Home

SWEN90017 2023 TEAM R - August Robotics

Project Brief

August Robotics is deploying its world leading floor marking robot Lionel in various countries of the world. Part of the Lionel floor marking business is o perating under Robot as a Service business model, which means August Robotics' global robot operation team brings the robot to the client site (e.g., exhibition centers), deploy the robot to complete certain client jobs (e.g., floor marking for several exhibitions), delivers the marked floor to the client and brings the robot back to the office.

In order to optimize the company's management efficiency and create an overview of the business for internal stakeholders, we need to establish a digi talized client/job database management platform as one of the foundations of the company process.

Product			
Project Background 	Requirements	JIRA Link Jira Software	GitHub

Process

Team	Meetings	Decisions
		
Slack channel	Useful Project course notes	Design & Architecture
		

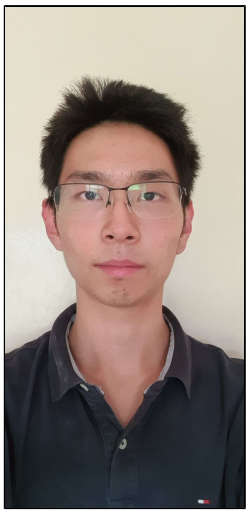
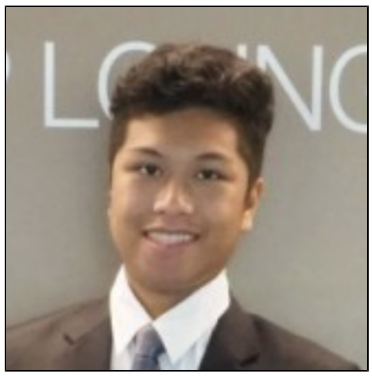
Contact Details

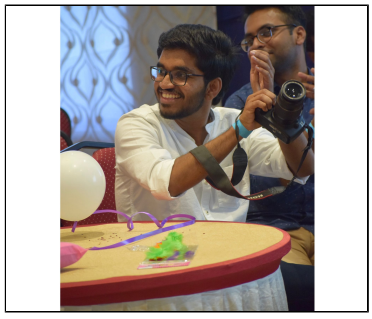
Stakeholders

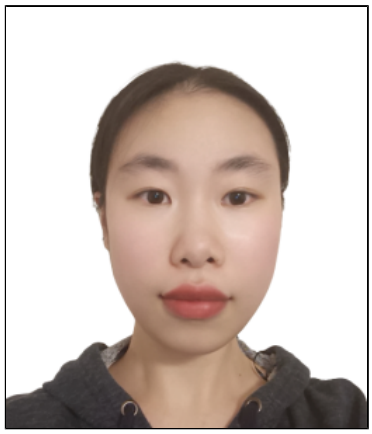
Name	Preferred Name	Photo	Contact	Role
Leo Li	Leo	-	Email: leo.li@augustrobotics.com	Client
Mingye Li	Mingye		Email: mingye.li1@unimelb.edu.au	Supervisor

Meet the Team

Name	Preferred Name	Photo	Contact	Role	Description

Andrew Liu	Andrew		Email: ahliu@student.unimelb.edu.au	Quality Assurance Lead	Responsible making sure product quality and standards are met. This involves developing and managing test plans. Responsible for organising testing plan and monitoring the execution of test procedures.
Michael Hannon	Michael		Email: mhannon@student.unimelb.edu.au	Product Owner	Responsible for managing and prioritising product backlog to maximise the value from the team's work. Also responsible for representing the team in communications with the client.
Kritnand Suwanna-Arj	Krit		Email: ksuwannaarj@student.unimelb.edu.au	Scrum Master	Responsible for managing the project. Also responsible for enacting Scrum values and practices as well as enabling cooperation across the team.
Lipeng Zhang	Lipeng		Email: lipengz@student.unimelb.edu.au	Developer	Responsible for web front-end initialization and collaborative development. Also responsible for managing the technical stack, ensuring proper code quality / coverage / documentation, and creating standards such as file / folder structures.

Wenxuan XIE	Ethan		Email: wexie2@student.unimelb.edu.au	Back-end Lead	Responsible for managing the interchange of data between the server and the users
Wei Zhao	Jasper		Email: weizhao1@student.unimelb.edu.au	Architecture Lead	Guiding the technical direction of the project and ensuring that the architecture of the software system is designed to meet the requirements of the client
Himaja Ramesh Kakade	Himaja		Email: hkakade@student.unimelb.edu.au	UI/UX Lead	Responsible for designing and managing the user interface and user experience of the Job Database platform
Poorvith Gowda Gavenahalli Divakar	Poorvith		Email: pgowda@student.unimelb.edu.au	Risk Management Lead	Responsible for creating, reviewing and assessing risk management policies and protocols necessary for proper and safe functioning

Haiyao Yan	Haiyao		Email: haiyao@student.unimelb.edu.au	Front-end Lead	Responsible for web front-end initialization and collaborative development. Also responsible for managing the technical stack, ensuring proper code quality / coverage / documentation, and creating standards such as file / folder structures.
Jiyang XIN	Jiyang		Email: jiyxin@student.unimelb.edu.au	Deployment Lead	Responsible for the technology used in the project, ensuring no technical problems and focused on system improvements and setting up deployment for making releases.

Road Map

Sprint	Timeline	Milestone	Status
Inception Phase	13th March - 29th March 2023	• Requirements Elicitation • User stories • Design and architecture diagrams • Low-Fidelity Prototype	COMPLETED
Sprint 1	30th March - 28th April 2023	• High-fidelity Prototype • Set Up GitHub Repository • More design and architecture diagrams • Quality Assurance documentation	COMPLETED
Sprint 2	1st May - 26th May 2023	• Update High-Fidelity Prototype • Review User stories • Review Design and architecture • Update Acceptance criteria and tests • More Testing documentation • Development of user stories • Deployment Guide	COMPLETED
Sprint 3	31st July - 21st August 2023	• Develop user stories • User acceptance and unit testing • Updating and adding to documentation	COMPLETED
Sprint 4	29st July - 17st August 2023	• Develop user stories • User acceptance and unit testing • Updating and adding to documentation	COMPLETED

Sprint 5	20th September - 15th October 2023	• Develop user stories • User acceptance and unit testing • Updating and adding to documentation	COMPLETED
Sprint 6	16th October 2023 - 29th October 2023	• Clean up and polishing code and features • User acceptance and unit testing • Updating final documentation • Handover to Client	COMPLETED

Requirements

Table of Contents

General Goals:

1. Understand client needs: The primary goal of producing requirements is to understand and document the needs of August Robotics. Requirements should be developed with the user in mind, ensuring that the product or project meets their needs and expectations.
2. Communicate effectively: Producing requirements helps to establish clear and effective communication between project stakeholders, including the development team, users, and other stakeholders. Clear communication ensures that everyone is on the same page and understands what is expected.
3. Define Job Database software platform scope: Producing requirements helps to define the scope of the project or product, including the features, functions, and capabilities required to meet user needs.
4. Ensure quality: Producing requirements helps to ensure that the product or project is of high quality and meets the necessary standards and regulations.
5. Manage resources: Producing requirements helps to manage project resources, including time, money, and personnel. By establishing clear requirements, it is easier to allocate resources effectively.
6. Facilitate testing: Producing requirements helps to facilitate testing and validation of the product or project. By establishing clear requirements, it is easier to determine whether the product or project meets the necessary criteria.
7. Support project management: Producing requirements helps to support project management by providing a clear roadmap for development and ensuring that everyone is working towards the same goals.

Background

Job Database Management Software Platform

Project Background

August Robotics is deploying its world leading floor marking robot Lionel in various countries of the world. Part of the Lionel floor marking business is operat ing under Robot as a Service business model, which means August Robotics' global robot operation team brings the robot to the client site (e.g., exhibition centers), deploy the robot to complete certain client jobs (e.g., floor marking for several exhibitions), delivers the marked floor to the client and brings the robot back to the office.

In order to optimize the company's management efficiency and create an overview of the business for internal stakeholders, we need to establish a digitaliz ed client/job database management platform as one of the foundations of the company process.

The platform is supposed to be used by company's internal users with different user group (access right) to input and maintain the data. There will be some fixed data that need to be maintained based on the client and venue (e.g., client information, size of the venue, etc.), and there will be some operation data need to be input by the user per job basis (e.g., duration of the job, number of marks, etc.). Finally, the data need to be processed by the sof tware, and presented in a customizable dashboard and list (e.g., trending of efficiency, total mark count by client/region, etc.) to help the user quickly grab the information needed, and export to an Excel spreadsheet if necessary. The platform also needs to consider future extendibility as the company will continue to use and extend the usage of the platform.

Scope of the Project

In scope features

- Web based platform running under defined corporate server
- Login and user group permission management
- Well-structured database design including future expendabilities
- A customizable dashboard to present different data in an efficient manner
- Import and export the data in Excel spreadsheet format

Out of scope features (for now)

- Other features like: auto-generating job reports, job QC report, auto-generating invoices, etc.
- Additional new features that are yet do be discovered/provided by client

Challenges and Opportunities for Project Team

- Learning new technologies that will be used in developing the Job database software platform
- Opportunity to learn and work with real world client
- Build reliable software that will be used to store work done by robots!
- Work with a large team of 10 and manage expectations, perspectives, ideas and issues of each member
- Show ability in carrying out sophisticated software engineering processes to design and deliver a secure system at every level
- Build strong relationships within team and carry out effective communication
- Follow logical and relevant developmental processes to aid frequent releases and deployment
- Create a safe environment by maintaining strong relationship with client to bring out issues and discussion for improvements through constructive feedback

Requirements Elicitation

Elicitation Summary

Elicitation Type	Interview
Interview with Client Time	8th March 2023, 2-3 PM (Melbourne Time)
Client	Leo Li
Interview Location	Zoom Meeting
Requirements Elicitation process	COMPLETED
Interview Done by (Facilitator)	Product Owner - Michael Hannon
Note Taker	Haiyao Yan
Attendees	Team
Roles and Responsibilities	Team Communication and Conflict Management
Interview Details and Resources for elicitation (includes link for questions asked for elicitation)	2023-03-08 (Week 2) Client with Question List for Client

Elicitation strategy

Elicitation technique	pros	cons
Interviews	• Can get direct user involvement • Able to ask further details	• Need to prepare questions in advance • Host need to ensure discussion stay in scope
Workshops	• Get diverse opinions from different stakeholders • Can be resource intensive	• Require multiple stakeholders involved • Need to keep everyone engaged
Focus groups	• Useful for exploring users' attitudes, impressions, preferences and needs	• Need to facilitate group of users

Observation	• Allow team have good understanding of workflow	• Could be time consuming • restricted by geographical location
Questionnaires	• Able to survey large groups of users efficiently	• Questions need to be well-designed • Need to analysis the results

After comparing the advantages and disadvantages of different elicitation techniques, we decide to use **interview** as our elicitation type based on the following reasons:

- The intended system is mainly for company internal use, should focus on employee only
- The client is located in Hong Kong and the team is located in Melbourne
- The client already had a clear image of how the system will be like

Elicitation Roles and Responsibilities

Team Member	Role	Responsibilities
Michael Hannon	Product Owner	• Facilitate the meeting with the client. • Ask questions on behalf of the team. • Ensure that the interview stays on topic/in scope
Haiyao Yan	Note Taker	• Record meeting minutes. • Record important requirements listed by the client.
Other Teammates	Team	• Listen carefully to the client and make additional notes. • Ask additional questions. • Assist in brainstorming questions prior to meeting.

Questions and Answers

Topic	Answers
introduction	• prefer communication - email • would like to join GitHub
Can we have a brief project background and purpose?	• provide service instead of product: • setting up/deploying robots for floor marking, then charge for completed service • require staff to control/manage robots(main cost) • optimates the usage of resource by august robotics for floor marking jobs
Venue information for robot to mark floors	1. input the floor plan (with details on # of halls, gross area) 2. transfer floor plan into json dataset 3. robot mark based on data ▪ dataset for client (floor marking customers) usually won't change unless re-construction

What is difference between fixed and operational data?	**Fixed data maintenance** • Every time when there's a new client or new venue, the administrator add the client data and venue dat as the fixed data to the database for future use • Only authorized user can modify the data. If the fixed data is modified, all the other areas that use the modified data will be changed accordingly. **Operational data mainatainence** • When the client books a new job, the user input the preliminary job information into the system • When the job is done, the job data maintainer will update the job data in the data base, some data can only be added after the job • Some data need to be selected from the fixed data database (e.g. venue, hall, client), some will need to be input (e.g. # of marks, # of marking days, # of FTE days, etc)
What tables/content to be included for Data analysis ?	• Some of the data analysis tables that could be graphically shown*:* 1. Client List 2. Total Marks marked by AR divided by clients 3. Total FTE (full-time equivalent) and Intern days worked on-site divided by client (only post-pandemic data) 4. Average # of marks per day (only post-pandemic data) 5. Average # of marks per job window (only post-pandemic data) 6. Marked # of Halls (only post-pandemic data)
What is the Feature Priorities?	• feature priority: 1. data maintenance data display (adding fixed and operational data) 2. light-weight usage (low requirement on efficiency) 3. currently have APIs for mapping 4. auto-generated report for job , export using excel sheets 5. Access control for 3 types of users - System admins, data maintainers and viewers such as financial teams for data analysis and strategizing company resources and cost • analysing the fault • integrate with the current invoicing process
Any sample/examples for data we can look at?	1. different floor marking colours available 2. billing based on net or gross area and # of halls 3. need to record the info for both before and after the job 4. only authorise member can modify data (different teams have different role) 5. would like to have query for analysing data

Data fields needed to be stored in the database (includes fixed data and operational data)	**230227_SAMPLE_Client Job Database.xlsx** 230227_SAMPLE_Client Job Database.xlsx 3 MB

Client Decisions

- Meetings with Product Owner every two weeks at least to show progress and ask questions, do sign-offs
- focus on building the platform for adding and maintaining Job data first then focus on access control

Elicitation risks

ID	Type	Description	Probability	Impact (out of 5)	Exposure	Justification
1	Project	The product does not meet all specifications by the client.	50%	3	1.5	It is likely that some of the specifications or new features will not be met due to time constraints, but we assume the most vital ones will be.
2	Project	There is a new issue from the changes in the technologies used. (not adaptable to the system that's already present)	5%	4	0.2	There is a low probability that one of the technologies we are using has significant changes that would fundamentally impact the project. However, if this does occur, large changes would be required.
3	Project	Client becomes unresponsive.	10%	3	0.3	Client so far has shown to be very responsive. The impact would be dependent on the what stage we are in at the moment and what we need from the client.

Note: *In the sections below the main functional and non-functional requirements are discovered from the requirements mentioned by our client Leo Li. This includes flushing out what information will be displayed on the job database platform, how to digitize and visualize this data through meaningful analysis and display through eg. graphs, how the floor marking process takes place and what details are required to be stored before and after the job gets done. Furthermore, important access control features for different types of users is mentioned who will have varying abilities to view and modify data.*

Table of Contents

Page Title **Content**

1. Functional Requirements — Key Functionalities/Requirements that are to be developed for the job database platform

2. Non-Functional Requirements — Quality attributes and constraints for the job database platform that will enhance the use, effectiveness and efficiency of the system

Functional Requirements

1. Data Digitization and Visualization:

1.1 Current job database (excel sheet) consists of job information such as start date, end date, no. of marking days, no. of halls, no. of corners, etc. This job information needs to be digitized to showcase for data analysis and building strategies for business development. The data needs to be processed by the software, and presented in a customizable dashboard and list (analyze through graphs) (e.g., trending of efficiency, total mark count by client/region, etc.) to help the user quickly grab the information needed.

1.2 The dashboard will present job key performance index. E.g.: select client Shepart, select venue Indianapolis, select show "WorkTruckWeek", the data of the job will be displayed

1.3 Some of the data analysis tables that could be graphically shown*:*

- Client List
- Total Marks marked by AR divided by clients
- Total FTE (full-time equivalent) and Intern days worked on-site divided by client (only post-pandemic data)
- Average # of marks per day (only post-pandemic data)
- Average # of marks per job window (only post-pandemic data)
- Marked # of Halls (only post-pandemic data)

2. Information and Workflow Requirements:

(For more information on the data fields refer to: Preview attachment 230227_SAMPLE_Client Job Database.xlsx and Client Meeting Notes)

2.1 Job input information practice will include adding data before and after the job is done

2.1 Fixed data maintenance

2.1.1 Every time when there's a new client or new venue, the administrator add the client data and venue data as the fixed data to the database for future use

2.1.2 Only authorized user can modify the data. If the fixed data is modified, all the other areas that use the modified data will be changed accordingly

2.2 Operational data maintenance (per job basis)

2.2.1 When the client books a new job, the user input the preliminary job information into the system

2.2.2 When the job is done, the job data maintainer will update the job data in the data base, some data can only be added after the job

2.2.3 Some data need to be selected from the fixed data database (e.g. venue, hall, client), some will need to be input (e.g. # of marks, # of marking days, # of FTE days, etc)

3. Access Control and Permission Management:

3.1 The software platform will be distributed across the organization (e.g. finance, management and technical team)

3.2 3 Types of user groups required to input, maintain and view the data:

3.2.1 Admin - Manage the Job Database Software platform and access control

3.2.2 Job Data Maintainer (Staff Login) - Authorized user to modify and add new data for a job

3.2.3 Viewer (Client/Staff Login) - Only able to view job data and graphical forms of the analyzed data

4. Import/Export and Reporting Requirements:

4.1 Import and export the analyzed data in Excel spreadsheet format

4.2 Platform could have additional features such as auto-generating job reports, job QC report, auto-generating invoices, etc.

Non-Functional Requirements

1. Extendibility

1.1 Well-structured database design including future expendabilities

1.2 Web based platform running under defined corporate server

1.3 Easily able to add new features to the platform

2. Maintainability

2.1 Job Database Platform should be well documented and easy to read.

2.2 Job Database Platform should be written such that it is easy to identify, locate and fix bugs.

3. Reliability

3.1 Job Database Platform should be able to operate as expected over long periods of time and should be robust and fault-free.

3.2 System will maintain backups of data in the event of data loss.

4. Availability

4.1 Job Database Platform should be available most of the time.

4.2 Database should be available for request most of the time.

5. Security & Integrity

5.1 User data should be stored in an encrypted manner.

5.2 Job and client data should be encrypted.

5.3 Input validation to database.

6. Performance

6.1 Space:

- 6.1.1 Job Database Platform should be able to handle a small amount of users.
- 6.1.2 Database should be able to handle a moderate size of data and read/writes from a small amount of users.

6.3 Time:

- 6.3.1 Web application should feel quick.
- 6.3.2 Web application should be intuitive to use.

7. User Interface

7.1 Usability: The interface for platform should be seamless and easy to navigate through to find relevant job based information and data analysis.

7.2 Understandability: Data should be easily understandable and clear to client and staff of August Robotics to use it for building business strategies

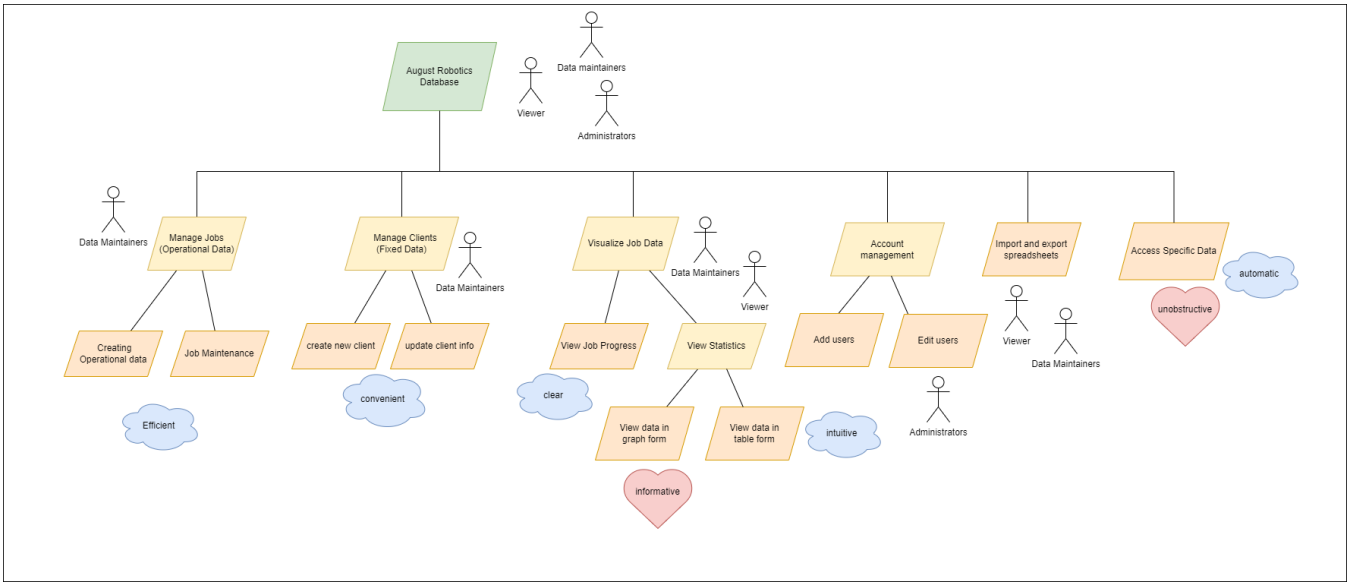
7.3 Look and feel: Similar to the operating robot application in terms of color and UI

Motivational Model

Do-Be-Feel List

Who	Do	Be	Feel
• Data Maintainers • Administrators • Viewer	• Creating Operational Data • Job Maintenance • Create New Client • Update Client Info • View Job Progress • View Data in a graph form • View Data in a table form • Add users • Edit users • Export Spreadsheets • Import Spreadsheets • Access Specific Data	• Convenient • Clear • Intuitive • Automatic • Efficient	• Informative • Unobstructive

Motivational Model V1.2



Link to Diagram: https://drive.google.com/file/d/1I6VbzsCuFVWcqOZKaoKfVVhVKdIMvTC9/view?usp=share_link

Description

The motivational model is meant to give a general overview of the product, users, and what the users are supposed to think and feel about each service. The orange represents *do's* that model should be able to complete. These could also be translated into epics for user stories. The blue represents the *be's* the service is, what the service should be like. The red represents the *feel's*, what the service should make feel like while using it.

Goals List

Functional Goals	Non-Functional Goals	Emotional Goals

• Input job data into system • Store client data in database • Add fixed data for each new client (venue) • Authorized users can edit fixed data • Dashboard to display data visually	• Security • Performance • Extendibility • Maintainability • Reliability • Availability • Usability • Understandability • Integrity	• Ability to visualize and analyze job data • Ability to view and modify the data before and after the job is done • Import/export data easily • Create invoices

Personas

Name	Job title/responsibilities	Goals	Frustrations
Alex Liu	Alex works for the start-up tech company. His primary responsibility is to manage and maintain the job database for the company. He is responsible for adding new jobs to the database, updating job information, and ensuring that the data is accurate and up-to-date. John is also responsible for generating job reports for clients and management.	• Ensure that the job data is correct and up-to-date. • Easily access and update the data when needed • Needs a platform that can help him efficiently manage job data and generate reports, while also providing a range of data analysis tools that can help him identify trends and insights in the data. • Needs a platform that is user-friendly and customizable, so that he can easily navigate through the data and generate customized reports for clients and management.	• Currently, Alex manages the job data using Excel spreadsheets, which is time-consuming and prone to errors. It can be challenging to keep track of all the data for each job, and it is easy to make mistakes when manually entering data. Generating reports is also a manual process, and it can take a lot of time to create reports that meet clients' specific needs.
Mikhaila Karstensen	Mikhaila Is an administrator working for August Robotics. She is responsible for managing the company's internal users' access to the venue and fixed data information including creating new users, removing users and managing user permissions. Mikhaila Is also responsible for adding new clients or new venues to the platform as fixed data.	• Responsible for managing the job management platform • Responsible for managing access to the job management platform. • Responsible for adding new client information to the platform. • Responsible for adding new venues to the platform.	• Currently has to manually client information manage using excel spreadsheets. This can be time consuming and confusing. • User access to client and venue information is also done by using excel spreadsheets. This is also time consuming and tedious
Ollie Rocha	Ollie is responsible for making financial decisions based on data/analytics from previous job data. Ollie is also responsible for processing job data into meaningful data from which to make decisions on.	• Make informed financial decisions based on analytical data from job data. • View and track meaningful job data.	• No access to an intuitive user interface to help track financial status. • Has to manually calculate analytical information and generate graphs from data on excel spreadsheets. This is time consuming and tiresome.

1. Job Data Maintainer

Alex Liu

AGE	29
EDUCATION	Masters in Business
STATUS	Single
OCCUPATION	Job Data Maintainer
LOCATION	HongKong
TECH LITERATE	Mid

> I'm looking for a platform that can automate and streamline my job data management processes

Personality

Introvert | Thinker

Platform

Website | Mobile App

Background

Alex works for the start-up tech company. His primary responsibility is to manage and maintain the job database for the company. He is responsible for adding new jobs to the database, updating job information, and ensuring that the data is accurate and up-to-date. John is also responsible for generating job reports for clients and management.

Goals

- Ensure that the job data is accurate and up-to-date.
- Easily access and update the data when needed.
- Needs a platform that can help him efficiently manage job data and generate reports, while also providing a range of data analysis tools that can help him identify trends and insights in the data.
- Needs a platform that is user-friendly and customizable, so that he can easily navigate through the data and generate customized reports for clients and management.

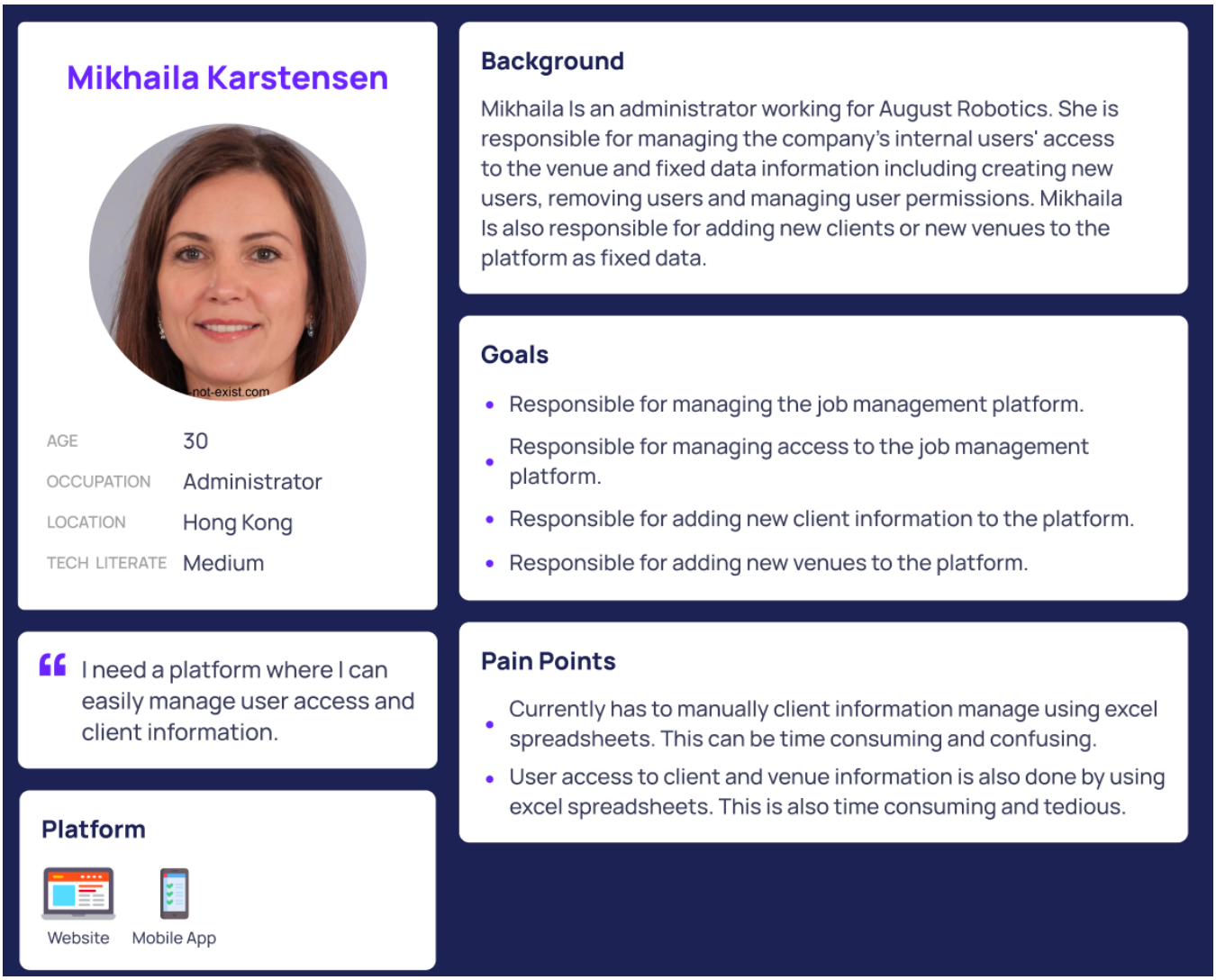
Pain Points

Currently, Alex manages the job data using Excel spreadsheets, which is time-consuming and prone to errors. It can be challenging to keep track of all the data for each job, and it is easy to make mistakes when manually entering data. Generating reports is also a manual process, and it can take a lot of time to create reports that meet clients' specific needs.

Brands

NASA | HUAWEI | facebook | YouTube | Instagram | BELLE

2. Administrator


Mikhaila Karstensen
not-exist.com
AGE 30
OCCUPATION Administrator
LOCATION Hong Kong
TECH LITERATE Medium
Background
Mikhaila Is an administrator working for August Robotics. She is responsible for managing the company's internal users' access to the venue and fixed data information including creating new users, removing users and managing user permissions. Mikhaila Is also responsible for adding new clients or new venues to the platform as fixed data.
Goals
Responsible for managing the job management platform.
Responsible for managing access to the job management platform.
Responsible for adding new client information to the platform.
Responsible for adding new venues to the platform.
I need a platform where I can easily manage user access and client information.
Pain Points
Currently has to manually client information manage using excel spreadsheets. This can be time consuming and confusing.
User access to client and venue information is also done by using excel spreadsheets. This is also time consuming and tedious.
Platform
Website
Mobile App

3. Viewer

Ollie Rocha

AGE	40
OCCUPATION	Finance Manager
LOCATION	Hong Kong
TECH LITERATE	Low

> I need a platform where I can view user friendly analytics on job data.

Platform

Website Mobile App

Bio

Ollie is responsible for making financial decisions based on data/ analytics from previous job data. Ollie is also responsible for processing job data into meaningful data from which to make decisions on.

Core needs

- Make informed financial decisions based on analytical data from job data.
- View and track meaningful job data.

Pain Points

- No access to an intuitive user interface to help financial status.
- Currently finds perect people from past work relations, family, friends and within my circle and online which is tedious

User Stories

This section consists of all our user stories that are to be developed and implemented into our system as a end goal for its users.

Notes
**Database Platform user - administrator/ job data maintainer/ viewer*
**AR - August Robotics*
**operational data - Floor marking Job details for a particular client*
**Fixed data - Client details*

Size Estimation for User Stories

Size	Description
Small	The task can be completed within 1-2 days.
Medium	The task can be completed in 3-5 days.
Large	The task can be completed over 6-10 days.

MoSCoW Priority

MoSCoW is a prioritization technique used in Agile software development and project management to categorise user stories or requirements based on their importance. The acronym MoSCoW stands for:

Priority	Description
Must Have	Features that must be delivered or the software will not create the expected value for the client.
Could Have	Features that the client considers nice to have but will not have a material impact to value, if not delivered.
Should Have	Features that have significant value to the client and should be delivered, but not considered crucial.
Wont Have	Out-of-scope features; useful as next steps for your project as potential improvements for future releases.

User Story Table 2.0

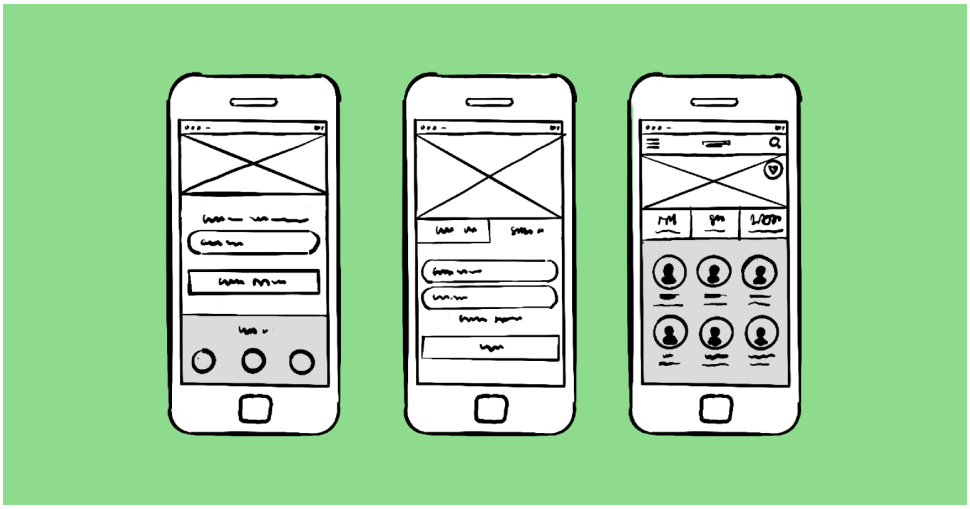
ID	Epic	As a	I Want To	So That	Size Estimation	MoSCoW Priority	Justification
1	**Job Data Digitization (Dashboard)**	Database Platform user	view a table of total Marks marked by AR divided by clients	I can look at the total marks for each client pre and post-pandemic monthly or quarterly	**M**	**Should-have**	Size: (Data analysis) Will analyze details of job data and represent it through tables and graphs (trends). For the smaller user stories there are less data fields to consider thus the difference with other medium sized user stories. MoSCow: Necessary to show analyzed data specific to different jobs and their requirements which can then be imported/exported/used to optimize and build stronger business strategies. But takes less precedence compared to adding and maintaining job data (fixed/operational)
2		Database Platform user	view a table/graph showing the Total FTE (full-time equivalent) and Intern days worked on-site divided by client (only post-pandemic data)	I can look at how many days were invested and efficiency performed per person on-site	**M**	**Should-have**	
3		Database Platform user	view a table showing average number of marks per day	I can look at the average marks for a client for the total job days	**S**	**Should-have**	
4		Database Platform user	view a table showing average number of marks per job window	I can assess the client behavior and needs to plan the capacity required on-site	**M**	**Should-have**	
5		Database Platform user	view a table showing the marked number of halls	I can check the total job days and average halls marked per day	**M**	**Should-have**	
6	**Access Control**	administrator	be able login to access fixed and operational data	I can maintain the database platform	**M**	**Must-have**	Size: Both fixed data and operational data will have important information and many fields so it has to ensured that only required personal will get the access to update or modify. MoSCow: Must provide proper access to the required people in the team and everyone cant have access to the data as some can only view the data.

7		data maintainer	be able login to access fixed and operational data	I can maintain the database platform	**M**	**Must-have**	
8		viewer	be able to just view the data and have no access to modify or update any data	I can view the necessary job data	**S**	**Must-have**	
9	**Operational Data**	administrator/ data maintainer	be able to input preliminary job data into the system when there is a new job	the system has the necessary data for a new job	**M**	**Must-have**	Size: The operational data will contain many fields. Will also require input validation to ensure there are no incorrect parameters inputted into database MoSCow: All of these functions are for mandatory data for each job, these must be implemented to achieve the necessary data visualisation.
10		administrator/ data maintainer	be able to input final job data for each hall marking job	I can update the floor marking data that is obtained during the completion of a job	**M**	**Must-have**	
11		Database Platform user	view processed floor marking job data (operational)	I can quickly access information for a ongoing or finished job	**M**	**Must-have**	Size: Will require several data fields for every floor marking job done such as start date, end date, no. of marking days, no. of halls, no. of corners, etc. MoSCow: Necessary to represent all floor marking job information that is ongoing or finished in a digitized format specific to different clients. This will optimize the company's floor marking job management efficiency and processes.
12		Database Platform User	view the performance index for jobs	I can analyse the performance for a job after it's completion	**M**	**Must-Have**	Size: Will require some calculations from data within a job to generate performance index for a job MoSCow: Necessary information for user's of the platform to analyse a job and it's performance
13		Database Platform User	view the general information for a job	I can see the people working on a job and estimated completion dates	**S**	**Must-Have**	Size: Will only require a few data fields to be entered into the database MoSCow: Mandatory for each job to display the general information
14		Database Platform User	generate a job summary	I can extract the information needed from a job to generate an invoice	**M**	**Should-Have**	Size: Will require drawing a number of data fields from the database and formatting them in a user-friendly format for analysis MoSCow: Not completely necessary for a user to obtain this information, a nice should-have as it will streamline this process
15		Database Platform User	view a version history of hall marking jobs within a job	I can track and view all changes made to specific hall marking jobs	**M**	**Could-Have**	Size: Creating a version log will require copies of job history stored in the database, handling and storing these copies will require some complexity MoSCow: Not necessary to achieve the goal of the platform, will be a nice to have as tracking these changes is important
16	**Fixed Data**	administrator/ data maintainer	be able to add the client data and venue data as the fixed data to the database	other parts can use the data in the database to manage the venue	**M**	**Must-have**	Size: It requires an effective link between the webpage and the database to save the data into the database. The data stored in the database can be display on the webpage, and the administrator can edit the data in the database. MoSCoW Priority: This functionality is one of the essential functions of the project, it requires an effective link between the frontend and the backend. In that case, the data edit and presentation can be achieved.
17		administrator/ data maintainer	be able to view client data	I can know the data in the database such as the client and venue data	**M**	**Must-have**	
18		administrator/ data maintainer	be able to edit the data of the client	I can modify the fixed data in the database	**M**	**Must-have**	
19	**Import and export data**	administrator/ data maintainer	be able to import the job duration, number of marks, and other relevant information	I can import existing data into the system easily	**L**	**Could-have**	Size: This functionality will require complex handling of large amounts of data. Importing this data correctly will most likely require a large amount of time. MoSCow: This functionality is not essential to the project. It would be useful to have it as It will be very inefficient to manually enter historical data one by one without being able to import data all in one but is not critical at the moment to implement.

20		Database Platform user	be able to export the data in Excel spreadsheet format	I can easily share the information with others and analyse it further.	**M**	**Could-have**	Size: This functionality will require handling large amounts of data, but unlike the import function the data will be formatted and easier to obtain. MoSCow: This functionality is essential but not critical to the project. It involves exporting the job/client data into an excel sheet for further analysis. However, this is low priority.
21	**Searching**	Database Platform user	be able to filter, sort and search the data based on various criteria such as client, region, or date	I can quickly find the information I need.	**M**	**Must-have**	Size: This functionality will require some complex database queries to function correctly, but overall should not require the addition of many backend functions MoSCow: This functionality is important in this project. Sorting and searching functions can make it easier for users to retrieve data, so that project is can easily visualize and display data.
22	**Admin User Management**	administrator	be able to create a new account for a user	a new employee can use the system	**S**	**Should-have**	Size: This functionality will only require the entry of a few data fields into the database and a simple web page MoSCow: This functionality is essential to the system, an admin must be able to make new user accounts for any new employees.
23		administrator	be able to delete an existing account	inactive accounts can be removed from the system	**S**	**Could-have**	Size: This functionality will only require a few database queries and simple web pages MoSCow: This functionality is recommended as not to have inactive accounts with privileges floating around the system. But is not essential.
24		administrator	be able to edit an existing user	a user's details can be changed.	**S**	**Could-have**	

Previous Version Here

Low-Fidelity Prototype



Below is the *Figma link* to our *low-fidelity prototype for Job Database Software Platform using wireframes*. The Screens are made for the following functions:

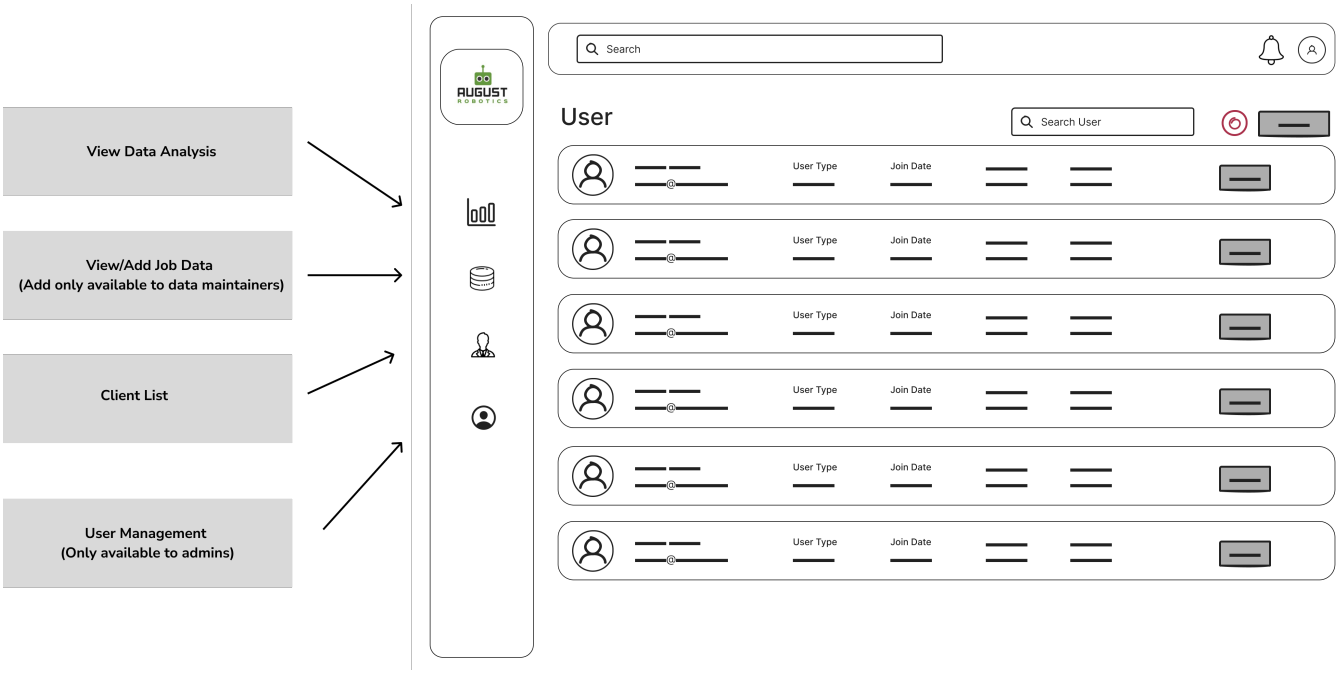
Screens
• Log In • Admin Login and user management (add/edit user) • Main Dashboard (Showing data analysis) • View Job lists • View a particular floor marking job details • View General information on it • Ability to filter and sort • View Client List • Add new client (fixed data) • Add new Job (operational data) • Import data • Export data (PDF/Excel Sheet) • Edit a Job

Figma Link: https://www.figma.com/file/BtQtU2LJW5Jc5I7ZFdH8qu/SWEN90017-%3A-Low-Fidelity-Prototype---Wireframes?node-id=709%3A14319&t=5FnSofst539qk7u7-1

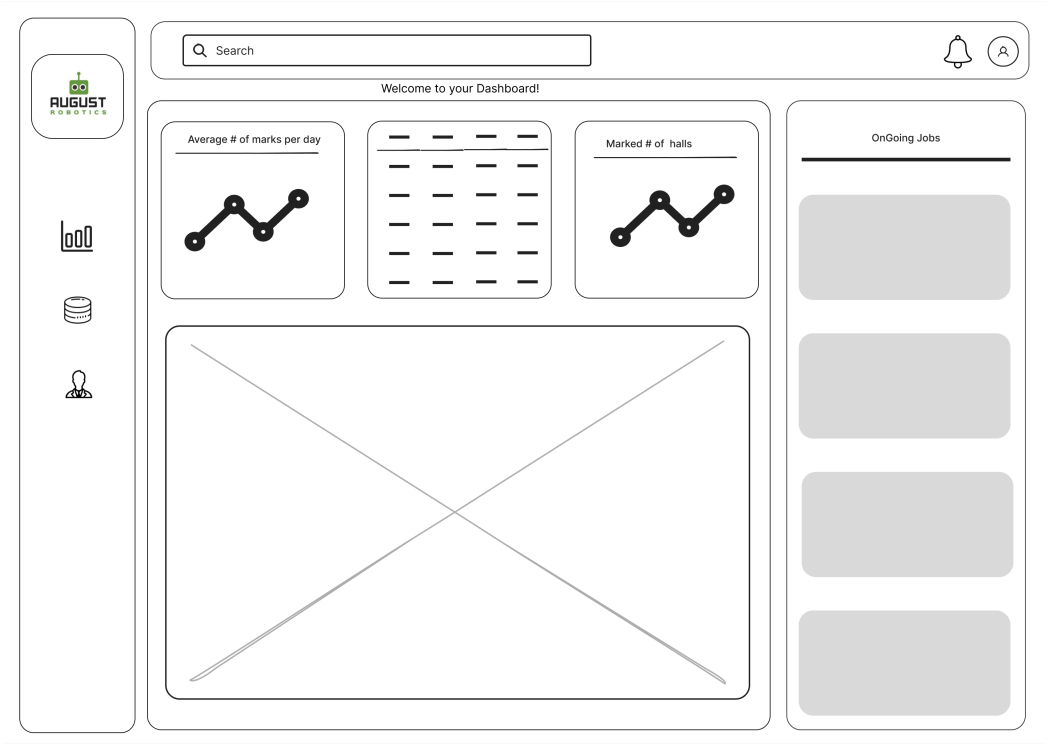
Note : Changes provided by the client have been reflected directly on the high fidelity prototype as per 2023-04-26 (Week 8) Client. However similar outline to the low fidelity prototype has been used.

