Faculty of Engineering, Computer
and Mathematical Sciences

SIVT 1

Initial Report

Software Engineering & Project

2020

Members:

Gabriel P.C.L De Almeida	a1673698
Jackson Dearing	a1686411
Rhett Hull	a1706579
Steve Monger	a1176574
Kristina Plavsa	a1723663
Colin Ross	a1704822
Joshua Tatton	a1690417
Elliot Wheatland	a1717968
Brian Yip	a1751117

Submitted on:

18^{th} August 2020

1 Project Vision

Propic is a company that provides analyst services for the Real Estate sector, using big data and artificial intelligence. Their business intelligence product, ReVeal AI, generates recommendations for real estate agents for listings and appraisals. Part of this development process involves analysts querying the system to facilitate answering business questions. The transformation process has been refined over time in order to facilitate answering increasingly sophisticated and abstract business questions. These refinements have introduced increased complexity, leading to difficulty in understanding, validating and risk-managing the transformation process.

The aim of the project is to provide a visualisation tool which allows analysts to easily understand the data transformation. For each SQL script provided by a user, this tool will derive the relevant table and data lineage. The information will be displayed in a web browser in a clear and intuitive manner. Interactive elements will allow users to highlight only the data lineage information which is related to an individual table. This will enable developers and the end user to better understand why, how and where recommendations are coming from. This will enable Propic to better refine their understanding and build more reliable and explanable systems.

2 Customer Q&A

The first customer Q&A was a general overview of the final product with the product owner and provided. The meeting provided the necessary starting information to begin the first sprint. The product owner described their final product as being a web application that allows developers and analysts to input a presto SQL file, from which an interactive data lineage will be visualized. For example, given an SQL script, it will be possible to trace the source and transformation of a column of data from a "Users" table with ease.

The following questions were asked during the kickoff meeting:

a. What is the domain of the provided input? (Aside from being SQL queries) Is there any other formal specification you have already?

The file input will be a SQL presto file.

b. What user action triggers the visualisation?

The output is automatically a visual representation, i.e. given an SQL file path, it should automatically load the visualization, without running other scripts.

c. Do you have a preference or disposition to any specific subset of available tools, programming languages or frameworks. If so, what are they?

No restrictions, open source is acceptable. Input scripts must be be Presto syntax.

d. Do you have a preference for the application to be available in the web browser or as a native application? If the latter, what operating system do Propic developers use?

Please use a system agnostic solution, such as a web application.

For the first sprint, the Product Owner recommended the development team focus on parsing the Presto SQL code. The first sprint iteration of the product should thus output a table of data showing which tables are using which data. As demonstrated in Section 7, the first sprint is dedicated to initial configuration of the system, as well as achieving this initial goal.

On reflection, the team determined the initial meeting was successful, as sufficient information was gathered to understand the Product Owner's vision, and the core objectives that we must achieve. All group members engaged with the product owner, asking insightful follow-up questions and collectively created a list of tasks and objectives.

However, the team determined that the scrum master should take primary lead during meetings as the primary spokesperson and chairman, to ease communication with a large development team. In particular, the scrum master should have been the person posing the questions formulated by the team, and delegating speaking responsibilities to other members in the group. This would have facilitated in-depth discussions with the product owner about their needs and allowed for everyone the chance to provide their input and pose relevant follow-up questions.

In hindsight, follow-up questions were posed to the client after internal discussions, which were as follows:

a. Who is the typical intended user of our tool? What kind of typical technical background do they come from? Are the staff low-level analysts or high-level management for example?

Users will be data analysts and data engineers whose role requires an in-depth understanding and frequent interaction of the data pipeline.

b. How is this tool going to assist and be integrated into their workflow?

It is going to be used in the maintenance of ETL (Extract, transform and Load Data) development process. This will be useful to see the complete data flow from origin to destination. The main goal is to provide a better understanding to the user of what happened to the data throughout the life cycle.

c. Can you confirm that a Web-based application is suitable for Propic?

Yes, a Web-based application is suitable. To make the deployment easier for us, I would prefer using technology like Docker. But I am no expert on it, happy to see what you recommend.

