

1. Home	2
1.1 Development	4
1.1.1 Quality Assurance Guidelines	5
1.1.2 Development Process	6
1.1.3 Frontend workspace structure proposal	8
1.2 Specifications	13
1.2.1 Project Overview	14
1.2.2 Personas	16
1.2.3 Motivational Model	19
1.2.3.1 Do/Be/Feel lists	20
1.2.3.2 Goal Model	21
1.2.4 User Stories	22
1.2.5 Prototypes	25
1.2.6 Notes on provided material	27
1.2.7 Acceptance Criteria	28
1.2.8 Test Cases	29
1.3 Sprint Artefacts	33
1.3.1 Retrospectives	34
1.3.1.1 Sprint 1 Retrospective	35
1.3.1.2 Sprint 2 Retrospective	36
1.3.2 Presentation	37
1.3.3 Sprint 2 summary	38
1.3.4 Sprint 3 Summary	39
1.3.5 Client Communications	40
1.4 Resources	43
1.4.1 Links and Tools	44
1.4.2 Contact Information	45
1.4.3 Learning Resources	46
1.4.3.1 Accessing workflow related data	47
1.4.3.2 Docker	48
1.4.3.3 Git	50
1.4.3.4 GitHub	51
1.4.4 Skills Assessment	52
1.5 Milestones & Deliverables	53

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 methodologies to compare candidate materials and make recommendations based on how they each satisfy the specific use case requirements. Some of these methodologies 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 engineers 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





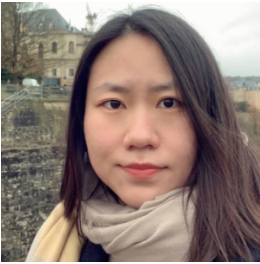
Alastair Davis
([Alastair Davis](#))

Team Representative



Sanjeevani
"Sanjee" Avasthi
([Sanjeevani Avasthi](#))

Developer

 <p>Mamta Lopes (Mamta Ritesh Lopes) Developer</p>	 <p>Ghina Yashar (Ghina Yashar) Frontend Development Lead</p>	 <p>Chunbaixue "Caitlyn" Yang (Chunbaixue Yang) Scrum Master</p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Supervisor: Mauro Mello Jr

Lecturer: Cristoph Treude

COMP90082 Software Project Subject

Coordinator: Eduardo Araújo Oliveira

* 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).

| Development

- | [Quality Assurance Guidelines](#)
- | [Development Process](#)
- | [Frontend workspace structure proposal](#)

| Quality Assurance Guidelines

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:
 - a. Be assigned reviewers within **24 hours** of being submitted
 - b. 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 summarise 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 Process

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
- 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.

Sprint Retrospectives

Each sprint should end with a sprint retrospective. Each team has done this slightly differently, in the case of RedBack we have used a "Start / Stop / Continue" model.

The team representatives also conduct separate retrospectives to discuss team interactions, which used a "What went well / What didn't go well / What was confusing" model.

Releases

Versioning

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

| Frontend workspace structure proposal

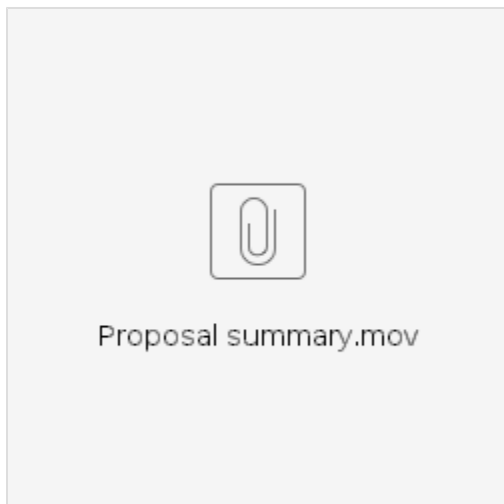
Proposed by	Ghina Yashar
Presented to	All frontend contributors from teams RedBack, BlueRing and BoxJelly
Proposal date	September 1 2022
Status	APPROVED - September 2 2022
Approvers	<ul style="list-style-type: none">▪ Mamta Lopes(RedBack): 1/9/2022▪ Felipe Lin (BoxJelly): 2/9/2022▪ Rui Zhang (BlueRing): 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 2022

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

Hacking Materials **SECTION: HEADER**

Step 2.1: Lorem ipsum

Make a selection

Execute

Step 2.2: Lorem ipsum

Execute

Step 2.3: Lorem ipsum

Make a selection

Execute

...

< Go back

Go Next >

Start

Prepare Data

Apply Machine Learning

Analyse the Results

2	Mat2	1A2B	2012	ABC
3	Mat3	ABC	1A2B	600
4	Mat4	1010	2009	1A2B
...

Citation: Lorem ipsum ndoeuif jhoifgn dofif hofsnfoeu

Step 2.1:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed tincidunt congue ligula in rutrum. Morbi nec lacus condimentum, hendrerit mi eu, feugiat.

Step 2.2:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed tincidunt congue ligula in rutrum. Morbi nec lacus condimentum, hendrerit mi eu, feugiat.

...

Hacking Materials

Step 2.1: Lorem ipsum

Make a selection

Execute

Step 2.2: Lorem ipsum

Execute

Step 2.3: Lorem ipsum

Make a selection

Execute

...