1. Log in

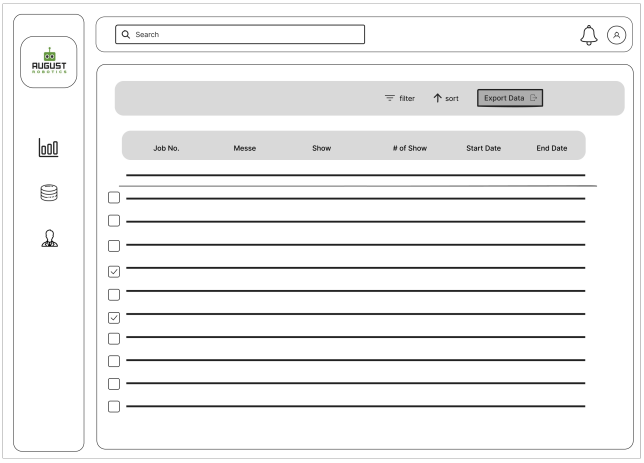
2. Admin Login and user management (add/edit user)

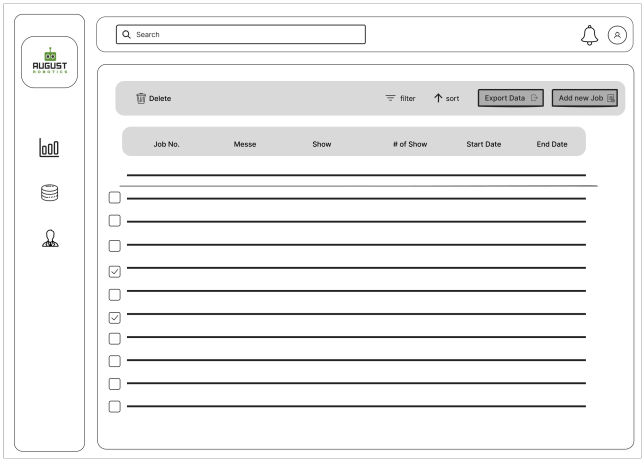


3. Main Dashboard

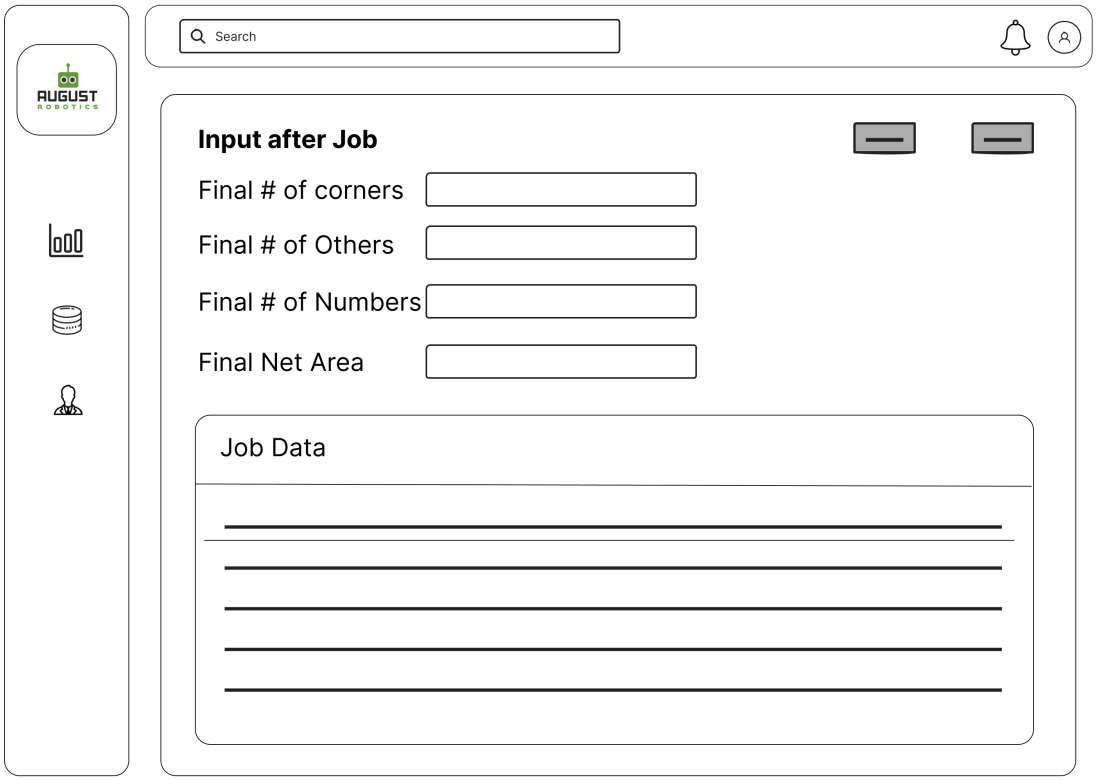


4. View Job Lists

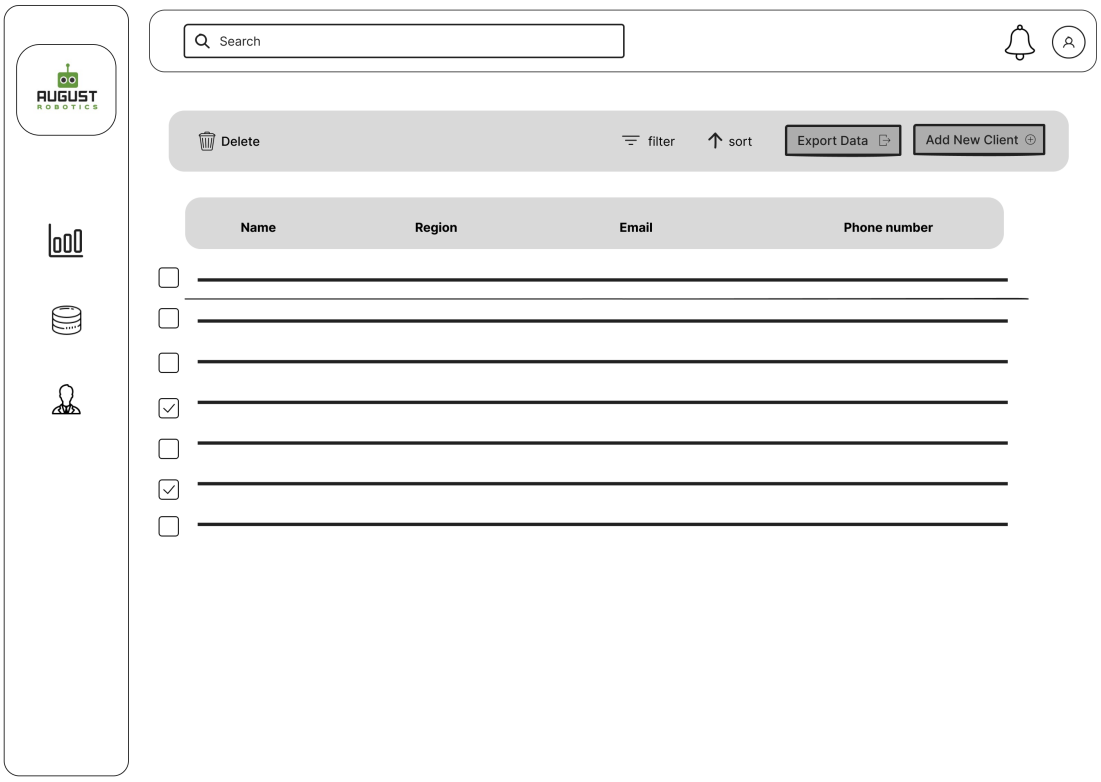




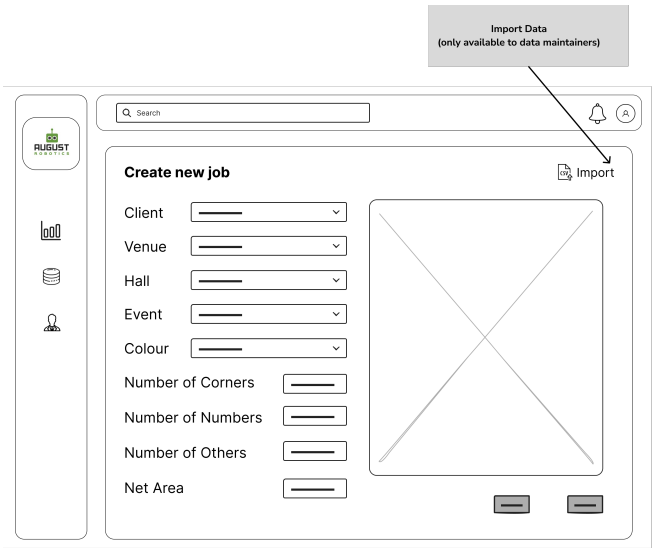
5. View Job details and general information including invoice

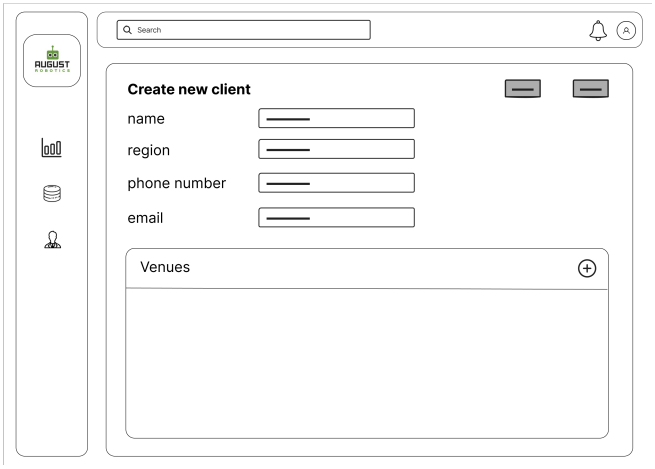


6. View Client List

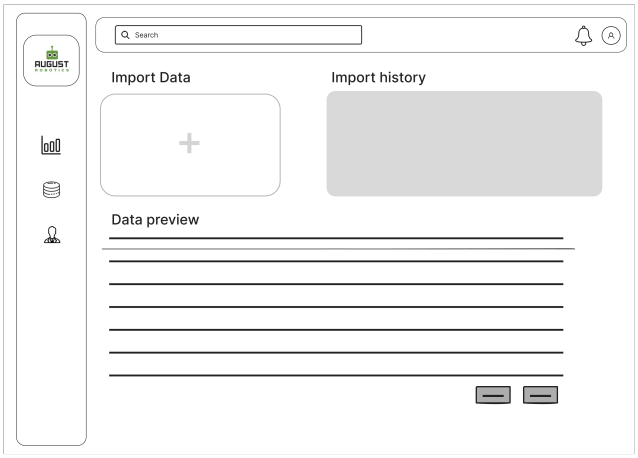


7. Add fixed (from add new client button) and operational data (from add new job button)





8. Import/Export data



Acceptance Criteria

Notes

**Database Platform user - administrator/ job data maintainer/ viewer*

**AR - August Robot*

**operational data - Floor marking Job details for a particular client*

**Fixed data - Client details*

Epic	User Story ID	User Story	Acceptance Criteria ID	Given	When	Then	Test Result (Pass / Fail)	Comments
Job Data Digitization (Dashboard)	1	As a Database Platform user, I want to view a table of total Marks marked by AR divided by clients so that I can look at the total marks for each client pre and post-pandemic monthly or quarterly.	1.1	I login to view the database platform dashboard	the dashboard presents the data analysis for the job key performance index	I want to select the table to see total Marks marked by AR divided by clients	PASS	
			1.2	I have selected to view the table	the table is opened in a bigger view	I want to view correctly calculated data displayed in it.	PASS	
	2	As a Database Platform user, I want to view a table/graph showing the Total FTE (full-time equivalent) and Intern days worked on-site divided by client so that I can look at how many days were invested and efficiency performed per person on-site.	2.1	I login to view the database platform dashboard	the dashboard presents the data analysis for the job key performance index	I want to select the table/graph showing the Total FTE (full-time equivalent) and Intern days worked on-site divided by client	PASS	
			2.2	I have selected to view the table	the table/graph is opened in a bigger view	I want to view correctly calculated data displayed in it.	PASS	
	3	As a Database Platform user, I want to view a table showing average number of marks per day so that I can look at the average marks for a client for the total job days.	3.1	I login to view the database platform dashboard	the dashboard presents the data analysis for the job key performance index	I want to select the table showing average number of marks	NOT IMPLEMENTED	
			3.2	I have selected to view the table	the table is opened in a bigger view	I want to view correctly calculated data displayed in it.	NOT IMPLEMENTED	
	4	As a Database Platform user, I want to view a table showing average number of marks per job window so that I can assess the client behavior and needs to plan the capacity required on-site.	4.1	I login to view the database platform dashboard	the dashboard presents the data analysis for the job key performance index	I want to select the table showing average number of marks per job window	NOT IMPLEMENTED	
			4.2	I have selected to view the table	the table is opened in a bigger view	I want to view correctly calculated data displayed in it.	NOT IMPLEMENTED	
	5	As a Database Platform user, I want to view a table showing the marked number of halls so that I can see the marked halls for each client to check the total job days and average halls marked per day.	5.1	I login to view the database platform dashboard	the dashboard presents the data analysis for the job key performance index	I am able to view the table showing the marked number of halls	NOT IMPLEMENTED	
			5.2	I have selected to view the table	the table is opened in a bigger view	I am able to check the total job days and average halls marked per day for each client' halls	NOT IMPLEMENTED	
Access Control	6	As a administrator, I want to be able login to access fixed and operational data, so that I can maintain the database platform	6.1	I have logged in as an administrator	there is a new client	I am able to access fixed data	PASS	
	7	As a data maintainer, I want to be able login to access fixed and operational data so that I can maintain the database platform.	7.1	I have logged in as job data maintainer	the job is done	I want to update operational data when the job is completed	PASS	
	8	As a viewer, I want to be able to just view the data and have no access to modify or update any data	8.1	I have logged in as viewer	I want to view data	I am able to view the relevant data	PASS	
			8.2	I have logged in as viewer	I try to modify or update some data	I am unable to modify or update any data.	PASS	

Operational Data	9	As an Administrator/data maintainer, I want to be able to input preliminary job data into the system when there is a new job, so that the system has the necessary data for a new job	9.1	I am a logged in as an administrator/data maintainer	a new job is created	I want to be able to input preliminary job data	PASS	
	10	As an Administrator/data maintainer, I want to be able to input final job data for each hall marking job, so that I can update the floor marking data that is obtained during the completion of a job	10.1	I am a logged in as an administrator/data maintainer	a completed job is viewed	I am able to update the job with data gathered during completion.	PASS	
	11	As a Database Platform user, I want to view processed floor marking job data (operational) so that I can quickly access information for a ongoing or finished job.	11.1	I login to view the database platform dashboard	the dashboard is displayed	I want to select the job lists on the navigation bar and view all floor marking jobs for clients	PASS	
			11.2	I am viewing the floor marking job lists	I select a particular job	I want to view the venue information and input details before/after the job is done.	PASS	
	12	As a Database Platform User, I want to view the performance index for jobs, so that I can analyse the performance for a job after it's completion.	12.1	I am viewing the job listings	I select the option to view the performance index	I can see the performance index for all jobs.	PASS	
	13	As a Database Platform User, I want to view the general information for a job, so that I can see the people working on a job and estimated completion dates	13.1	I am viewing the job listing	I select the job	I can view the detailed general information of the job	PASS	
	14	As a Database Platform User, I want to view the generate a job summary, so that I can extract the information needed from a job to generate an invoice	14.1	I am in a job's details page	I select the option to generate summary	a summary is generated that can be downloaded.	PASS	
	15	As a Database Platform User, I want to view a version history of hall marking jobs within a job, so that I can track and view all changes made to specific hall marking jobs	15.1	I am in the job's details page	I can select the option to view history	I see a graph/table showing all changes made to that hall marking job.	NOT IMPLEMENTED	
Fixed Data	16	As an Administrator/data maintainer, I want to be able to add the client data and venue data as the fixed data to the database, so that other parts can use the data in the database to manage the venue	16.1	I am logged in as an administrator/data maintainer	Some client data needs to be added to the database as fixed data	The client data can be able to used by other parts	PASS	
			16.2	I am logged in as an administrator/data maintainer	Some venue data needs to be added to the database as fixed data	The venue data can be able to used by other parts	PASS	
	17	As an Administrator/data maintainer, I want to be able to view client data, so that I can know the data in the database such as the client and venue data	17.1	I am logged in as an administrator/data maintainer	I want to see the fixed data about the client	I am able to see the data about the client and the venue on the page	PASS	
	18	As an Administrator/data maintainer, I want to I want to be able to edit the data of the client, so that I can modify the fixed data in the database	18.1	I am logged in as an administrator/data maintainer	the data of client needs to be updated	I can edit the client data stored in the database	PASS	
			18.2	I am logged in as an administrator/data maintainer	the data of client needs to be updated	the other areas that use the modified data will be changed accordingly after I edit the data of client	PASS	
Import and Export Data	19	As an administrator/ data maintainer, I want to be able to import the job duration, number of marks, and other relevant information, so that I can import existing data into the system easily	19.1	I am a logged in as an administrator/data maintainer and selecting the selected job import page	a job import page is opened	I am able to input the job data in different data through excel sheet	NOT IMPLEMENTED	
			19.2	I am a logged in as an administrator/data maintainer and have imported a correct job excel sheet	the save button is clicked	the job data is imported correctly and has been saved in the database.	NOT IMPLEMENTED	

			19.3	I am a logged in as an administrator/data maintainer and have imported a job excel sheet that lacks key data or contains incorrect data (such as incorrect data type).	The job excel sheet has been verified by the system	the job data can't be saved in the database and an error notification will be sent to users	NOT IMPLEMENTED	
	20	As a Database Platform user, I want to be able to export the data in Excel spreadsheet format, so that I can easily share the information with others and analyse it further.	20.1	I am a logged in as an administrator/data maintainer and viewing the dashboard	the export excel button is clicked	I am able to see the browser is downloading the spreadsheet	NOT IMPLEMENTED	
Searching	21	As a Database Platform user, I want to be able to filter, sort and search the data based on various criteria such as client, region, or date, so that I can quickly find the data I need.	21.1	I am a logged in as an administrator/data maintainer and viewing the dashboard	I select keywords such as client name and venue name or time range, and click the filter button	I am able to see and select the results filtered based on the key words I select	PASS	
			21.2	I am a logged in as an administrator/data maintainer and viewing the dashboard	I type keywords such as client name and venue name, and click the search button	I am able to see and select the results searched based on the key words I type	PASS	
			21.3	I am a logged in as an administrator/data maintainer and viewing the dashboard	Click one of the sorting buttons (sort by date, sort by alphabet)	I am able to see the results sorted based on the criteria I chose	PASS	
Admin User Management	22	As an administrator, I want to be able to create a new account for a user, so that a new employee can use the system	22.1	I am logged in as an administrator	the user management page is opened	I can create a new user.	NOT IMPLEMENTED	
			22.2	I am creating a new user	the user management page is opened	I can set the user name, email address, password, and user (privilege) type for new users.	NOT IMPLEMENTED	
			22.3	I am creating a new user	user name or mailbox already exists	the user can not be created.	NOT IMPLEMENTED	
	23	As an administrator, I want to be able to delete an existing account, so that inactive accounts can be removed from the system	23.1	I am logged in as an administrator	the user management page is open	I can delete users whose type of privilege is job data maintainer or viewer.	NOT IMPLEMENTED	
			23.2	I am logged in as an administrator	the user management page is open	I can delete other administrator users.	NOT IMPLEMENTED	
	24	As an administrator, I want to be able to edit an existing user, so that a user's details can be changed.	24.1	I am viewing an account in the user management page	I click edit	I can change the user's permission level	NOT IMPLEMENTED	

High-Fidelity Prototype

Table of Contents

High Fidelity Design for AR Database Platform

Figma Prototype Design Link: https://www.figma.com/file/wppLyVtOXpqCN2MCv252zQ/SWEN90017-%3A-High-Fidelity-Prototype?node-id=0-1&t=i3J8DvdrnyFQCogZ-0

High Fidelity Prototype for AR Database Platform

Figma Interactive Prototype Link: https://www.figma.com/proto/wppLyVtOXpqCN2MCv252zQ/SWEN90017-%3A-High-Fidelity-Prototype?type=design&node-id=812-2&scaling=scale-down&page-id=0%3A1&starting-point-node-id=812%3A2

Login and Dashboard Workflow

Login and Dashboard page user stories: 1, 2, 3, 4, 5, 6, 7, 8.

Login page

1.User Accesses Login Page
  1.1. The user visits the web application and is directed to the login page.

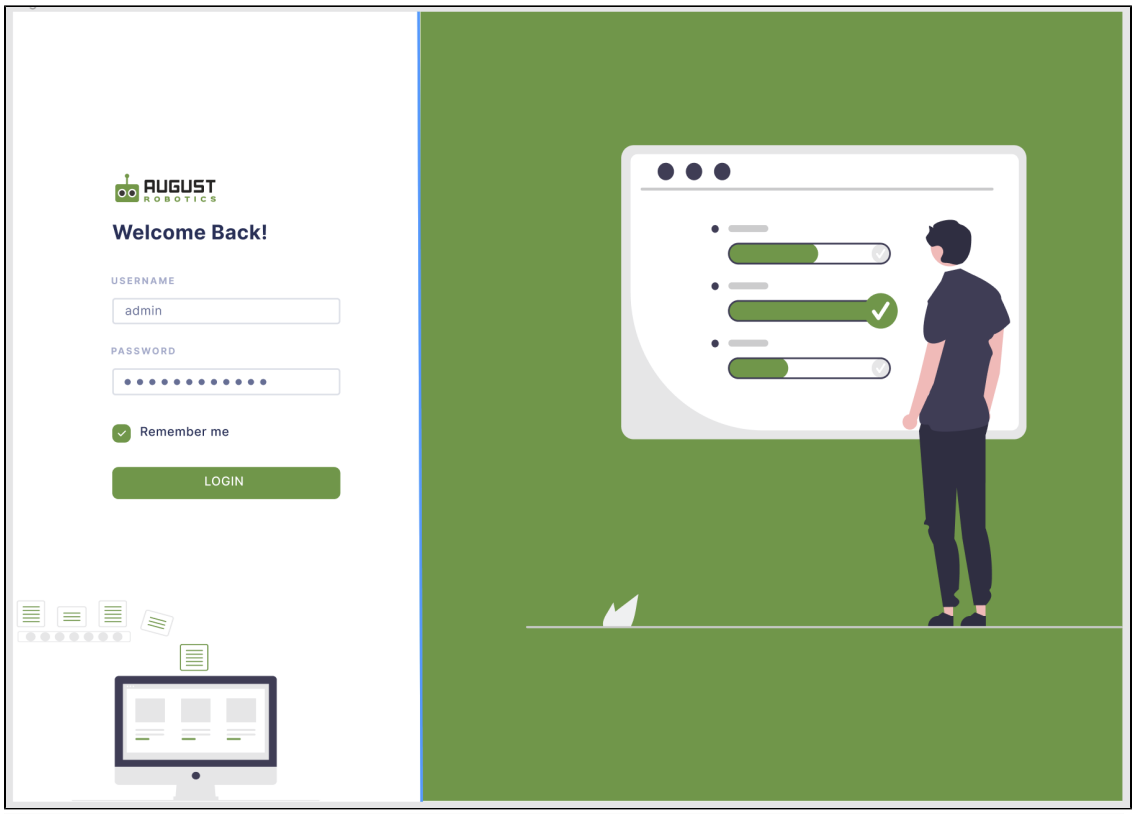
2. User Authentication
  2.1. The user enters their username and password.
  2.2. The web application verifies the user's credentials against the stored information in the database.
  2.3. If the credentials match, the user is authenticated, and the web application retrieves the user's role (viewer, operator, or administrator) from the database. If the credentials do not match, an error message is displayed, and the user is prompted to retry.

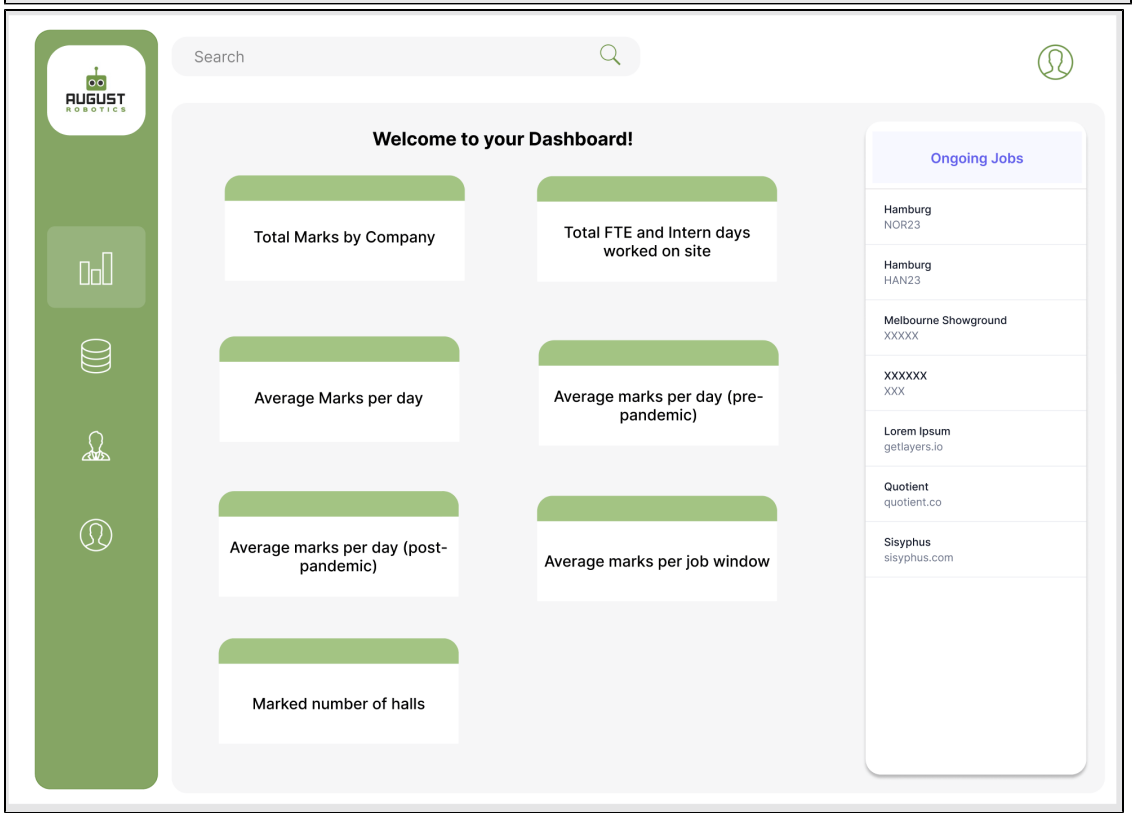
3. Role-based Permissions
  3.1. Viewer (US - 8)
    3.1.1. Access to view-only sections of the web application.
    3.1.2. No ability to create, edit, or delete content.

  3.2. Operator (US - 7)
    3.2.1. Access to view, create, and edit content within the web application.
    3.2.2. Limited ability to delete content or manage certain sections of the web application.
    3.2.3. Access to certain advanced features or tools within the web application.

  3.3. Administrator (US - 6)
    3.3.1. Access to all sections of the web application.
    3.3.2. Full ability to create, edit, and delete content.
    3.3.3. Access to manage users and their permissions.

4. Jump to Dashboard page





Dashboard page

1. User Accesses Dashboard Page
   1.1. After successful login, the user is directed to the dashboard page, which displays the 7 cards with summarized data.

2. Dashboard Cards
  2.1. The web application retrieves the relevant data for each card from the database and displays it as a summary.

3. Card Interaction
  3.1. The user clicks on one of the cards to view detailed information.

4. Display Related Content
  4.1. "Total Marks by Company" (US-1)

  4.2. "Total FTE and Intern days worked on site" (US-2)

  4.3. "Average marks per day"   (US-3)

  4.4. "Average marks per day (pre-pandemic)" (US-3)

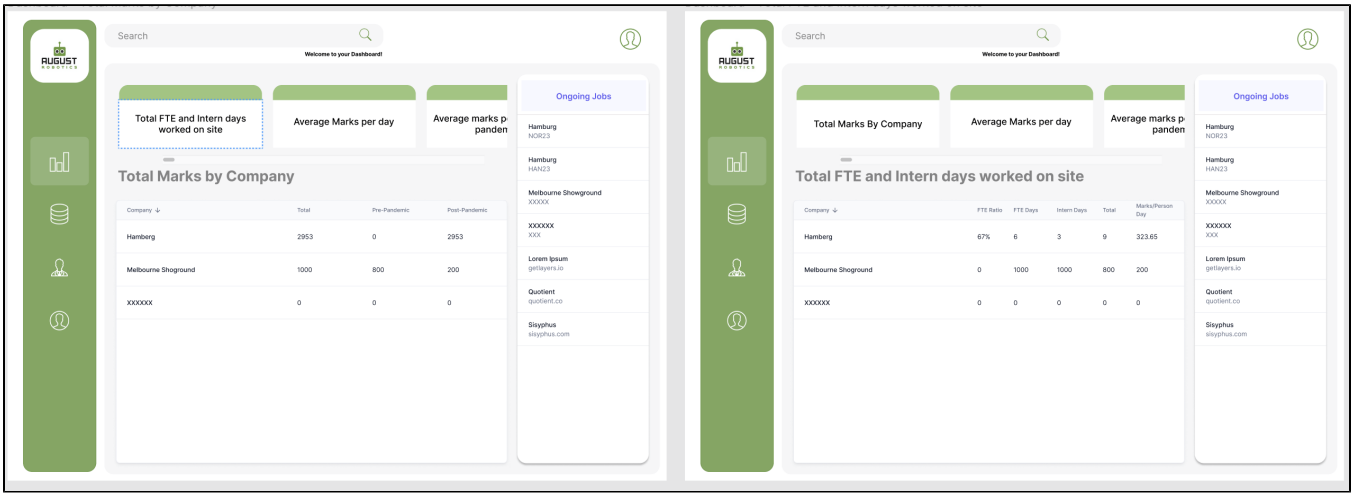
  4.5. "Average marks per day (post-pandemic)" (US-3)

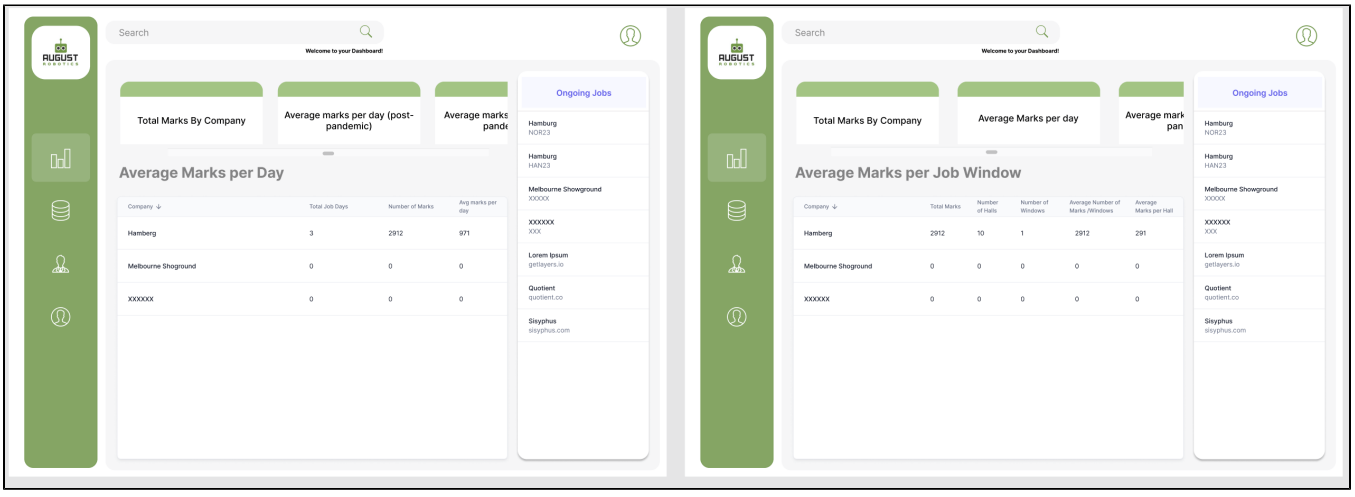
  4.6. "Average marks per job window" (US-4)

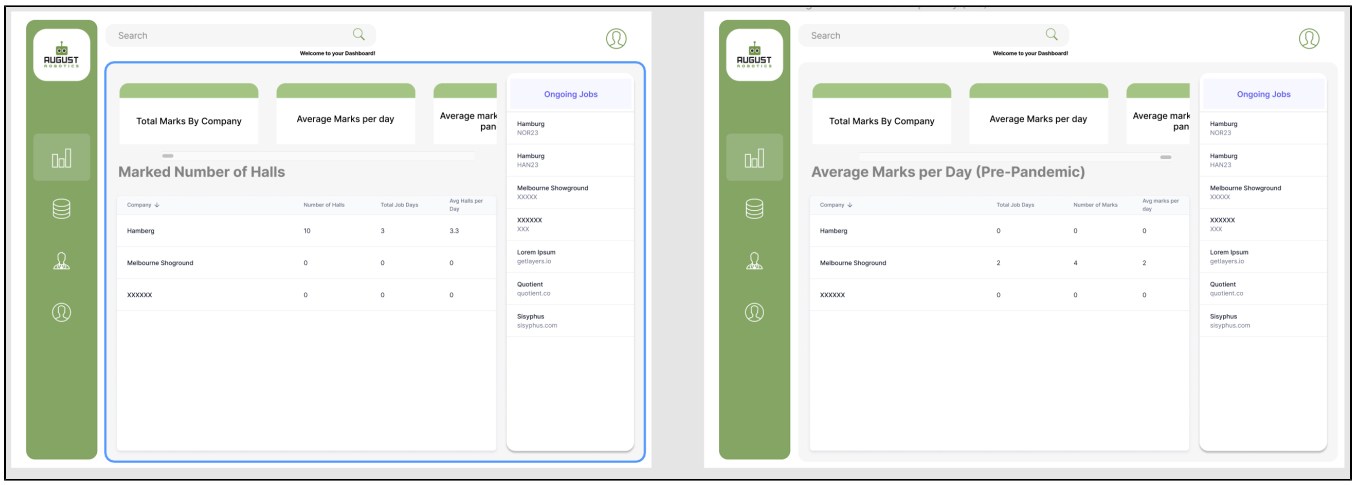
  4.7. "Marked number of halls" (US-5)

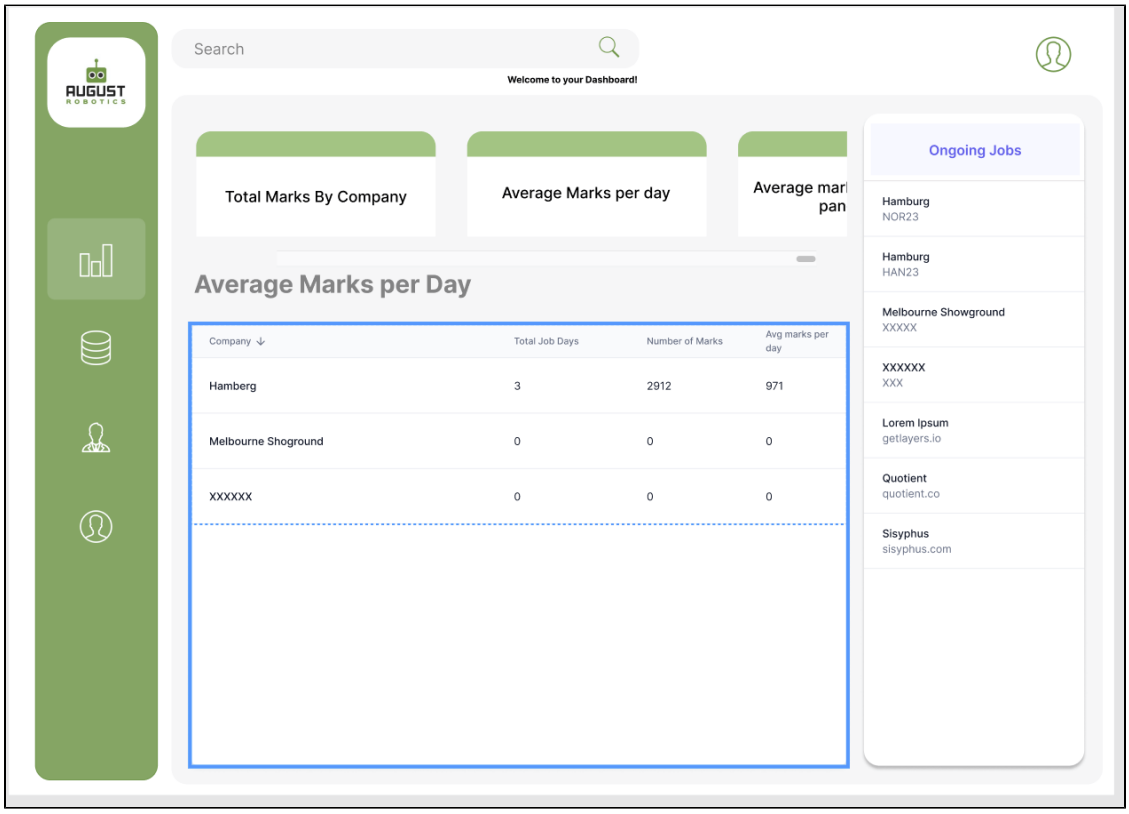
5. Navigation
  5.1. The user can return to the dashboard page by clicking a "Back" button or the dashboard navigation link, which will display the 7 cards again.

6. Refresh Data
  6.1. If the user wants to refresh the data displayed on the cards or in the detailed tables, they can click a "Refresh" button or link, which will retrieve the most recent data from the database and update the display.






