review guidelines
- Has to be reviewed by 1 member from other 2 teams
- Pull request should have assigned reviewers within 24h once it's no longer DRAFT and should be reviewed (either approve or require changes) within 48h
- Pull requests should include commit messages describing the work you've done and steps you did to verify your work
- Pull request will be reviewed by using the Review Changes button under the Files changed tab
- Pull request should only be merged/rebased by the creator of pull request
- Branch that has been successfully merged/rebased to main should be deleted by the creator of the pull request
- Reviewers should follow the work quality guideline to review the code

Acceptance criteria definition guidelines
- It should be defined from the user's point of view
- It should contain a list of steps to test the desired functionality

Definition of done (for user story)
- Acceptance criteria should be defined for the user story and pass
- All related code has passed code review and merged to main

| Development Process

- All tests are required to pass in CI before landing a pull request
- Sprint lifecycle:
    - Sprint Kickoff:
        - Review and re-estimate tasks: user stories get t-shirt size and priority
    - Development:
        - Feature kickoff:
            - Specify test cases and acceptance criteria
            - Tasks are estimated in number of days to complete using the magic estimation approach
        - Code reviews:
            - Require tests pass in CI before merging
            - At least one other RedBack member must approve the pull request before it can be merged
            - At least one BoxJelly member working on the same piece of technology must approve the pull request before it can be merged
            - At least one BlueRing member working on the same piece of technology must approve the pull request before it can be merged
            - All test cases and acceptance criteria identified in kickoff must be satisfied
        - Use auto-formatters to maintain code quality
        - Branching
            - Use the format `feature/t-<ticket>` as a feature branch template, where `<ticket>` is the Trello card number
            - `<username>/idea` for scratch / experimenting branches
            - `main` is the main branch
            - We will follow the following guidelines: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow (we probably only need main, release, scratch, and feature branches)
    - Release management:
        - Deploy as required

| Frontend workspace structure proposal

Proposed by	Ghina Yashar
Presented to	All frontend contributors from teams RedBack, BlueRing and BoxJelly
Proposal date	September 1 2022
Status	APPROVED - September 2 2022
Approvers	▪ Mamta Lopes(RedBack): 1/9/2022 ▪ Felipe Lin (BoxJelly): 2/9/2022 ▪ Rui Zhang (BlueRing): 2/9/2022

Proposed Structure

Please note: All the names used below can be replaced if needed, the focus of this proposal is more on the structure rather than the naming.

Summary video

If you don't like reading, please watch the video below for a quick overview of the proposed structure. The sample code snippets shown in the video are copied below as well.

Summary in writing

The structure I'm proposing would follow this rough directory tree:

Sample directory structure

```
|_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:

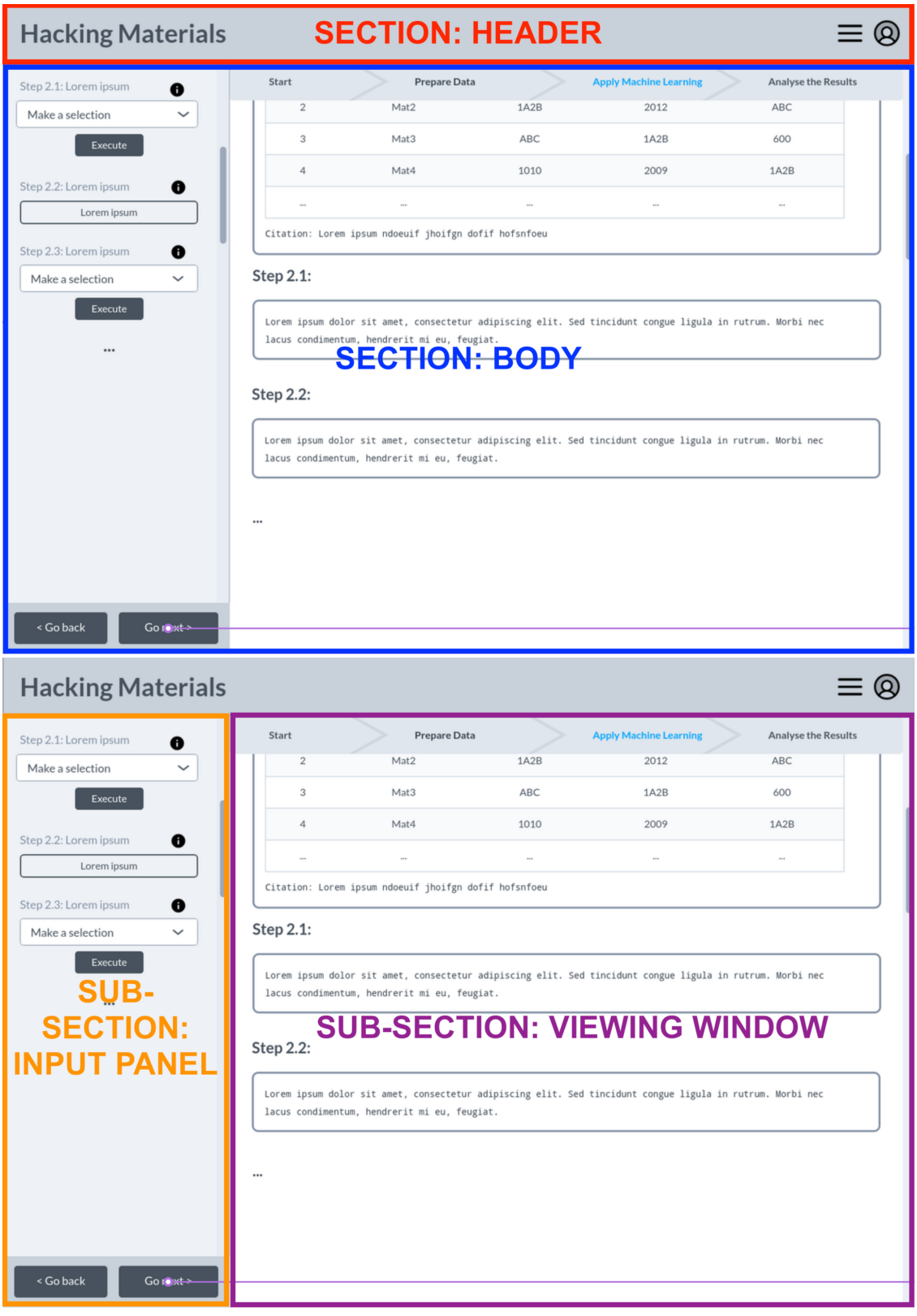
`Sections`

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`

AMENDMENT - 2 SEPTEMBER 2 2022

Felipe Lin (HA-BoxJelly) suggests potentially adding an `appFooter` as well. No objections to this so far.



`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

Sample src/steps/datasetSelection/index.tsx