< Go back

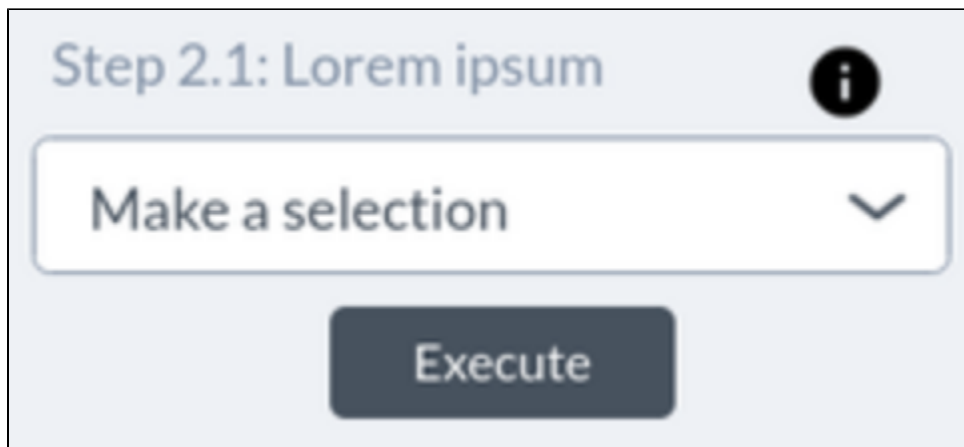
Go Next >

SUB-SECTION: VIEWING WINDOW

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>
    );
  }
};
```

| Specifications

- | [Project Overview](#)
- | [Personas](#)
- | [Motivational Model](#)
- | [User Stories](#)
- | [Prototypes](#)
- | [Notes on provided material](#)
- | [Acceptance Criteria](#)
- | [Test Cases](#)

| 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 engineers 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 methods 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 academic 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.

Project Scope

In this first semester of development (through the end of Sprint 3), 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.

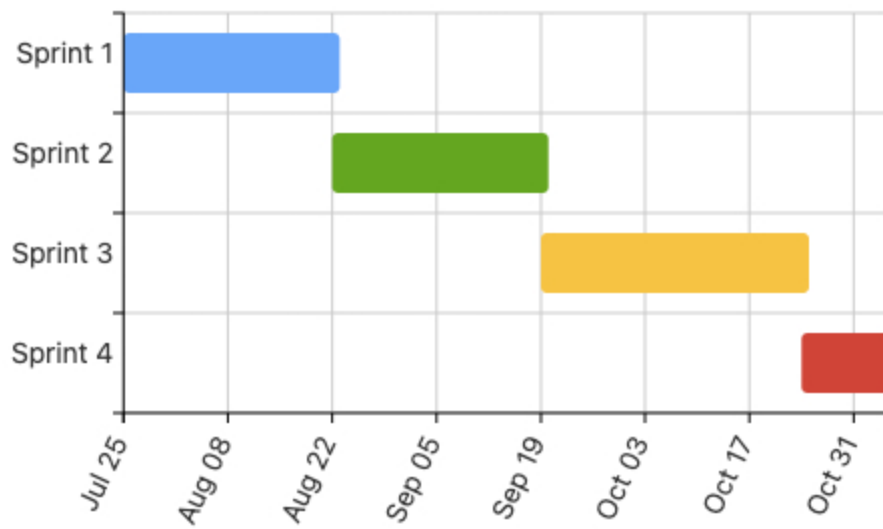
The development language is Python. We're also using Pytest and Sphinx for running unit tests and generating documentation.

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.

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: Master's 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



MATERIALS 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

Values

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

Wants

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

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: Materials Science and Engineering

occupation: Student

marital status: Single

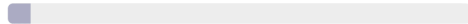


"It's SO time consuming to do material research and get decent results through just a semester."

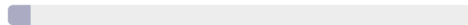
Motivation: As a materials engineering graduate student, Assol gets frustrated and demotivated when she can't make sense of the data she has and feels that she is not really making real progress with her studies. She needs a tool that can speed up materials data retrieval and processing so she can focus in analysing the results to better understand the theory and concepts of materials engineering.

Comfort With Technology

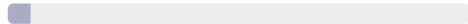
PROGRAMMING WITH PYTHON



MACHINE LEARNING



CLOUD BASED STORAGE



MATERIALS 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 materials 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

Values

- Convenience
- Quickness
- Safety
- Understandable

Wants

- A data mining application that helps her research projects
- 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

Fears

- Spends hours working on a research project with very little progress 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

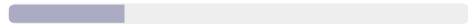


"It is fantastic to apply a multi-function online tool with ML methods if it is efficient and reliable. Nobody will refuse a tool that can save time"

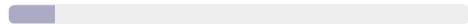
Motivation: Gray Zhou is a R & D Engineer of polymers in a battery factory. His work is searching for better materials for battery production. Gray spends a lot of time testing different materials, but some of tests are waste of time because of the poor performance observed. He needs a system that can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

Comfort With Technology

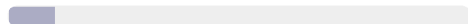
PROGRAMMING WITH PYTHON



MACHINE LEARNING



CLOUD BASED STORAGE



MATERIALS 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

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

Wants

- Ability to interact with the graph to further compare several 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

Fears

- Not enough guidance in the web or tool so him may feel confused to find functions he wants.
- Lacking understand of what the system done, then reducing the confidence level of his report
- Frequently unable to access the system

Change log

Version date	Version	Editor	Comment
21 Aug 2022	1.3	Ghina Yashar	Set up the page.
22 Sep 2022	1.6	Chunbaixue Yang	Added the personas.
19 Sep 2022	1.16	Sanjeevani Avasthi	Added updated personas.
20 Oct 2022	1.17	Sanjeevani Avasthi	Added updated personas for BlueRing and BoxJelly.
20 Oct 2022	1.18	Alastair Daivis	Added updated Redback's persona.

| Motivational Model

- [Do/Be/Feel lists](#)
- [Goal Model](#)

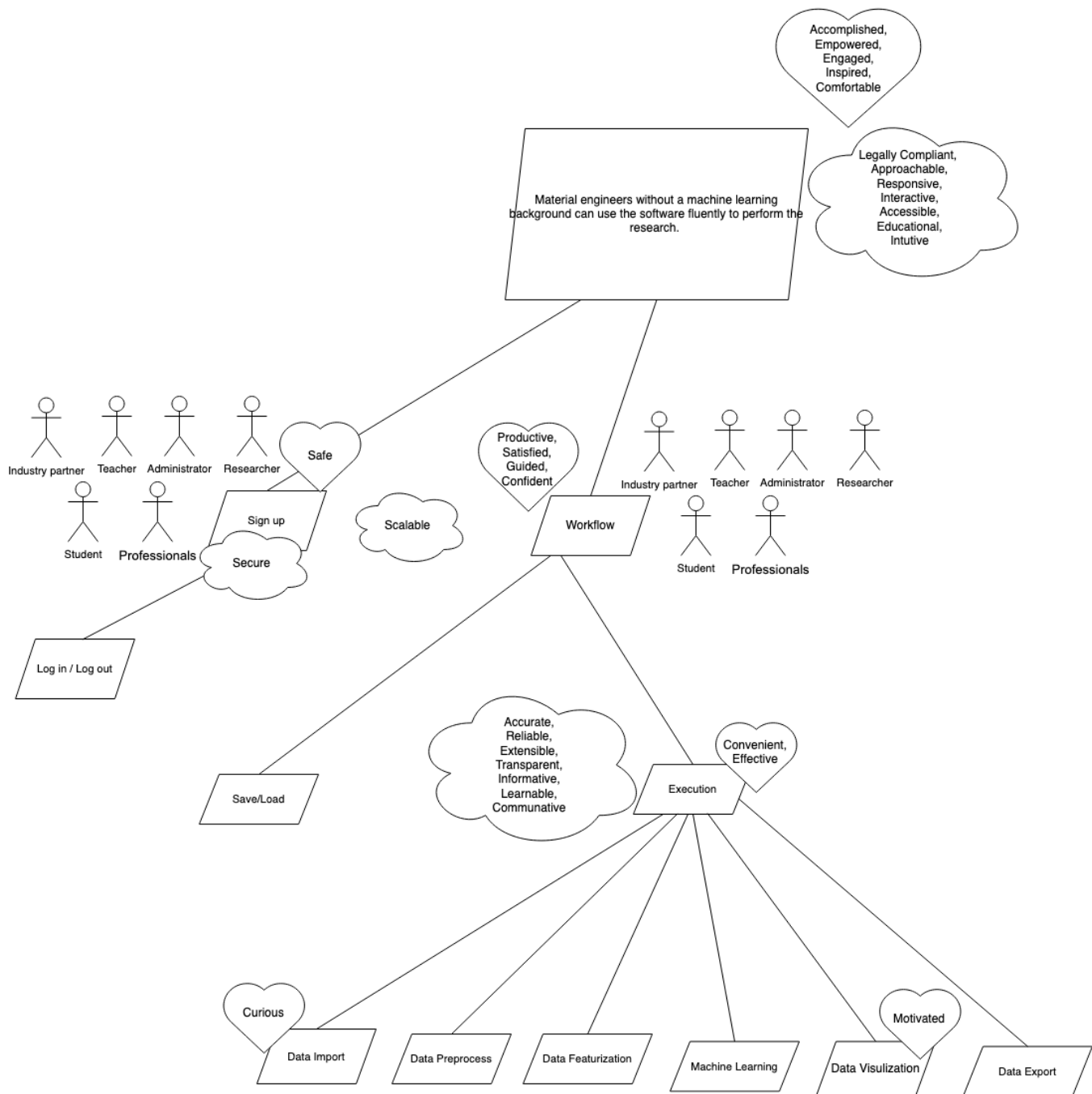
Do/Be/Feel lists

Version 2.0 - (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
Researchers	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: Create 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 2.0 - Sprint 3 Final (cross-team effort)