Search
Welcome to your Dashboard!
Total Marks By Company
Average Marks per day
Ongoing Jobs
Marked Number of Halls
Company
Number of Halls
Total Job Days
Avg Halls per Day
Hamberg
10
3
3.3
Melbourne Shoground
0
0
0
XXXXXX
0
0
0
Hamburg
NOR23
Hamburg
HAN23
Melbourne Showground
XXXXX
XXXXXX
XXX
Lorem Ipsum
getlayers.io
Quotient
quotient.co
Sisyphus
sisyphus.com
Search
Welcome to your Dashboard!
Total Marks By Company
Average Marks per day
Ongoing Jobs
Average Marks per Day (Pre-Pandemic)
Company
Total Job Days
Number of Marks
Avg marks per day
Hamberg
0
0
0
Melbourne Shoground
2
4
2
XXXXXX
0
0
0
Hamburg
NOR23
Hamburg
HAN23
Melbourne Showground
XXXXX
XXXXXX
XXX
Lorem Ipsum
getlayers.io
Quotient
quotient.co
Sisyphus
sisyphus.com

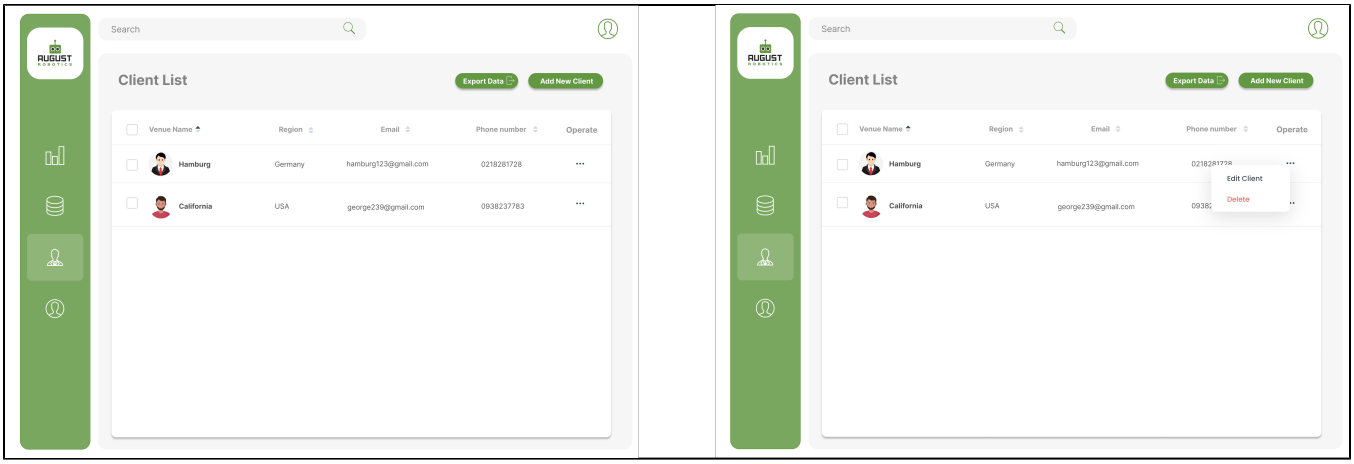

Search
Welcome to your Dashboard!
Total Marks By Company
Average Marks per day
Ongoing Jobs
Average Marks per Day
Company
Total Job Days
Number of Marks
Avg marks per day
Hamberg
3
2912
971
Melbourne Shoground
0
0
0
XXXXXX
0
0
0
Hamburg
NOR23
Hamburg
HAN23
Melbourne Showground
XXXXX
XXXXXX
XXX
Lorem Ipsum
getlayers.io
Quotient
quotient.co
Sisyphus
sisyphus.com

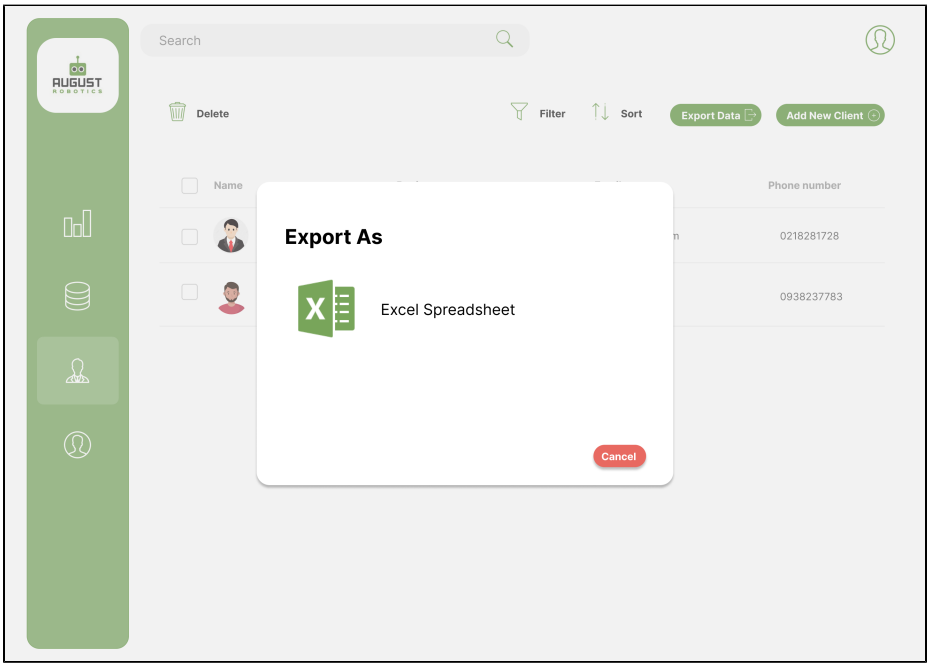
Fixed Data Workflow

Fixed data user stories: US 16, 17, 18, 20.

Client list (Fixed Data)

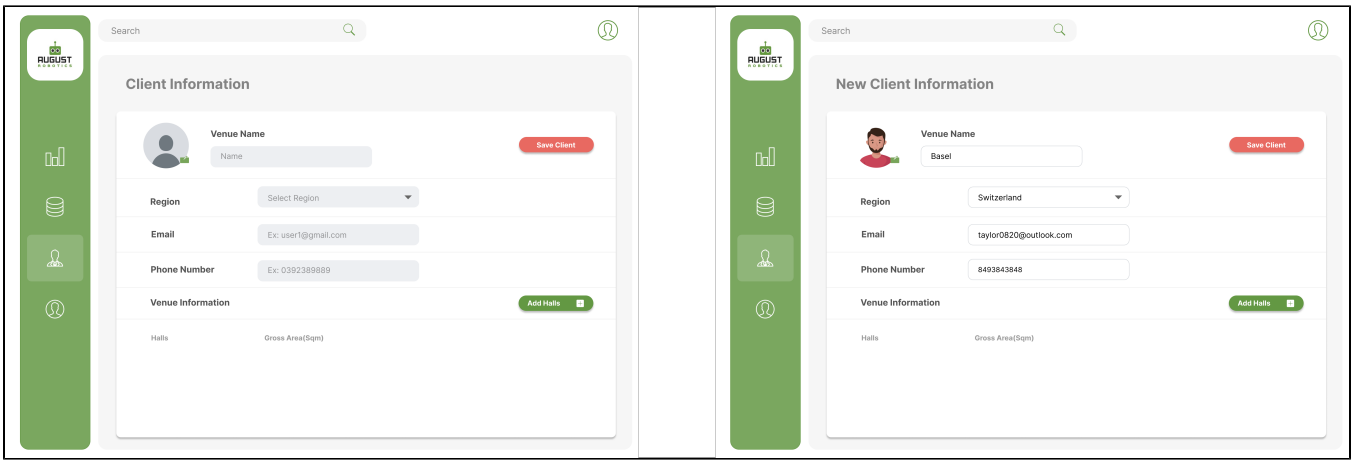
1. When the user clicks on the client icon they can view the client list this includes venue name (to identify client), region, email and phone number for contact details. It also includes operate details. The relates to US-17 to view client details. The operate button shows options to edit/update(US-18) and delete a client. The maintainer can also add a new client by clicking the add new client and also export selected data (US-20).





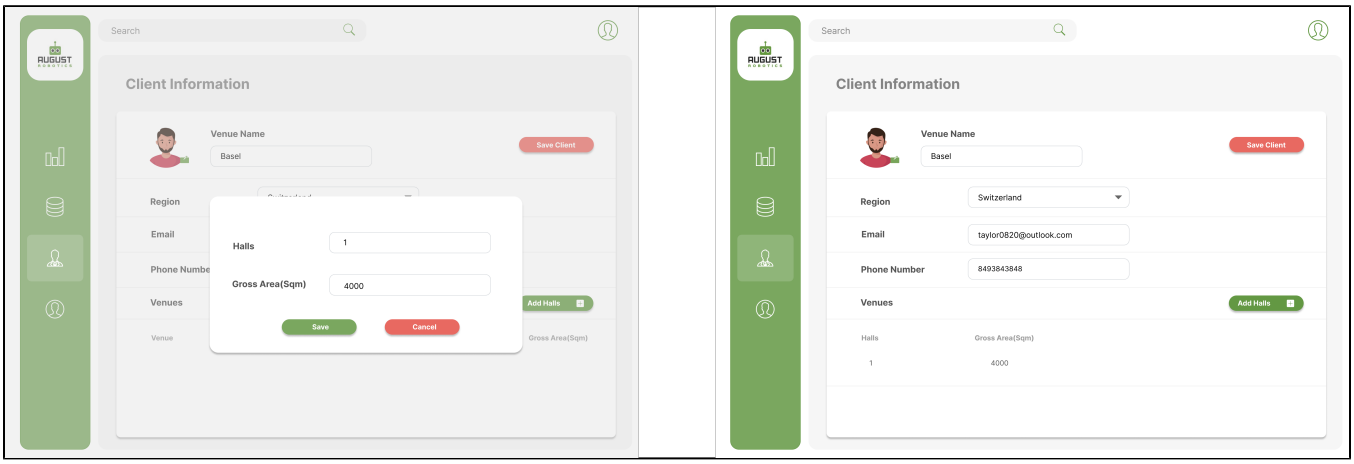
Add New Client Information

2. When the user clicks on add new client they view this following page to add new client information, same as the information displayed in the client list above. This relates to US-16, add new client.



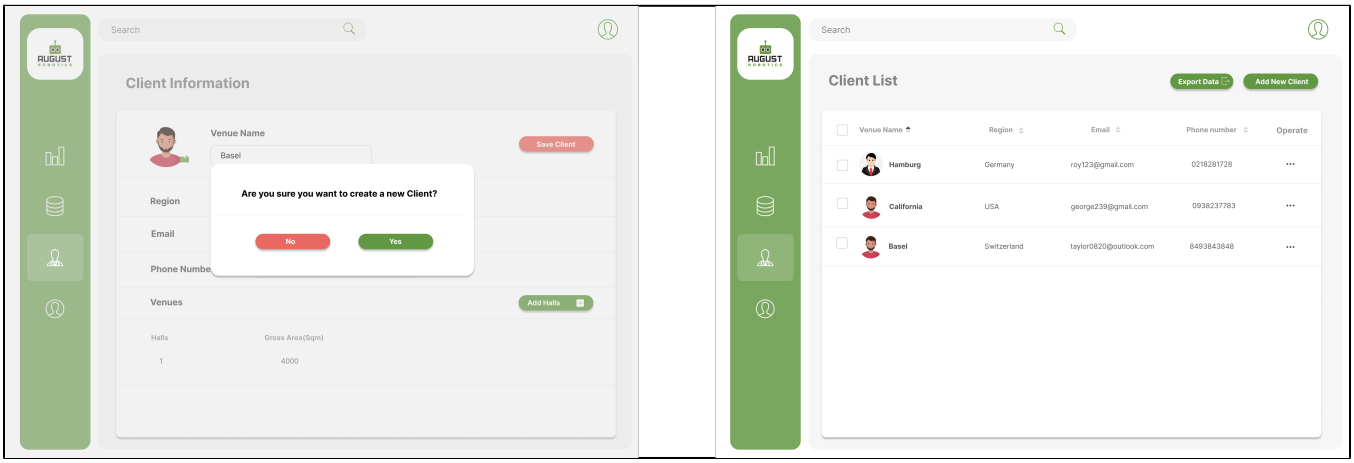
Add Venue Information

3. Continuation of US-16, The maintainer can add halls for the venue and gross area (sqm) to estimate size and resources required for the venue job. The maintainer can then save the data and view it on the screen. The second part relates to US-17, view all information on the client stored in the database.



Save and View New Client on the Client List

4. Finally, the maintainer can save the client and view the new saved client on the client list/fixed data page.

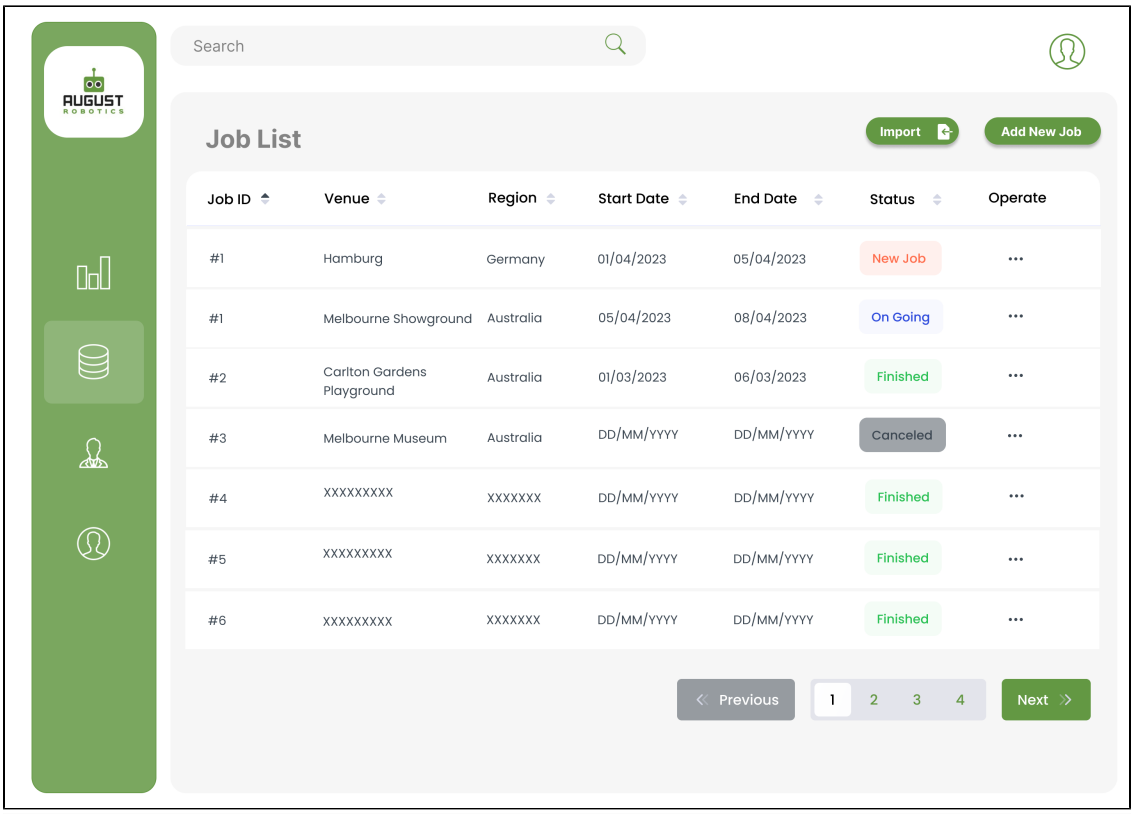


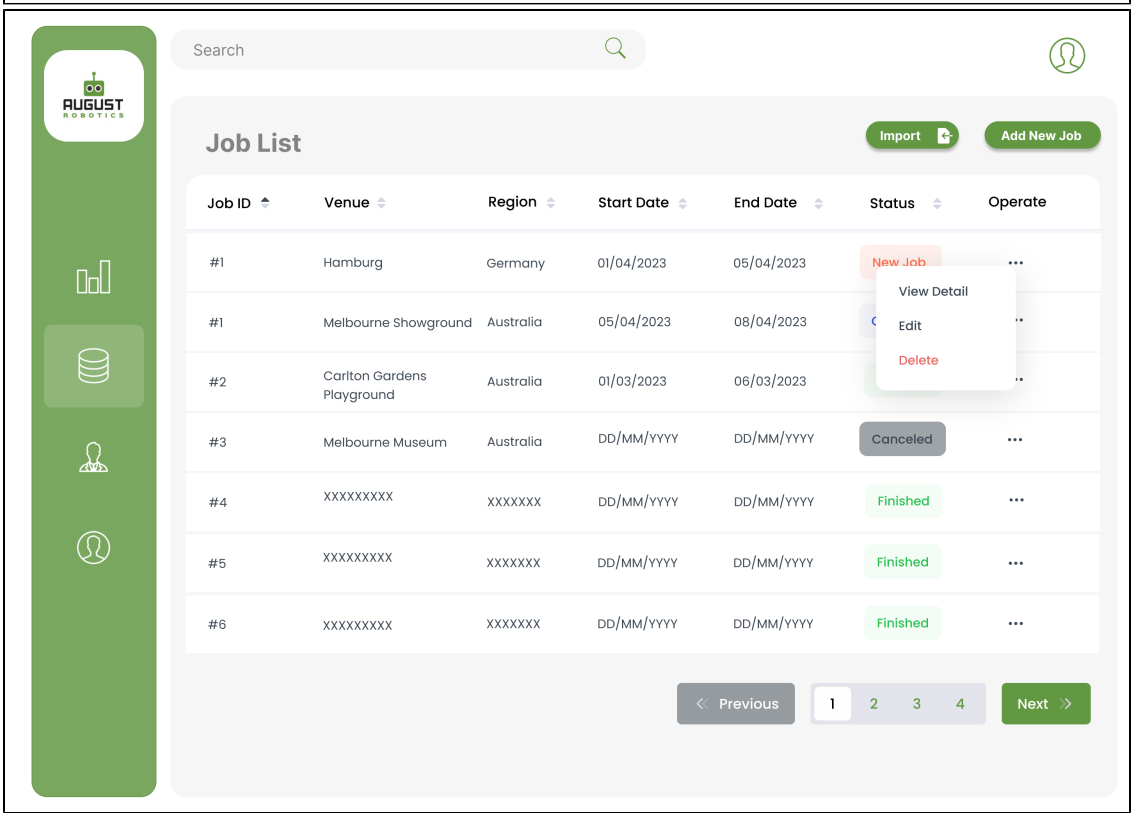
Operational Data Workflow

Operational Data User Stories: 9, 10, 11, 12, 13, 14, 15, 19.

View Job List:

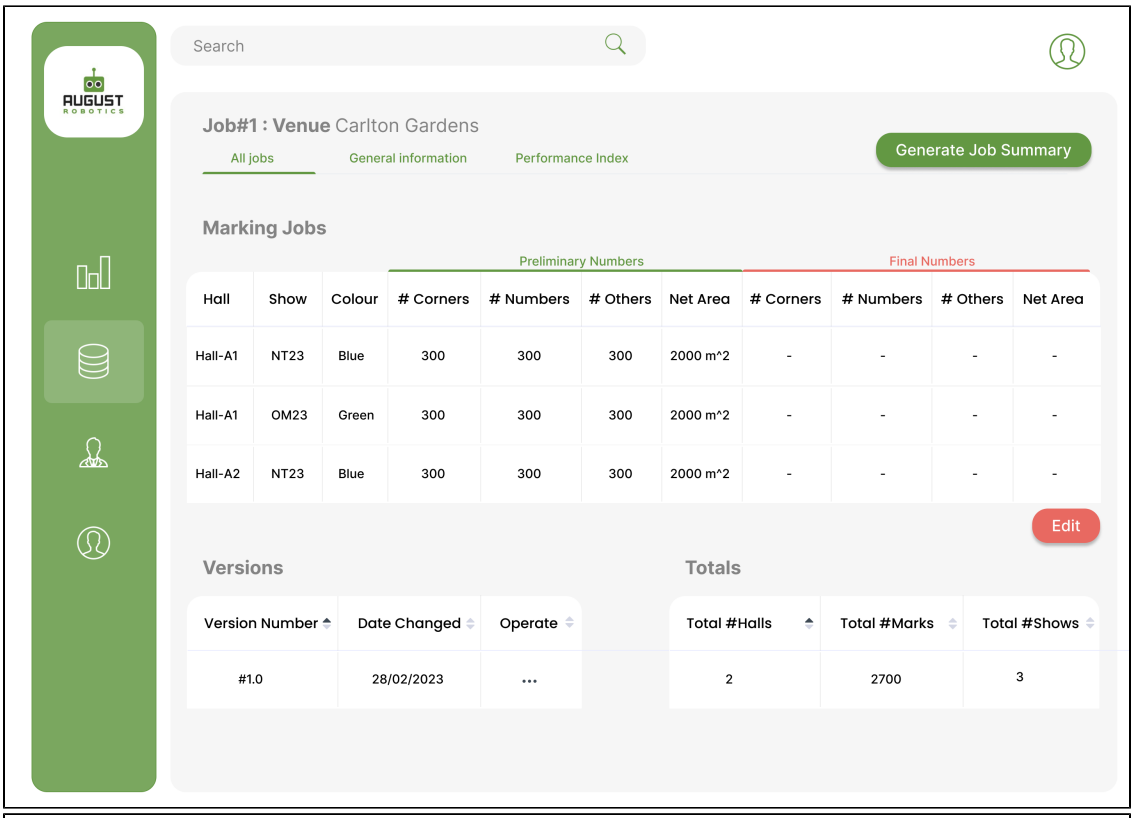
1. When a user clicks on the job list icon on the sidebar, they are navigated to a page listing all current jobs. They are able to view current jobs, their status and perform operations on them by clicking the 3 dots.

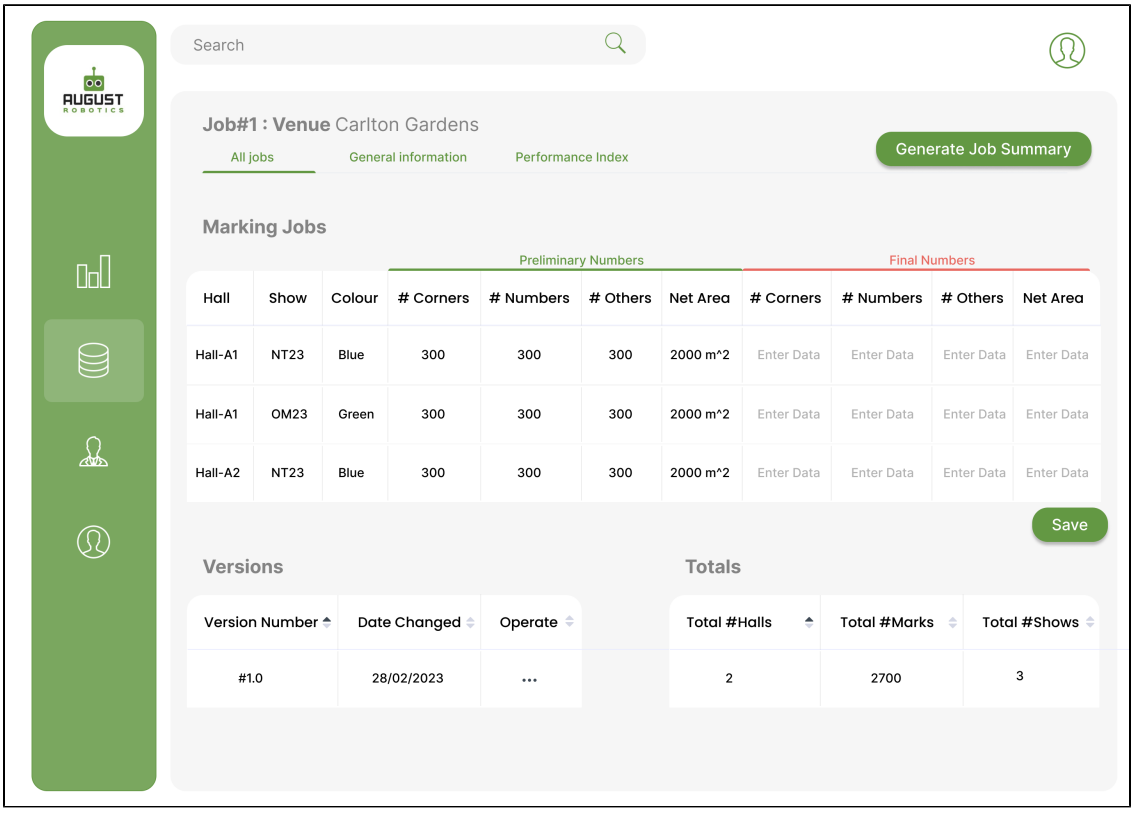


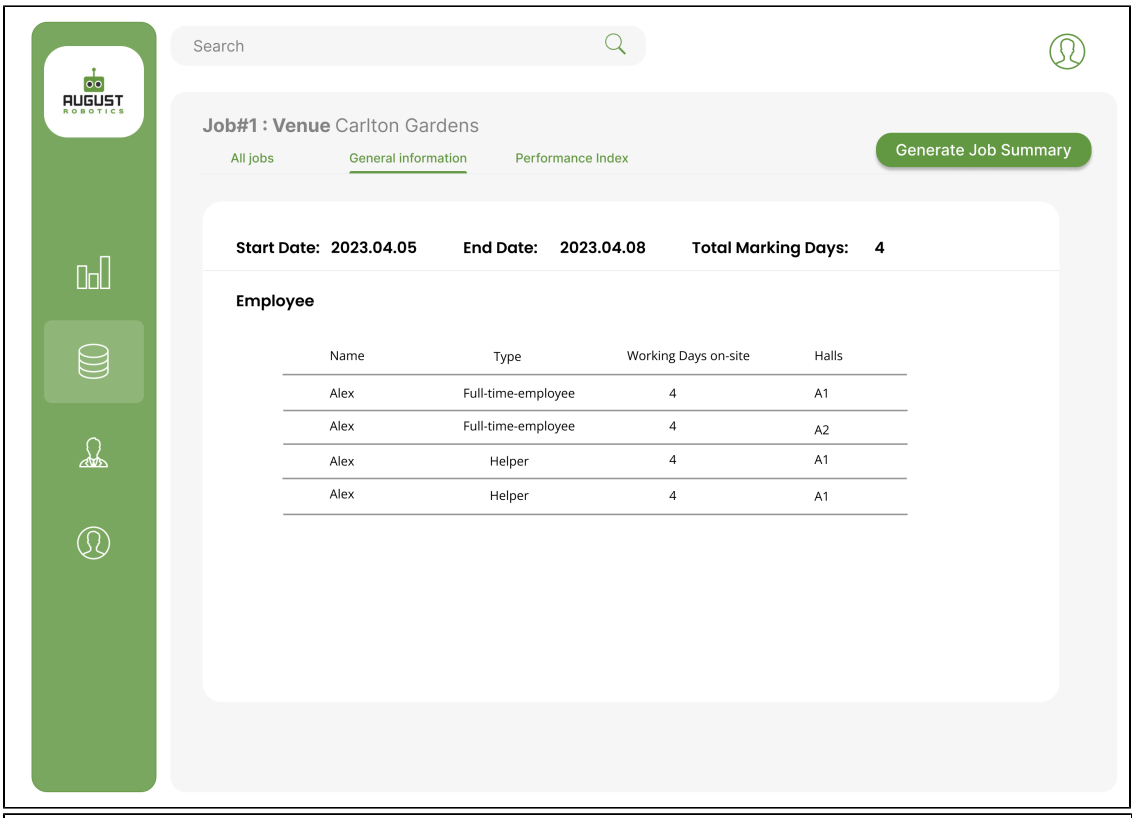


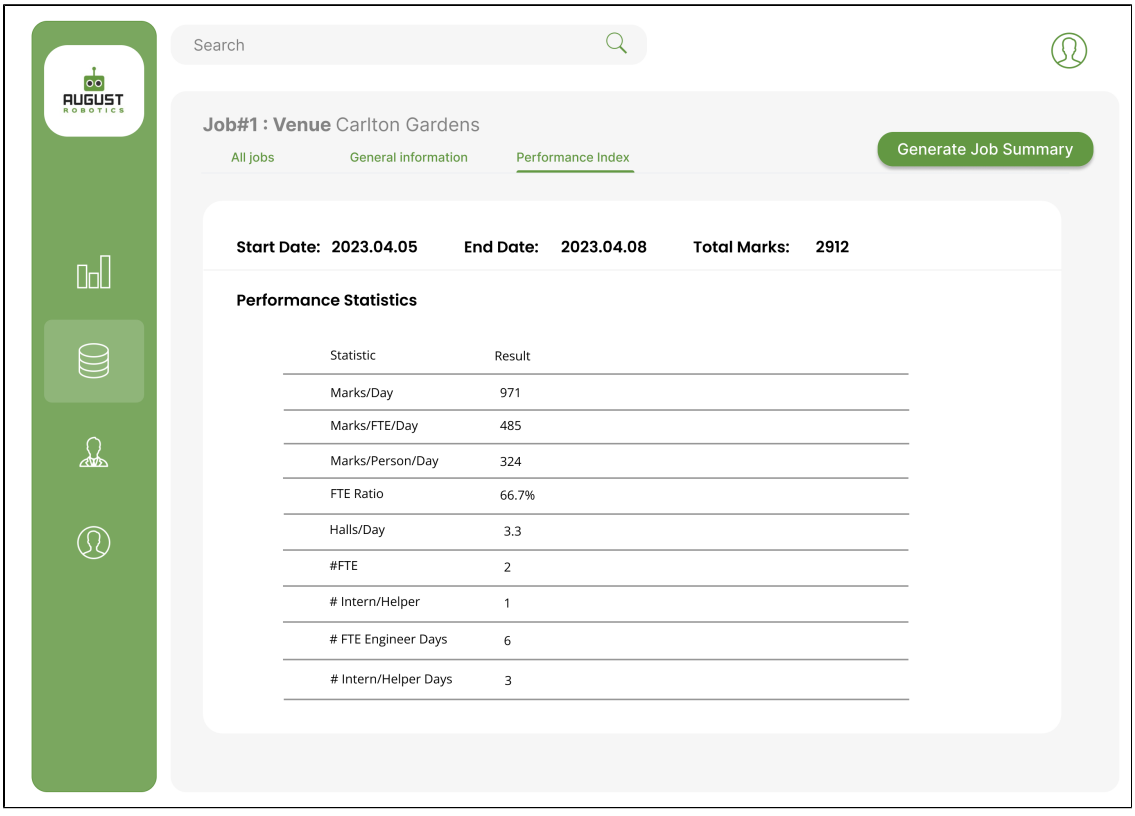
View Job Details:

2. When a user clicks the view details button, they are navigated to a page containing all the jobs details. Users can view all the halls within the job, general information and performance index about the job.


Search
Job#1 : Venue Carlton Gardens
All jobs
General information
Performance Index
Generate Job Summary
Marking Jobs
Preliminary Numbers
Final Numbers
Hall | Show | Colour | # Corners | # Numbers | # Others | Net Area | # Corners | # Numbers | # Others | Net Area
Hall-A1 | NT23 | Blue | 300 | 300 | 300 | 2000 m^2 | - | - | - | -
Hall-A1 | OM23 | Green | 300 | 300 | 300 | 2000 m^2 | - | - | - | -
Hall-A2 | NT23 | Blue | 300 | 300 | 300 | 2000 m^2 | - | - | - | -
Edit
Versions
Version Number | Date Changed | Operate
#1.0 | 28/02/2023 | ...
Totals
Total #Halls | Total #Marks | Total #Shows
2 | 2700 | 3

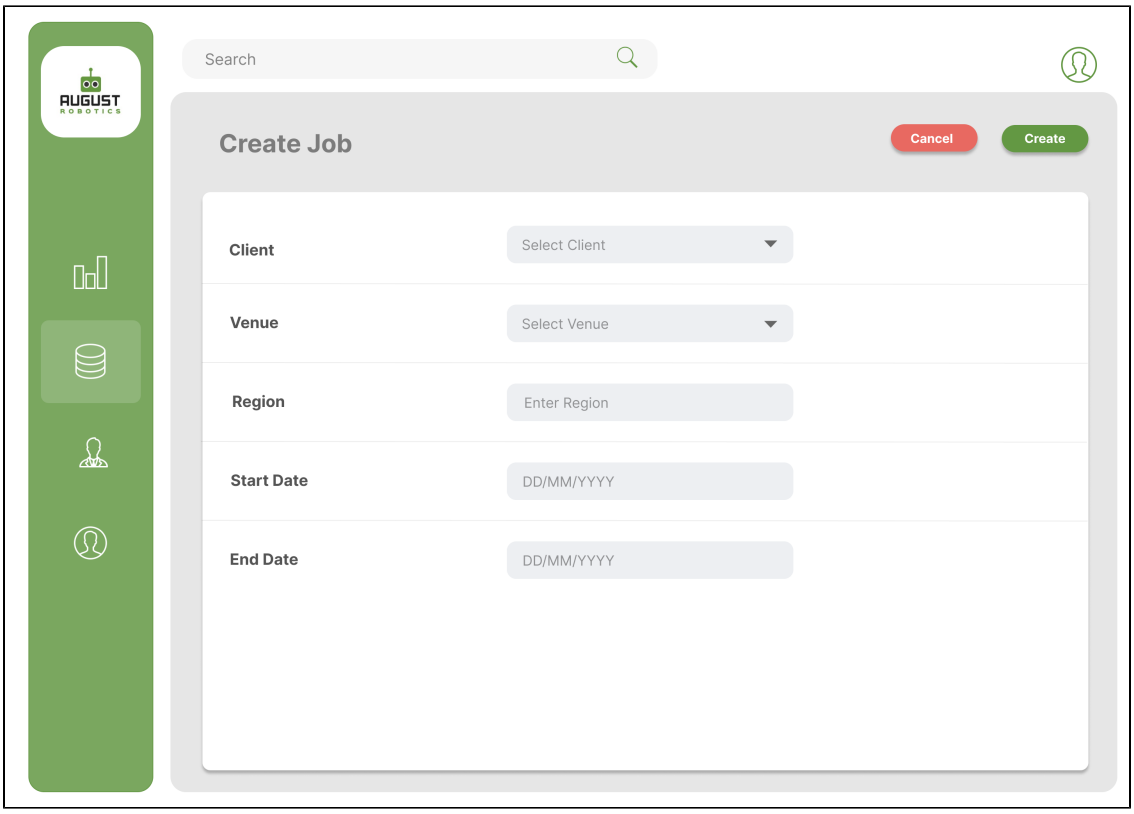

Search
Job#1 : Venue Carlton Gardens
All jobs
General information
Performance Index
Generate Job Summary
Marking Jobs
Preliminary Numbers
Final Numbers
Hall | Show | Colour | # Corners | # Numbers | # Others | Net Area | # Corners | # Numbers | # Others | Net Area
Hall-A1 | NT23 | Blue | 300 | 300 | 300 | 2000 m^2 | Enter Data | Enter Data | Enter Data | Enter Data
Hall-A1 | OM23 | Green | 300 | 300 | 300 | 2000 m^2 | Enter Data | Enter Data | Enter Data | Enter Data
Hall-A2 | NT23 | Blue | 300 | 300 | 300 | 2000 m^2 | Enter Data | Enter Data | Enter Data | Enter Data
Save
Versions
Version Number | Date Changed | Operate
#1.0 | 28/02/2023 | ...
Totals
Total #Halls | Total #Marks | Total #Shows
2 | 2700 | 3





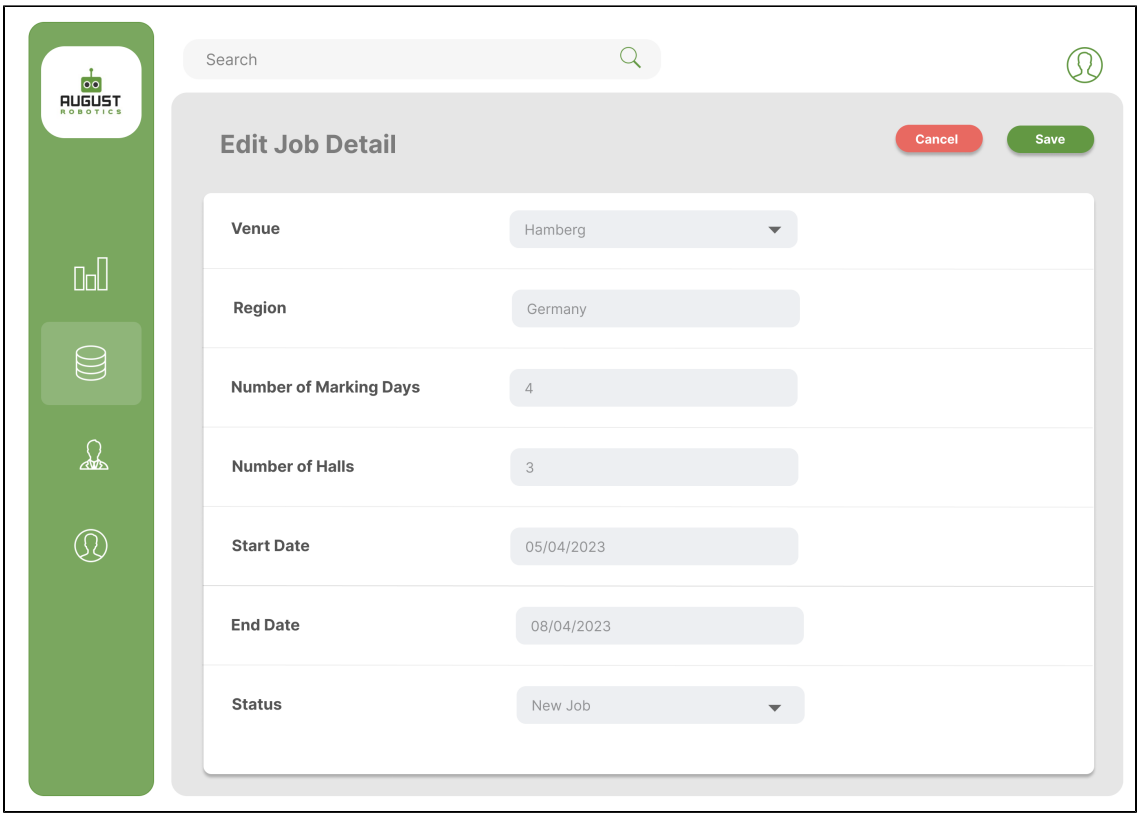
Create a New Job:

3. From the job list page, users can click on the add new job button to create a new event. Some of this data includes fixed data such as the client, venue, region, and number of halls. This will navigate them to a separate page where users can enter event information including the client, venue, preliminary start and end dates etc. When a new job is created, its status is "New Job" and it is empty. When a job has this status users can add hall marking jobs (preliminary data) within the job details. This provides details on the the no. of different types of floor markers that are needed initially for the job.



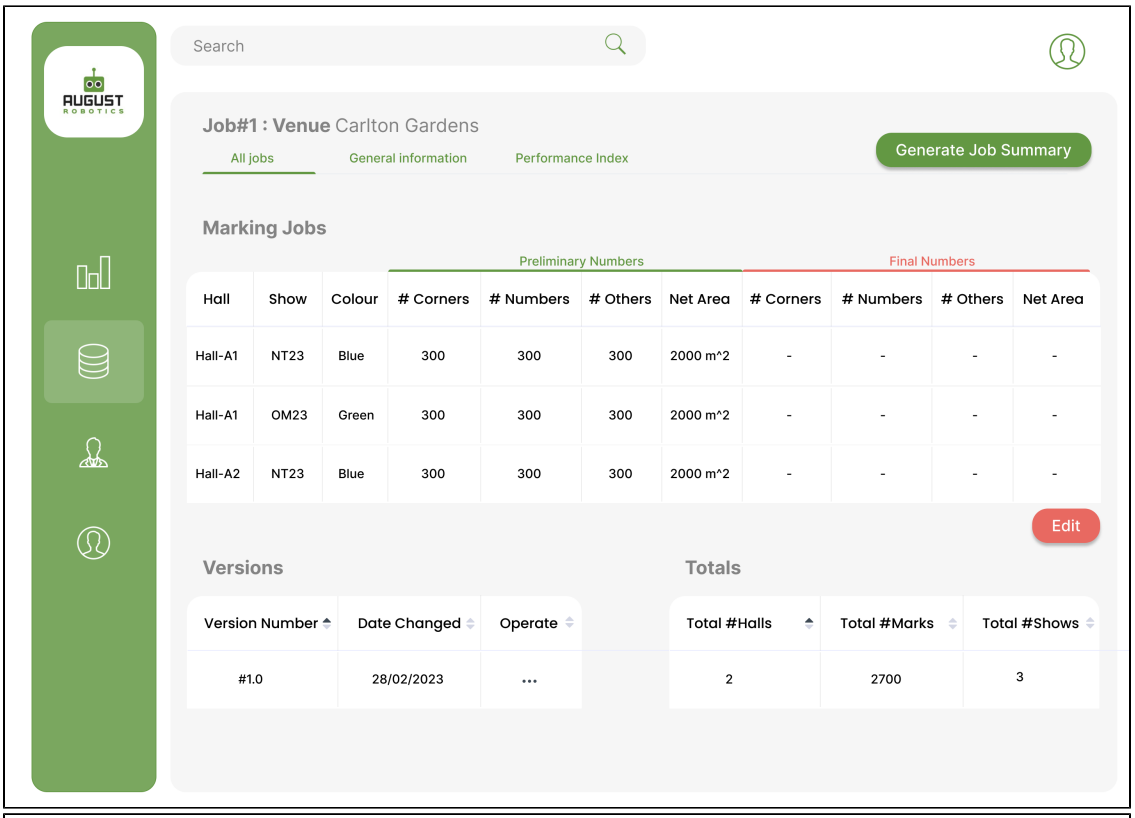
Edit Job

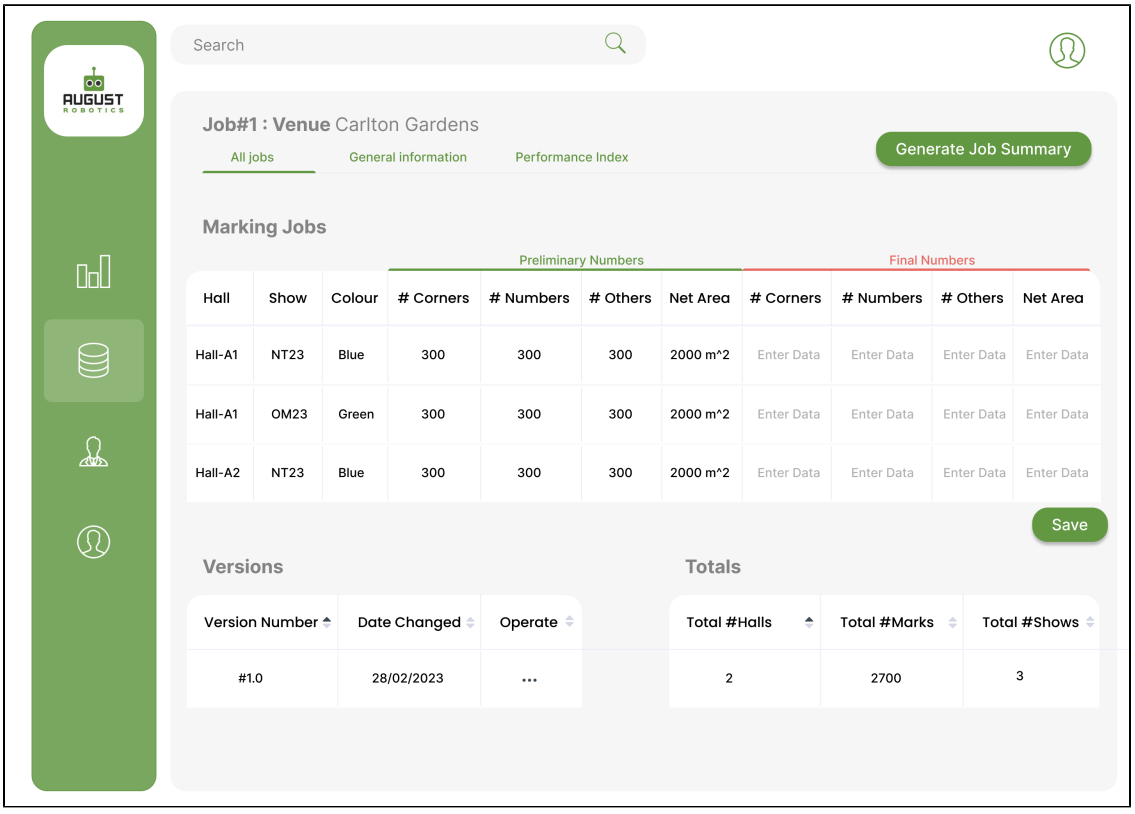
4. For each Job new Job on the Job list page, the following details can be edited, all the fields remain the same, the number of marking days, Number of halls, start date and end date are editable as well. One more field has been added as per client request. This includes the Job Status, to know if the Job is new, ongoing, cancelled or finished. The Maintainer can then save these changes.