```
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}
                />
        );
};
```

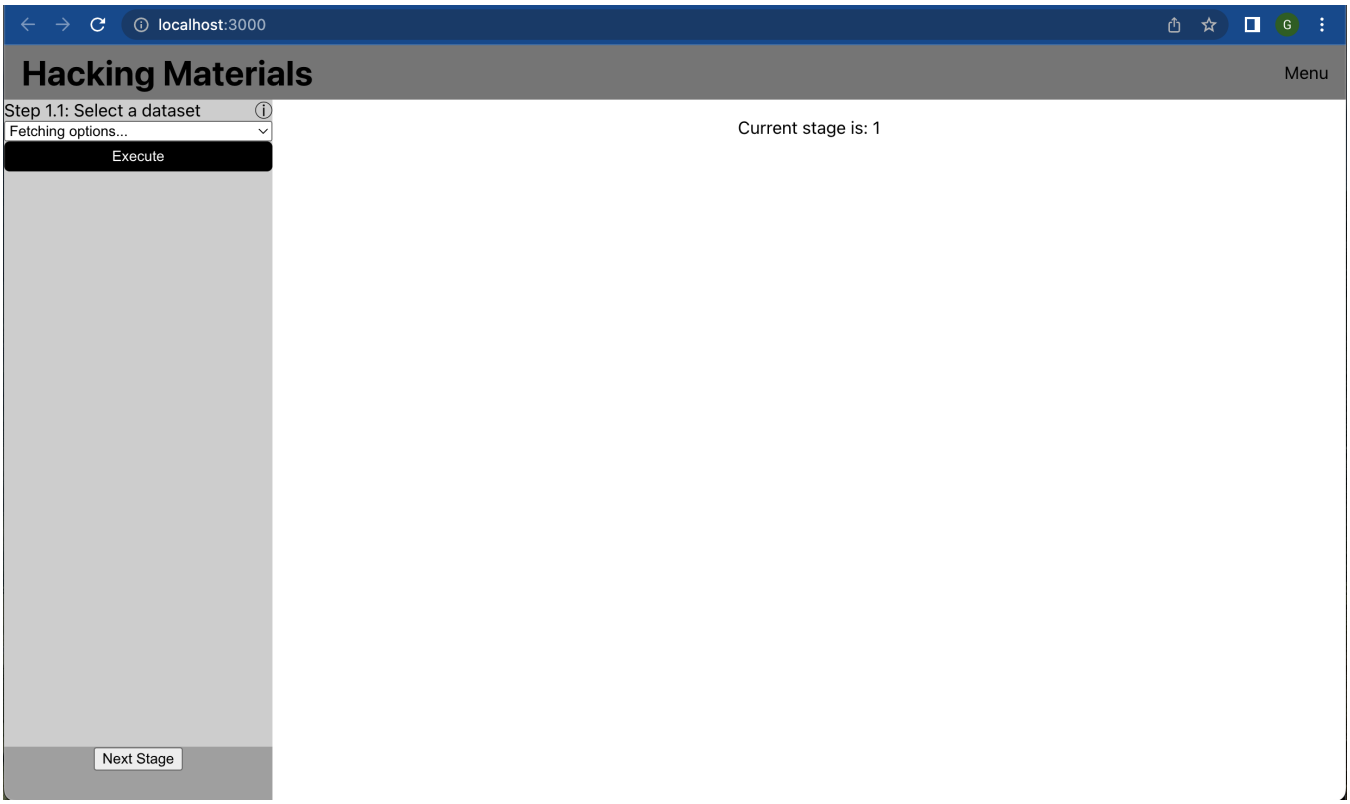
Sample src/sections/appBody/InputPanel/index.tsx

```
import DatasetSelectionStep from '../../../steps/datasetSelectionStep';
import FeatuirzerSelectionStep from '../../../steps/featuirzerSelectionStep';
...

const InputPanel = (props) => {
        ...
        const { stage } = props;

        if (stage === 1) {
                return (
                        <div>
                                <DatasetSelectionStep
                                        stepNumber="1.1"
                                        data={data}
                                        handleChange={handleChange}
                                />
                                <FeatuirzerSelectionStep
                                        stepNumber="1.1"
                                        data={data}
                                        handleChange={handleChange}
                                />
                                ...
                        </div>
                );
        } else if (stage === 2) {
                return (
                        <div>
                                ...
                        </div>
                );
        }
};
```

Sprint 2 summary

Overview

- User stories 21-selecting a dataset, 32-browsing featurizers, 34-browsing datasets are in progress and close to completion
- Majority of the foundational development has been completed, including:
  - [ ] Running development services for both frontend and backend
  - [ ] Running postgress database
  - [ ] Test framework
  - [ ] Docker set up for backend and database services
  - [ ] Documentation generation
  - [ ] Page skeleton
  - [ ] Frontend input collection architecture
  - [ ] Frontend components Storybooks (for automated documentation on reusable components)



Analysis

Most of the tasks derived from our assigned user stories are either done or under review. However, we are slightly behind schedule for multiple reasons.

- One of them is that we found there are varying levels of experience with the technologies we chose. As such, a lot of time were spent on learning and gathering tutorials for members across all 3 teams. Moreover, the experienced team members dedicated a lot of their time to make sure everyone can get to the level required for making meaningful contribution. Going forward, we expect that this should be less of an issue as team members become more comfortable with the technologies.
- As we were getting started with the development, we realised we need to spend some time to standardise processes and practices. Since this work has been completed now, it should no longer be a problem for the next sprint.
- During the sprint, code reviews were taking longer than expected as people were initially hesitant about code reviews, because they did't feel confident enough to assess the work of others. The situation was getting better as people gaining more knowledge about the technologies. We are trying to address this issue by introducing code review guidelines.