| User Stories

- [Version 3.0 - \(cross-team effort\)](#)
 - [Epics & Owning team allocation](#)
 - [User Stories](#)
- [Change log](#)



Prioritization Technique

We used the MoSCoW prioritization classification.

Must have - must be included in the scope of the project, we defined this all the must have user stories can create a minimum viable product

Should Have - should be included in the scope of the project

Could Have - could be included in the scope of the project

Won't Have - will not be included in the scope of the project

Version 3.0 - (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
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
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
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	2 - Should have
21	As a	general user	I want to	be able to select datasets from existing databases	so that	I do not have to worry about how the data is loaded		3	1 - Must have
37	As a	general user	I want to	be able to preview the output of each execution	so that	I could explore the data		1	2 - Should have
41	As a	general user	I want to	view the columns of the saved dataset and select the column I want to featurize	so that	I can provide the correct input to the featurizer		3	1 - Must 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
40	As a	general user	I want to	select features used for x and y axis	so that	I can plot the chart based on features I'm interested in		1	1 - Must 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 additional 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

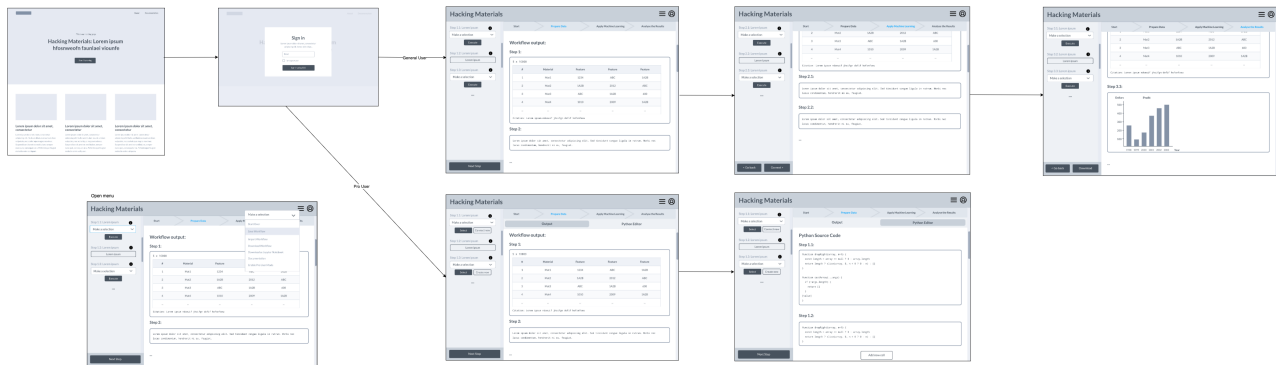
Change log

Version date	Version	Editor	Comment
19 Sep 2022	1.3	Chunbaixue Yang	Update US 32 to include featurizer selection as well
23 Sep 2022	1.4	Chunbaixue Yang	Update user stories after cross team meeting
20 Oct 2022	1.5	Mamta Ritesh Lopes	Changed the order of US40 and 22 according to spreadsheet and fixed typos
21 Oct 2022	3.0	Sanjeevani Avasthi	Updated the version number for consistency.

| Prototypes

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/product
 - 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.

User Testing Prototypes

After we divided the workflow steps into three stages, it became unclear when and how the user would execute the workflow. After asking the client what he preferred, we realised that this is a more complex decision than we expected and it would need more consideration. To help the client reach a decision, we prepared three functional low-fi prototypes for user testing. Each prototype represented a different approach. The approaches were as follows:

- Approach A - Run workflow steps individually. Prototype: <https://marvelapp.com/prototype/2g445gdg/screen/88805093>
- Approach B - Run the full workflow at once at the end. Prototype: <https://marvelapp.com/prototype/2g445gdg/screen/88805697>
- Approach C - Run each stage (which includes a number of steps) individually. Prototype: <https://marvelapp.com/prototype/2g445gdg/screen/88805736>

The prototypes were shared with the client, but as he was unavailable at the time, we decided to preemptively adopt Approach C as it offered the best balance between . Once the client was able to respond, he agreed with our decision. We then implemented Approach C, using the prototype as a guide.

| Notes on provided material



Note

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).

Organized Wishlist

As described in [the provided notebook](#) and the rough prioritization 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

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

Unknown priority features:

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

General characteristic:

- Should be usable by engineer or postgraduate 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 <i>matminer.dataset</i>	UI to select dataset from available options
1.0 B	Select a dataset from an online resource with <i>matminer.data_retrieval</i>	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 <i>describe()</i>	"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 <i>citations()</i>	List required citation (<i>.citation()</i> 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?

| Acceptance Criteria

Acceptance Criteria

USER STORY ID	USER STORY	GIVEN	WHEN	THEN
32	Browse 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
32	Select built-in featurizers	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
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