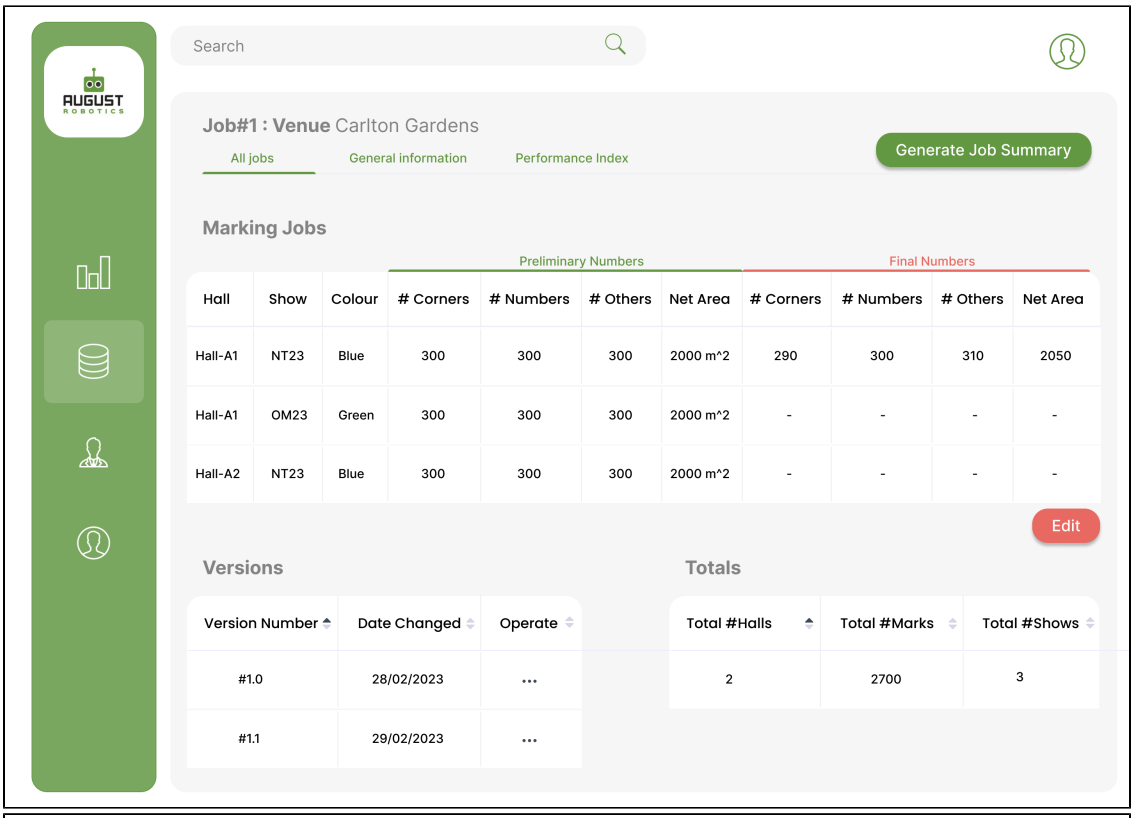


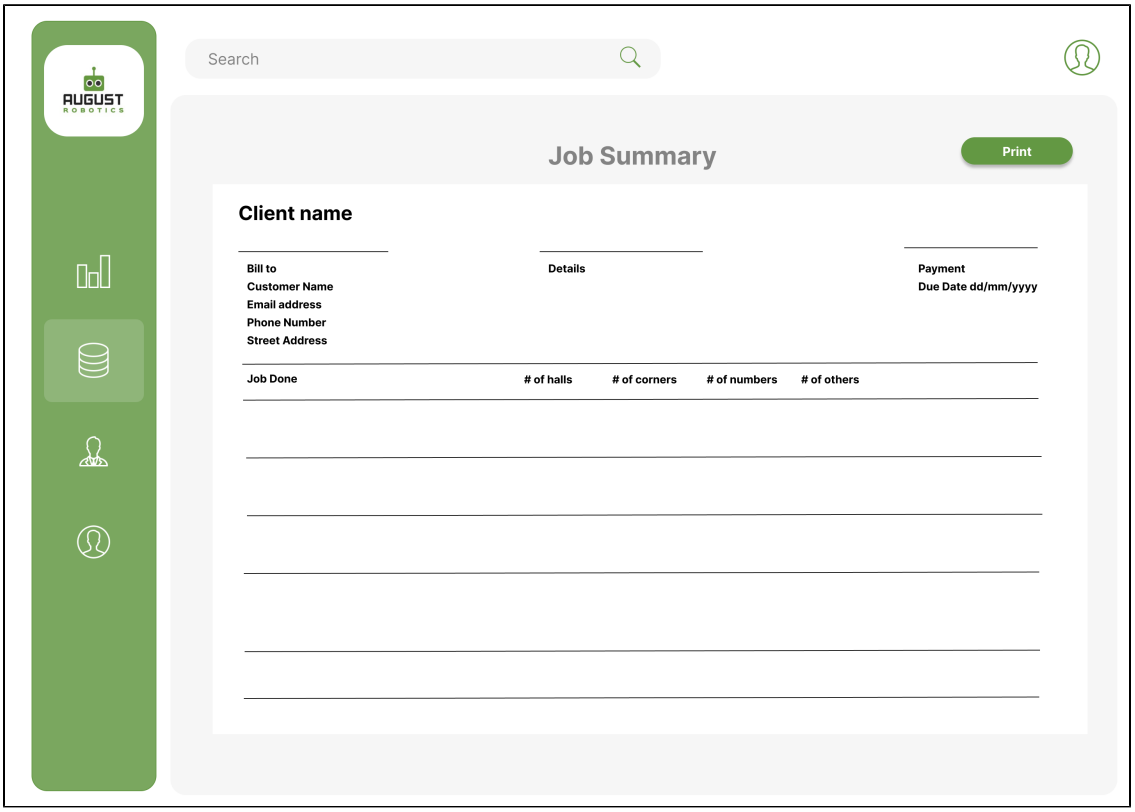
Add Data after Job Done and Job Summary

5. After a job is completed, final number of floor markings need to be added and any changes to the preliminary job data can be made here. This can be done by selecting edit on the Job Information page and then adding the value to the cells for the final numbers or preliminary numbers (like an excel sheet). By clicking on the generate Job Summary the maintainer or viewer can view the summarized details of the job which can then be used to create invoices for the client by the business team.


Search
Job#1 : Venue Carlton Gardens
All jobs
General information
Performance Index
Generate Job Summary
Marking Jobs
Preliminary Numbers
Final Numbers
Hall | Show | Colour | # Corners | # Numbers | # Others | Net Area | # Corners | # Numbers | # Others | Net Area
Hall-A1 | NT23 | Blue | 300 | 300 | 300 | 2000 m^2 | - | - | - | -
Hall-A1 | OM23 | Green | 300 | 300 | 300 | 2000 m^2 | - | - | - | -
Hall-A2 | NT23 | Blue | 300 | 300 | 300 | 2000 m^2 | - | - | - | -
Edit
Versions
Version Number | Date Changed | Operate
#1.0 | 28/02/2023 | ...
Totals
Total #Halls | Total #Marks | Total #Shows
2 | 2700 | 3

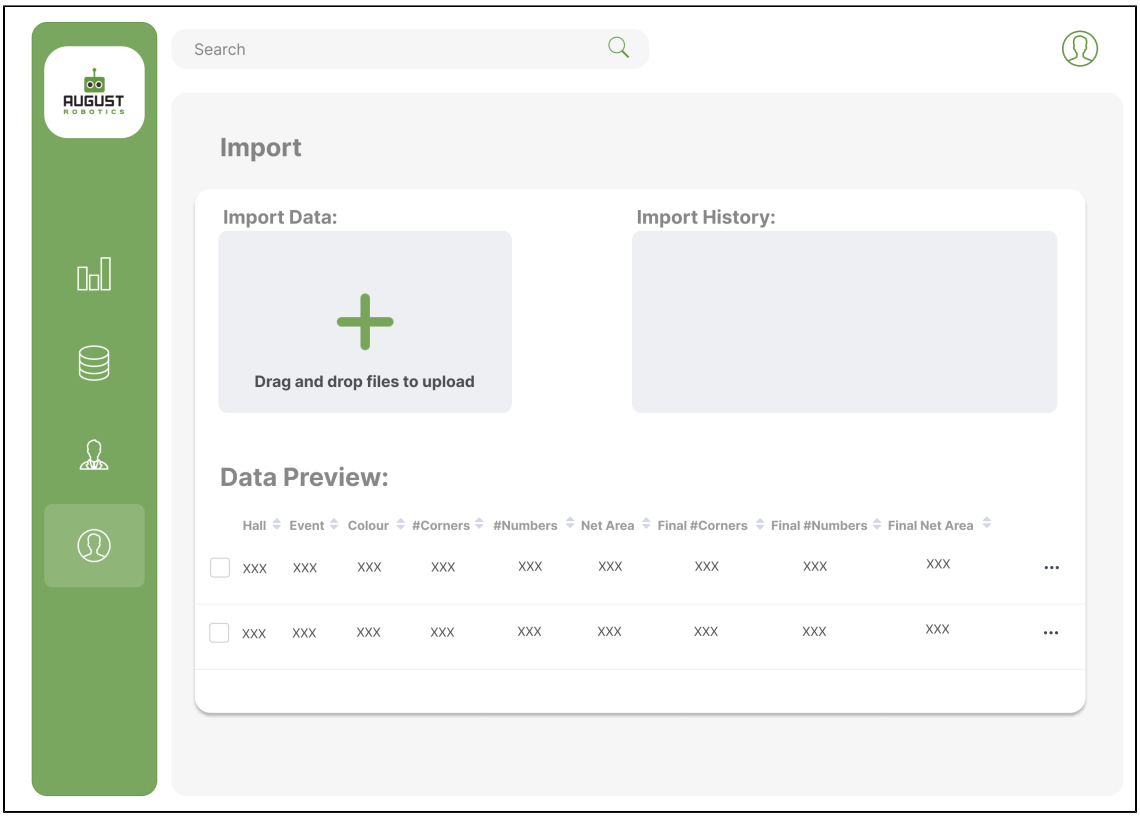

Search
Job#1 : Venue Carlton Gardens
All jobs
General information
Performance Index
Generate Job Summary
Marking Jobs
Preliminary Numbers
Final Numbers
Hall | Show | Colour | # Corners | # Numbers | # Others | Net Area | # Corners | # Numbers | # Others | Net Area
Hall-A1 | NT23 | Blue | 300 | 300 | 300 | 2000 m^2 | Enter Data | Enter Data | Enter Data | Enter Data
Hall-A1 | OM23 | Green | 300 | 300 | 300 | 2000 m^2 | Enter Data | Enter Data | Enter Data | Enter Data
Hall-A2 | NT23 | Blue | 300 | 300 | 300 | 2000 m^2 | Enter Data | Enter Data | Enter Data | Enter Data
Save
Versions
Version Number | Date Changed | Operate
#1.0 | 28/02/2023 | ...
Totals
Total #Halls | Total #Marks | Total #Shows
2 | 2700 | 3





Import Data

6. A job data can also be imported from an excel sheet to be added to the operational data. This includes all the same details as above to input the details onto the platform. This feature is mainly useful for historical data that may have to be added to the AR database platform.

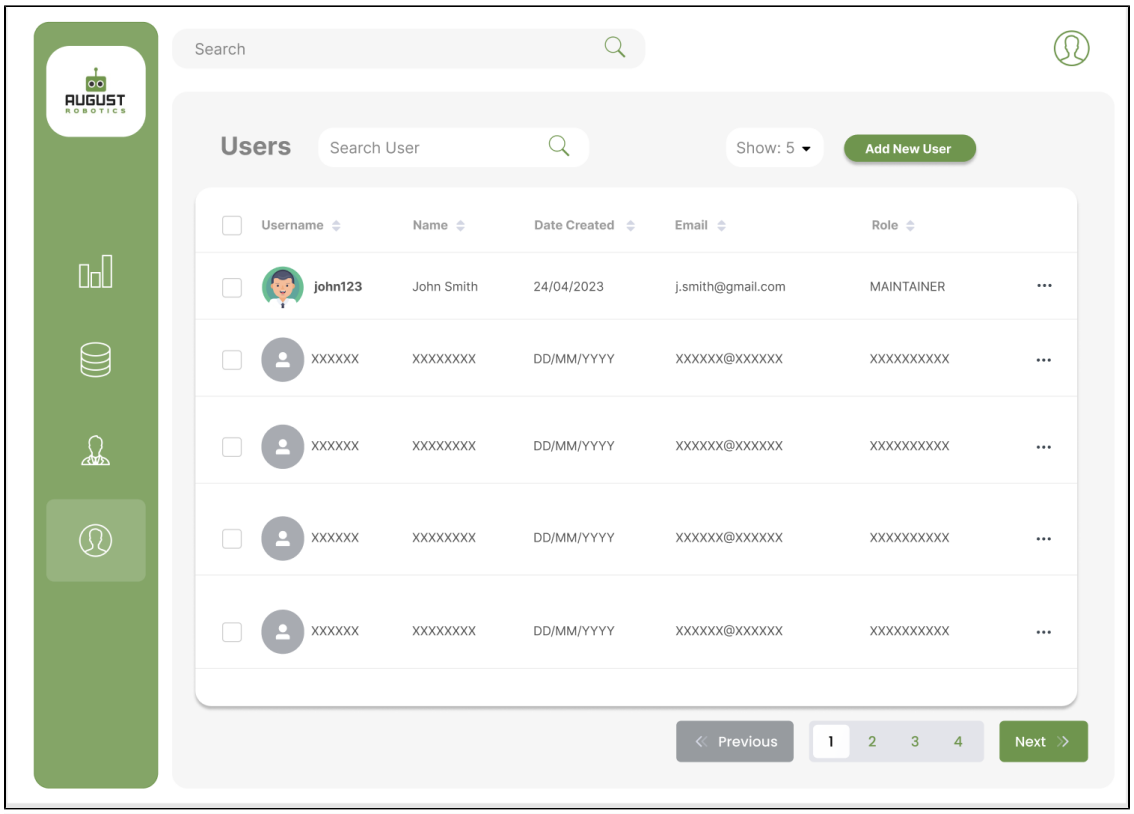

AUGUST
ROBOTICS
Search
Import
Import Data:
Drag and drop files to upload
Import History:
Data Preview:
Hall
Event
Colour
#Corners
#Numbers
Net Area
Final #Corners
Final #Numbers
Final Net Area
XXX

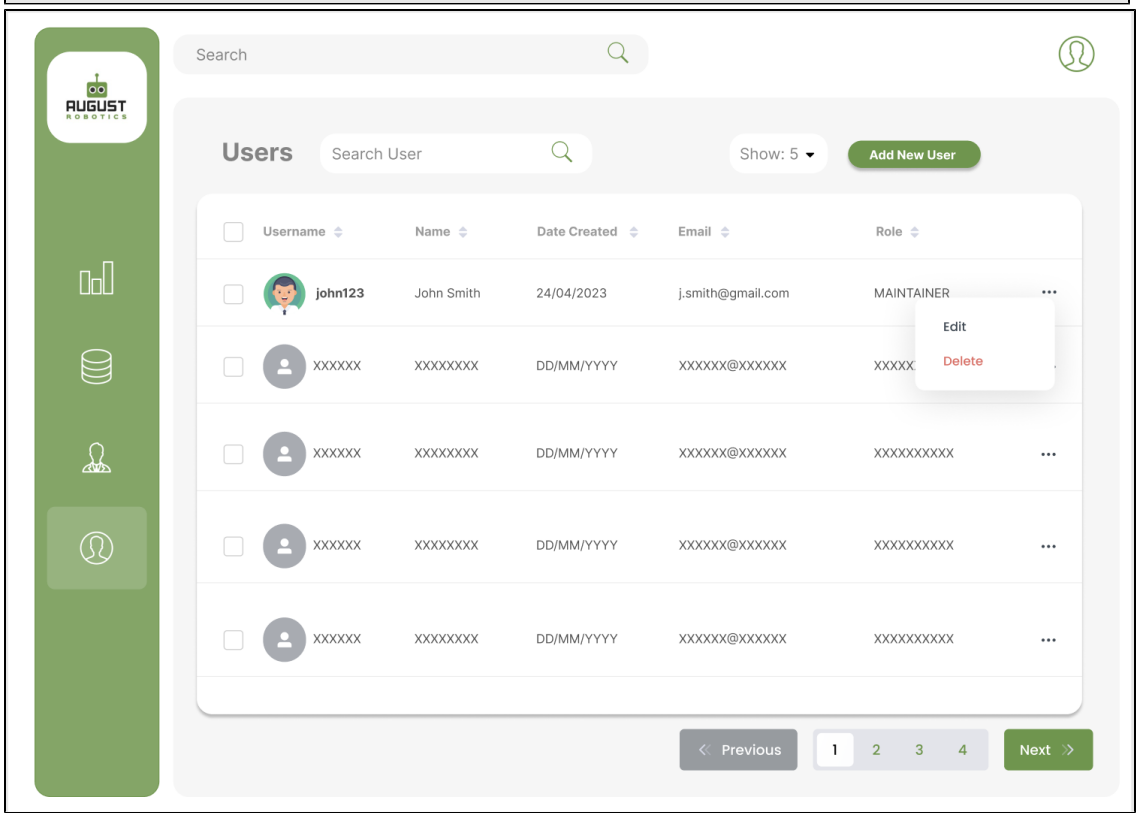
User Management Workflow

User Management User Stories: 22, 23, 24

View User List:

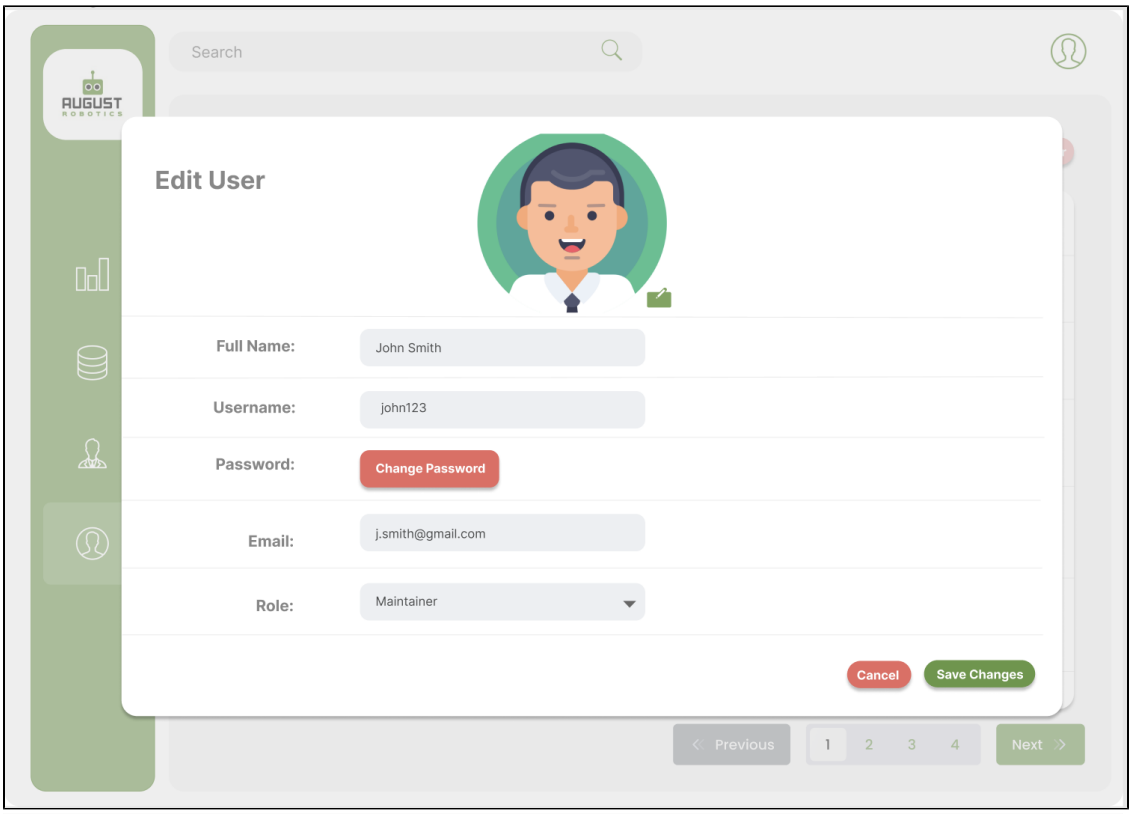
1. When the user management icon is clicked on the sidebar, users are navigated to the user list page. On this page users can view all users and their details. The admin can choose to edit or delete these users (U-23).

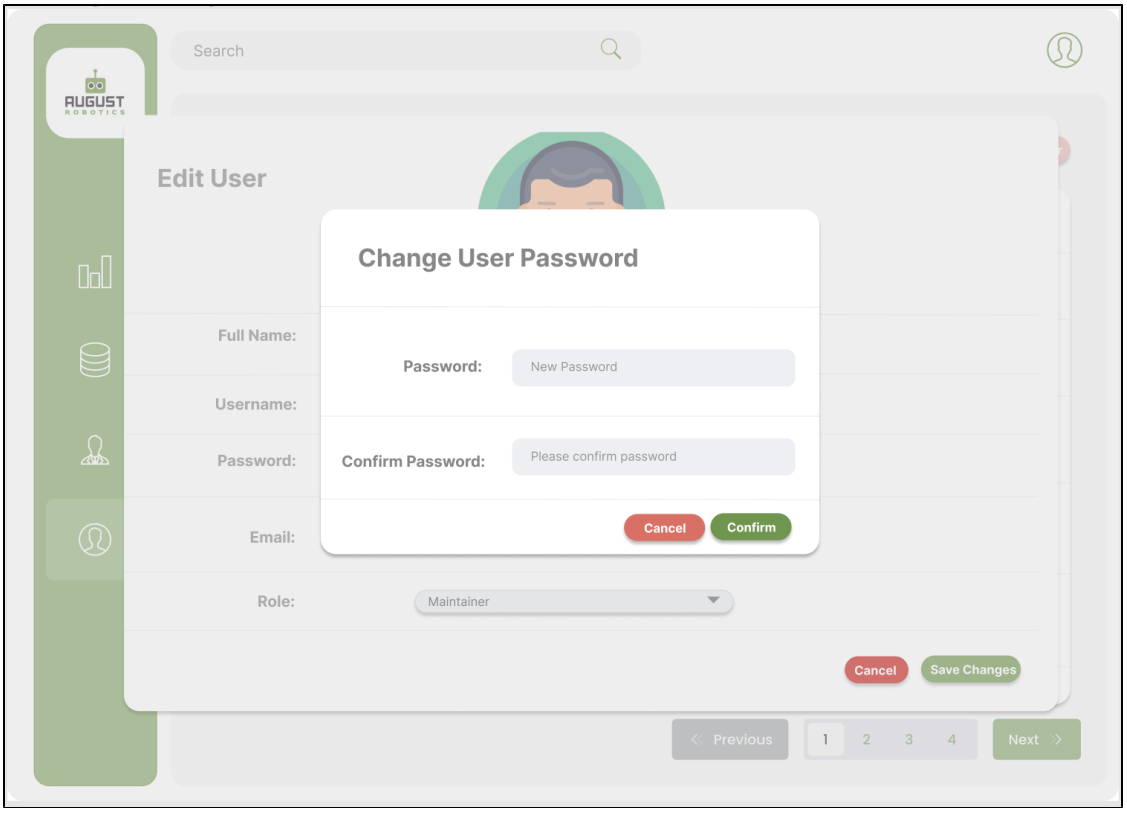




Edit User Details:

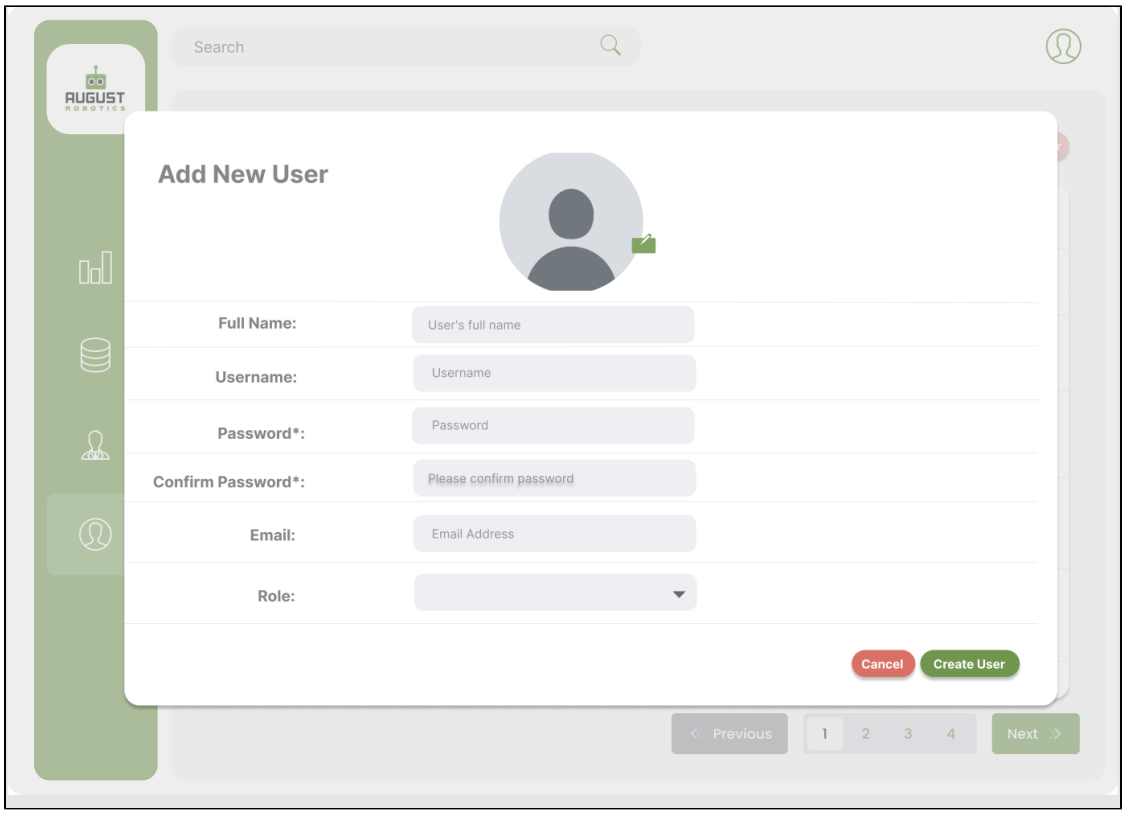
2. When a user clicks on the 3 dots next to a user, they can select the edit button. In this page a user can edit user details, or change a user's password (U-24).

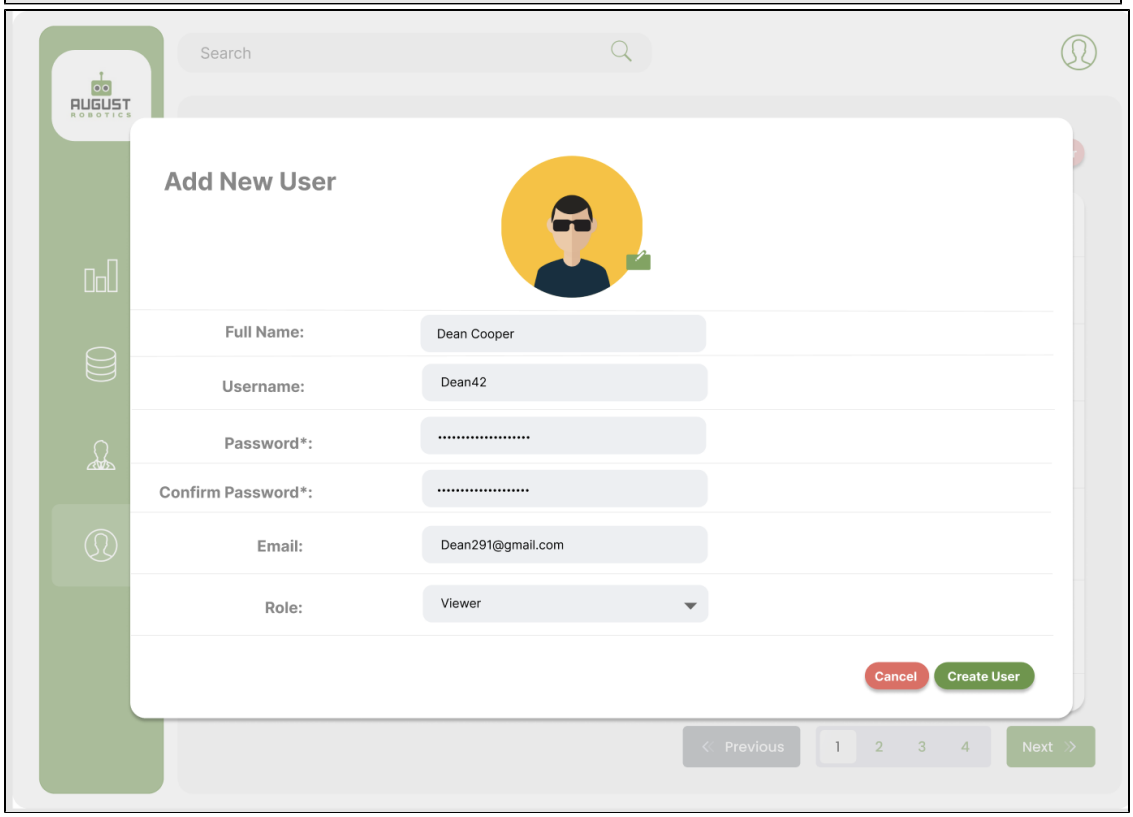




Add a new user:

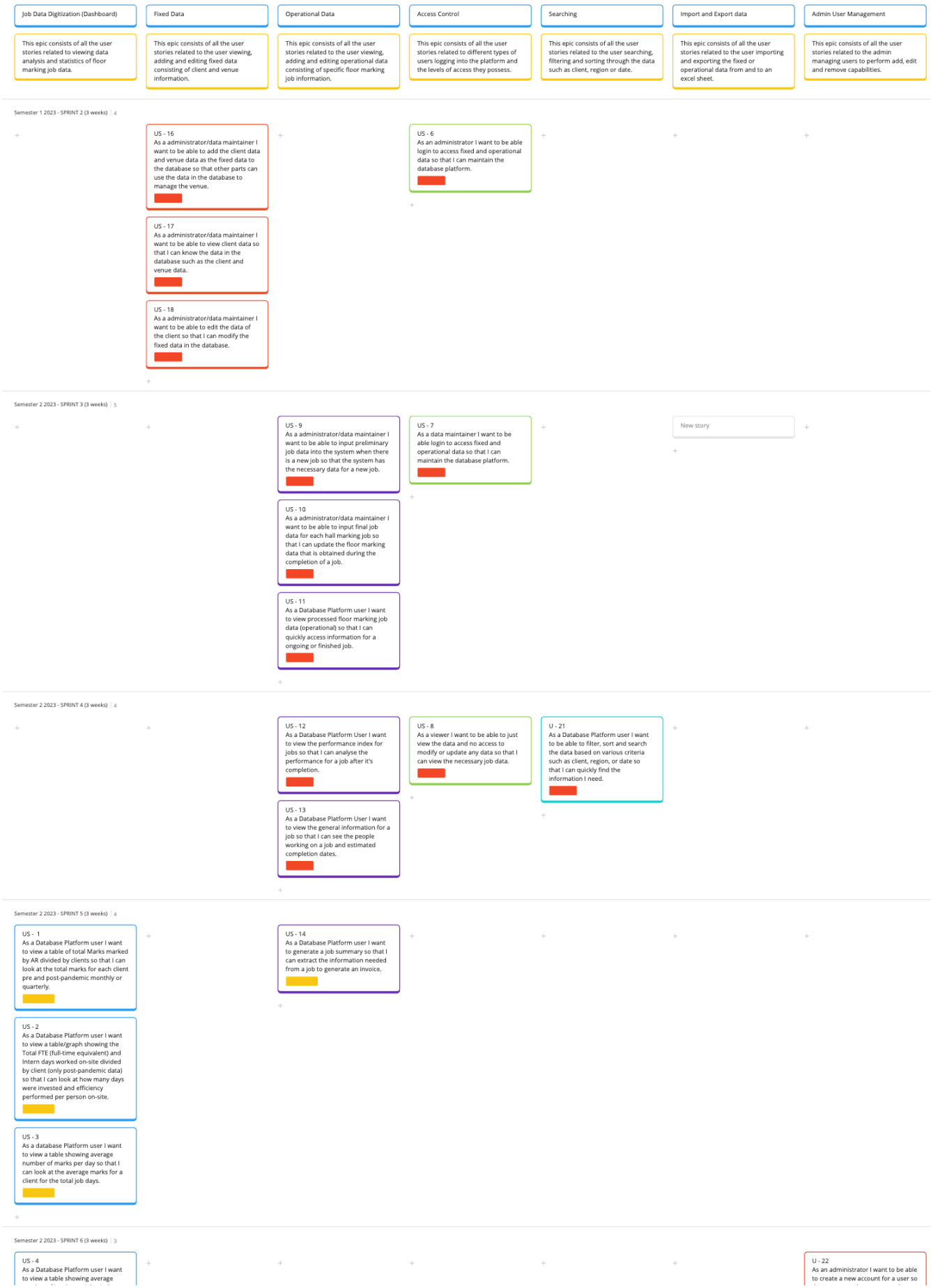
3. From the user list page, an admin can click the add new user button to add a new user (U-22). This will navigate the admin to a separate page where new user details are filled out.

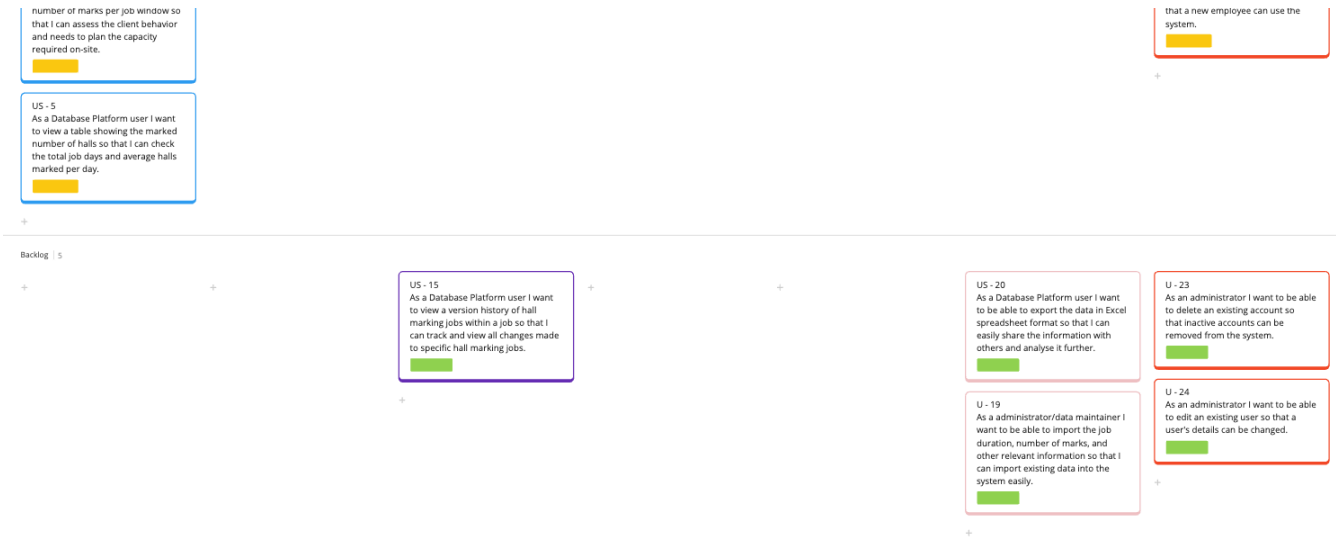

Search
AUGUST
ROBOTICS
Add New User
Full Name:
User's full name
Username:
Username
Password*:
Password
Confirm Password*:
Please confirm password
Email:
Email Address
Role:
Cancel
Create User
Previous
1
2
3
4
Next


Search
AUGUST
ROBOTICS
Add New User
Full Name:
Dean Cooper
Username:
Dean42
Password*:
Confirm Password*:
Email:
Dean291@gmail.com
Role:
Viewer
Cancel
Create User
Previous
1
2
3
4
Next

User Story Map

Miro Link To User story map: https://miro.com/app/board/uXjVMWqellw=/?moveToWidget=3458764553140373954&cot=14


Job Data Digitization (Dashboard)
This epic consists of all the user stories related to viewing data analysis and statistics of floor marking job data.
Fixed Data
This epic consists of all the user stories related to the user viewing, adding and editing fixed data consisting of client and venue information.
Operational Data
This epic consists of all the user stories related to the user viewing, adding and editing operational data consisting of specific floor marking job information.
Access Control
This epic consists of all the user stories related to different types of users logging into the platform and the levels of access they possess.
Searching
This epic consists of all the user stories related to the user searching, filtering and sorting through the data such as client, region or date.
Import and Export data
This epic consists of all the user stories related to the user importing and exporting the fixed or operational data from and to an excel sheet.
Admin User Management
This epic consists of all the user stories related to the admin managing users to perform add, edit and remove capabilities.
Semester 1 2023 - SPRINT 2 (3 weeks) | 4
US - 16
As a administrator/data maintainer I want to be able to add the client data and venue data as the fixed data to the database so that other parts can use the data in the database to manage the venue.
US - 6
As an administrator I want to be able login to access fixed and operational data so that I can maintain the database platform.
US - 17
As a administrator/data maintainer I want to be able to view client data so that I can know the data in the database such as the client and venue data.
US - 18
As a administrator/data maintainer I want to be able to edit the data of the client so that I can modify the fixed data in the database.
Semester 2 2023 - SPRINT 3 (3 weeks) | 5
US - 9
As a administrator/data maintainer I want to be able to input preliminary job data into the system when there is a new job so that the system has the necessary data for a new job.
US - 7
As a data maintainer I want to be able login to access fixed and operational data so that I can maintain the database platform.
New story
US - 10
As a administrator/data maintainer I want to be able to input final job data for each hall marking job so that I can update the floor marking data that is obtained during the completion of a job.
US - 11
As a Database Platform user I want to view processed floor marking job data (operational) so that I can quickly access information for a ongoing or finished job.
Semester 2 2023 - SPRINT 4 (3 weeks) | 4
US - 12
As a Database Platform User I want to view the performance index for jobs so that I can analyse the performance for a job after it's completion.
US - 8
As a viewer I want to be able to just view the data and no access to modify or update any data so that I can view the necessary job data.
U - 21
As a Database Platform user I want to be able to filter, sort and search the data based on various criteria such as client, region, or date so that I can quickly find the information I need.
US - 13
As a Database Platform User I want to view the general information for a job so that I can see the people working on a job and estimated completion dates.
Semester 2 2023 - SPRINT 5 (3 weeks) | 4
US - 1
As a Database Platform user I want to view a table of total Marks marked by AR divided by clients so that I can look at the total marks for each client pre and post-pandemic monthly or quarterly.
US - 14
As a Database Platform user I want to generate a job summary so that I can extract the information needed from a job to generate an invoice.
US - 2
As a Database Platform user I want to view a table/graph showing the Total FTE (full-time equivalent) and Intern days worked on-site divided by client (only post-pandemic data) so that I can look at how many days were invested and efficiency performed per person on-site.
US - 3
As a database Platform user I want to view a table showing average number of marks per day so that I can look at the average marks for a client for the total job days.
Semester 2 2023 - SPRINT 6 (3 weeks) | 3
US - 4
As a Database Platform user I want to view a table showing average
U - 22
As an administrator I want to be able to create a new account for a user so



- In Semester 1 - Sprint 2, the must-have user stories relate to mainly fixed data as we need to add client/venue information before we can add jobs for them, thus this epic needs to be completed first. It also includes access control for the system admin as that is the first user to be created with all privileges.
- In Semester 2 - Sprint 3, the must-have user stories relate to mainly operational data as this is priority by our client Leo and is the natural workflow that after adding a new venue/client, the floor marking job will be created next. It also includes providing access to data maintainer to view and modify all information.
- In Semester 2 - Sprint 4, the must-have user stories are a continuation of operational data, mainly implementing additional detailed pages such as general job information and performance index for each individual at the job. It also includes features like searching, filtering/sorting data for ease of user view.
- In Semester 2 - Sprint 5, the should-have user stories relate to the dashboard, mainly showing statistics and data analysis of the floor marking jobs, generating job summary of a floor marking job and being able to export details to an excel sheet. These were the next in priority for our client of the database platform after the successful completion of the fixed and operational data.
- In Semester 2 - Sprint 6, the should-have user stories are a continuation of data analysis graphs/tables on the dashboard. It also includes adding a new user by the admin.
- For the backlog there are could-have features like viewing version history for job information and being able to import details for a new job. Other low priority user stories include being able to edit and delete a user by an admin for user management.