3 Users

The users identified by the development team after the kickoff meeting are Data Analysts and Developers.

Data Analysts: Typically, data analysts have a mathematical background with some software and computer science knowledge and experience. Propic data analysts are responsible for querying the system, and determining how the data was transformed, why the data was transformed and leveraging this knowledge to provide Real-Estate recommendations. The analyst shall be able to use the data visualisation tool to quickly identify the data lineage and its relations by inputting their necessary scripts. They will then be able to interact with the tool to further explore the relationships.

Developers: Similar to data analysts, developers have some mathematical background, although they have much more experience in the field of computer science. Propic developers develop and maintain the code and database queries for the tool. Mapping the lineage in an interactive way will provide them with a reliable method of validating and maintaining the extract, transform, and load process and reduce the time needed to analyse and understand the scripts.

4 Software Architecture

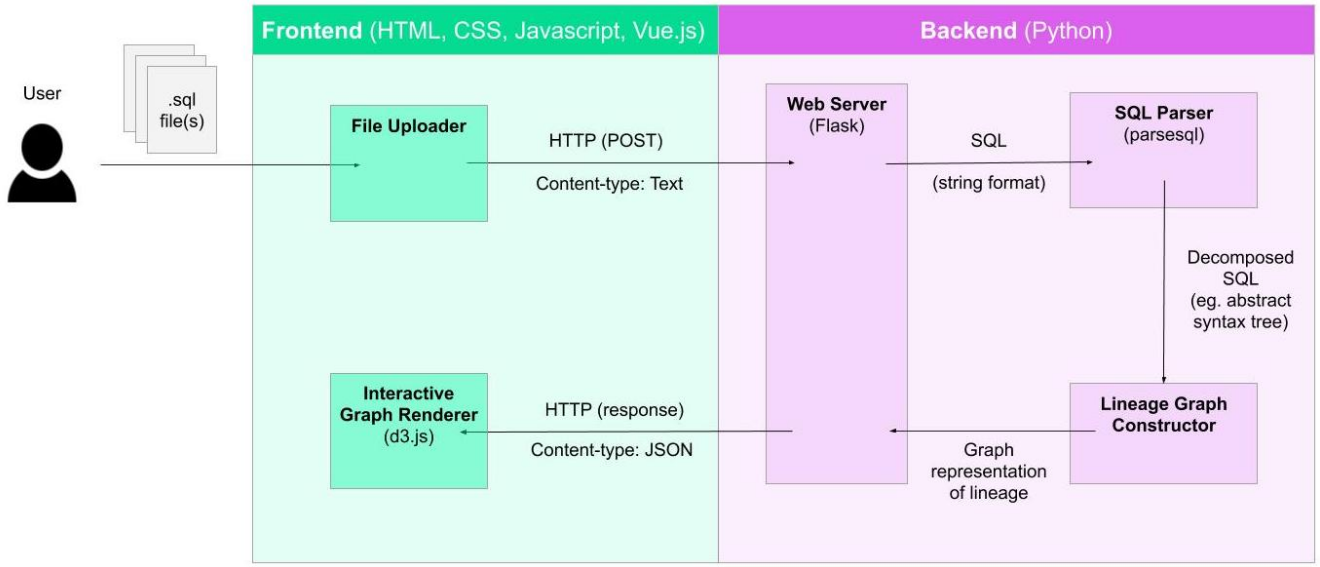


Figure 1: Application Architecture Schematic

The architecture for the application seen in Figure 1 represents a Client-Server web application. A web application was selected based on the following considerations:

- Existence of relevant, open-source data visualisation libraries in JavaScript
- Ability to implement back-end logic in Python (aligning with team member skills)
- Maximise applications' availability - running through the web browser makes it independent of any particular operating system

The Client-Server model allows for separation of the SQL analysis logic from the visualisation components. This separation facilitates incremental development and increases modularity which is ideal for large, collaborative projects.