41	Browse the columns of the saved dataset	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
41	Select the column I want to featurize	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
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
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

Change Log

Date	Version	Author	Comment
14 Oct 2022	1	Chunbaixue Yang	Moved acceptance criteria from User Story to a separate page, also merged BoxJelly's work
20 Oct 2022	1.1	Chunbaixue Yang	Removed user stories we didn't work on

| 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

Test Type	Functional
Execution Type	Manual
Objective	Users are able to view citations, authors, and help text for featurizers
Setup	Standard setup
Execution Steps	<ol style="list-style-type: none"> 1. Log in using a Melbourne Uni student account 2. Select a featurizer from the featurizers dropdown menu 3. Click the "Featurizer Details" button 4. Verify: <ol style="list-style-type: none"> a. Citations are shown for the selected featurizer b. Implementors are shown for the selected featurizer c. Help text is shown for the selected featurizer d. A list of generated labels is shown for the featurizer
Notes	<p>A good featurizer for testing this is "Miedema"</p> <p>There's no guarantee that every featurizer has all four of the items to be verified, but that one has all of them.</p> <p>Remember to check the browser and backend logs for errors.</p>
Time constraint	1m per tested featurizer

US32

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

US34 and US21

Test Type	Functional
------------------	------------

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

US19 - Not kicked off**US37 - Not kicked off****US41**

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

US25 - Not kicked off**US13 - Not kicked off****US28 - Not kicked off****US18 - Not kicked off****US1 - Not kicked off****Foundation**

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

Execution Steps	<ol style="list-style-type: none"> 1. Log in using a Melbourne Uni student account 2. In step 1.1, select a dataset and click "Save" 3. Verify: a "Your selection was successfully saved" message appears 4. Step through to stage 2 5. Verify: the output from step 3 is still visible 6. Step through to stage 3 7. Verify: the output from step 3 is still visible 8. Step back to stage 1 9. Verify: the dataset previously selected is still selected
Notes	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.
Time constraint	2m

Foundation

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

US23

Test Type	Functional
Execution Type	Manual
Objective	Users can execute a featurizer
Setup	Standard setup
Execution Steps	<ol style="list-style-type: none"> 1. Log in using a Melbourne Uni student account 2. Select a dataset from the step 1.1 dropdown, and click "Save" 3. Verify: <ol style="list-style-type: none"> a. A "Your selection was successfully saved" message appears b. The dataset columns dropdown is displayed in step 1.2, and has content c. The list of columns matches those listed by Matminer 4. Select a dataset column from the step 1.2 dropdown, and click "Save" 5. Verify: The dataset column selection is communicated to the backend and the database is updated 6. Select a featurizer column from the step 1.3 dropdown, and click "Save" 7. Verify: The featurizer selection is communicated to the backend and the database is updated 8. Click "Run Stage 1" 9. Verify: <ol style="list-style-type: none"> a. The "Run Stage 1" button becomes disabled b. Job execution begins (Use Docker logs or Flower to verify this) c. Job execution ends successfully d. The "Run Stage 1" button becomes enabled again e. A success message appears in the output pane

Notes	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: <ul style="list-style-type: none">▪ Dataset: Elastic Tensor 2015▪ Column: Formula▪ Featurizer: StrToComposition
Time constraint	12m

| Sprint Artefacts

- | [Retrospectives](#)
- [Presentation](#)
- [Sprint 2 summary](#)
- [Sprint 3 Summary](#)
- | [Client Communications](#)

| Retrospectives

- [Sprint 1 Retrospective](#)
- [Sprint 2 Retrospective](#)

Sprint 1 Retrospective

Start	Stop	Continue	Actions
developing!	Doing things instead of asking others (when appropriate)	Communicating	Team to identify the tasks that can be divided
Having a clear division of tasks	Pushing directly to main branch	Taking initiative with meeting notes	Sanj to set up slackbot for stand up reminder, include specific message we define