Research

This section consists of the research done for the technology and requirements for our Job Database Software Platform for August Robotics. This is a rough space for posting links and information on research.

Table of Contents

Technology Front-end and Back-end

Front-End Framework

	Framework	Introduction	Advantage	Disadvantage
1	React	React: Is a front-end JavaScript library for building user interfaces. Its modular architecture, declarative syntax, and reusability make it a popular choice among developers.	• High performance due to its virtual DOM implementation, which reduces the number of DOM updates and improves the rendering speed of the application. • Flexibility, as React can be used for both web and mobile app development using React Native. • Large community and ecosystem with many third-party libraries and tools available. • Supports server-side rendering for improved SEO and initial load times.	• Steep learning curve for beginners, especially those without prior knowledge of JavaScript and its advanced concepts. • Lacks built-in features such as routing, form handling, and state management, which may require additional third-party libraries to be used. • JSX syntax may require some getting used to, and some developers may prefer to use traditional HTML templates.
2	Next.js	Next.js is a popular framework for building server-side rendered (SSR) web applications using React.	• Server-side rendering: Next.js provides built-in support for server-side rendering (SSR) of React components, which can improve performance and user experience by delivering pre-rendered HTML to the client. • File-based routing: Next.js allows developers to create pages as individual files, making it easier to organize and manage the application's routing and pages. • Easy to use: Next.js provides a simple and intuitive API, making it easy to get started and build complex applications quickly.	• Dependencies: Next.js requires several dependencies to be installed, which can increase the size and complexity of the application. • Learning curve: nextJS takes time to learn if you are not familiar with server-side rendering or React.
3	Angular	Angular is a popular open-source front-end web application framework developed and maintained by Google. It is used for building dynamic single-page web applications (SPAs) and mobile applications.	• Full-featured framework with built-in support for routing, forms, state management, and other commonly used features. • Strongly-typed language (TypeScript) and strict syntax helps catch errors at compile time, leading to more robust code. • Large community and ecosystem with many third-party libraries and tools available. • Good performance and scalability for large, complex applications.	• Steep learning curve, especially for developers new to TypeScript and Angular's advanced concepts. • May be too heavy-handed for small projects, leading to increased development time and complexity. • More verbose syntax compared to React and Vue.
4	Vue	Vue is a lightweight and flexible front-end JavaScript framework that provides a wide range of tools and features for building scalable and performant web applications.	• Lightweight and easy to learn, making it a good choice for beginners or small projects. • Flexible and can be used for both small and large applications. • Good performance and scalability, with a virtual DOM implementation similar to React's. • Offers a gentle learning curve with clear documentation and an intuitive API.	• Smaller community and ecosystem compared to React and Angular, with fewer third-party libraries and tools available. • May not be suitable for large, complex applications due to its simpler design. • Limited official support from large tech companies compared to React and Angular.

Back-end Framework

	Framework	Introduction	Advantage	Disadvantage
1	Express.js - JavaScript	Express is a minimalist web framework that's designed to make building web applications in Node.js easier and faster. It follows a middleware-based approach, allowing developers to build applications by chaining together middleware functions. Express provides features such as routing, templating, and database integration, making it a versatile framework for building a wide range of web applications.	• Simplicity: Express is a simple and lightweight framework, making it easy to learn and use. Its minimalist approach allows developers to build web applications quickly and easily. • Flexibility: Express is a flexible framework, allowing developers to use their preferred libraries and tools. It also provides a simple and easy-to-use API for building web applications. • Scalability: Express is a scalable framework, allowing developers to build high-performance web applications with ease. It provides features such as asynchronous I/O and clustering, making it a great choice for building high-traffic web applications.	• Limited functionality: Express is a minimalist framework, so it may not have all the features required for complex web applications. • Security: Express doesn't provide built-in security features, so developers need to be careful when building secure web applications.

2	Django - Python	Django is a high-level web framework that's designed to make web development easier and faster. It follows the "batteries included" approach, providing a large set of tools and libraries for building complex web applications. Django provides features such as URL routing, database integration, authentication and authorization, and administration panels, making it a versatile framework for building a wide range of web applications.	• Robustness: Django is a robust and mature framework, with a large and active community. It's well-documented and has strong testing support, making it easy to write robust and maintainable code. • Scalability: Django is a scalable framework, allowing developers to build large-scale web applications with ease. It provides features such as caching, load balancing, and asynchronous tasks, making it a great choice for building high-traffic web applications. • Security: Django provides built-in security features, such as password hashing, cross-site scripting (XSS) protection, and clickjacking protection, making it easy to build secure web applications.	• Complexity: Django can be complex, especially for beginners, due to its many configuration options and features. • Learning curve: requiring a solid understanding of Python and web development concepts. • Flexibility: Django is a flexible framework, but it can be difficult to integrate with other libraries and frameworks, especially if they don't follow Django's conventions.
3	Flask - Python	Flask is a lightweight web framework that's easy to learn and use. It follows a minimalist approach, allowing developers to build web applications quickly and easily. Flask provides features such as routing, templating, and database integration, making it a versatile framework for building a wide range of web applications.	• Simplicity: Flask is a lightweight framework that's easy to learn and use. It has a small and simple core, with extensions available for additional features. • Flexibility: Flask is a flexible framework, allowing developers to use their preferred libraries and tools. It also provides a simple and easy-to-use API for building web applications. • Modularity: Flask is a modular framework, allowing developers to build their applications in a modular and scalable way.	• Limited functionality: Flask is a lightweight framework, so it may not have all the features required for complex web applications. • Scalability: Flask may not be the best choice for large-scale applications, as it may require more configuration and setup compared to other backend frameworks. • Security: Flask doesn't provide built-in security features, so developers need to be careful when building secure web applications.
4	Sprint Boot - Java	Spring Boot provides a streamlined and opinionated approach to building Java-based web applications. It comes with a set of preconfigured libraries and tools, making it easy to get started quickly. Spring Boot also provides features for building microservices, such as support for building and consuming REST APIs, messaging, and event-driven architectures.	• Rapid development: Spring Boot provides a set of preconfigured libraries and tools, making it easy to get started quickly and develop applications faster. • Robustness: Spring Boot is built on top of the Spring Framework, which has a large and active community. It's also well-documented and has strong testing support, making it easy to write robust and maintainable code. • Flexibility: Spring Boot provides a flexible and modular architecture, allowing developers to easily integrate with other libraries and frameworks.	• Complexity: Spring Boot can be complex, especially for beginners, due to its many configuration options and features. • Performance: Spring Boot may have lower performance compared to other backend frameworks due to its use of reflection and dynamic code generation. • Learning curve: requiring a solid understanding of Java and the Spring Framework.
	Sprint MVC - Java	Spring MVC is a Web MVC framework for developing or building web applications. It contains many configuration files for various capabilities. Spring MVC is an HTTP-oriented web application development framework.	• Apps using Spring MVC are highly scalable • There's a large community with documentation • It covers an extensive ecosystem • It allows you to decouple the framework for easier execution	• The architecture is simple, but it's not beginner friendly • It is bulky with legacy DI API and spaghetti code

Database:

	Database	Introduction	Advantage	Disadvantage

1	**Firebase**	Firebase is a cloud-based, real-time NoSQL database platform developed by Google. It is designed to store and sync data in real-time across multiple clients, including web and mobile applications.	• Serverless architecture, meaning that users do not need to manage infrastructure • Real-time data synchronization across multiple devices and platforms • Easy to set up and integrate with other Google services • Built-in security and scalability features • Flexible data modeling using JSON-like documents	• Limited support for complex data models and relationships • Limited query capabilities compared to other database types • Data storage costs can become expensive for larger applications or datasets
2	**MongoDB**	MongoDB is a document-oriented NoSQL database that uses a JSON-like data model to store data in collections rather than tables. It is designed to be flexible, scalable, and developer-friendly.	• Flexible data modeling allows for dynamic schemas and easy updates • High scalability and availability, with built-in sharding and replica set features • Supports ad hoc queries and indexing for faster query performance • Offers robust support for geospatial data and text search • Open source and has a large, active community	• Data consistency may not be guaranteed in all cases, as MongoDB uses eventual consistency • Limited support for transactions across multiple documents or collections • Indexing can be complex and time-consuming for large datasets • May require more technical expertise to use and maintain compared to some other databases
3	**MySQL**	MySQL is an open-source relational database management system (RDBMS) that uses a traditional table-based data model to store and organize data. It is designed to be fast, reliable, and easy to use.	• Fast and efficient, with strong support for indexing and query optimization • Easy to set up and use, with a simple, intuitive interface • Supports a wide range of programming languages and APIs • Highly scalable, with built-in support for replication and partitioning • Large community of users and developers, with a wide range of resources and support available	• Limited support for advanced SQL features compared to some other databases • May not be as reliable or durable as some other databases, particularly in high-traffic or high-concurrency environments • Limited support for geospatial data and other advanced data types • May require more resources to run effectively compared to some other databases

GitHub Research Documents

https://www.gitkraken.com/learn/git/best-practices/git-branch-strategy

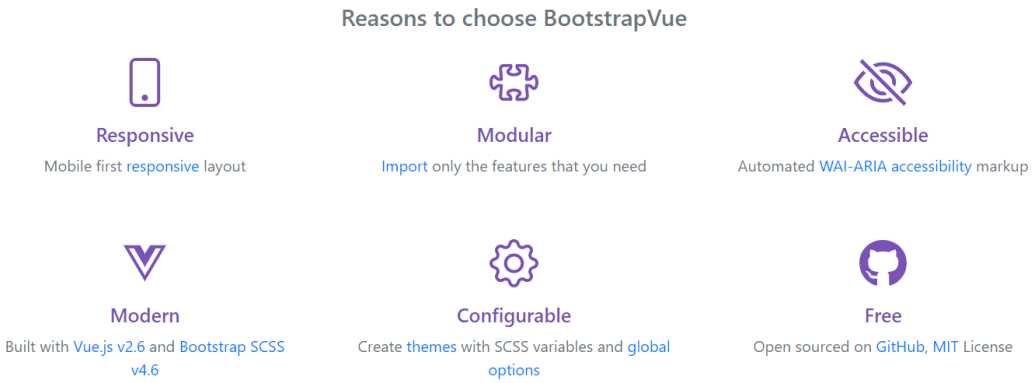
https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

https://gist.github.com/digitaljhelms/4287848

https://blog.mergify.com/the-best-git-branching-strategies/

Front-end UI frameworks research

Bootstrap:



https://vuejsexamples.com/provide-plugin-for-grids-table-with-editable-cells/

https://bootstrap-vue.org/docs/components

Vuetify:

Why Vuetify?

Vuetify is a powerful Vue Component Framework built from the ground up to be easy to learn and rewarding to master. Our collection of UI components maintain a consistent style throughout your application with enough customization options to meet any use-case.

#It's free

Vuetify is an Open Source project available for free under the MIT licensed. Additionally, Vuetify's source code is available on GitHub, allowing developers to modify and contribute to its development if they choose to do so.

#Flexible components

Every component in Vuetify is handcrafted under the guise of Google's Material Design specification and comes with hundreds of customization options that fit any style or design; even if it's not Material. Write Vue templates that are as concise or verbose as you want using exclusively or a combination of **pr ops**, **slots**, and **components**.

#Tooling

Vuetify has a large ecosystem of supporting tools that enrich the development experience that range from project creation to design UI kits.

- Figma UI Component Kit
- First-party Vite support
- Pre-configured Vue 3 applications for TypeScript and JavaScript
- Intellisense & autocomplete support for VSCode and JetBrains products
- Wireframe examples

#Community

When you develop with Vuetify, you are never alone. Stuck on a problem? Take advantage of our massive Discord community and collaborate with other Vuetify developers in one of our public help channels. Need a more personalized support solution? Vuetify offers Enterprise support with options tailored to individuals and businesses. Check out some other ways to Sponsor Vuetify development.

Reference: https://vuetifyjs.com/en/components/all/

Tailwind:

https://tailwindui.com/documentation

Decisions

This section consists of the important decisions that has been taken by the team and incorporated into our product/process journey.

Decision Making Strategy used is here: DACI Decision-Making Framework

Table of Contents

DACI Decision-Making Framework

The decision-making process is a crucial component of any project's lifecycle. It's generally an ongoing activity that requires the cooperation of every project stakeholder, from organizational leaders to individual project stakeholders, and it's a process that should never be taken lightly (Johnivan, 2022). We plan to use the DACI framework in our decision making process to ensure streamlined process and smooth transition to our decisions.

https://project-management.com/project-management-phases/

What is the DACI Model?

An acronym that stands for Driver, Approver, Contributor, and Informed, DACI provides a structured approach to project management. It achieves this through the naming clear roles that cover the senior levels of a project management team. The DACI model is often utilized in areas like product development, software development, or any project that requires a streamlined and versatile approach.

Each role has clearly defined expectations for the person or people occupying that position in the model. By understanding what's expected, communication and decision-making are streamlined as duplicated effort is largely eliminated. (Johnivan, 2022)

Refer https://www.atlassian.com/team-playbook/plays/daci for more explanation on the following:

https://www.thecloudtutorial.com/daci-model/

DACI Decision Making Template

DACI: Decision documentation

Add your comments directly to the page. Include links to any relevant research, data, or feedback.

Summarize this decision in the table below. Type **/date** to quickly add the due date and @mention the driver, approver, contributors, and informed to keep everyone on the same page.

Status	UNDECIDED
Impact	HIGH / MEDIUM / LOW
Driver	
Approver	
Contributors	
Informed	
Due date	31 Oct 2019
Outcome	

Background

Provide context on the decision the team needs to make. Include links to relevant research, pages, and related decisions, as well as information on constraints or challenges that may impact the outcome.

Relevant data

Add any data or feedback the team should consider when making this decision

Options considered

	Option 1:	Option 2:	Option 3:
Description			
Pros and cons			
Estimated cost	HIGH	MEDIUM	LOW

Action Items

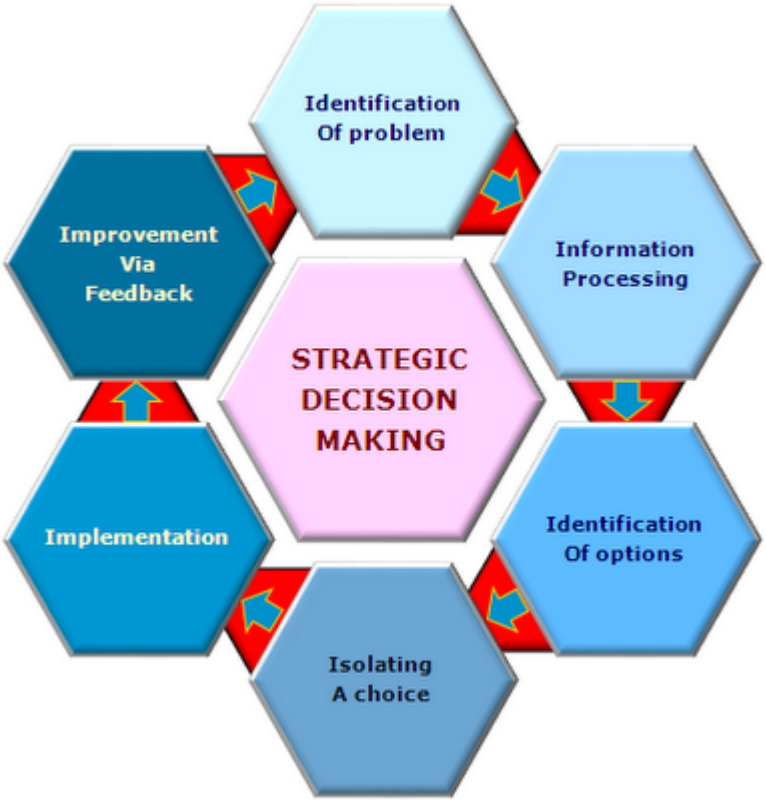
Add action items to close the loop on open questions or concerns

Outcome

Summarize the outcome below (type **/decision** to add another one)

https://www.atlassian.com/software/confluence/templates/decision

Decision Making Strategy: The 6 I's



6 I's are:

1. **Identification of problem:** During this stage the problem for which the strategic decision has to be made is identified. he output of this stage would be the problem statement.

2. **Information processing:** This is the stage where data gathering is done and information is processed. Referring to my model of strategy generation, this is the Strategic Assessment stage/phase. We analyze all external and internal factors, conduct appreciative enquiry and arrive at various objectives.

3. **Identification of options:** The identified objectives will act as an input for identifying various options. From IT strategy perspective this would be the second phase of my strategy generation model, SITP Planning Process (rather even the 4th and 5th Is are related to it).
Otherwise for identifying any strategic option, the objectives will be analyzed to identify the various ways or options by which it can be accomplished. The focus should be on identifying as many options that may be possible.

4. **Isolating a choice:** After identifying various available options, the best one needs to be identified. There are various qualitative and quantitative techniques that may be used to isolate the choice. These methods would be discussed in my next post.This would also give measurable targets for the strategy or objectives.

5. **Implementation:** After the choice has been identified/isolated, the implementation plan has to be formulated. Mintzberg's Plan and Pattern will act as an catalyst for formulating the Implementation plan. Thereafter, steps for implementation of the plan is performed, which would include allocation of required resources. Thus, resources and capabilities of the organization will enable in eventual implementation.

6. **Improvement via feedback:** This is the feedback mechanism. Whether the implementation is inline with identified measurable targets or not is determined with regular feedback, gaps and corrective actions identified and implemented. Eventually when the required target is achieved, it would mean that the strategic decision has been able to successfully resolve the IT/business strategic problem.

(Sumit, 2023)

References

Johnivan J., 2022, DACI: Top Decision-Making Framework, Project-Management.com, https://project-management.com/daci-model/

Sumit K., 2023, 6 ‘I’s Of Strategic Decision Making, Process Consultant, http://process-consultant.blogspot.com/2011/12/6-is-of-strategic-decision-making.html

Team Role Allocations

Status	DECIDED
Stakeholders	Team
Outcome	Home
Date	27 Jul 2023

Note: *These are the current roles for our team, which have been randomly allocated. The team will help each other in the tasks for the lead roles if any issues are faced and they will be in charge of managing the completion and progress of that product process while doing regular team tasks.*

Roles	Scrum Master	Product Owner	Quality assurance lead	Architecture Lead	Front-end Lead	Back-end Lead	UI/UX Lead	Risk Management Lead	Testing lead	Deployment Lead
Andrew Liu										
Michael										
Kritnand Suwanna-Arj										
Lipeng Zhang										
Wei Zhao										
Himaja Ramesh Kakade										
Poorvith Gowda Gavenahalli Divaka										
Haiyao Yan										
Wenxuan XIE										
Jiyang XIN										

Yellow - Semester 1 roles

Red - Semester 2 updated roles (if not changed then same role as Semester 1)

Green - Backup roles

Chosen Technology

Status	DECIDED
Impact	HIGH
Driver	Team
Approver	Team
Contributors	Team **2023-Mar-13 17:15 - 18:15 Team-meeting**
Informed	Team
Due Date	03 Apr 2023
Outcome	Use Vue for front-end, Django for Backend and PostgreSQL for Database

Background

Need to decide which Technologies to use for development.

Relevant Data

For Technology research see page: Technology Front-end and Back-end

*Highlighted in bold is the chosen technology

Options Considered

	Front end	**Back-end**	**Database**
Description	React/**Vue**	Java Sprint Boot/**Django**	Firebase/**PostgreSQL**
Pros and cons	⊕ React and Vue have been used by team members in prior projects and have experience in both. Vue has been used by the client but it is still to decide which would be most ideal to develop functionality and its use thus we decided to go with it.	⊕ After discussions among the team members, the backend leader thought of using Java Spring Boot as the framework for our project's backend. An important reason is that none of our team members have used this framework before, but they all have experience in Java programming. ⊕ However, Django ended up being the ideal choice as it has been used by our client for their robot application and would be easier to for them to extend to.	⊕ Through the requirements analysis, we found that the data types we need to store and use are not very complex. Using a JSON-like data modeling approach is enough to meet our needs. Therefore, we chose the NoSQL database Firebase, which is very easy to set up and convenient to use, it also store the data in cloud so we can backup the data easily. ⊕ However, we decided PostgreSQL would be an ideal choice as well as it has been used by our client for their robot application and would be easier to for them to extend to and team members also have experience with using it in previous projects.
Estimated Cost	N/A	N/A	N/A

Action Items

- [x] Setup front-end and back-end in repository

Outcome

- Vue is chosen for front-end
- PostgreSQL chosen for database
- Django chosen for back-end

Front-End UI Framework Decision

Status	DECIDED
Impact	MEDIUM
Driver	Himaja Ramesh Kakade Haiyao Yan Poorvith Gowda Gavenahalli Divakar
Approver	Haiyao Yan
Contributors	Himaja Ramesh Kakade Haiyao Yan Poorvith Gowda Gavenahalli Divakar
Informed	Team
Due Date	15 May 2023
Outcome	Use Vuetify as the UI library.

Background

Need to decide which UI framework to use for front-end development of Database Platform.

Relevant Data

References:

https://stackshare.io/stackups/tailwind-css-vs-vuetify

https://stackshare.io/stackups/bootstrap-vs-vuetify#:~:text=Bootstrap%20and%20Vuetify%20can%20be,key%20factor%20in%20picking%20Vuetify.

Research Links : Front-end UI frameworks research

Options Considered

	Option 1: Vuetify	**Option 2: TailwindCSS**	**Option 3: Bootstrap**
Description	*Material Component Framework for VueJS 2*. Vuetify is a component framework for Vue.js 2. It aims to provide clean, semantic and reusable components that make building your application a breeze. Vuetify utilizes Google's Material Design design pattern, taking cues from other popular frameworks such as Materialize.css, Material Design Lite, Semantic UI and Bootstrap 4.	*A utility-first CSS framework for rapid UI development*. Tailwind is different from frameworks like Bootstrap, Foundation, or Bulma in that it's not a UI kit. It doesn't have a default theme, and there are no build-in UI components. It comes with a menu of predesigned widgets to build your site with, but doesn't impose design decisions that are difficult to undo.	*Simple and flexible HTML, CSS, and JS for popular UI components and interactions*. Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web;
Pros and cons	⊕ wide range of components and active development ⊕ More team members have used material UI and thus have some familiarity for Vuetify which used the design and icons from material UI. ⊕ Extensive documentation that would help us use components easily leading to faster development	⊕ **"Highly customizable"** is the primary reason why developers consider Tailwind CSS over the competitors, whereas **"Wide range of components and active development"** was stated as the key factor in picking Vuetify. It is also known for its sleek design ⊖ Not component based more of a CSS framework as we need more reusability in our platform this is not ideal.	⊕ High Responsiveness, Grid based, has all major components that we would need for tables, buttons, and alignment of different grid containers ⊕ Extensive documentation and community based as it has been in use for a long time
Estimated Cost	N/A	N/A	N/A

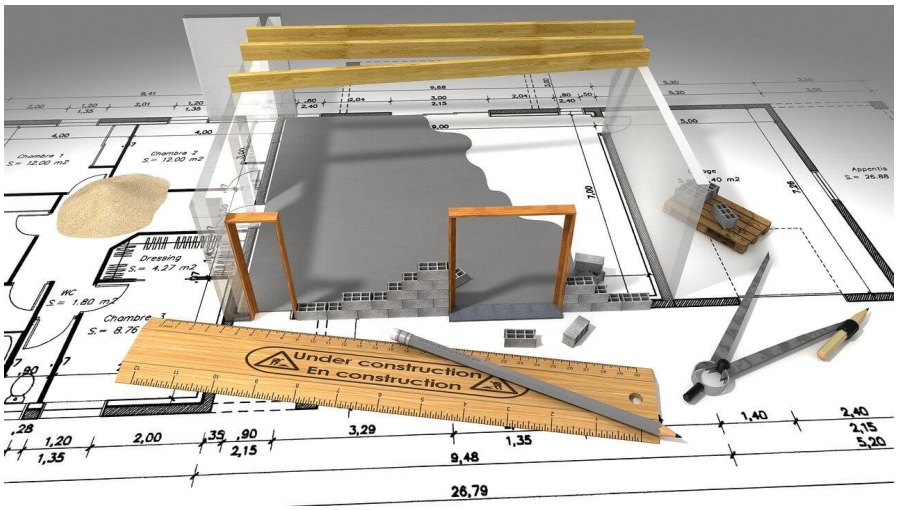
Action Items

- [x] Install Vuetify for use

Outcome

- Vuetify is the decided framework as it is more component based than TailwindCSS and better suited for our project, most team members also have experience with MaterialUI and Vuetify is an extension to it, thus is a better fit for overall development.

Design & Architecture



Outcomes for this section

- **Software design:** Create a plan or blueprint for the construction for the Job database system. It involves defining the system's components, their interactions, and the overall structure of the system. The software design process typically involves breaking down the system into smaller, more manageable components and designing each component in detail. This process helps ensure that the software is designed to meet its functional and non-functional requirements, and is scalable, maintainable, and testable.
- **Software architecture:** Define the overall structure of the Job database system, including its components, their relationships, and the principles and guidelines governing their design and evolution over time. It involves identifying the system's key stakeholders, their needs and goals, and designing the system in a way that meets those needs and goals while balancing competing constraints such as performance, scalability, maintainability, and security. Software architecture provides a high-level view of the system that helps guide the design and development process and enables the system to evolve over time in response to changing requirements.

Diagrams

This section consists of all our systems diagrams to understand the logic and architecture of how the system will be built and designed.

All Diagrams

No.	Title
1	Domain Diagram
2	Architecture Diagram
3	Use Case Diagram
4	ER Diagram
5	Sequence Diagram
6	Component Diagram
7	Communication Diagram

Diagrams Google Drive: https://drive.google.com/file/d/1RwIBe0gwUcriHGsH-9jWLT4oDbNiRjDs/view?usp=sharing

Use Case Diagram

Description:

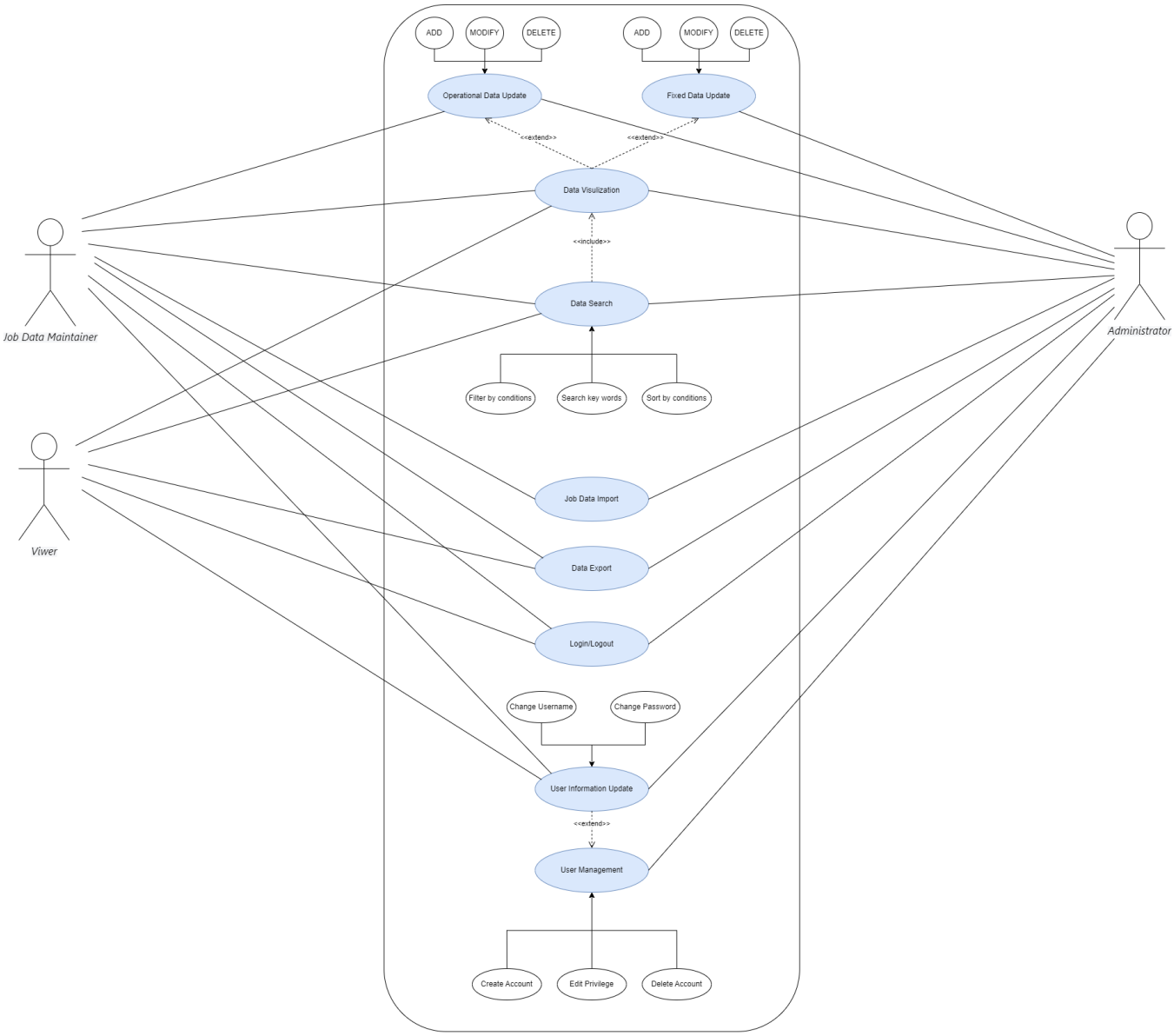
A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well.

Relations Description:

- **The white circle:** The specific function belongs to the use case in the blue circle.
- **Extend relationship:** The use case is optional and comes after the base use case.
- **Include relationship:** The use case is mandatory and part of the base use case.

V2.2

V2.0


ADD
MODIFY
DELETE
ADD
MODIFY
DELETE
Operational Data Update
Fixed Data Update
<<extend>>
<<extend>>
Data Visulization
<<include>>
Data Search
Filter by conditions
Search key words
Sort by conditions
Job Data Maintainer
Administrator
Viwer
Job Data Import
Data Export
Login/Logout
Change Username
Change Password
User Information Update
<<extend>>
User Management
Create Account
Edit Privilege
Delete Account

V1.0

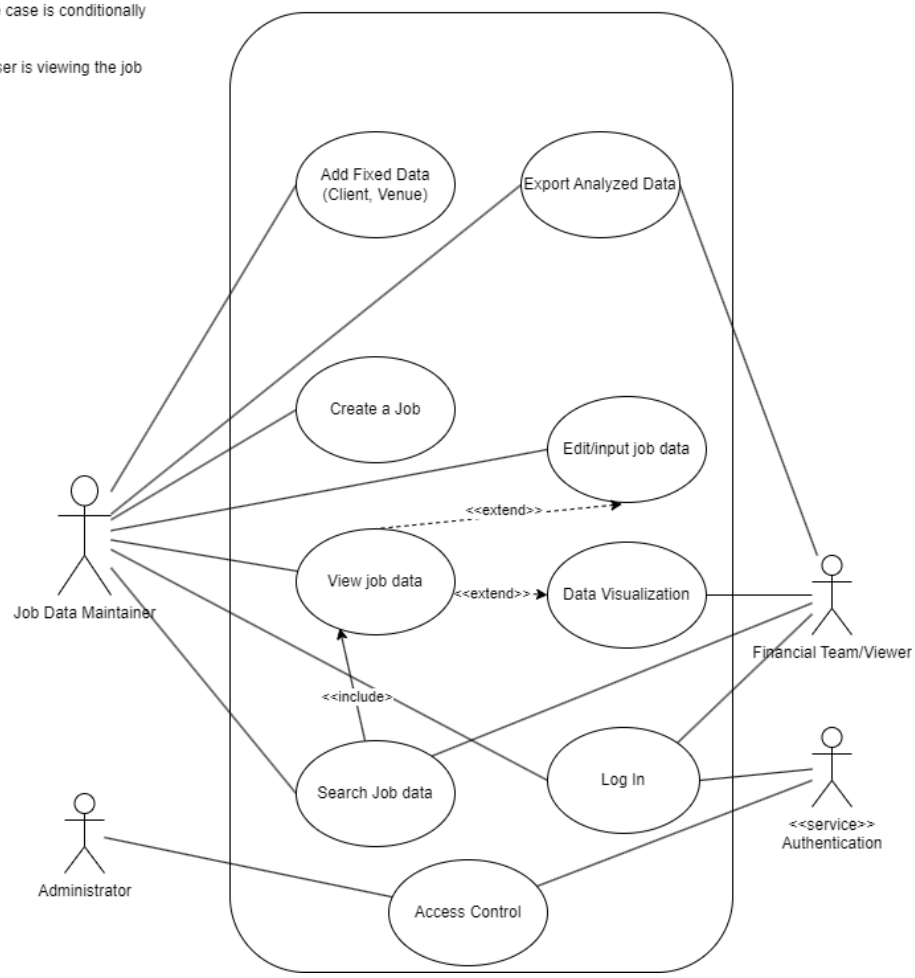
Use Case Diagram

notes:

"search job data" ---> (include) "view job data"means "search job data" use case may include a "view job data" use case.
In this situation, every time a user search the job data, the system shows user the job data view of what they search

The "extend" relationship is used when the behavior of a use case is conditionally extended by another use case.

In the situation, the system shows the edit page only if the user is viewing the job data and want to do some modify on the data.



link to the diagram https://drive.google.com/file/d/1RwIBe0gwUcriHGsH-9jWLT4oDbNiRjDs/view?usp=sharing

Notes that the links contains all the architecture artifacts, please check the sub page and find the corresponding diagram.

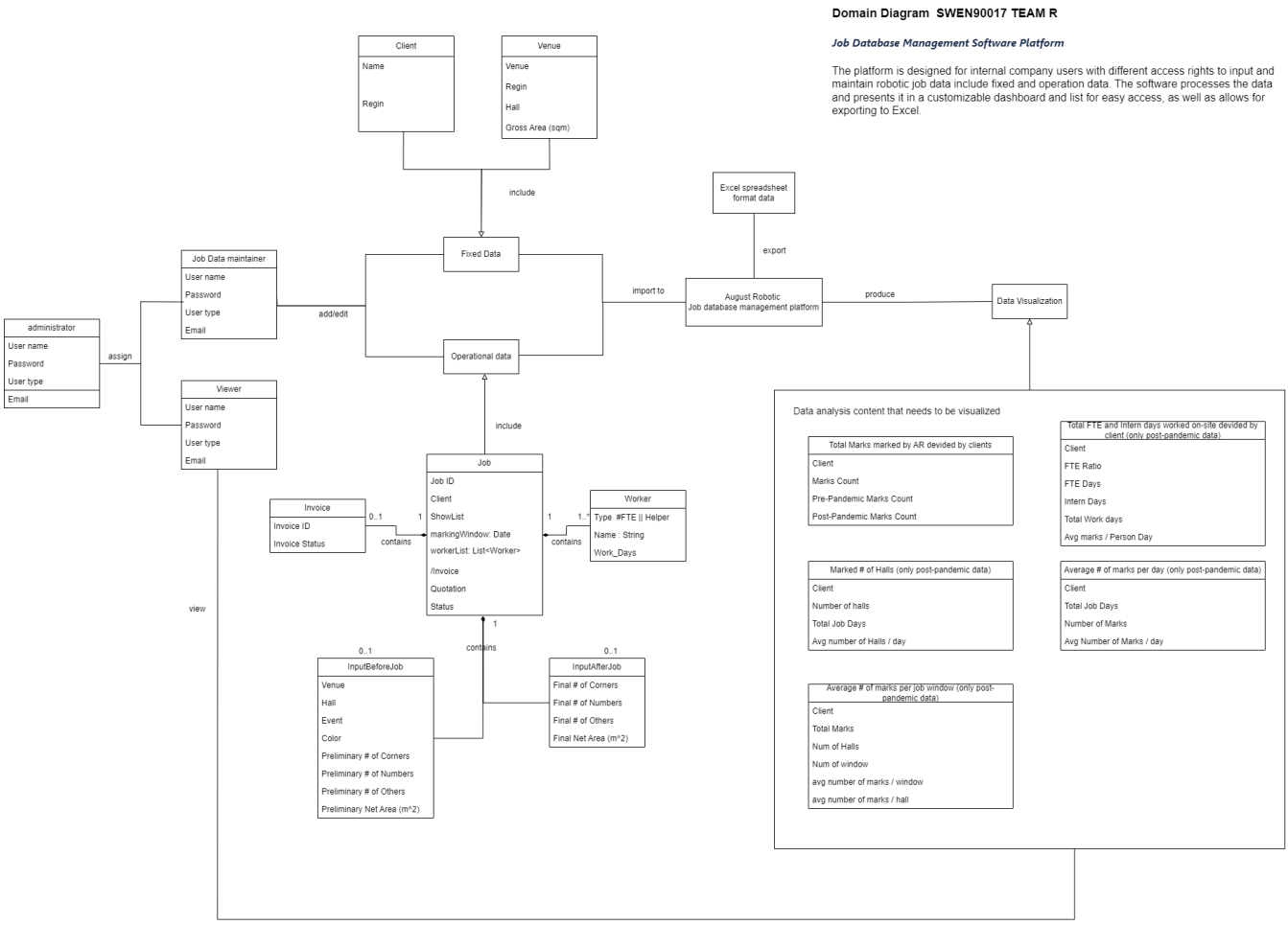
Domain Diagram

Description:

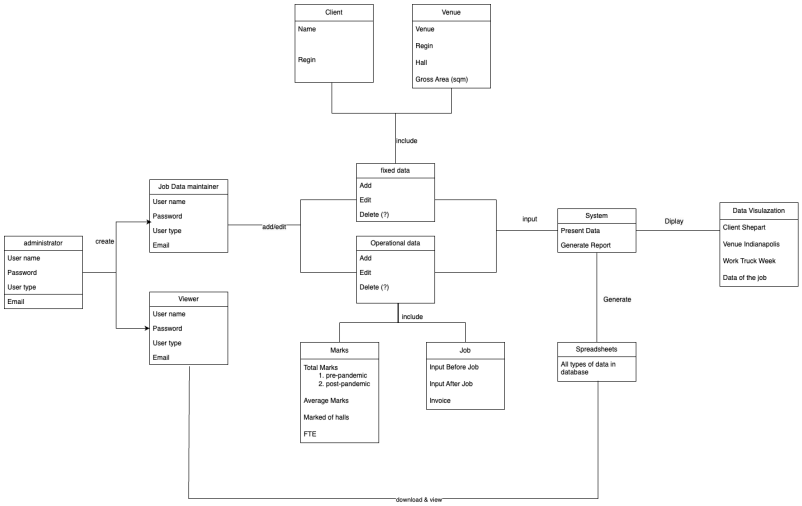
The domain diagram is a representation of meaningful real-world concepts pertinent to the domain that need to be modeled in software. The concepts include the data involved in the business and rules the business uses in relation to that data. In domain diagram, there are four different types of data that can be accessed by users to modify(each type of user has different operation permissions).The data will be displayed in the fornt-end by the system and consolidated into spreadsheets documents for users to download.

Link to the domain diagram: https://drive.google.com/file/d/1RwlBe0gwUcriHGsH-9jWLT4oDbNiRjDs/view?usp=sharing

Domain Diagram V1.1



Domain Diagram V1.0


Client
Name
Regin
Venue
Venue
Regin
Hall
Gross Area (sqm)
include
fixed data
Add
Edit
Delete (?)
Job Data maintainer
User name
Password
User type
Email
add/edit
administrator
User name
Password
User type
Email
create
Viewer
User name
Password
User type
Email
Operational data
Add
Edit
Delete (?)
include
Marks
Total Marks
1. pre-pandemic
2. post-pandemic
Average Marks
Marked of halls
FTE
Job
Input Before Job
Input After Job
Invoice
input
System
Present Data
Generate Report
Diplay
Data Visulazation
Client Shepart
Venue Indianapolis
Work Truck Week
Data of the job
Generate
Spreadsheets
All types of data in database
download & view

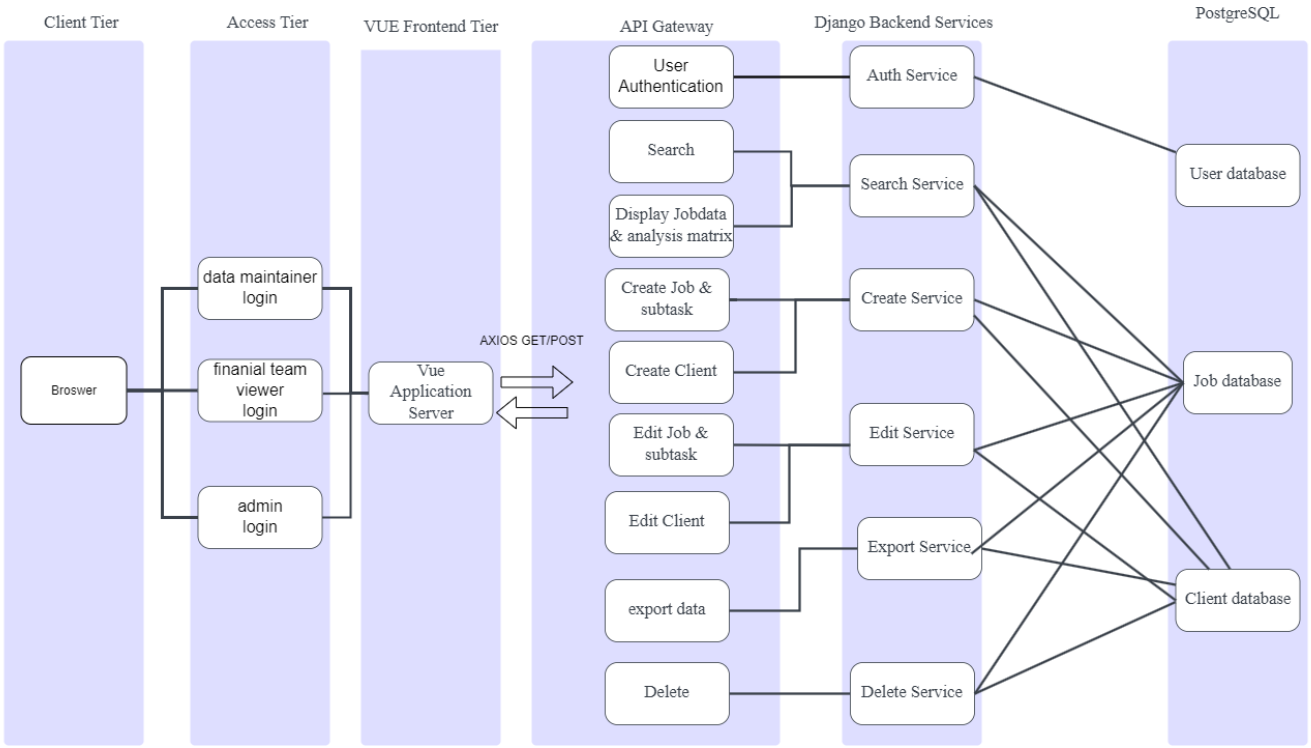
Architecture Diagram

Architecutre diagram identifies the key components of application and how they interact with each other.