For the average application usage, a user shall upload an sql file as input to the front-end File Uploader which will upload and convert the file content to string format. The file content shall be sent via HTTP POST to the Web Server which will forward the SQL content to a dedicated SQL Parser. In the back-end, the SQL Parser and Lineage Graph Constructor are responsible for

decomposing the SQL input and formulating a programmatic representation of a graph containing the lineage relationships. This graph of lineage information shall be sent back to the front end in JSON format. The Interactive Graph Renderer shall take the lineage information and produce an interactive visual in the web browser.

5 Tech Stack and Standards

5.1 Team Communications

Technologies Considered: Slack, Facebook Messenger, Email, Discord

Decision Criteria:

- Instant messaging
- Multiple text channels
- Voice channel support
- GitHub integration
- Currently used services

Chosen: Discord

Justification: Slack and Discord rated highest for instant messaging, text channel support and reliable voice channel service. Despite Slack having better integration capabilities with GitHub, Discord was chosen as it was already regularly used by team members. Adoption of yet another communication service may have over-saturated the communication space for team members and been detrimental to responsiveness.

5.2 Version Control and Remote Repository

Chosen: git with GitHub

Justification: Using git for version control and GitHub to host the remote repository was mandated for the project. This also extended to the use of GitHub Projects for project planning operations including issue tracking and sprint planning.

5.3 Front-End Technologies

Technologies Considered: Vue.js, Angular, React

Decision Criteria:

- Learning curve
- Development team experience
- Flexibility
- Scalability
- Performance
- Size

Chosen: Vue.js

Justification: In order to navigate the numerous criteria, a data-driven approach was taken by constructing a decision matrix seen in Table 1 which resulted in Vue.js being the most optimal front end framework. Vue is a light-weight JavaScript library used to develop interactive web interfaces. It has extensive and excellent documentation that makes the front-end development a smooth process for developers of all levels. One of Vues' features is its loosely coupled components, which allows the development team to create custom elements and re-use them later if needed. It also allows for imports of existing components from external sources, therefore decreasing development time. An important library that will be used in the front-end for http communication is called Axios.

5.4 Back-End Technologies

Technologies Considered: Python with either Flask or Django, Javascript with Express.js or Node.js

Table 1: Front-end Framework Decision Matrix

		Options					
Criteria	**Weight**	**Angular**		**React**		**Vue**	
		Score	Total	Score	Total	Score	Total
Learning Curve	6	3	18	4	24	5	30
DT Experience	6	3	18	2	12	5	30
Flexibility	5	2	10	6	30	6	30
Scalability	4	6	24	5	20	2	8
Performance	4	5	20	6	24	6	24
Size	2	3	6	2	4	6	12
	Total	**98**		**114**		**134**	

Decision Criteria:

- Ease of Use
- Development team experience
- Simplicity
- Compatibility with Front End Technology
- Size

Chosen: Python, Flask

Justification: The use of Python as the back-end language of choice was heavily influenced by the development teams' collective aggregate experience with Python. As shown in Figure 2, Python is one of the most commonly known languages amongst the team. Coupled with the fact that Python has a shallow learning curve in comparison to other languages, makes it the language of choice for back-end technology. The library chosen to be used with python is Flask. Flask excels as a lightweight back-end for simple web applications, whereas Django has a large number of features that are unlikely to be used for the project. This makes Flask an ideal choice for the scope of this project. These choices are summarized in the decision matrix shown in Table 2. **??**.