Using development processes	Assuming all people know the technology you are talking about	Showing leadership	Add actional tasks on Trello for set up work
Async standup in slack		having a well thought out plan	Suggest on slack that one member from the host team should take notes and share it after the meetings
Using our trello board more		taking initiative in any manner	Set up test runner for our repo
Regular email communication with client		Keeping our Confluence space on point	Come up with a list of tutorials that can be useful for all by Thursday 1 Sep
collaborating with people from respective development team		sharing knowledge between teams	Ask Mauro again about how we will be assessed with the confluence submission
Documenting learnings / reflections in confluence		take turns in organising the meetings	Suggest on slack about using team emojis for pull request and setting up profile
Researching and learning new technology			
Finish setting up repository			
Collecting questions from all team members			
Schedule the second recurring meeting			
Telling Mauro what we want instead of asking			

Sprint 2 Retrospective

Start	Stop	Continue	Actions
Acceptance criteria	Reviewing submission criteria only just before due date	Grinding out work	Start doing feature kickoff: define acceptance criteria(PO involved) + both frontend and backend developers working on the feature should have a discussion, notes about decisions made during this meeting should be shared among all teams
Start logging the documentation	Merge work from other feature branches	helping others when they are stuck	Add change logs to confluence pages
More design discussions + up-front design	Tag last minute	reviewing PR's	Estimate tasks and define deadlines during sprint kickoff, re-estimate during feature kickoff
Assigning timelines to tasks		Reach out to other team members directly	Go through sprint3 checklist every stand up
Seeing the submission checklist in the beginning			Keep confluence pages up to date, for example, publish meeting notes right after meetings
Documenting earlier /simultaneously			Message Mauro to ask about definition of "test cases"
Cover more details in kickoffs so everyone agrees on the implementation			Start meetings by picking someone to a) run the meeting and b) take notes. They are responsible for making sure the meeting finishes on time.
writing test cases			
define timelines for the tasks			
Breaking down the work into smaller independent PRs			
Timebox the meetings			
Define acceptance criteria			
Have both frontend and backend people involved in design discussion			
Estimate tasks			
Ask questions about checklist in the beginning of the sprint			

Presentation

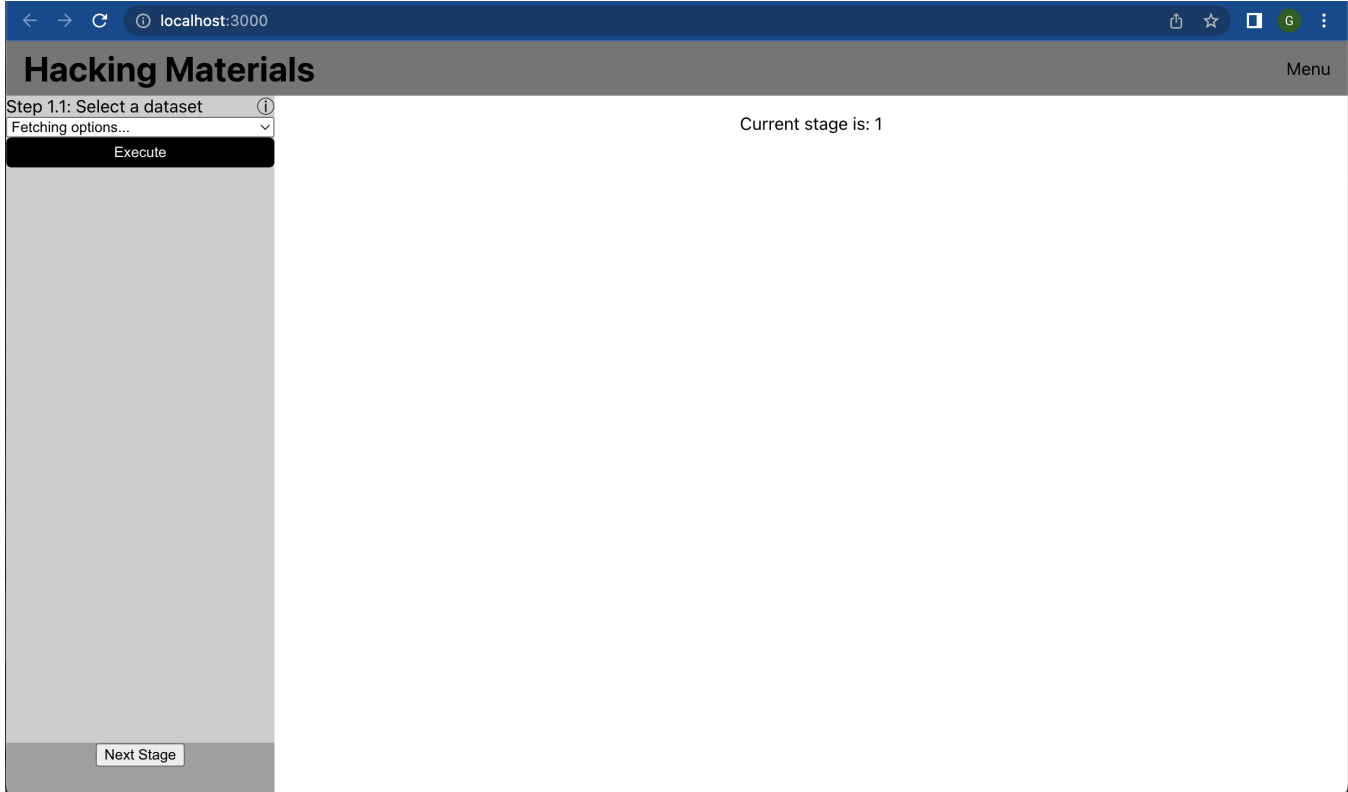


HA final presentation.pdf

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 postgres 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 was 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 not 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.

Sprint 3 Summary



Overview




- User stories 21-selecting a dataset, 32-browsing featurizers, 34-browsing datasets have been completed in sprint 3, in addition, user stories 23-stage 1 execution, 41-selecting dataset column and 30-view featurizer citations are completed. Our team member also helped BlueRing with their user story 35 for model selection
- We also did additional tasks needed for the project including:
 - Bug fixes for both frontend and backend
 - Styling for the website
 - Spike on how session management work in Flask
 - Backend support for CORS in development environments
 - Design common output container for the frontend
 - Configure code linters for the frontend with pre-commit hook
 - Created additional prototypes for the client to choose from
 - Set up Github Action for running backend unit tests
 - Created docker container for the frontend
 - Worked on code integration
 - Assisted other teams with coordination, planning and development
 - Prepare the final presentation for the client



In this sprint, we caught up with the development and finished all the features we initially wanted. We also deployed the website and demonstrated its functionalities to the client in the final presentation. The client was happy with the results we delivered.

For more detailed implementation progress, please refer to our Trello board: <https://trello.com/b/jZLcqPMi/ha-redback-project-board>

| Client Communications

Topic	Date	Archive
Kick-off Meeting	05 Aug 2022	 Kick-Off Meeting.pdf
Kick-off Meeting Agenda	12 Aug 2022	 Kick-Off Meeting Agenda.pdf