The diagram is divide into three main sections: Client-side(frontend), Server-side(backend), and Database.

the draw.io link: https://drive.google.com/file/d/1RwIBe0gwUcriHGsH-9jWLT4oDbNiRjDs/view?usp=sharing

V2.0 VUE + Django + ProstgreSQL

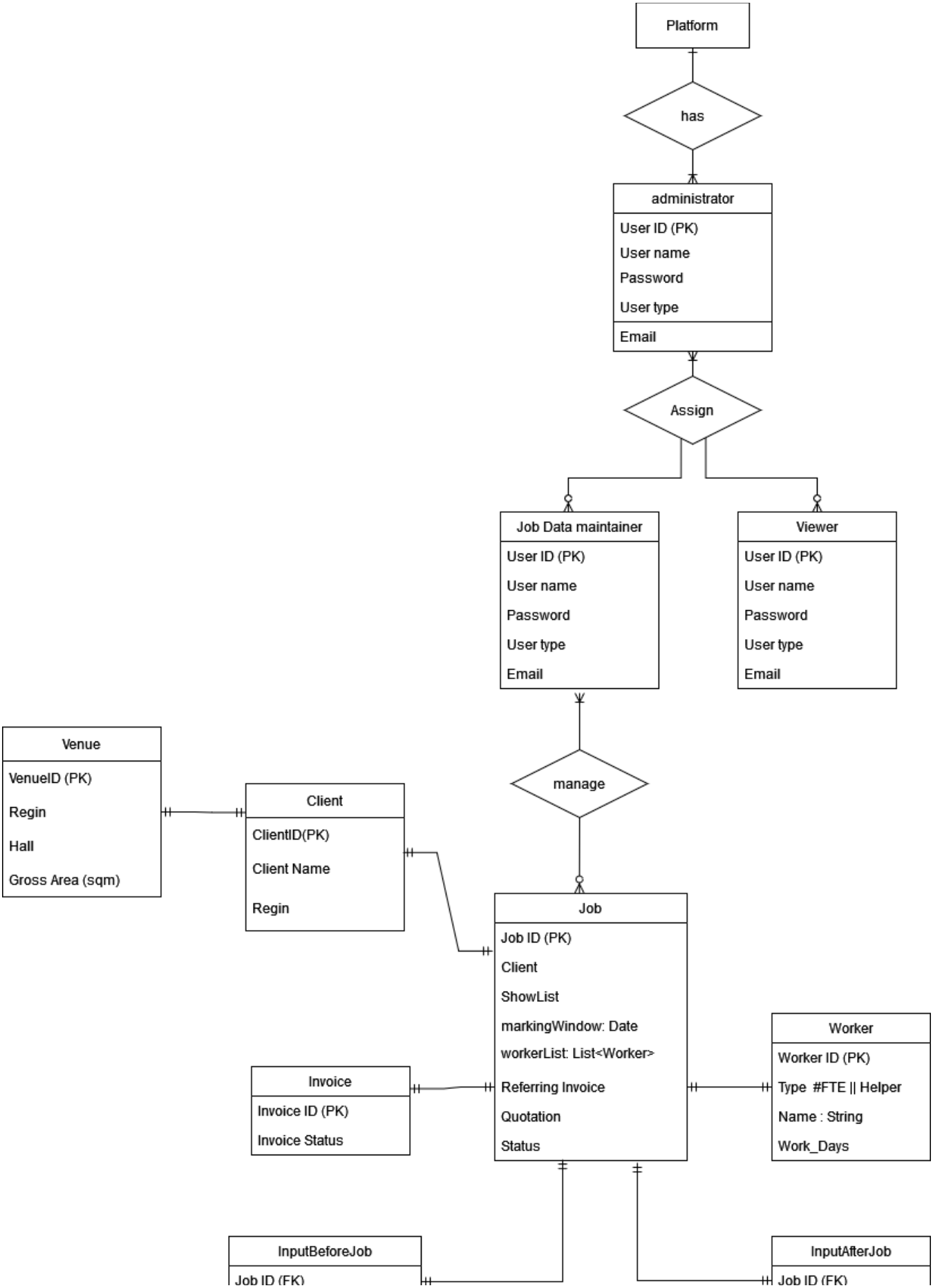


ER Diagram

An entity relationship diagram is responsible for visualising the relationship between different entities in a system.

It is useful for displaying the logical structure of a database and the relationship between different tables in it.

V1.0: https://drive.google.com/file/d/1RwIBe0gwUcriHGsH-9jWLT4oDbNiRjDs/view?usp=sharing


Platform
has
administrator
User ID (PK)
User name
Password
User type
Email
Assign
Job Data maintainer
User ID (PK)
User name
Password
User type
Email
Viewer
User ID (PK)
User name
Password
User type
Email
Venue
VenueID (PK)
Regin
Hall
Gross Area (sqm)
Client
ClientID(PK)
Client Name
Regin
manage
Job
Job ID (PK)
Client
ShowList
markingWindow: Date
workerList: List<Worker>
Referring Invoice
Quotation
Status
Worker
Worker ID (PK)
Type #FTE || Helper
Name : String
Work_Days
Invoice
Invoice ID (PK)
Invoice Status
InputBeforeJob
Job ID (FK)
InputAfterJob
Job ID (FK)

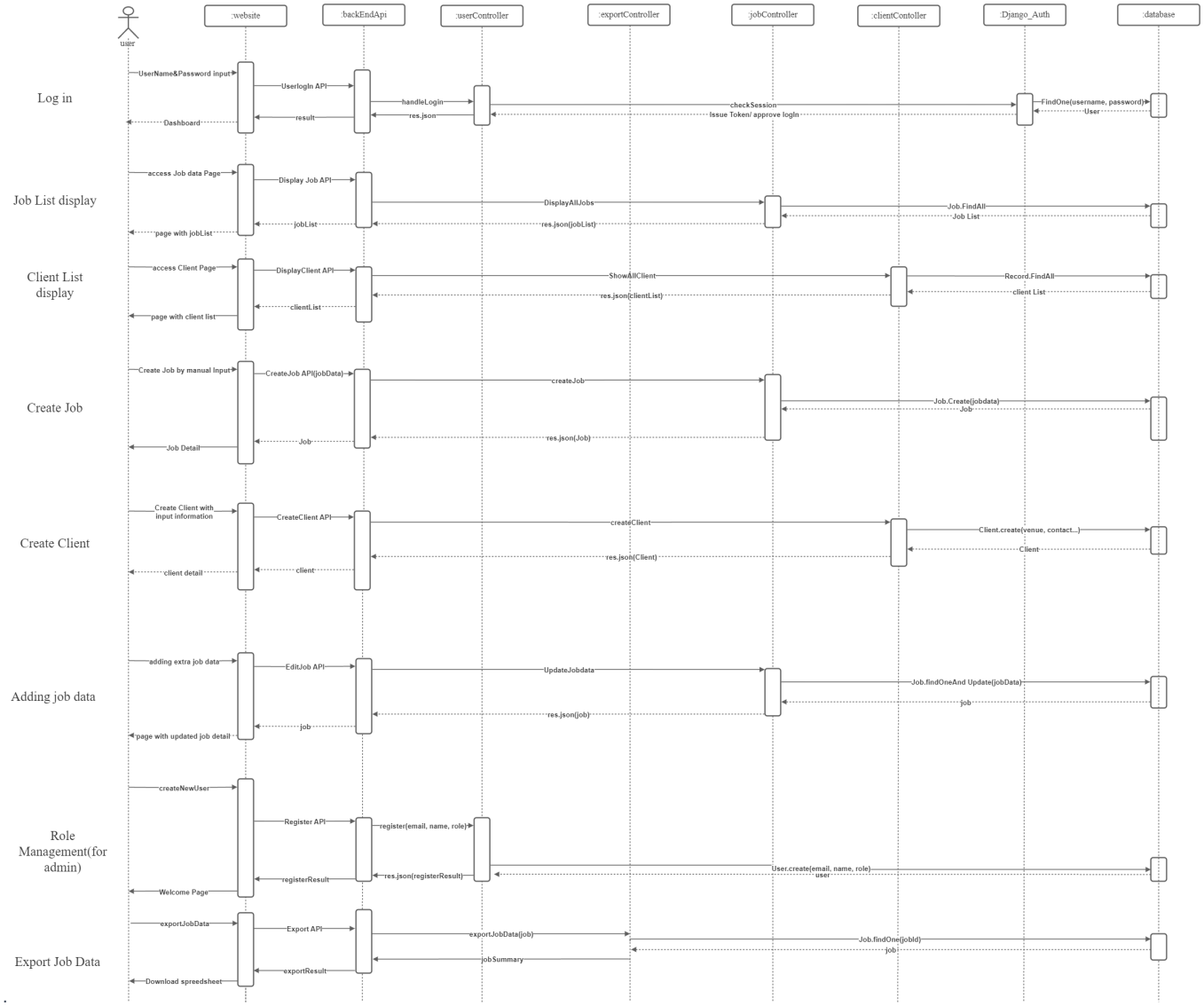
Venue
Hall
Event
Color
Preliminary # of Corners
Preliminary # of Numbers
Preliminary # of Others
Preliminary Net Area (m^2)

Final # of Corners
Final # of Numbers
Final # of Others
Final Net Area (m^2)

Sequence Diagram

A sequence diagram is a diagram that represents the interactions between user and our job management system in a system over time.
It shows the sequence of messages exchanged between the different system components involved in a particular use case or scenario

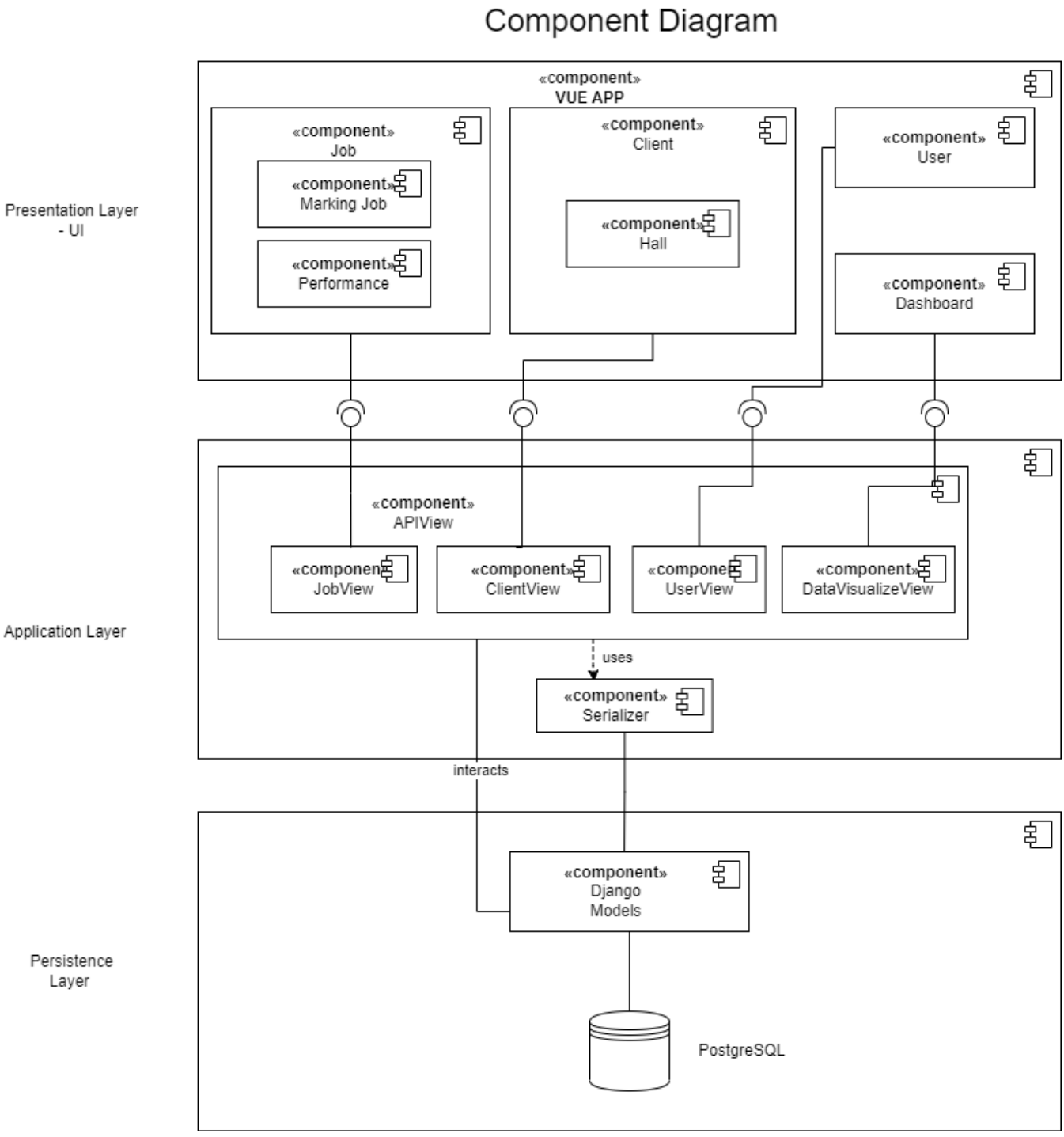
V1.0 : https://drive.google.com/file/d/1RwIBe0gwUcriHGsH-9jWLT4oDbNiRjDs/view?usp=sharing



.

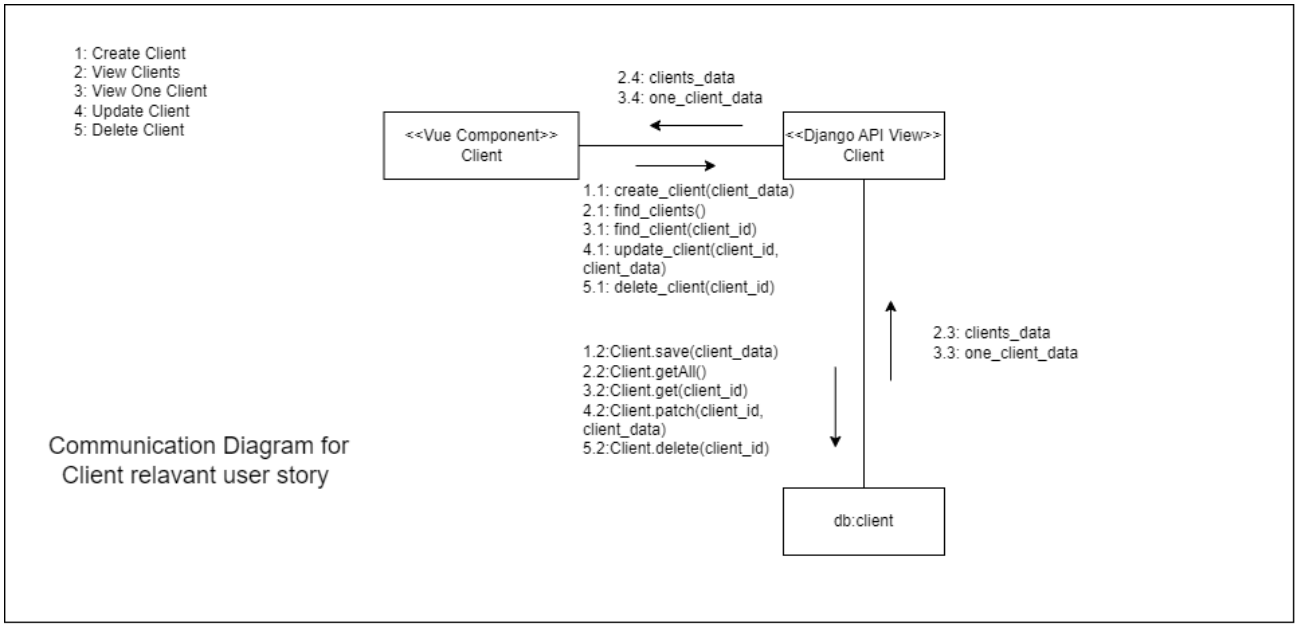
Component Diagram

https://drive.google.com/file/d/1RwIBe0gwUcriHGsH-9jWLT4oDbNiRjDs/view?usp=sharing

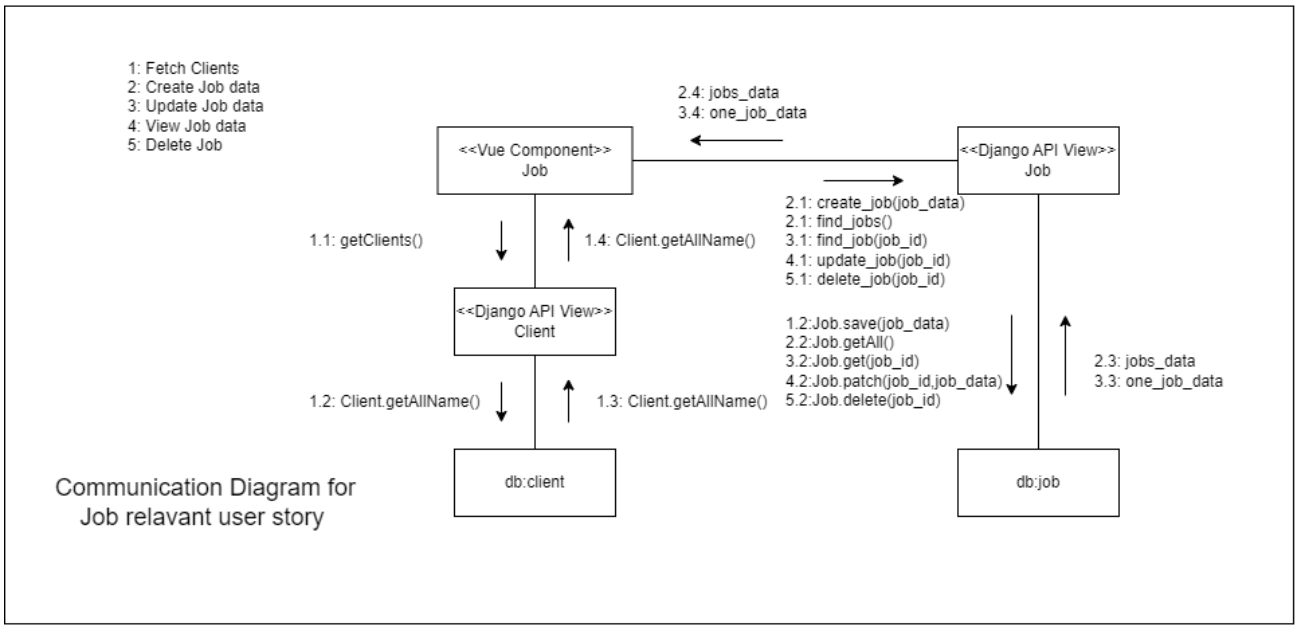


Communication Diagram

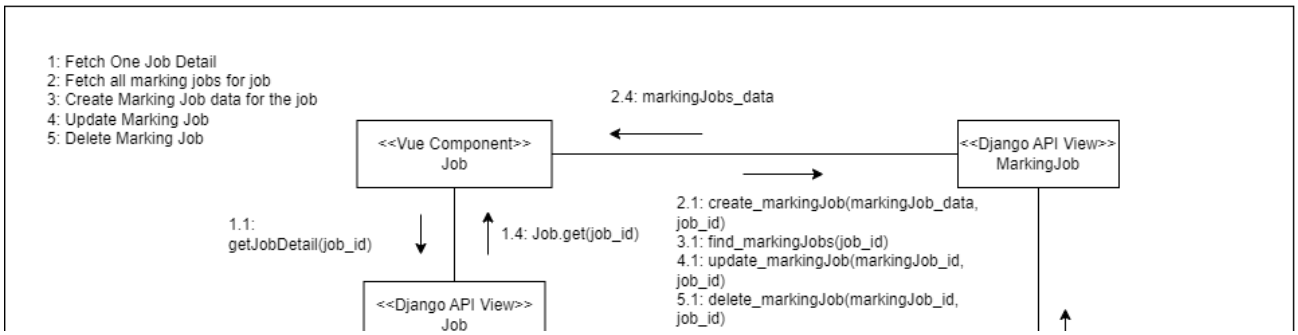
A communication diagram is used to visualize the interactions between objects or parts. Because we have a lot of user stories, we create a communication diagram for each user story or feature. This makes the charts simpler and more focused.

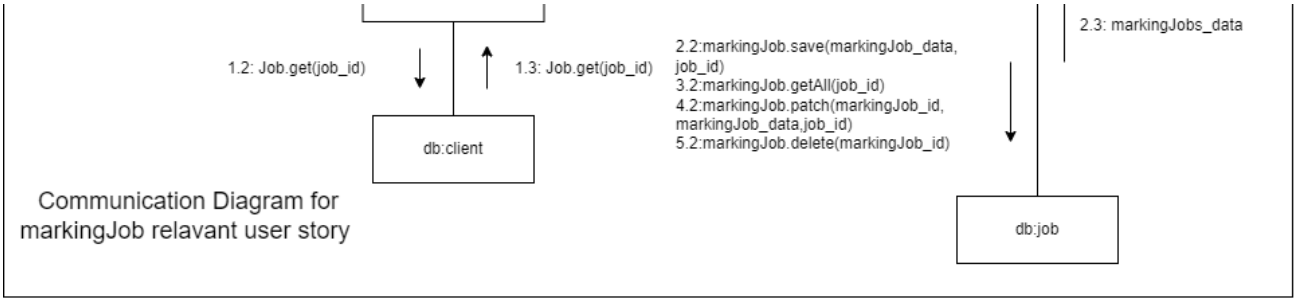


Communication Diagram for Client relavant user story



Communication Diagram for Job relavant user story





Communication Diagram for markingJob relavant user story

Quality

This section consists of the quality assurance techniques and plan undertaken by the team to assure that high standard and satisfying product is built.

Table of Contents

Page Title

Coding Standards

There are multiple technologies and languages to be used in this project, so several sets of coding standard have to be used. This document will list the technologies and languages used, and their respective coding standards.

Django/Python

- The backend will use the Django, so Python coding standards are needed along with standards for Django itself.
- Below is the recommended coding standards for provided by Django.
- Django Coding Standards

Vue.js/JavaScript

- The frontend will use Vue.js and therefore, JavaScript.
- Below is the recommended Style Guide provided by Vue.
- Vue.js Coding Standards

Postgres

- We will be using Postgres as our database. There are several conventions involved using Postgres including formatting, and proper error messages.
- Below is a link to a contents for coding conventions for Postgres 15.
- Postgres Coding Conventions

Risk Management

Purpose

This document is meant layout how the team plans on dealing with unexpected events that occur during development to reduce losses.

Definitions

Uncertainty

An event that cannot be guaranteed to happen.

Risk

A risk is an event that does not have a guarantee to happen. The effects of the risk can be positive or negative.

Types:

- Business
- Project
- Product

Risk Management Process

There are 5 steps to the risk management plane:

Identify

The method have been using to identify risks is brainstorming. We think of possible risks that the project may face.

Analyse

We analyse the risk by determining its likelihood of occurring and impact it would have on the project. We put these risks into risk register while categorising the types of risk it could be. We then write out possible mitigation strategies for the risks.

Evaluate and Prioritise

We evaluate these risks via a risk matrix. This so we understand the exposure these risks present and prioritise them.

Treat

We put our mitigation strategies into work.

Possible strategies include:

- Avoidance: stop activities that cause the risk.
- Reduction: reduce likelihood of risk
- Acceptance: acknowledge that if risk occurs, we will have to accept the consequences.

Monitor

We will regularly monitor, track, and review our risks to determine if they are occurring. We will also continue to review our plan to make sure it is adequate. The register will also be reviewed and go under the risk management process again, if there are additional risk that are identified.

Risk Management Plan

Risk Register

ID	Type	Description	Probability	Impact (out of 5)	Exposure	Justification
1	Project	A teammate leaves the project.	5%	2	0.1	There's a low probability of a teammate leaving the project this far into the semester. The impact would not be too significant because other members are still present. However, if this occurs multiple times then the impact would be higher.
2	Project	There is a new issue from the changes in the technologies used.	5%	4	0.2	There is a low probability that one of the technologies we are using has significant changes that would fundamentally impact the project. However, if this does occur, large changes would be required.
3	Product	The system breaks down.	20%	5	1.0	It is possible the system will break at some point that we are not aware of, but if it is designed well it is not likely to happen. The system breaking down is catastrophic and would require large changes.
4	Business /Product	The product does not meet all specifications by the client.	50%	3	1.5	It is likely that some of the specifications will not be met due to time constraints, but we assume the most vital ones will be.
5	Project	The technologies prove difficult to learn or use.	40%	3	1.2	It is possible that the technologies will be difficult to use, but is still the best choice for our product. This will also impact development for a time until the team understands how to use it.
6	Project	Hard to manage the team	30%	3	1.0	The team contains 10 members and sometimes some of the team members might not be able to attend all the meetings and which might hinder their understanding of the project requirements or client requirements.
7	Product	Breakdown or malfunction in Access control	20%	3	0.8	As there are 3 types of Access control, it is possible that sometimes the administrator or data maintainer might not be able to update the data when required or viewer might face difficulty viewing the data.
8	Project	Lack of language proficiency required for documentation and communication	60%	2	1.0	As some of the team member's first language is not English, it might affect the documentation process and communication with other team members or the client.
9	Product	Private data of the users or clients will be acquired by unauthorised hackers	20%	1	0.8	Private data of the clients gets acquired by hackers which may lead to fatal consequence
10	Project	Client becomes unresponsive.	10%	3	0.3	Client so far has shown to be very responsive. The impact would be dependent on the what stage we are in at the moment and what we need from the client.

Risk Matrix

High (5)	3		
Medium (3-4)	2,7,10	4,5,6	
Low (1-2)	1,9		8
Impact / Likehood	**Low(<=30%)**	**Medium(30%-60%)**	**High(>=60%)**

Response to Risk

ID	Trigger	Owner	Response	Resources Required
1	A teammate leaves the project.	Scrum Master	Prior to leaving they have to coordinate and communicate with team members and client, so that work can be redistributed.	None
2	New changes implemented are not adaptable to the system that's already present	Product Owner	Test the new changes implemented before releasing it and use the technology that is already being implemented if possible	
3	Increase of Users	Product Owner	Modify the existing database design allowing for more number users to access and replicating the servers to handle this	More fund to setup extra servers
4	Did not understand client requirements	Product Owner/ Scrum Master	Communicate better with clients regarding the requirements	Communication protocol

5	Unable to implement the required functionality because of lack of experience or knowledge in it	Team	Every team member who does not know the required technologies should learn them as soon as possible	
6	When some of the team members are not able to attend the meetings regularly	Team	Whoever cannot attend the meeting should watch the recording of the ones they missed and clarify doubts with other members of the team	Communication protocol
7	When the users cant access the system	Product Owner	Proper mechanisms must be implemented for all 3 types of user access and should be tested frequently	
8	Communication and documentation	Team	Speak slowly and clarify with others if they cant understand and take help from other resources and team members while documenting	Communication protocol
9	Product release and real user register	Product Owner	Avoid the data leak and fix the leak or bug as soon as possible to avoid any further damage	

Testing Plan

Objectives

The testing of the system should validate from both the requirements perspective and business perspective that:

- Security controls are in place to prevent unauthorized system access.
- Data can be visiualised in required format.
- Data integrity is maintained.
- The system is easy to use by end-users.
- Data importing and exporting functions work correctly.
- Sync data with low delay.

The objective of testing is to validate the system operation as a whole and with other systems. At the conclusion of testing, the project team and the test team will have a high level of confidence that the system will work according to user requirements and will meet business needs.

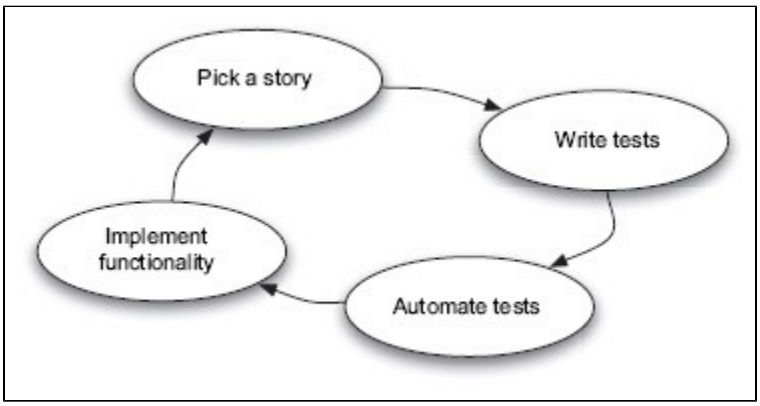
Scope

The testing of the system will include Data Digitization and Visualization, Information and Workflow Requirements, Access Control and Permission Management and Import/Export and Reporting functions.
The testing of the system will not include report or invoice auto-generating functions.

Testing strategy

Since the system is developed in agile, to align testing with incremental development, we use Acceptance Test-Driven Development (ATDD) as testing approach. This approach can be highly automated which makes testing efficient and flexible, and it verify that the features against user's expectations to ensure the product meet the requirements. To increase the overall quality of the system, we also conduct unit testing and performance testing as the complement of acceptance testing.

The acceptance TDD cycle



(resource: https://www.methodsandtools.com/archive/archive.php?id=72)

The ATDD cycle is consist of the following steps:

1. For each user story, create corresponding user acceptance criteria
2. Transform user acceptance criteria into test scripts for automated testing
3. Then implement functionality based on user acceptance criterias
4. Validate the code against the test
    - If pass, continue the cycle with another user story
    - If fail, modify the code until it pass the test

Testing approach

Acceptance testing

Acceptance testing tests whether the system meets the requirements as specified by the client.

Participants:

Testing team is responsible for creating acceptance tests and developer should implement functions based on acceptance tests.

Methodology:

1. Testing team creates acceptance tests based on acceptance criteria.
2. Developer read acceptance tests and implement function correspondingly.
3. After implementation, developer upload the evidence of asscoiated acceptance test is satisified in **Acceptance tests** in the confluence.
4. In the testing phrase of sprint, testing team validate the testing result again and document the outcomes in the testing report of the sprint.

Unit testing

Unit Testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. In this project, we define one unit as one function. For example, assuming one client is an instance, adding new client is an unit, editing client information is also an unit.

Participants:

Each developer who participate in coding should write unit tests for the functionalities that they developed to ensure it work correctly.

Methodology:

1. After developer implemented one function of a component, developer write a unit test in the test file of the component with reasonable justification.
    - For example, after developer implemented the adding function of client, developer should write a unit test named ***test_client_add*** in ***client_test*** file.
2. Developer then run the unit test and perform action based on testing outcome
    a. If test fail, developer should modifiy the code until the test pass.
    b. If test pass, developer should add passed unit test to the ***Unit tests*** in the confluence.
3. In the testing phrase of sprint, testing team validate the testing result again and document the outcomes in the testing report of the sprint.

Performance testing

Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload.

Participants:

Testing lead and testing team members should be responsible for the implementation and documentation of performance testings.

Methodology:

1. The testing team define acceptable performance criteria based on client's requirement through discussion.
2. The testing team write performance tests based on criteria.
3. After executing performance tests, act based on the testing outcome:
    a. If criteria is not met, notifiy the development team and discuss how to resolve it based on the current timeframe.
    b. If criteria is satifised, create a report that document how the outcome satisfy the performance criteria.

Problem Reporting

If any bug is detected during the testing phrase of each sprint, the tester should create a issue in Jira to notify the responsible developer the existance of bug with following information:

- The location of the bug (Which section of code contain this bug?)
- A description that specify why it is a bug
- Suggestions of possible modification

Tool

Cypress - Acceptance testing

Cypress is a purely JavaScript-based front-end testing tool built for the modern web. It aims to address the pain points developers or QA engineers face while testing an application. Cypress is a more developer-friendly tool that uses a unique DOM manipulation technique and operates directly in the browser. Cypress also provides a unique interactive test runner in which it executes all commands.

Cypress vs Selenium

In this project, we want to focus our limited resource on development to ensure the quality of product. Despite Selenium has long history with well-established documentation and firm community support, it is time-consuming to create test case in Selenium. On the other hand, Cypress as an fast-growing automation tool, only require simple set-up and provides user-friendly interface to help testers get on hands. This property of Cypress makes it preferable in this project.

Jest - Unit testing

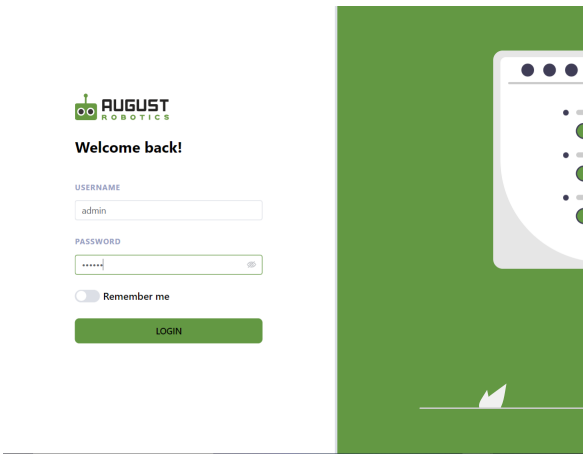
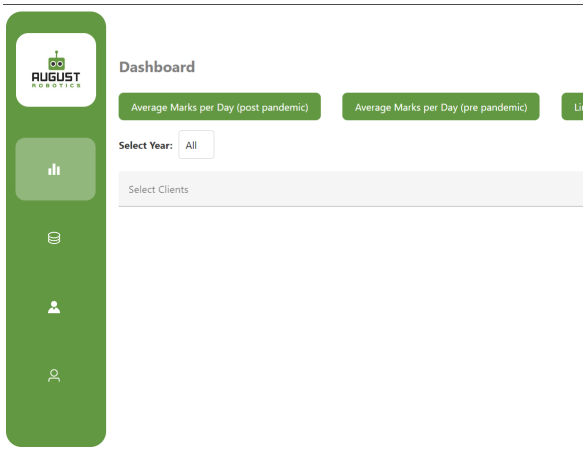
Jest is a testing framework of JavaScript mainly designed to ensure the JavaScript codebase preciseness and accuracy. It provides functionality to write tests with an approachable, familiar and feature-rich API (Application Programmable Interface) that will provide test results quickly for the given code. In addition, it also has well-defined documentations and only requires little configuration.

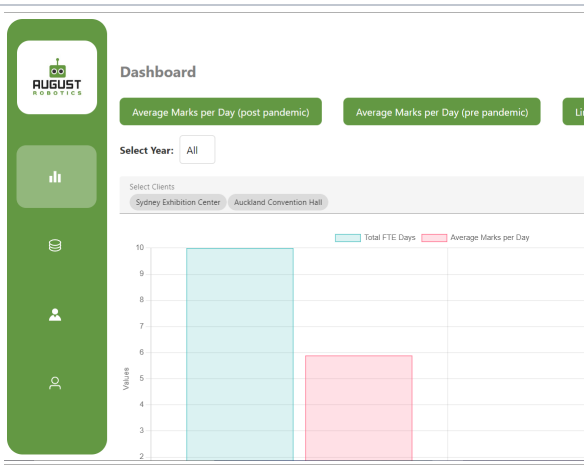
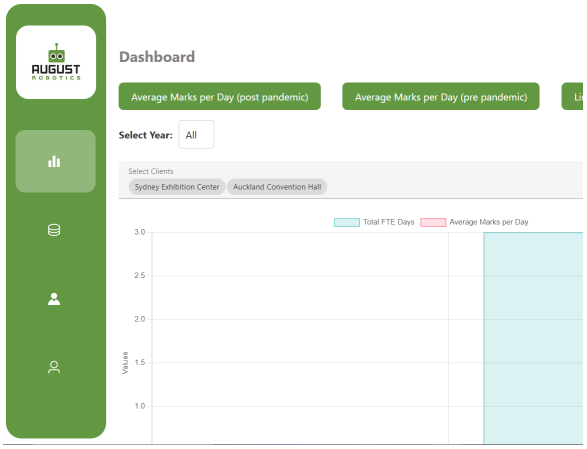
Acceptance Tests

**Screenshots added for Access Control, Fixed data epic user stories, Searching user stories, Operational data and some Sprint 6 job data dashboard user stories*

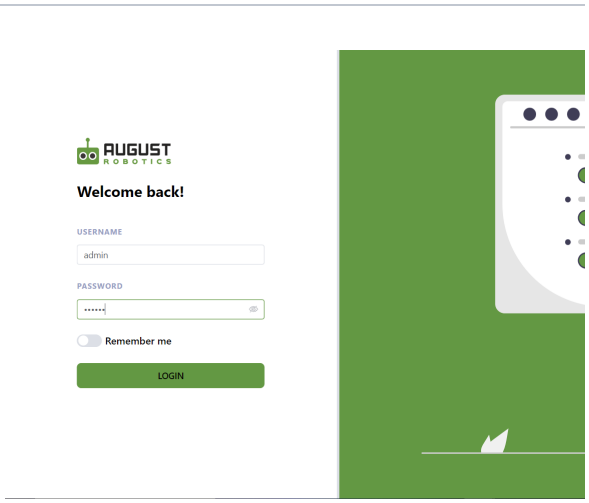
*Synchronous with Acceptance Criteria

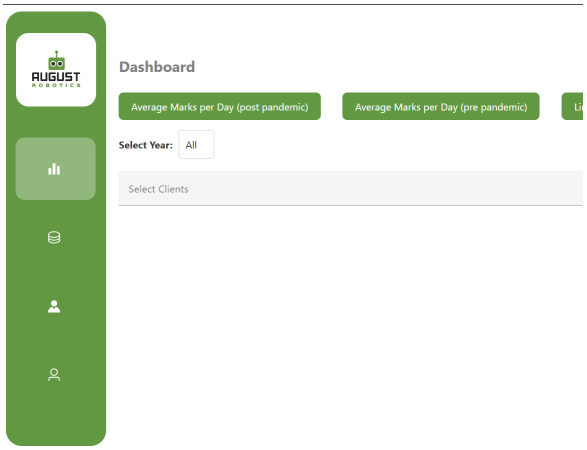
- Job Data Digitization (Dashboard)
- Access Control
- Operational Data
- Fixed Data
- Import and export data
- Searching
- Admin User Management

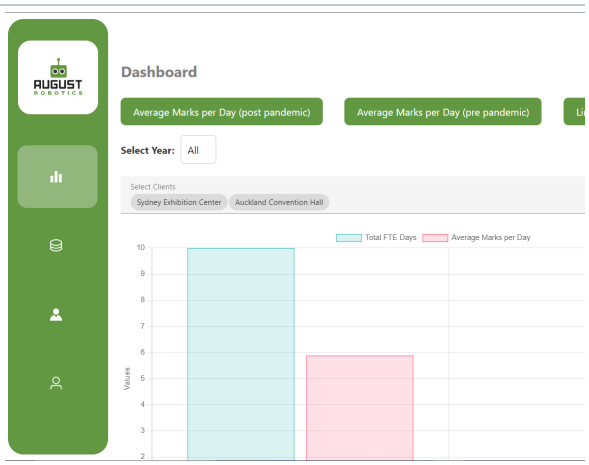
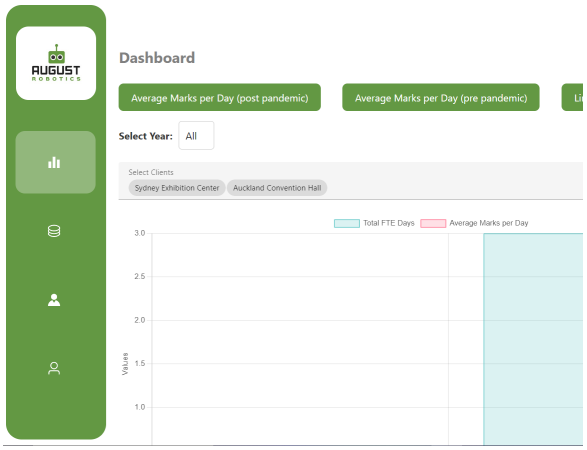
Epic	User Story	AC ID	Acceptance Criteria	Input (action that user perform)	Output (expected outcome for performed action)	Result (pass/fail)	Screenshots (with comments for failed/updated passed test cases)
Job Data Digitization (Dashboard)	As a Database Platform user, I want to view a table of total Marks marked by AR divided by clients so that I can look at the total marks for each client pre and post-pandemic monthly or quarterly.	1.1	Given I login to view the database platform dashboard, when the dashboard presents the data analysis for the job key performance index, then I want to select the table to see total Marks marked by AR divided by clients	Login using correct Database platform user account	On main dashboard page	PASS	 