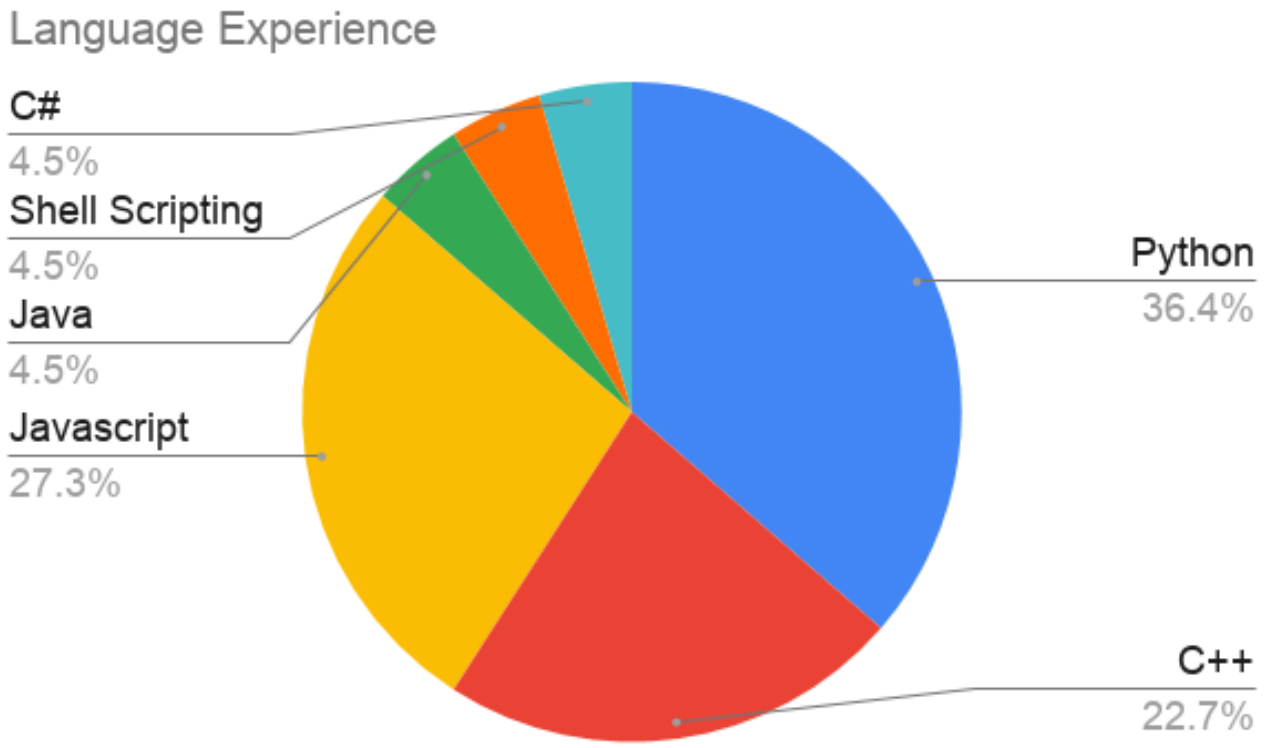


Figure 2: Development Team Language Experience

Table 2: Backend Technology Decision Matrix

		Python				**Javascript**			
Criteria	**Weight**	**Flask**		**Django**		**Express.js**		**Node.js**	
		Score	Total	Score	Total	Score	Total	Score	Total
Ease of Use	6	6	36	6	36	4	24	4	24
DT Experience	6	6	36	6	36	4	24	4	24
Simplicity	5	4	20	5	25	5	25	5	25
Size	4	6	24	4	16	6	24	5	20
Compatability*	6	6	36	5	30	6	36	6	36
	Total	**152**		**143**		**133**		**129**	

5.5 Coding Standards

Although Propic does not require the adoption of any specific technology or standard, the development team agreed that it would be good practice to still enforce coding standards to optimise

consistency, readability and maintainability. Given the tech stack decisions, this was predominantly relevant to Python code (back-end) and JavaScript code (front-end). For these, the PEP style guidelines will be used for python with a small addition for function commenting (available on the project wiki) as well as the Airbnb JavaScript style guide which will be assisted with static analysis by ESLint.

5.6 Integrated Development Environment

Technologies Considered: VSCode, PyCharm, Jupiter Notebook, Vim, Atom, Sublime

Decision Criteria:

- Compatibility for languages and frameworks to be used
- Static analysis extensions (specifically for PEP and Airbnb style guides)
- Development team experience and preference
- Version control integration

Chosen: VSCode (preferred guideline)

Justification: Given the above decisions on languages, frameworks and coding standards, it was decided that the preferred IDE for development will be VSCode. This is because it is free, straightforward to use and has good support for Python as well as JavaScript in terms of syntax highlighting and extensions for static analysis. It is also extremely versatile with a rich library of extensions which make it highly customisable. The universal use of a VSCode throughout the team is not being mandated as a range of other IDEs can also be configured in accordance with the projects' style preferences. This allows any developer to use their own favourite IDE, if preferred.