Client Meeting -2 Scheduled	15 Aug 2022	 Client-Meeting-2 .pdf

Updates - 1	25 Aug 2022	 Updates.pdf
Updates - 2	09 Sep 2022	 Updates-2.pdf
Client Meeting-3 Scheduled	13 Sep 2022	 Client-Meeting-4.pdf

Prototypes shared with 3 approaches	11 Oct 2022	 Approaches.pdf
Presentation Invite	18 Oct 2022	 Presentation Invite.pdf

| Resources

- | [Links and Tools](#)
- | [Contact Information](#)
- | [Learning Resources](#)
 - [Accessing workflow related data](#)
 - [Docker](#)
 - [Git](#)
 - [GitHub](#)
- [Skills Assessment](#)

| Links and Tools

- GitHub organisation: <https://github.com/orgs/COMP90082-2022-SM2/teams>
- GitHub team: <https://github.com/orgs/COMP90082-2022-SM2/teams/ha-redback>
- Confluence: <https://confluence.cis.unimelb.edu.au:8443/display/COMP900822022SM2HARedBack/Home>
- When2Meet: <https://www.when2meet.com>
 - Our pre-filled one based on our regular schedules: <https://www.when2meet.com/?16293461-QLKH3>
- Trello board: <https://trello.com/b/jZLcqPMi/agile-sprint-board>
- Other tools not yet set up.
- <https://edstem.org/au/courses/8857/discussion/>
- This Person Does Not Exist: <https://thispersondoesnotexist.com/>
- Persona design tools: Xtensio: <https://xtensio.com/>, UXpressia: <https://uxpressia.com/>

- <https://www.conventionalcommits.org/en/v1.0.0/> for a guide on writing commit messages
- <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow> for a guide on "Git Flow", a branching strategy

| Contact Information

Name	Calling name	Email	GitHub	Student ID	Discord
Alastair Davis (Team rep)	Alastair	adavis@student.unimelb.edu.au	alaroldai	359 101	alaroldai
Sanjeevani Avasthi	Sanjee	savasthi@student.unimelb.edu.au	sanjeevania	1101265	sanjeevani
Mamta Ritesh Lopes	Mamta	mlopes@student.unimelb.edu.au	MamtaL	1157314	Mamta
Ghina Yashar	Ghina	gyashar@student.unimelb.edu.au	GhinaY	1274878	Ghinaenae
Chunbaixue Yang	Caitlyn	chunbaixue@student.unimelb.edu.au	ycbx1993	1208333	KitKat

Client

Name	Email
Dr Christian Brandl	christian.brandl@unimelb.edu.au

Staff

Name	Email	GitHub
Mauro Mello Jr (supervisor)	mauro.mellojr@unimelb.edu.au	
Eduardo Oliveira	eduardo.oliveira@unimelb.edu.au	agogear

Sibling team contacts: BlueRing

Name	Email
Yanan Liu (Team rep)	yananl7@student.unimelb.edu.au
Hongpeil Lu	hongpeil@student.unimelb.edu.au
Jiahao Ju	jiahaoj1@student.unimelb.edu.au
Xinle Yu	xinley@student.unimelb.edu.au
Rui Zhang	rzzhan2@student.unimelb.edu.au

Sibling team contacts: BoxJelly

Name	Email
Zhaoqi Wang (Team rep)	zhaoqi@student.unimelb.edu.au
Dara O hEidhin	doheidhin@student.unimelb.edu.au
Yaoming Xuan	yaomingx@student.unimelb.edu.au
Radhimas Djan	djanr@student.unimelb.edu.au
Felipe Leefu Huang Li	fleefuhuangl@student.unimelb.edu.au

| 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(Quick guide from setup, creating virtual env similar to the Developer.md , connecting to SQLAlchemy and the frontend)
 - <https://github.com/realpython/discover-flask>
-

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](#)

Accessing workflow related data

WorkflowConfigurations 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 user 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 to 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 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.

Docker

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 to 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` & `.pgpassfile` is 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>

Git

Git

Introduction

I strongly recommend learning how to use Git from the command-line. You can find good resources for doing this on the internet, but one of the best places to start in my opinion is the Atlassian Git tutorial.

The initial part of this tutorial is mainly about setting up a repository, which we have already done and might not be so useful to you. I would recommend starting with [Saving Changes](#), which introduces the "three trees" of Git. Understanding how those trees work together is fundamental to understanding how Git works and how to use it effectively.

A second, also excellent resource is [Thoughtbot's Git Tutorial](#), which comes with written tutorials, recorded videos, and example repositories. In particular, their [end-to-end feature example](#) very closely matches how I like to use Git.

Branching strategy

Branches are key to using Git effectively. While the tutorials above give some good advice on using them, every team has a slightly different branching strategy so I'll mention some good guidelines here:

- **Always use a branch.** Branches keep you safe by making sure you can get back to a given state of the repository. If you're about to do something risky (for example, a `git amend` or `git rebase` command), make a branch first so that if it goes wrong you can check out the "before" version of your code and undo the mistake.
- **Do not modify the history of shared branches.** If you are sharing work on a branch with other developers, using a command like `git commit --amend` or `git rebase` can cause your version of the branch to become different from the version that the other developers have.
- Adopt a **common strategy for naming branches**. I like to use the following formula:
 - Use a slash (/) to separate different parts of the branch name
 - A brief description of the **kind of work** being done on the branch (`feat` for new feature development, `style` for code style changes, `doc` for adding documentation, `tests` for adding unit tests, and so on)
 - A short description of the **actual work** being done on the branch, for example `feat/backend-featurizer-list-endpoint`

Resources

- The official [git tutorial](#)
- Atlassian's [git cheatsheet](#)
- [Dangit, Git!?!](#) is a good resource for figuring out how to fix a mistake you've made using Git

Tools

While your IDE may have built-in Git support, I recommend using a stand-alone tool. They tend to be more stable and provide a good overview of the state of the repository.

I use [Sublime Merge](#), but other popular tools include:

- [SourceTree](#), by Atlassian
- [Tower](#)
- [tig](#), a command-line UI

Again though, I strongly recommend becoming familiar with Git's command-line interface. Often the GUI interfaces are convenient but hide information that can help you actually understand what they are doing, and knowing the command-line interface will help you use them more effectively.

Related to the above, if you are not yet familiar with a command-line editor I would strongly recommend learning one.

By far the most popular of these are Vim and Emacs, but both of these have quite steep learning curves. If you are looking for a command-line editor to get started with, the easiest is probably `nano`, which comes installed on most MacOS and Linux machines (I'm not sure about Windows, sorry!).

For those of you who are already confident with Git, some other tools you might find interesting / useful include:

- [Interactive Rebase Tool](#), a more convenient interface for using `git rebase -i`.
- [Trail](#), a tool I developed for rebasing feature branches on top of an amended commit after responding to code review feedback.

GitHub

GitHub

Introduction

GitHub fills two key roles in our project:

- Code hosting server: it acts as a Git server from which we can push and pull commits, and by doing so share our work
- Code review platform: it acts as a way for us to share feedback on each other's work, facilitating quality control, collaboration, and knowledge distribution.

If you're familiar with Git already, most of GitHub should not be new to you. The exception to this is GitHub's **Pull Request workflow**, which covers the process of merging changes from a feature branch into the main branch.



With this in mind, I recommend reading GitHub's [documentation on this topic](#).

Tools

- GitHub's [command-line interface](#) is very good
- [GitHub Desktop](#) is a desktop application, which you may find more convenient than using GitHub's website.

Skills Assessment

Email Address	Team	Matminer	React	Flask	Pandas	Scikit-learn	Numpy	Matplotlib	Seaborn	Plotly	Bokeh	Tensorflow	Keras	Pytorch	NodeJS	Pure HTML/CSS	Angular	Vue	Django	NodeJS	Any other suggestions for backend
Hongpei Lu	Bluering	0	0	0	2	3	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0
Jiahao Ju	Bluering	0	0	0	4	4	4	4	0	0	0	1	0	2	0	3	0	0	0	0	0
Xinle Yu	Bluering	0	0	0	4	4	4	4	4	4	2	3	3	3	1	1	0	0	0	0	1
Rui ZHANG	Bluering	1	2	3	4	3	4	0	0	1	0	1	3	3	2	1	0	0	0	0	0
Yanan Liu	Bluering	0	0	0	4	4	4	4	2	2	0	3	3	3	0	0	0	0	0	0	0
Dara O'Hedhin	Boxjelly	0	2	1	5	5	5	3	3	2	0	4	4	1	2	2	1	0	0	0	2
Yaoming Xuan	Boxjelly	0	0	0	4	5	5	0	0	0	0	1	1	3	0	3	0	0	0	0	0
Felipe Leolu Huang Lin	Boxjelly	0	4	2	1	1	1	1	1	0	0	0	0	0	4	4	3	0	3	3	0
Zhang WANG	Boxjelly	0	1	0	4	2	4	3	2	2	0	3	0	2	0	2	0	0	1	1	0
Radhima Djan	Boxjelly	0	0	3	4	4	4	3	3	1	1	3	3	3	2	2	0	0	3	0	0
Mama Ritesh Lopes	Redback	0	3	0	3	5	5	4	0	0	0	1	1	0	0	4	0	0	0	0	0
Ghina Yashar	Redback	1	3	0	4	4	4	4	0	4	0	4	4	4	2	4	2	1	0	2	0
Chunshuwei Yang	Redback	1	0	0	3	4	3	2	2	0	0	3	3	1	0	1	0	0	0	0	0
Sanjeevani Awasthi	Redback	2	1	1	5	5	5	4	2	4	0	3	3	4	0	3	0	1	0	0	0