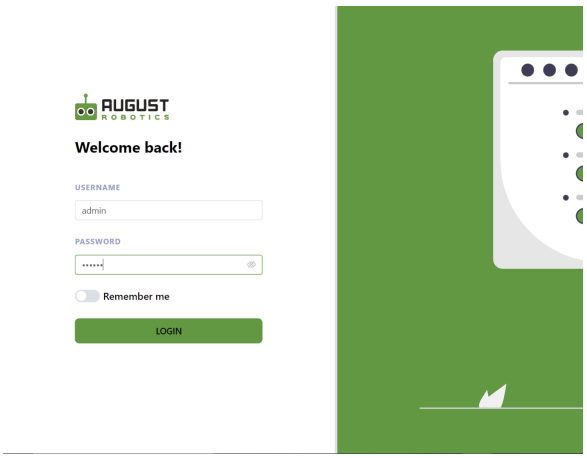
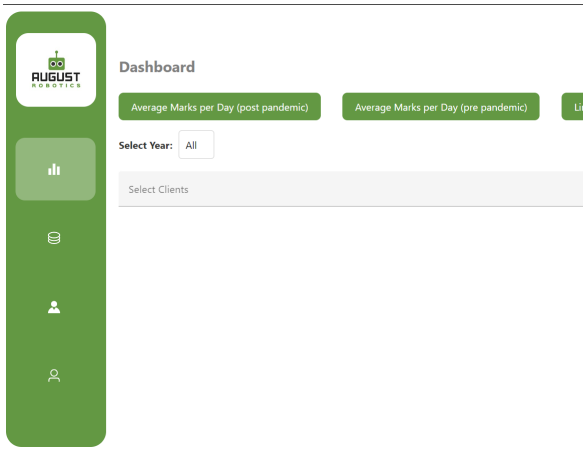
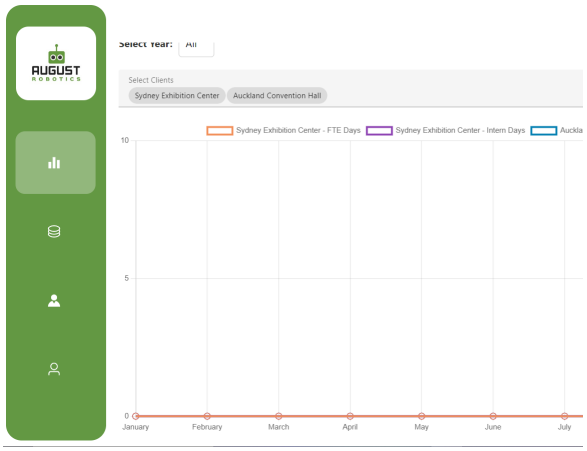
			Selects "total Marks marked by AR" table	Sees table	PASS	 

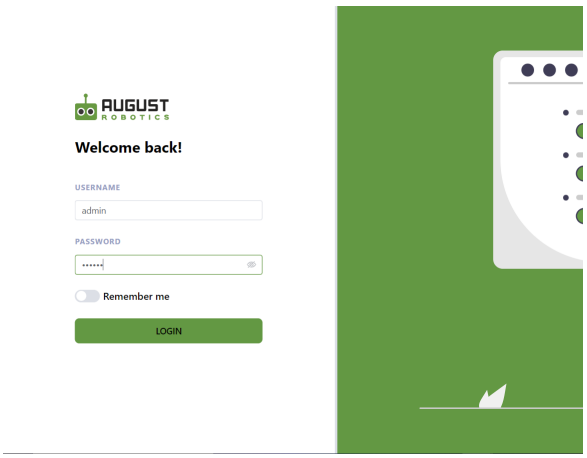
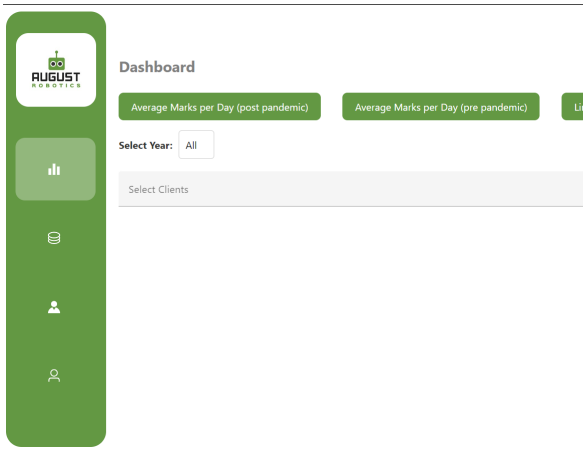
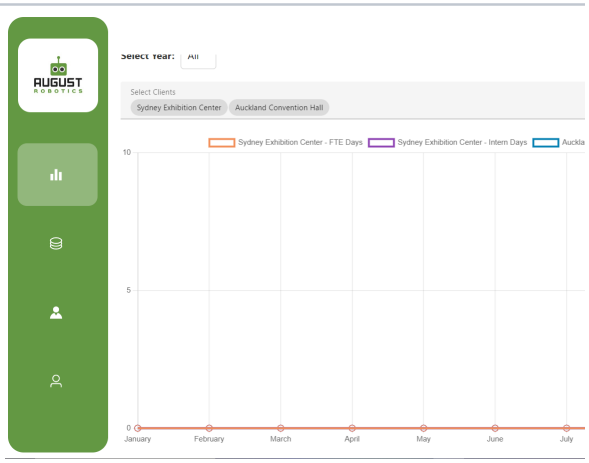
		1.2	Given I have selected to view the table, when the table is opened in a bigger view, then I want to view correctly calculated data displayed in it.	Login using correct Database platform user account	On main dashboard page	PASS



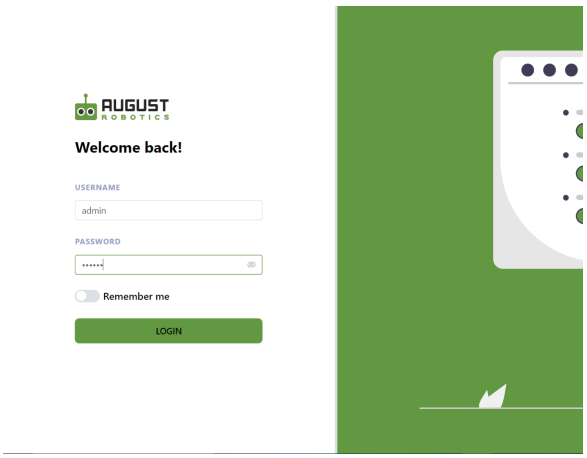
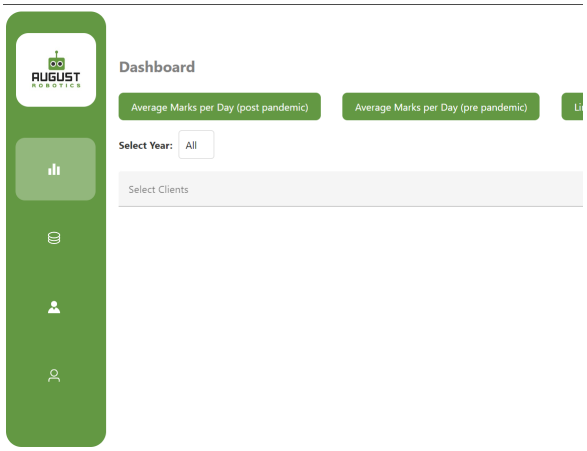
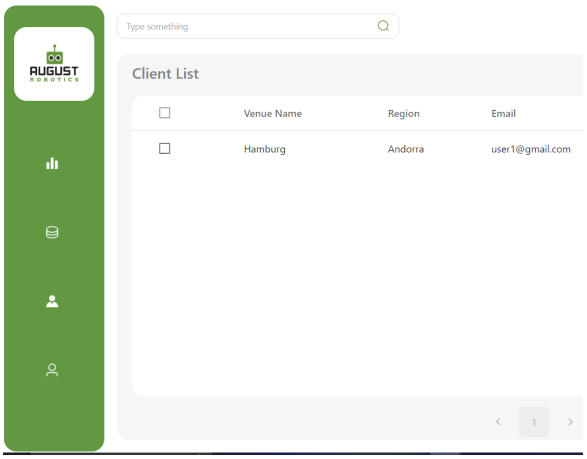


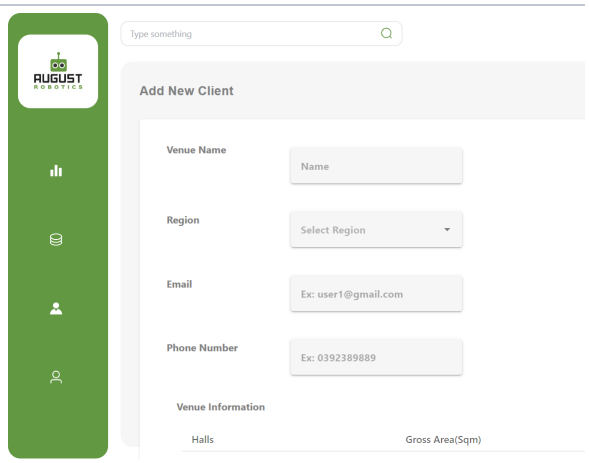
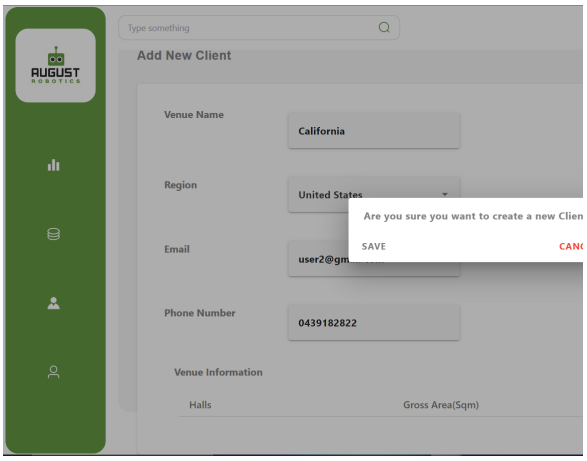
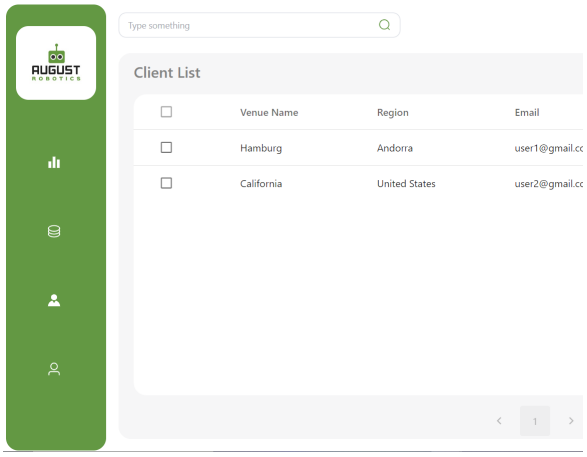
			Selects "total Marks marked by AR" table	Sees table	PASS	 

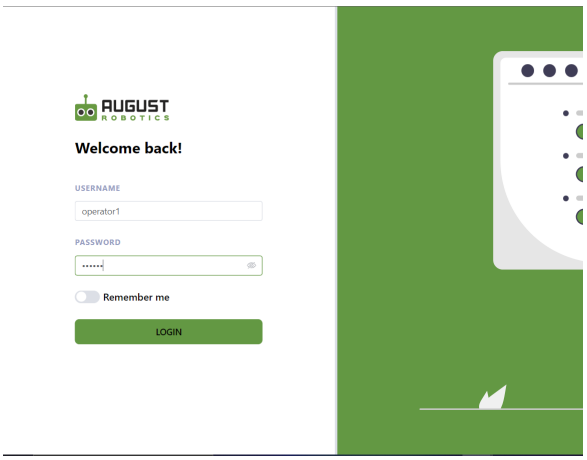
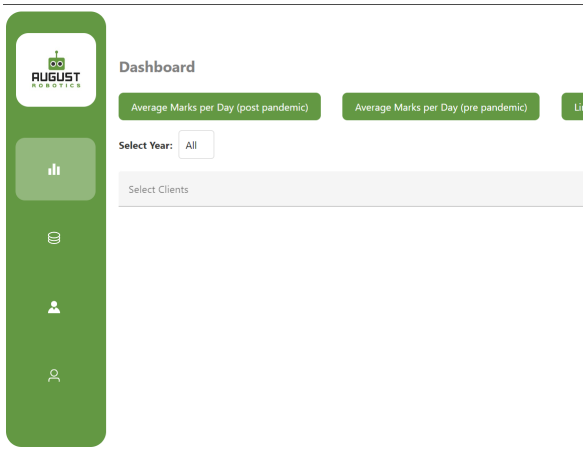
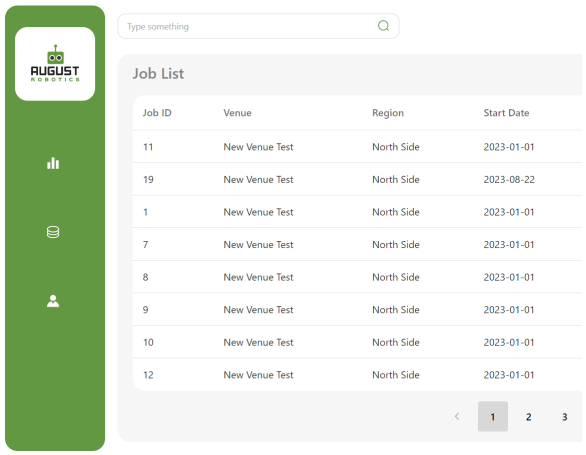
As a Database Platform user, I want to view a table /graph showing the Total FTE (full-time equivalent) and Intern days worked on-site divided by client so that I can look at how many days were invested and efficiency performed per person on-site.	2.1	Given I login to view the database platform dashboard, when the dashboard presents the data analysis for the job key performance index, then I want to select the table/graph showing the Total FTE (full-time equivalent) and Intern days worked on-site divided by client.	Login using correct Database platform user account	On main dashboard page	PASS	 
			Selects "Total FTE and Intern days worked" table	Sees table	PASS	

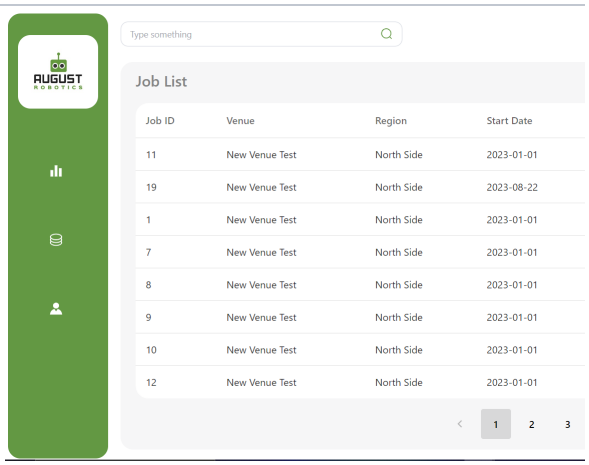
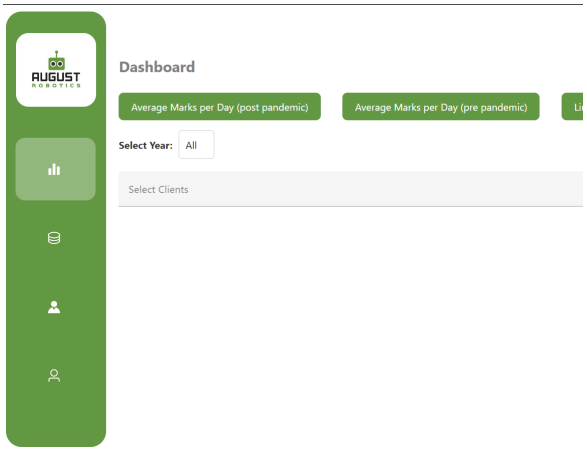
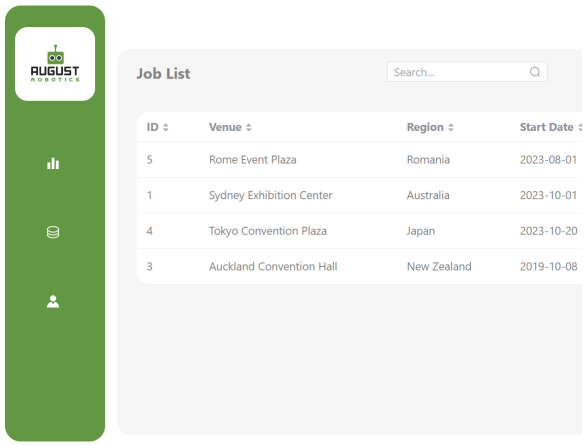
	2.2	Given I have selected to view the table, when the table/graph is opened in a bigger view, then I want to view correctly calculated data displayed in it.	Login using correct Database platform user account	On main dashboard page	PASS	 
			Selects "Total FTE and Intern days worked" table	Sees table	PASS	
As a Database Platform user, I want to view a table showing average number of marks per day so that I can look at the average marks for a client for the total job days	3.1	Given I login to view the database platform dashboard, when the dashboard presents the data analysis for the job key performance index, then I want to select the table showing average number of marks.	Login using correct Database platform user account	On main dashboard page	NOT IMPLEMENTED	
			Selects "average number of marks"	Sees table	NOT IMPLEMENTED	
	3.2	Given I have selected to view the table, when the table is opened in a bigger view,	Login using correct Database platform user account	On main dashboard page	NOT IMPLEMENTED	

			then I want to view correctly calculated data displayed in it.	Selects "average number of marks"	Sees table	NOT IMPLEMENTED	
	As a Database Platform user, I want to view a table showing average number of marks per job window so that I can assess the client behavior and needs to plan the capacity required on-site.	4.1	Given I login to view the database platform dashboard, when the dashboard presents the data analysis for the job key performance index, then I want to select the table showing average number of marks per job window	Login using correct Database platform user account	On main dashboard page	NOT IMPLEMENTED	
				Selects "average number of marks per job window"	Sees table	NOT IMPLEMENTED	
		4.2	Given I have selected to view the table, when the table is opened in a bigger view, then I want to view correctly calculated data displayed in it.	Login using correct Database platform user account	On main dashboard page	NOT IMPLEMENTED	
				Selects "average number of marks per job window"	Sees table	NOT IMPLEMENTED	
	As a Database Platform user, I want to view a table showing the marked number of halls so that I can see the marked halls for each client to check the total job days and average halls marked per day.	5.1	Given I login to view the database platform dashboard, when the dashboard presents the data analysis for the job key performance index, then I am able to view the table showing the marked number of halls	Login using correct Database platform user account	On main dashboard page	NOT IMPLEMENTED	
				Selects "marked number of halls"	Sees table	NOT IMPLEMENTED	
		5.2	Given I have selected to view the table, when the table is opened in a bigger view, then I am able to check the total job days and average halls marked per day for each client' halls	Login using correct Database platform user account	On main dashboard page	NOT IMPLEMENTED	
				Selects "marked number of halls"	Sees table	NOT IMPLEMENTED	

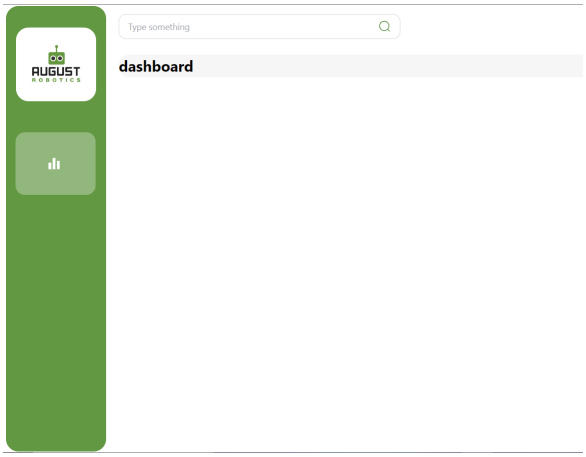
Access Control	As a administrator, I want to be able login to access fixed and operational data, so that I can maintain the database platform	6.1	Given I have logged in as an administrator, when there is a new client, then I am able to access fixed data.	Login using correct Database platform Administrator account	On main dashboard page for admins	PASS	 
				Select Client List page	On client list page	PASS	

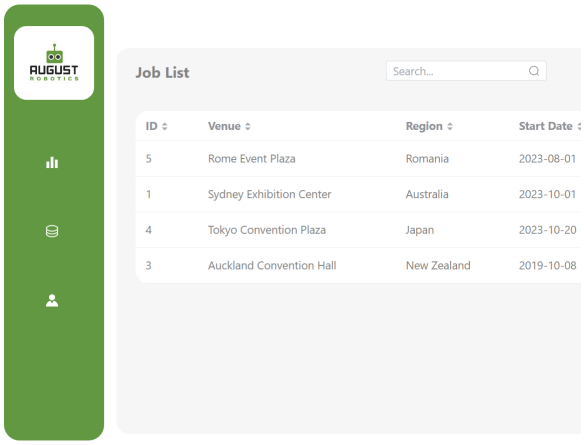
			Press "add new client"	On client creation page	PASS	
			Enter client data	Client Data is inputted	PASS	
			Press "Save client", and confirm	Redirects to client list page with new client.	PASS	

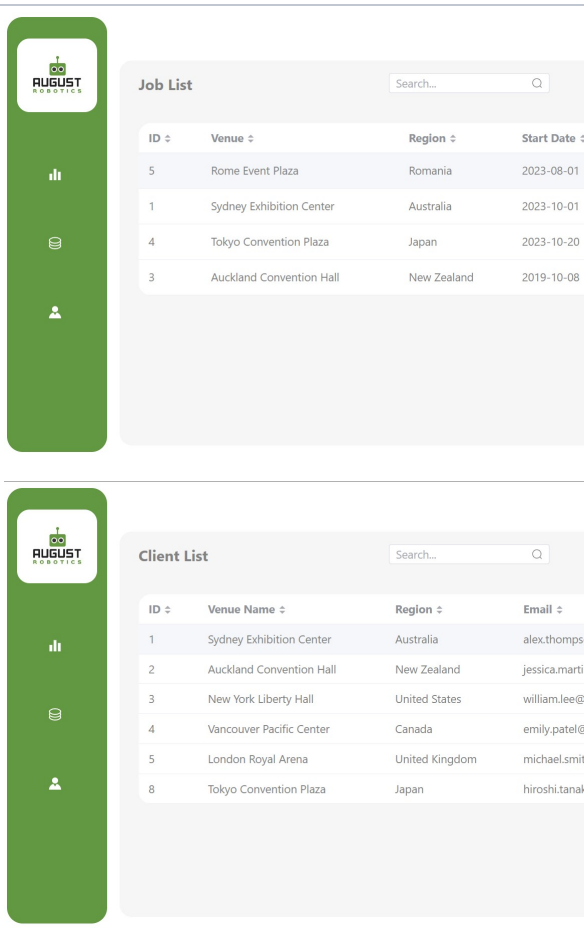
As a data maintainer, I want to be able login to access fixed and operational data so that I can maintain the database platform.	7.1	Given I have logged in as job data maintainer, when the job is done, then I want to update operational data when the job is completed.	Login using correct Database platform Maintainer account	On main dashboard page for maintainer	PASS	 
			Select Job List page	On job list page	PASS	

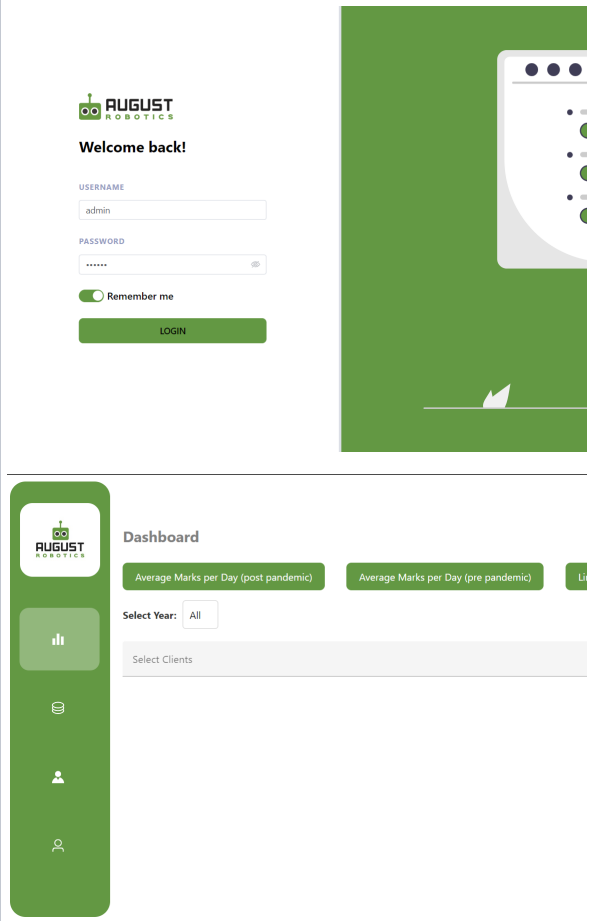
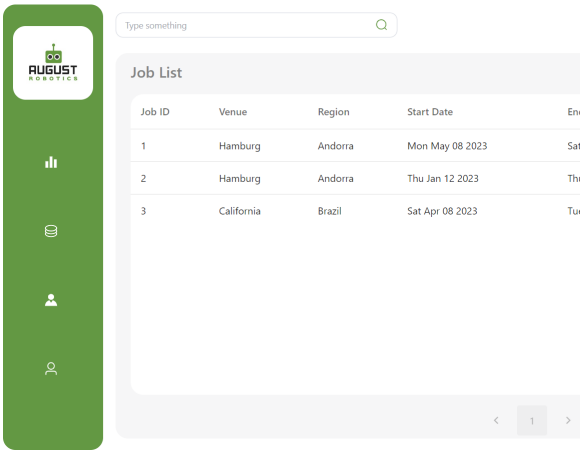
			Select an ongoing job's "Operate" column and select edit	Options to edit job open	PASS	
As a viewer, I want to be able to just view the data and have no access to modify or update any data	8.1	Given I have logged in as a viewer, when I want to view data, when I want to view data, I am able to view the relevant data.	Login using correct Database platform Viewer account	On main dashboard page for viewer	PASS	
			Select Job List page (any page would work)	On job list page (or relevant page)	PASS	

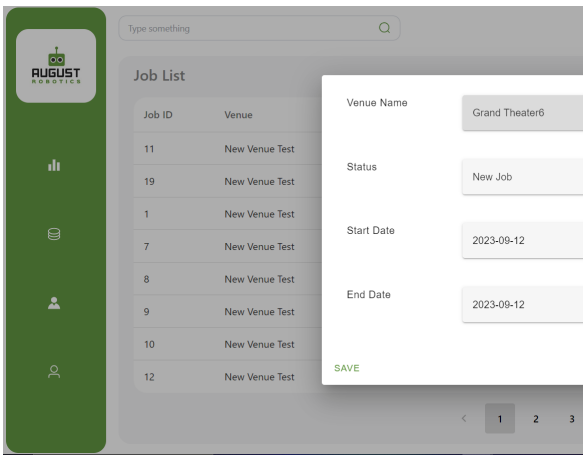
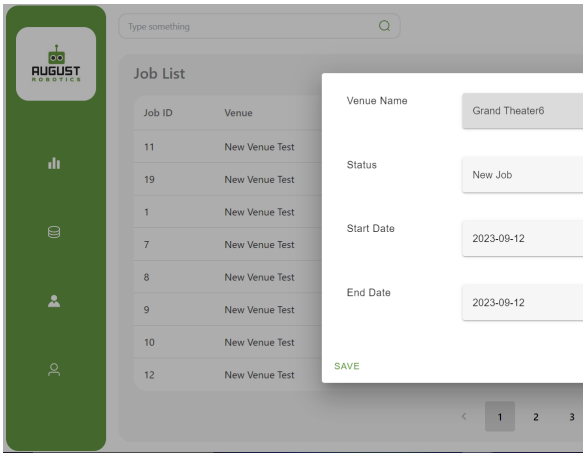
		8.2	Given I have logged in as a viewer, when I try to modify or update some data, I am unable to modify or update any data.	Login using correct Database platform Viewer account	On main dashboard page for viewer	PASS
				Select Job List page (any page would work)	On job list page (or relevant page)	PASS

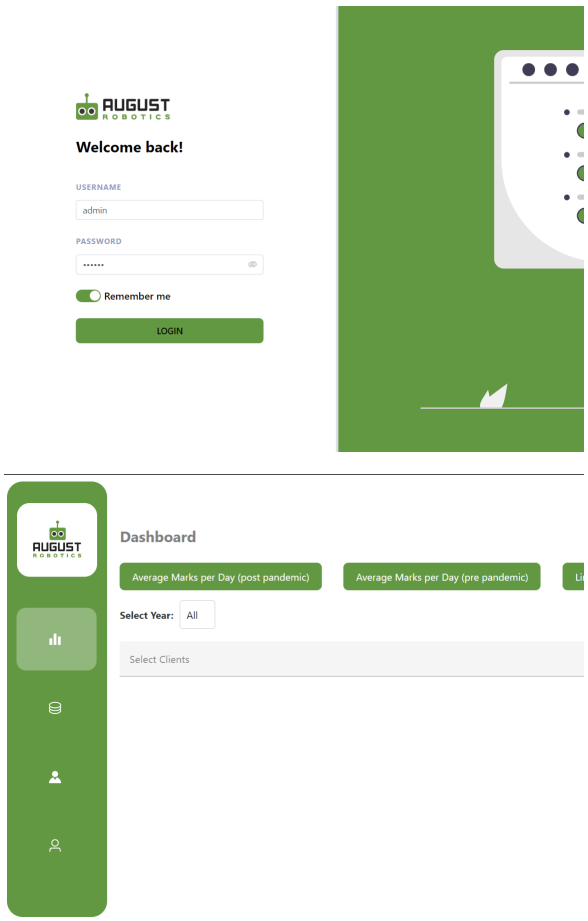
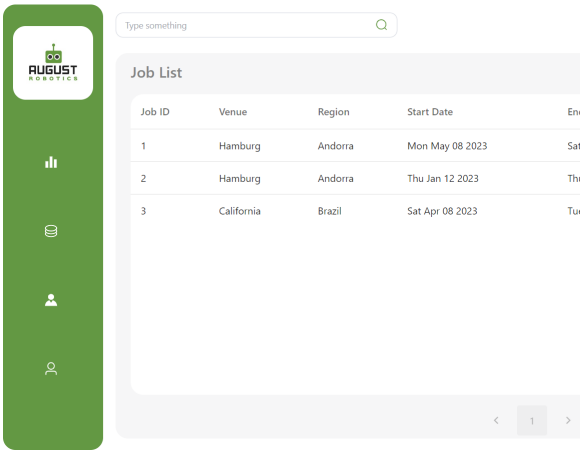




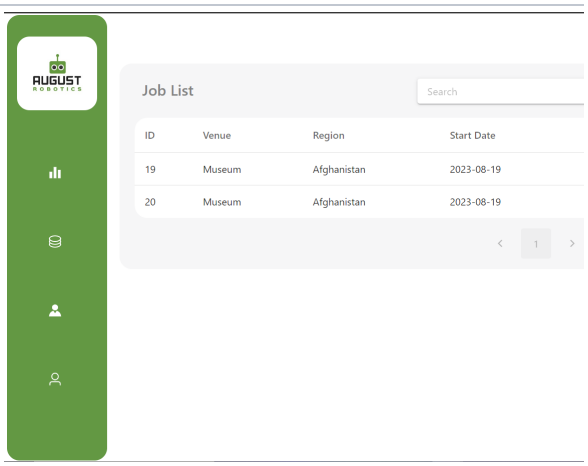
				Select a job's "Operate" column	Should not be available or cannot be selected	PASS	

Operational Data	As an Administrator/data maintainer, I want to be able to input preliminary job data into the system when there is a new job, so that the system has the necessary data for a new job	9.1	Given I am a logged in as an administrator /data maintainer, when a new job is created, then I want to be able to input preliminary job data.	Login using correct Database platform admin/data maintainer account.	On main dashboard page for admin.	PASS	
				Select Job List page	On Job List page	PASS	

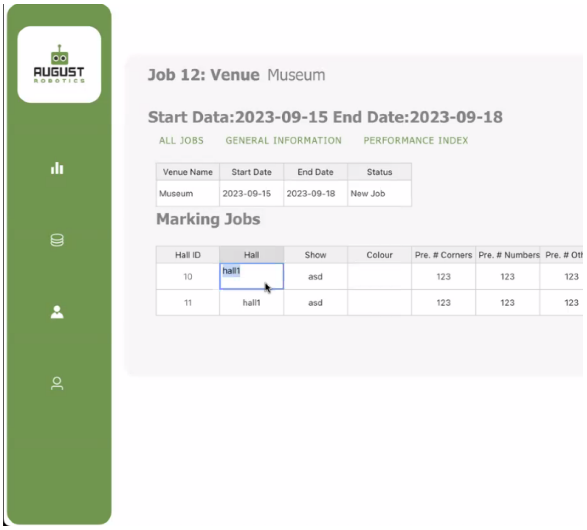
			Select add new job	On Create Job popup	PASS	 Previously failed, as user cannot input data, kept getting erased, and data not beir
			Input initial information and create	Able to select venue name from popup dropdown	PASS	 Previously failed, as the popup didn't have dropdown for venue name to chose clie

As an Administrator/data maintainer, I want to be able to input final job data for each hall marking job, so that I can update the floor marking data that is obtained during the completion of a job	10.1	Given I am a logged in as an administrator /data maintainer, when a completed job is viewed, then I am able to update the job with data gathered during completion.	Login using correct Database platform admin/data maintainer account.	On main dashboard page for admin.	PASS	
			Select Job List page	On Job List page	PASS	

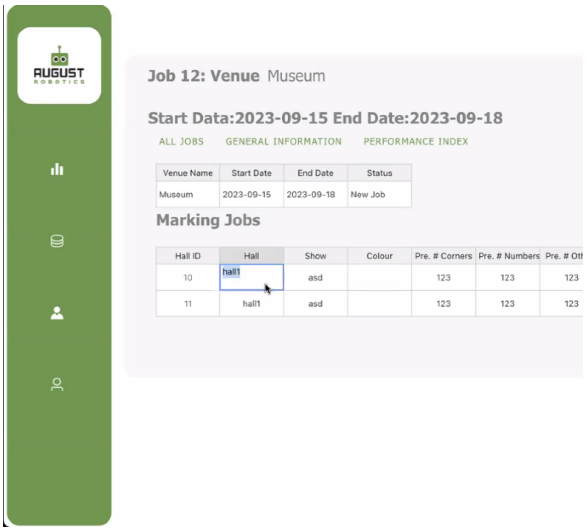
				Select an ongoing job's "Operate" column and select view in detail	Go to job's detail page	PASS
				Select edit	Data becomes modifiable	PASS
				Change job data and save	Returns to job's detail page	PASS



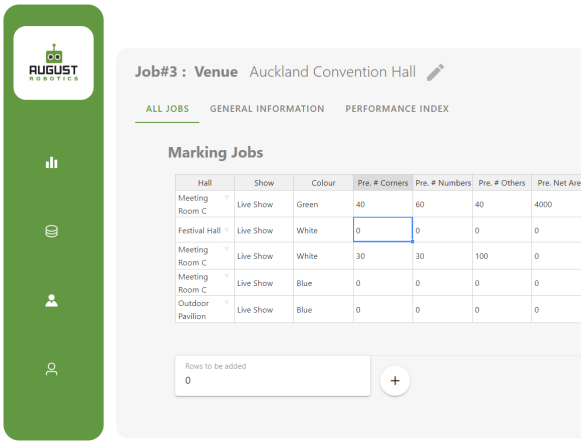
(The page below is still under development, changes will be reflected soon)

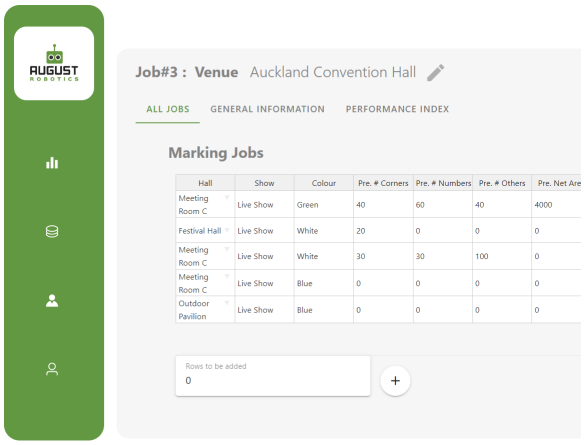


Changes to UI been done since last tested, thus new test carried out.

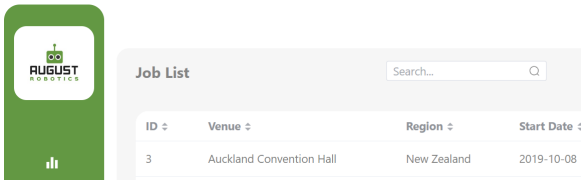


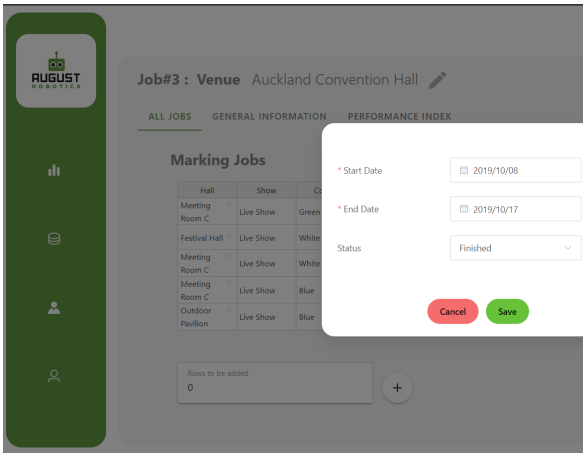
Previously failed, as the user couldn't edit the whole table at once and was one ro

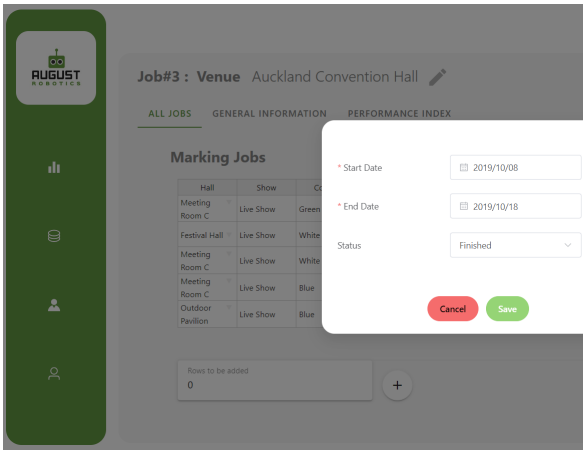


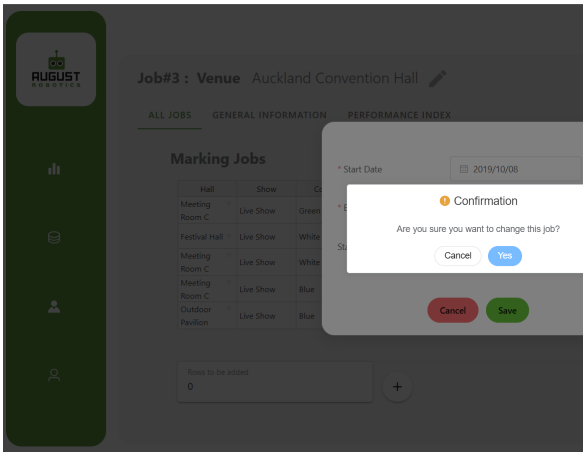


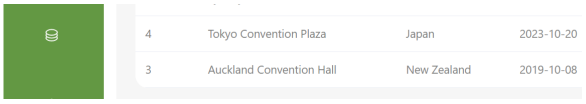
Job detail modification:

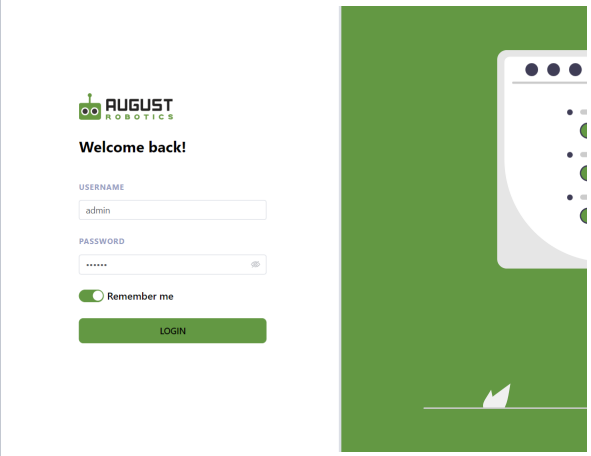
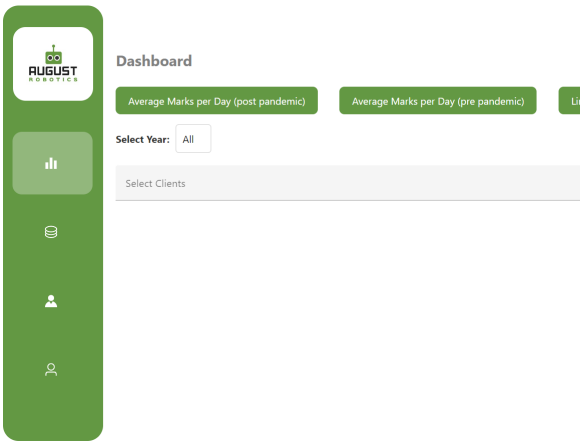
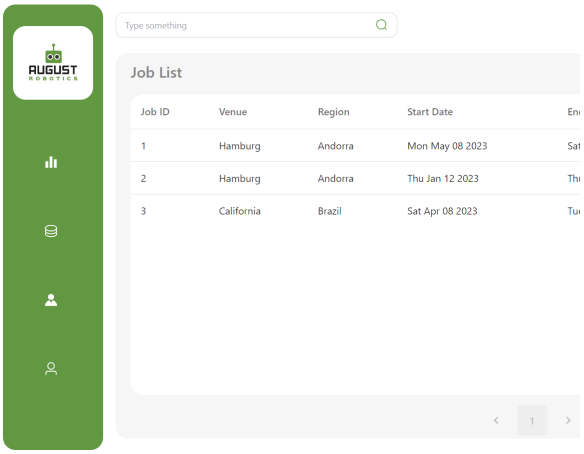