6 Group Meetings and Team Member Roles

Stand-up meetings are currently scheduled for two meetings per week. These are to be held every Monday and Friday until the completion of the project. During the later stages of the project, once the team has settled into their roles, a third stand-up meeting may be added. These stand-up

meetings are time-boxed to 15 minutes. This ensures that there is sufficient time for all team members to share their progress, intended work and blockers, whilst keeping the meeting short. Should any problems arise during the stand-up meetings, individual team members may schedule 'offline' meetings to work through potential solutions.

Client meetings, including the Sprint review and planning meetings, will be held with the tutor bi-weekly on Fridays at 1pm. The review and planning meetings will be combined to a single meeting due to the availability of contact time with the tutor (and client). This single meeting will be time-boxed to 25 minutes.

Sprint retrospectives are usually held between the sprint review and sprint planning meetings, however due to the nature of the single review-planning meeting, this will be held at a separate time. This retrospective meeting will be held after the review-planning in place of the 'daily' stand-up.

6.1 Scrum Masters

The scrum masters for each sprint are listed in Table 3. Scrum masters were selected through self-nomination and election and assigned at random to sprints.

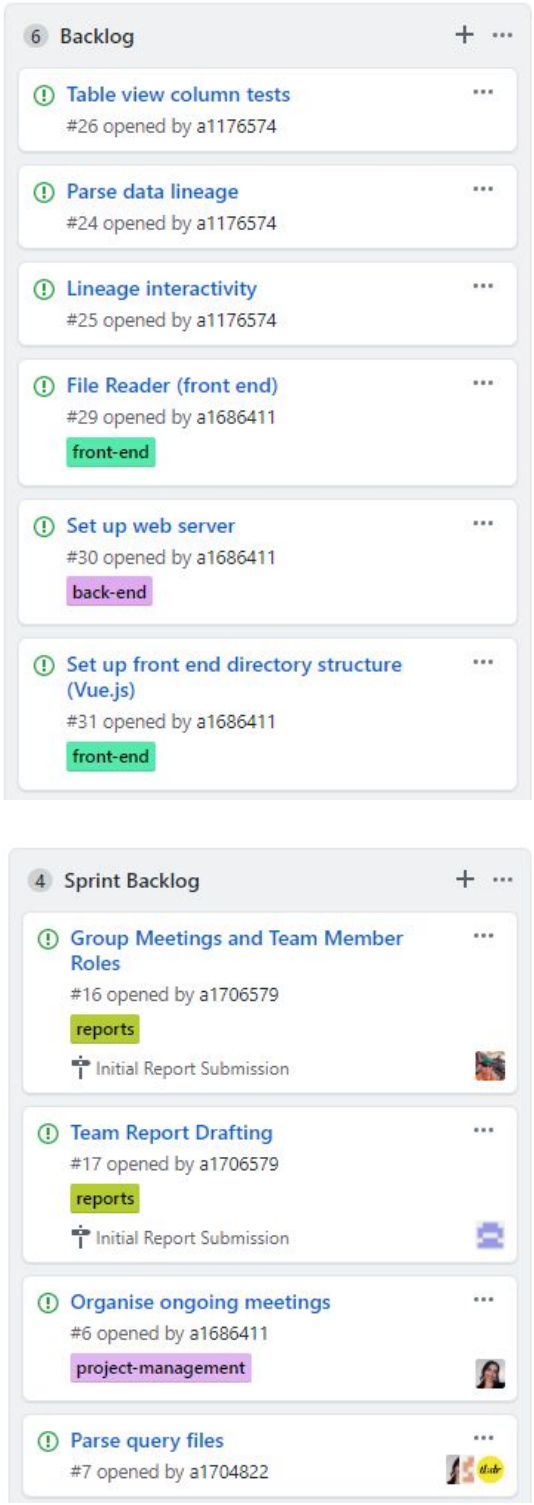
Table 3: Scrum Masters

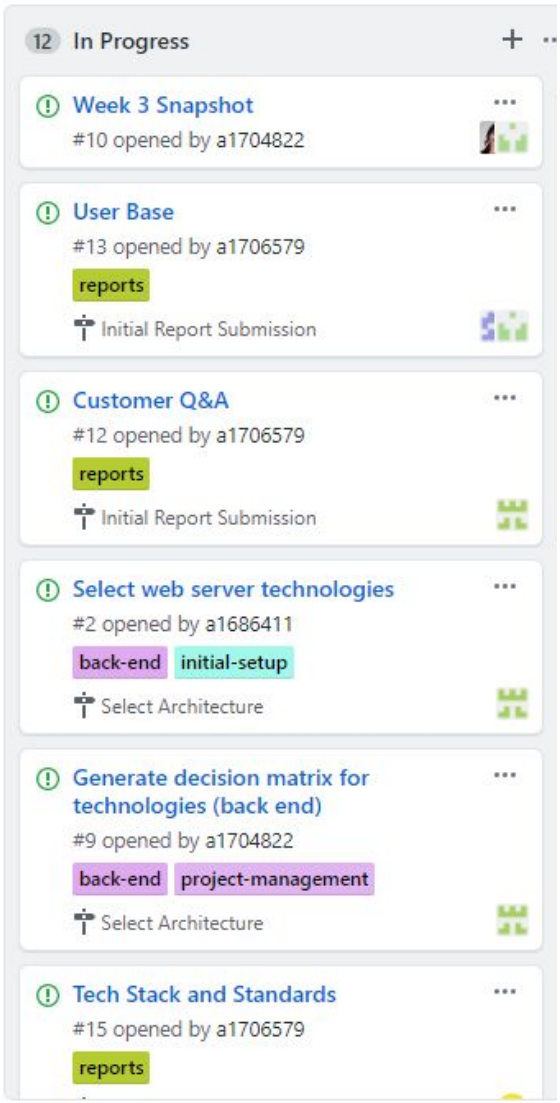
Sprint	Scrum Master
1	Colin Ross
2	Jackson Dearing
3	Joshua Tatton
4	Elliot Wheatland
5	Steven Monger