Alastair Davis	Redback	1	1	2	2	2	3	2	2	2	0	1	1	3	2	2	1	2	1	2	0

Description	File
Skills Assessment Matrix	 <p>Team HA - Skills ...- Team Skills.pdf</p>
Skill Distribution chart	 <p>Team HA - Skills Assessment.pdf</p>

| Milestones & Deliverables

Tasks

Milestone 1: End of Sprint 1 by **Aug 21**

- ✓ Setup project tools (Confluence, Trello/Jira, GitHub, Slack)
- ✓ Research to prepare project background & overview
- ✓ Establish development & project processes & document it in README and Confluence
- ✓ Establish requirements with the client:
 - ✓ Motivational model
 - ✓ Goal model
 - ✓ Do-Be-Feel
 - ✓ Personas: 2-3 personas
 - ✓ User stories (including estimation and prioritisation)
 - ✓ Prototype
- ✓ Develop a plan for sprints 2 & 3:
 - ✓ Define scope
 - ✓ Migrate user stories to product backlog in task management tool
- ✓ Sprint retrospective

Milestone 2: First Presentation by **Sep 12**

- ✓ Presented

Milestone 3: End of Sprint 2 by **Sep 18**

- ✓ Sprint kickoff
- ✓ Development process:
 - ✓ Prepare test cases before starting each task
 - ✓ Updating Trello/Jira regularly
 - ✓ Standups
 - ✓ Code reviews
 - ✓ Testing
 - ✓ etc (as per established development process)
- ✓ Maintain documentation:
 - ✓ Meeting records - **NOT** to be uploaded to GitHub
 - ✓ Update estimates & priorities where applicable
 - ✓ Record decisions, events, etc
- ✓ Release management:
 - ✓ Deploy as required
 - ✓ Upload sprint documentation to GitHub
 - ✓ Update README with new release
 - ✓ Create baseline tag
- ✓ Sprint retrospective
- ✓ Sprint review with client (if required)

Milestone 4: End of Sprint 3 by **Oct 21**

Tasks:

- ☒ Sprint kickoff
- ☒ Development process:
 - ☒ Prepare test cases before starting each task
 - ☒ Updating Trello/Jira regularly
 - ☒ Standups
 - ☒ Code reviews
 - ☒ Testing
 - ☒ etc (as per established development process)
- ☒ Maintain documentation:
 - ☒ Meeting records - **NOT** to be uploaded to GitHub
 - ☒ Update estimates & priorities where applicable
 - ☒ Record decisions, events, etc
- ☒ Release management:
 - ☒ Deploy as required
 - ☐ Upload sprint documentation to GitHub
 - ☒ Update README with new release
 - ☒ Create baseline tag
- ☐ Sprint retrospective
- ☐ Sprint review with client (if required)

Milestone 5: Final Presentation by Oct 21

- ☒ Demonstration + Presentation

Milestone 6: End of Sprint 4 by Nov 4

- ☐ Sprint kickoff
- ☐ Development process:
 - ☐ Prepare test cases before starting each task
 - ☐ Updating Trello/Jira regularly
 - ☐ Standups
 - ☐ Code reviews
 - ☐ Testing
 - ☐ etc (as per established development process)
- ☐ Maintain documentation:
 - ☐ Meeting records - **NOT** to be uploaded to GitHub
 - ☐ Update estimates & priorities where applicable
 - ☐ Record decisions, events, etc
- ☐ Release management:
 - ☐ Release notes
 - ☐ Release TAG
 - ☐ ZIP File
 - ☐ Final Presentation Slides
 - ☐ High Fidelity Digital Prototype and Data Sample
 - ☐ Self-reflection
- ☐ Project retrospective

Deliverables

Milestone 1: End of Sprint 1 by **Aug 21**

GitHub repository with:

- Exported Confluence content
 - Cover page
 - Project details: background, overview, scope.
 - Requirements in the form of models, personas, user stories and prototypes
 - Technical details and decisions
 - Sprints 2 & 3 plans
 - Overview of the team's established processes and workflows
- [README file explaining the project details and workflows](#)
- A baseline tag for Sprint 1

Milestone 2: First Presentation by **Sep 12**

- Presentation slides

Milestone 3: End of Sprint 2 by **Sep 18**

GitHub repository with:

- Exported Confluence content
 - Updated project details
 - Updated requirements
 - Updated technical details and decisions
- Updated README file explaining the new release
- A baseline tag for Sprint 2

Milestone 4: End of Sprint 3 by **Oct 21**

GitHub repository with:

- Exported Confluence content
 - Updated project details
 - Updated requirements
 - Updated technical details and decisions
- Updated README file explaining the new release
- A baseline tag for Sprint 3

Milestone 5: Final Presentation by **Oct 21**

- Presentation slides

Milestone 6: End of Sprint 4 by **Nov 4**

- High fidelity digital prototype and data sample (the product)
- Self-reflection
- ZIP file containing:
 - Release notes
 - Downloaded release from the GitHub repository with:
 - Release tag
 - Updated README file
 - Final presentation slides
 - All project resources, including
 - Documentation
 - Tests
 - Data samples
 - Prototypes
 - Images and other assets