7 Snapshot

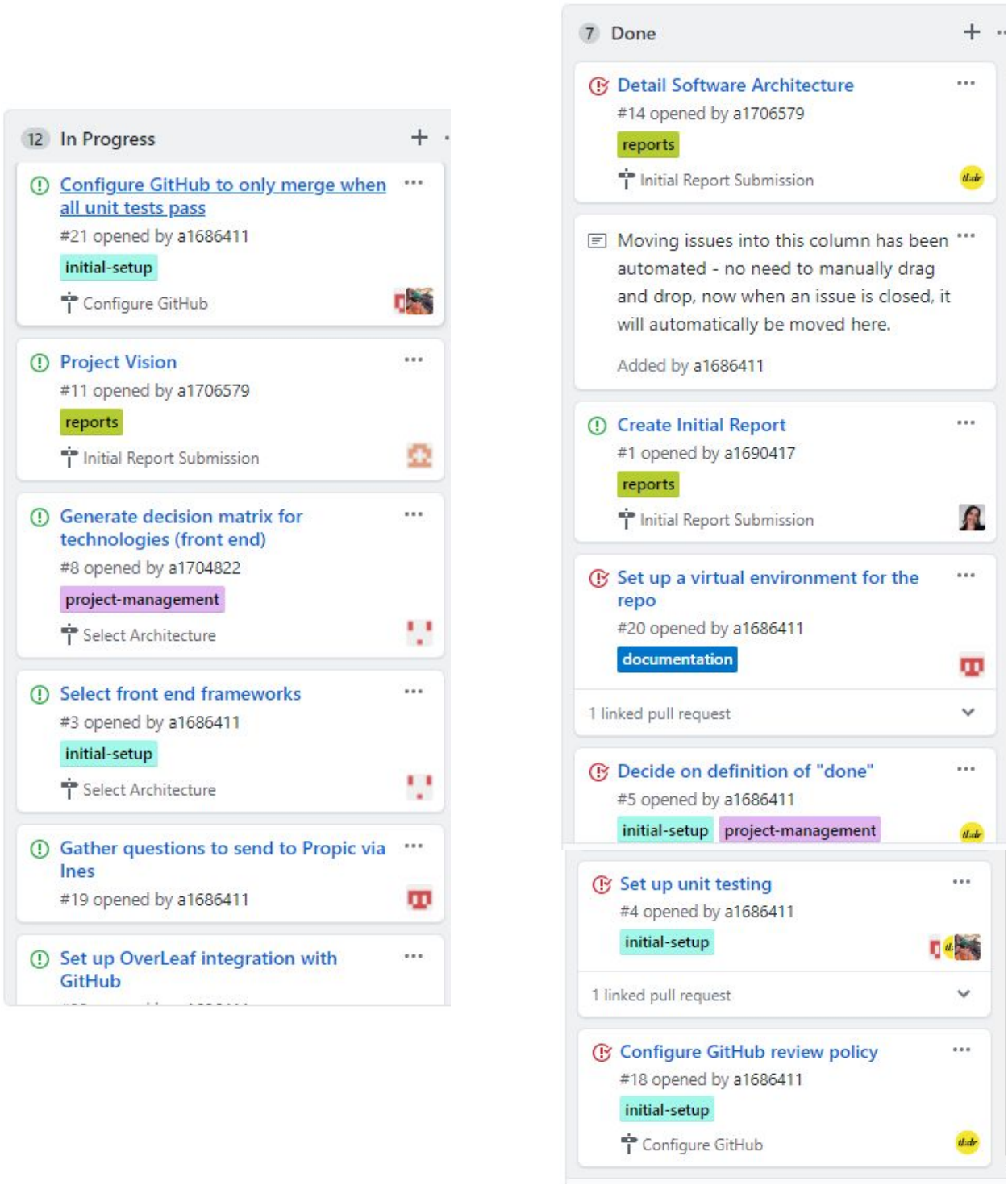
See pages 11 - 14.

Task Board Snapshot





Task Board Snapshot



User Stories In Scope for Sprint 1

Sprint 1 Goals

- To configure the GitHub repository, and set up the project backlog with tasks.
- To develop the necessary backend to receive and parse an SQL script to output the associated columns, tables and views.
- To compose a preliminary report describing the proposed architecture and workflows.

Primary Tasks:

- Configure python virtual environment for the repository.
- Set up unit testing with a minimal working example.
- Generate decision matrix for frontend and backend frameworks.
- Select frontend and backend frameworks.
- Determine coding standards.
- Parse query files.

Statement of work completed

The work items and tasks completed up to 1700 hours on 14/08/20 is shown in the task-board snapshots, and detailed further here. The primary goal for week 1 of sprint 1 was to configure all project management tools and resources including:

- Initialising the git repository.
- Creating a backlog of tasks and issues
- Agreeing upon and setting recurring sprint meetings (kick-off and retrospective) and stand-ups.

The frontend decision matrix was developed assessing three main technologies: Angular, React and Vue. These technologies were assessed on 6 criteria. From this it has been decided that Vue will be used. Coding standards have also been agreed upon and style guides will include PEP for Python and Airbnb for JavaScript with corresponding linters being configured.

Additionally, we organised an all hands for week 2 of sprint 1 to determine how to parse query files, and also aimed to break the files down and create a set of tasks to complete.

Work Remaining

- Break down the ‘parse query files’ task
- Stand up backend to retrieve input SQL script and produce appropriate output.
- Complete initial report

Blockers

- Github wiki down (minor)

Definition of done

For an issue to be considered to be done, where appropriate and necessary it must pass all linting, be reviewed by at least two other team members, unit tests, and have appropriate test coverage. We nominally aim for at least 80% test coverage of code.