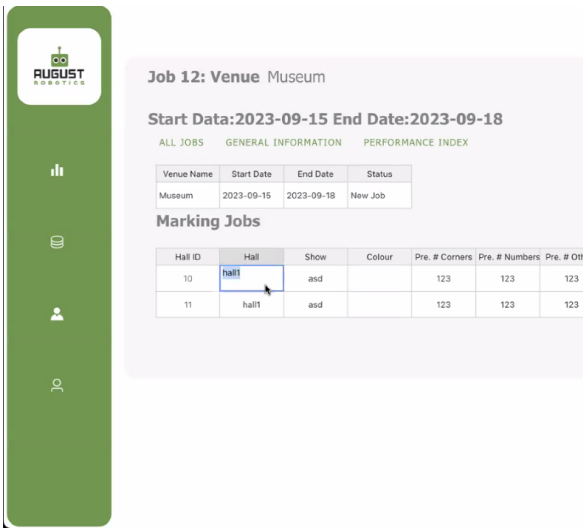
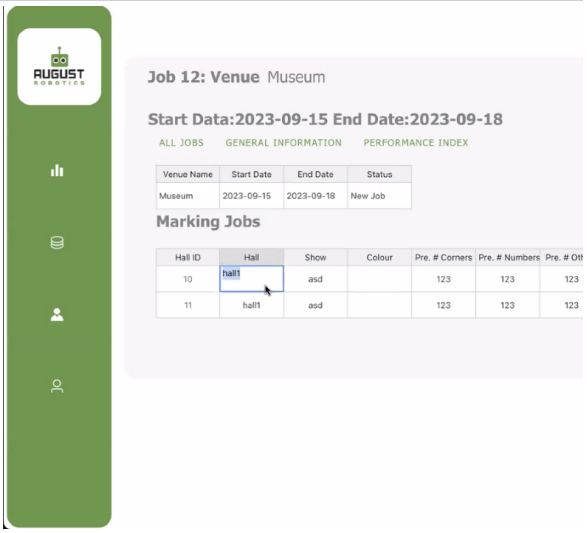
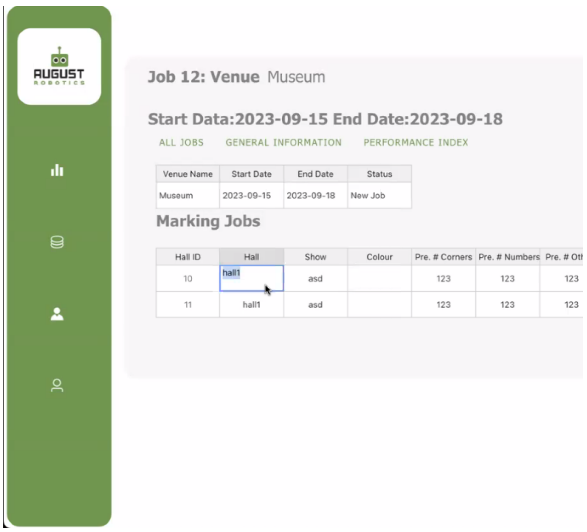


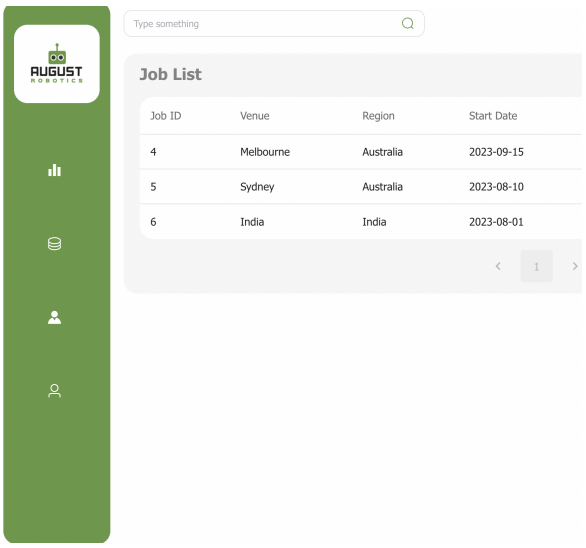
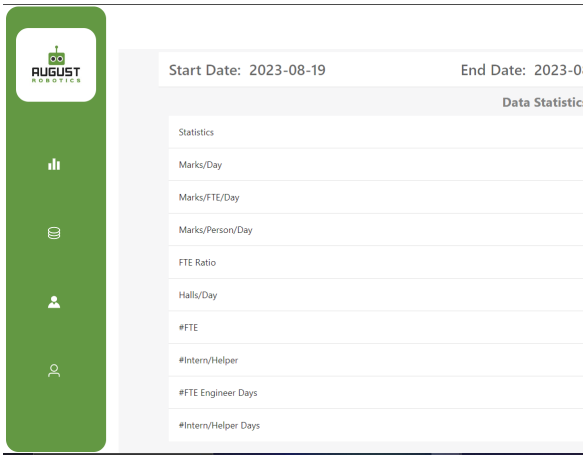
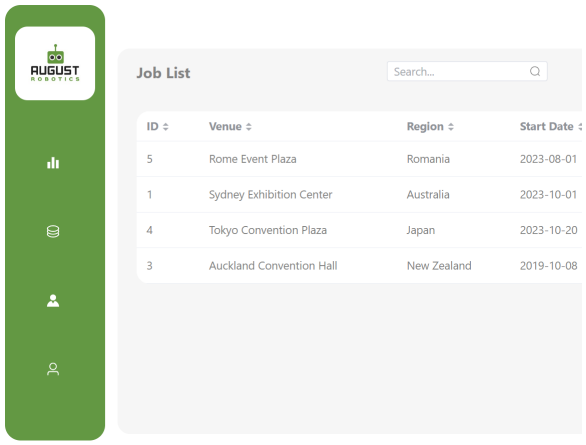

AUGUST ROBOTICS
Job#3 : Venue Auckland Convention Hall
ALL JOBS
GENERAL INFORMATION
PERFORMANCE INDEX
Marking Jobs
Hall
Show
Meeting Room C
Live Show
Green
Festival Hall
Live Show
White
Meeting Room C
Live Show
White
Meeting Room C
Live Show
Blue
Outdoor Pavilion
Live Show
Blue
* Start Date
2019/10/08
* End Date
2019/10/18
Status
Finished
Cancel
Save
Rows to be added
0

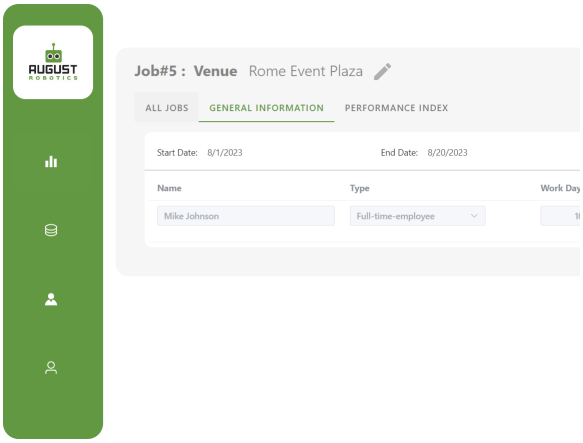

AUGUST ROBOTICS
Job#3 : Venue Auckland Convention Hall
ALL JOBS
GENERAL INFORMATION
PERFORMANCE INDEX
Marking Jobs
Hall
Show
Meeting Room C
Live Show
Festival Hall
Live Show
Meeting Room C
Live Show
Meeting Room C
Live Show
Blue
Outdoor Pavilion
Live Show
Blue
* Start Date
2019/10/08
Confirmation
Are you sure you want to change this job?
Cancel
Yes
Cancel
Save
Rows to be added
0

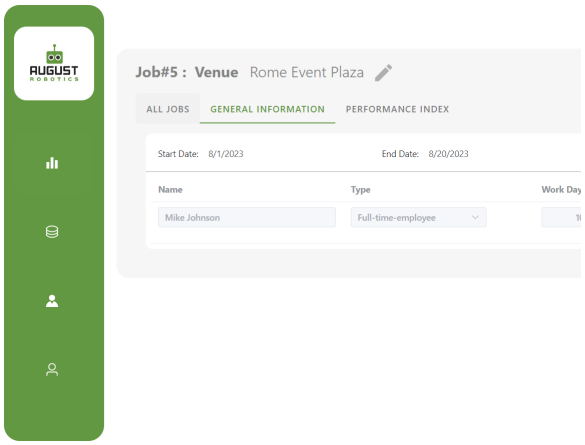

4
Tokyo Convention Plaza
Japan
2023-10-20
3
Auckland Convention Hall
New Zealand
2019-10-08

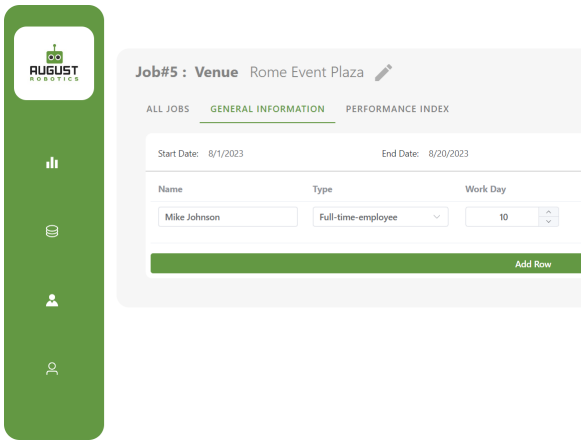
As a Database Platform user, I want to view processed floor marking job data (operational) so that I can quickly access information for an ongoing or finished job.	11.1	Given I log in to view the database platform dashboard, when the dashboard is displayed, then I want to select the job lists on the navigation bar and view all floor marking jobs for clients.	User navigates to the dashboard option	User manages to the Dashboard	PASS	 
			User click 'Job List' option on the Navigation bar in Dashboard	List of floor marking jobs will be displayed (venue name, region, start date, end date, status)	PASS	

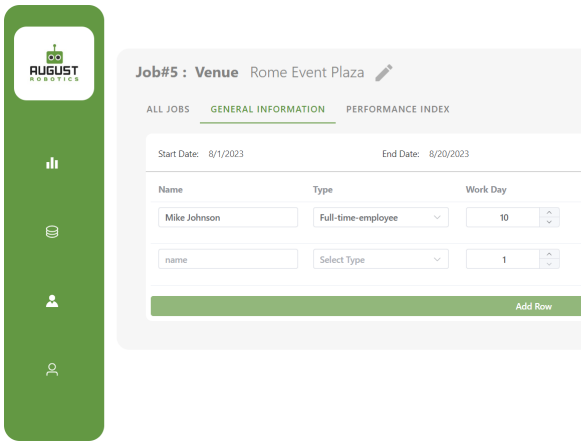
			View floor marking jobs done	Display Data representing floor marking job	PASS	
	11.2	Given I am viewing the floor marking job lists, when I select a particular job, then I want to view the venue information and input details before/after the job is done.	User select a particular Job	Display venue information	PASS	 Previously failed, as Venue details such as venue name, region, status, start date data. Test case now passed.
				Display input details from before and after the job	PASS	Job details can be viewed as required. 

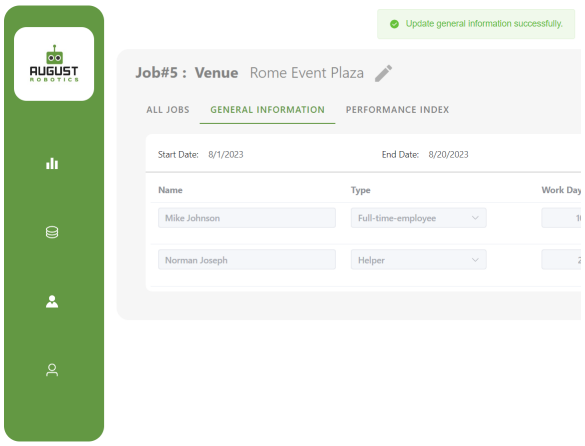
As a Database Platform User, I want to view the performance index for jobs, so that I can analyse the performance for a job after it's completion.	12.1	Given I am viewing the the job listings , when I select the option to view the performance index, then I can see the performance index for all jobs.	User navigate to Job List page	Display the Job List page	PASS	
			User select 'view performance index'	Display the performance index of all the jobs	PASS	
As a Database Platform User, I want to view the general information for a job, so that I can see the people working on a job and estimated completion dates	13.1	Given I am viewing the job listings for a client, when I select the job, then I can view the detailed general information of the job.	User navigate to Job List page	Display Job List page	PASS	

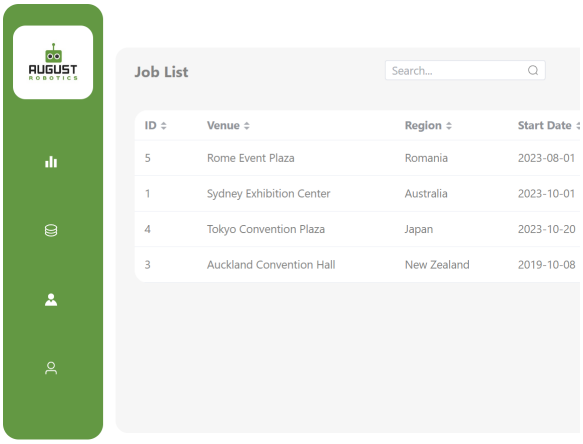
				User select 'General information'	General information will be displayed	PASS	
				User select edit and add new row	New row added and details saved	PASS	

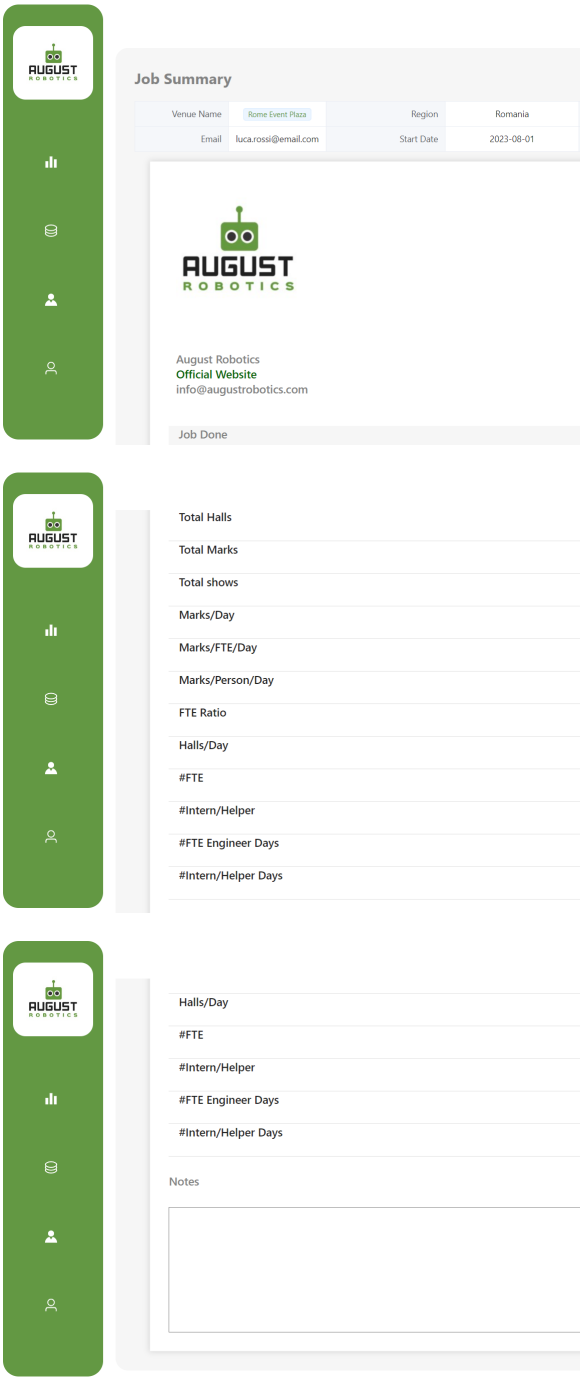

AUGUST ROBOTICS
Job#5 : Venue Rome Event Plaza
ALL JOBS
GENERAL INFORMATION
PERFORMANCE INDEX
Start Date: 8/1/2023
End Date: 8/20/2023
Name
Type
Work Day
Mike Johnson
Full-time-employee

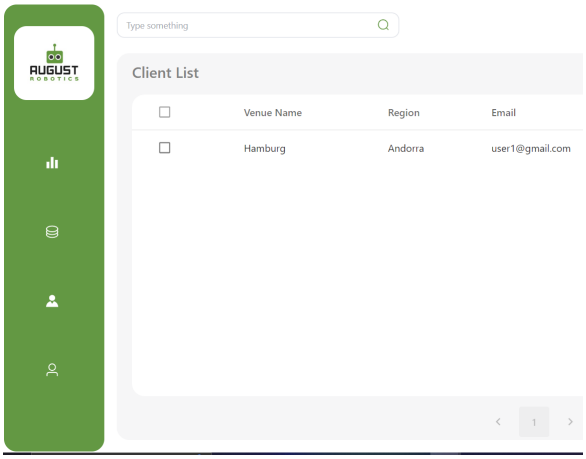
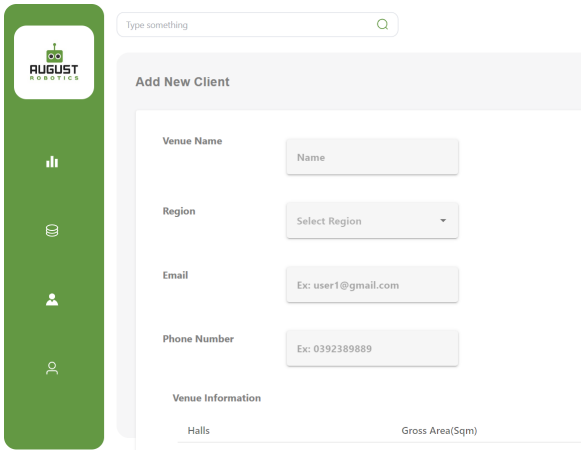
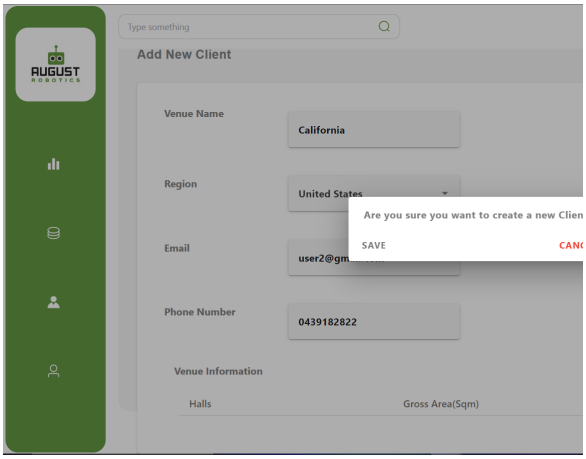

AUGUST ROBOTICS
Job#5 : Venue Rome Event Plaza
ALL JOBS
GENERAL INFORMATION
PERFORMANCE INDEX
Start Date: 8/1/2023
End Date: 8/20/2023
Name
Type
Work Day
Mike Johnson
Full-time-employee
10
Add Row

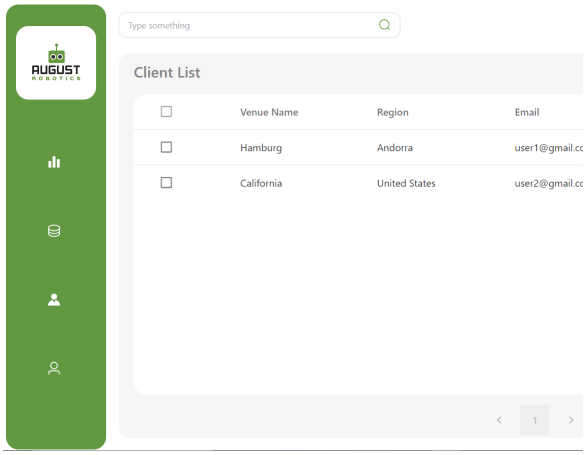
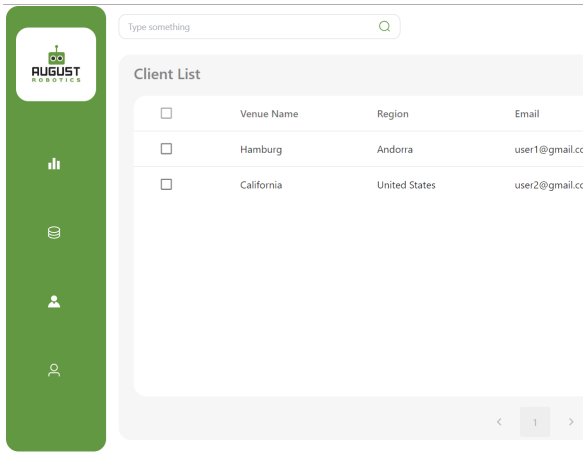
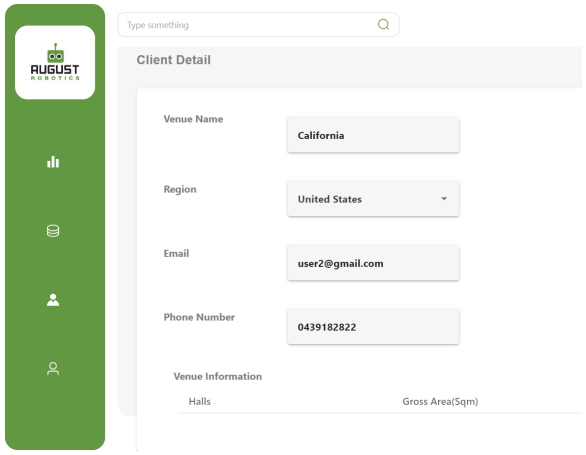

AUGUST ROBOTICS
Job#5 : Venue Rome Event Plaza
ALL JOBS
GENERAL INFORMATION
PERFORMANCE INDEX
Start Date: 8/1/2023
End Date: 8/20/2023
Name
Type
Work Day
Mike Johnson
Full-time-employee
10
name
Select Type
1
Add Row

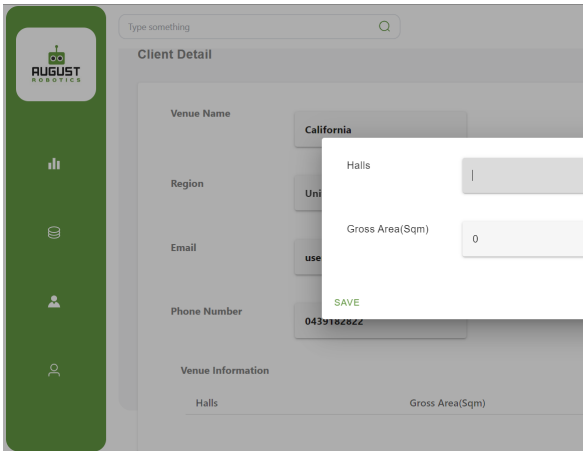
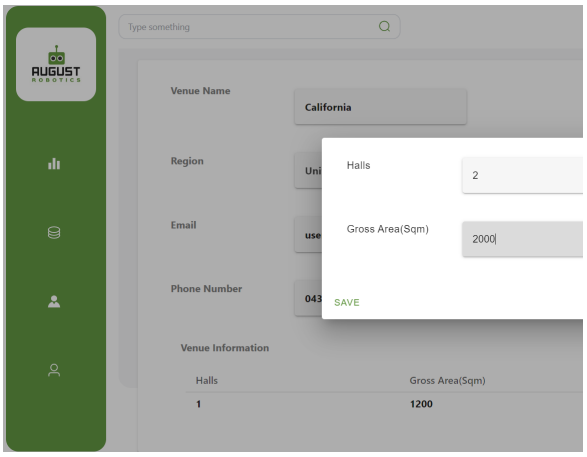
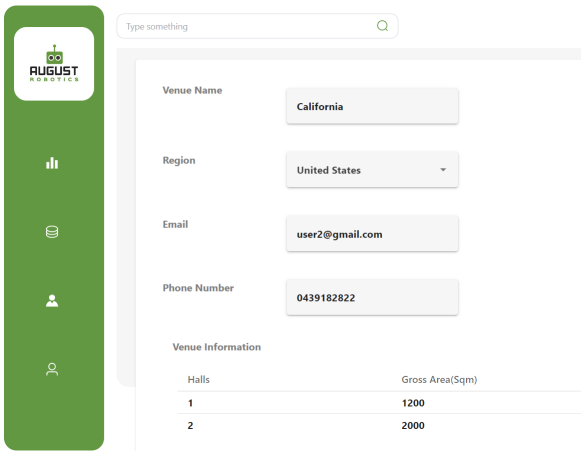

Update general information successfully.
AUGUST ROBOTICS
Job#5 : Venue Rome Event Plaza
ALL JOBS
GENERAL INFORMATION
PERFORMANCE INDEX
Start Date: 8/1/2023
End Date: 8/20/2023
Name
Type
Work Day
Mike Johnson
Full-time-employee
Norman Joseph
Helper

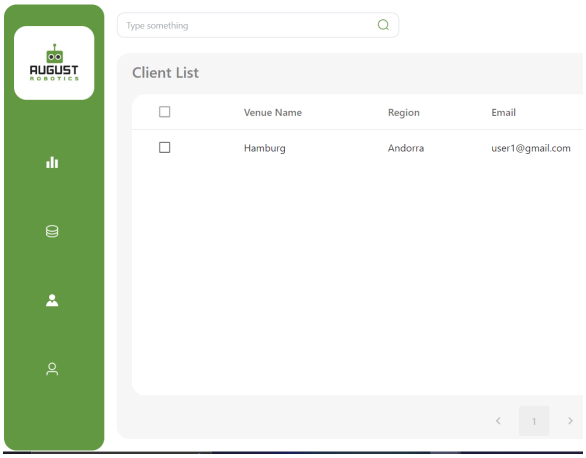
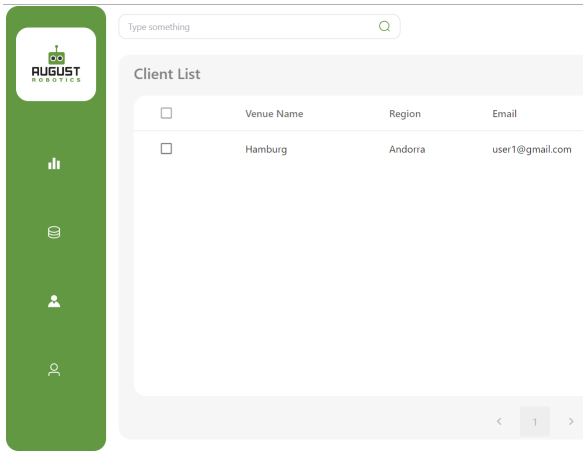
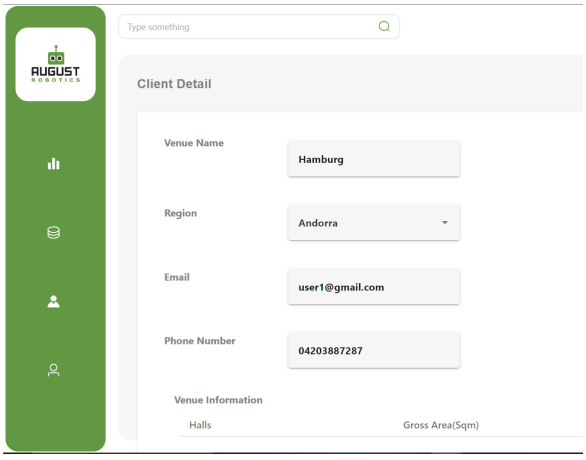
As a Database Platform User, I want to view the generate a job summary, so that I can extract the information needed from a job to generate an invoice	14.1	Given I am in a job's details page, when I select the option to generate summary, then a summary is generated that can be downloaded.	User navigate to Job List page	Display Job List page	PASS	

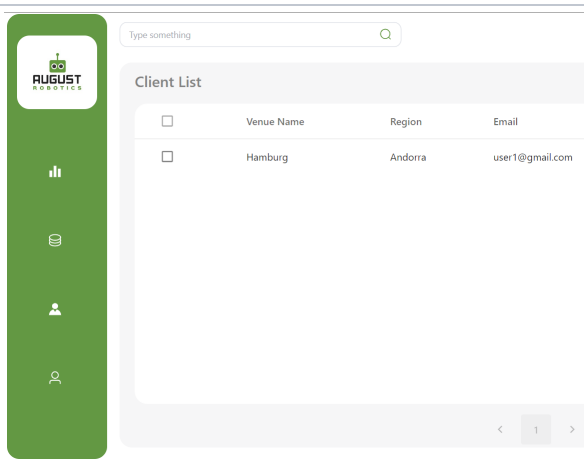
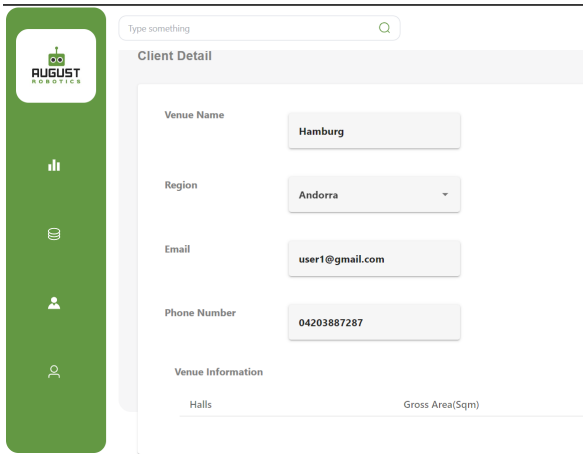
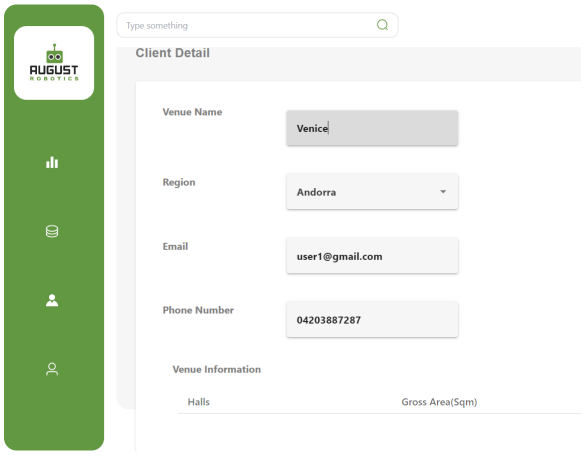
				User select 'Generate Summary' option	Summary will be generated	PASS	
				Click on the download icon to download	The summary will be downloaded	NOT IMPLEMENTED	*Not implemented yet (part of export functionality)
	As a Database Platform User, I want to view a version history of hall marking jobs within a job, so that I can track and view all changes	15.1	Given I am in the job's details page, when I can select the option to view history, then I see a graph /table showing all changes made to that hall marking job.	User navigate to Job List page	Display Job List page	NOT IMPLEMENTED	

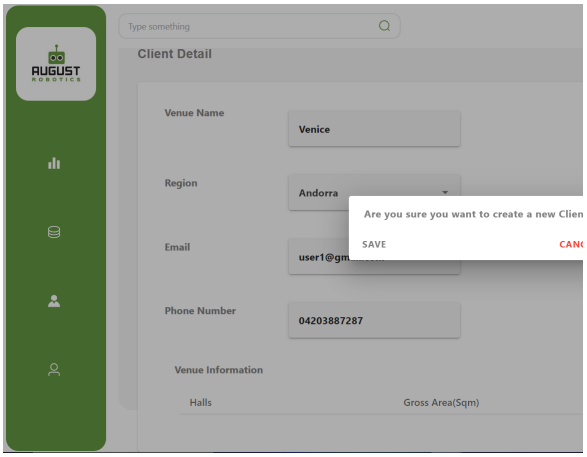
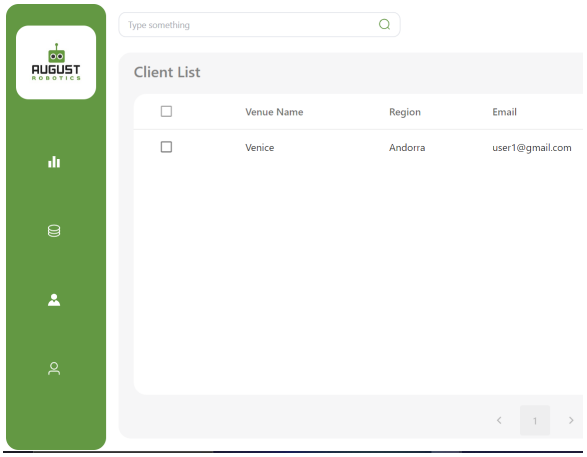
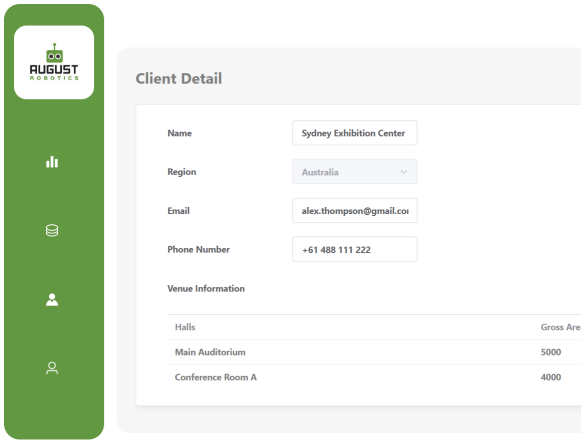
	made to specific hall marking jobs			User select 'View History' option	Display a graph showing all the changes made to the hall	NOT IMPLEMENTED	
Fixed Data	As an Administrator/data maintainer, be able to add the client data and venue data as the fixed data to the database	16.1	Given I am logged in as an administrator /data maintainer, when some client data needs to be added to the database as fixed data, then the client data can be able to used by other parts.	User navigate to client list page	User manages to navigate to page	PASS	
				User click *"Add New Client"*	Add new user widget is opened	PASS	
				User input data for new client	All fields can be saved	PASS	

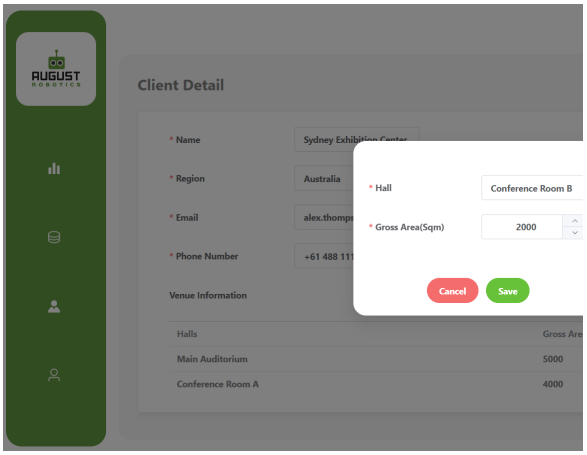
				User click *"Save Client"*	Clients gets saved into the system and can be viewed	PASS	
		16.2	Given I am logged in as an administrator /data maintainer, when some venue data needs to be added to the database as fixed data, then the venue data can be able to used by other parts.	User navigate to client detail page	User manages to navigate to page	PASS	 

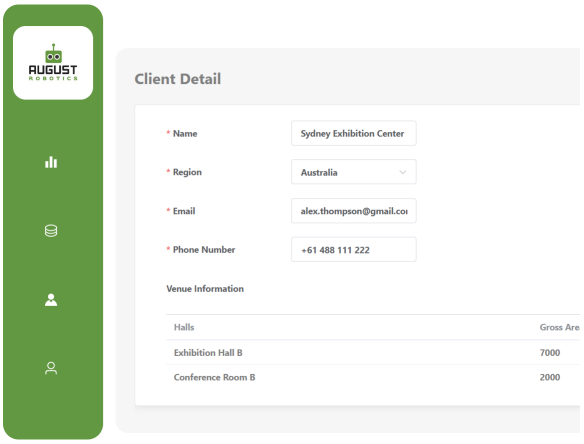
			User click *"Add Halls"*	The edit widget is opened	PASS	
			User input data for venue	New data can be inputted	PASS	
			User click *"Save"*	Additional halls are added to the venue data.	PASS	

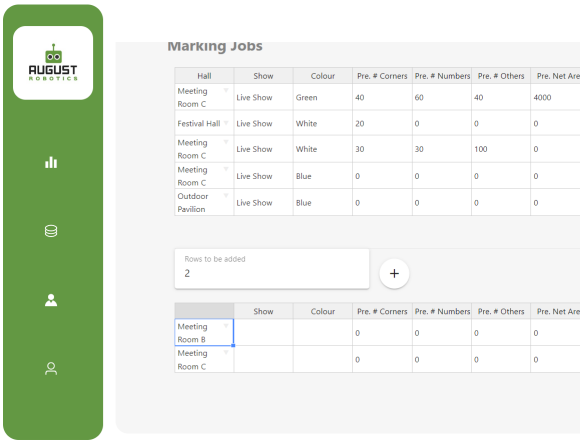
As an Administrator/data maintainer, be able to know the data of the client stored in the database	17.1	Given I am logged in as an administrator /data maintainer, when some venue data needs to be added to the database as fixed data, then t he venue data can be able to used by other parts.	User navigate to client list page	User manages to reach client list page.	PASS	
			User click operation button for one client	The client's operations widget pops up.	PASS	
			User click *"View Detail"* option in pop-up menu	User is now on the client details page for that client.	PASS	

As an Administrator/data maintainer, be able to edit the data of the client in the database	18.1	Given I am logged in as an administrator /data maintainer, when the data of client needs to be updated, then I can edit the client data stored in the database.	User click operation button for one client	The client's operations widget pops up.	PASS	
			User click *"Edit Client"* option in pop-up menu	User is redirected to the client's edit page.	PASS	
			User update client data	User can alter any field.	PASS	

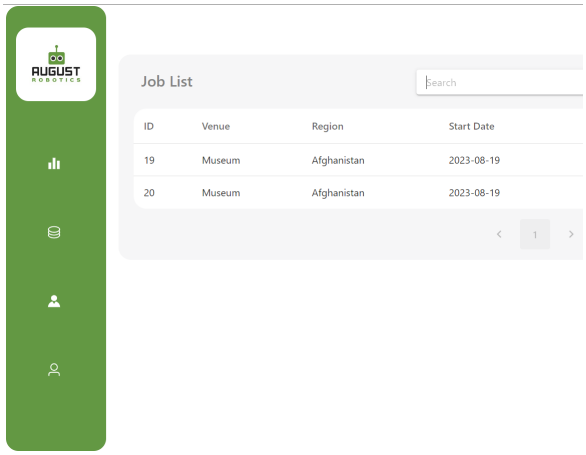
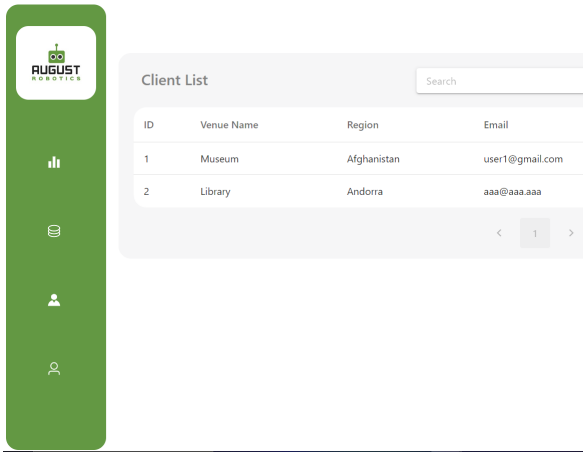
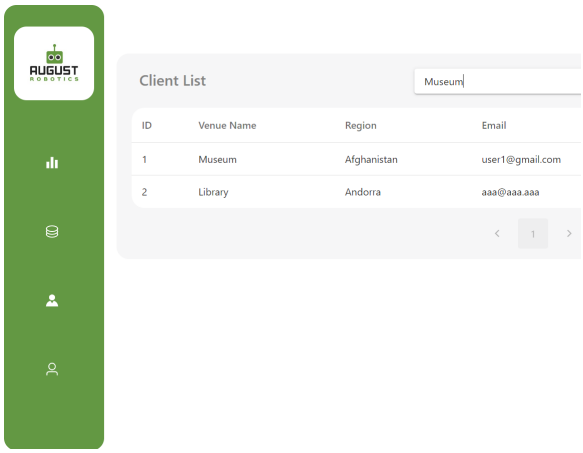
				User click *"Save"*	The changes are saved into the database and is now viewable on client details page.	PASS	 
		18.2	Given I am logged in as an administrator /data maintainer, when the data of client needs to be updated, then the other areas that use the modified data will be changed accordingly after I edit the data of client.	(*Possible to test only after operational job data has been created) User edits Client details	The changes are showed on the job list page for a job done for the same client that was updated	PASS	

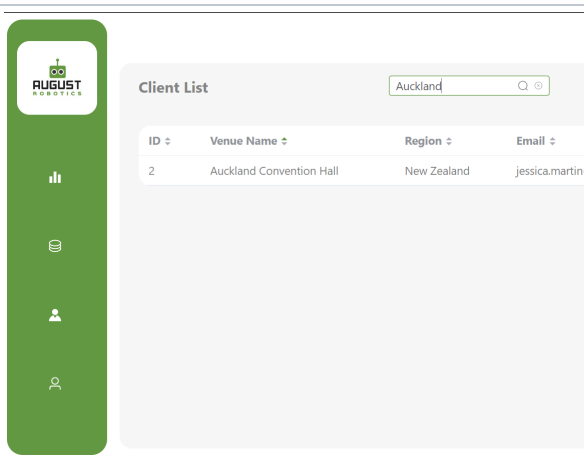
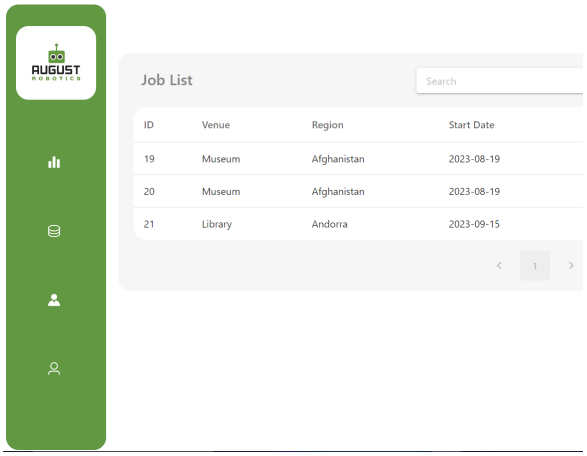
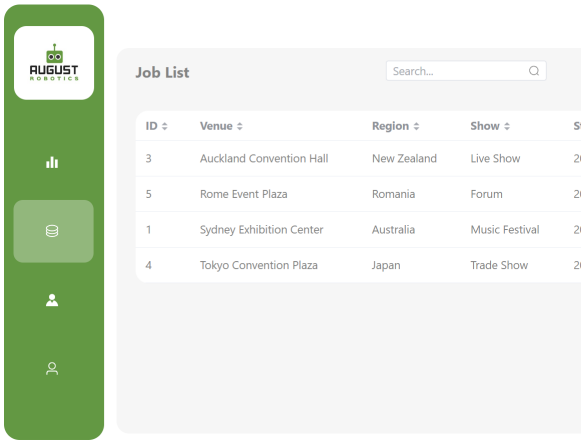

Client Detail
* Name
Sydney Exhibition Center
* Region
Australia
* Email
* Phone Number
+61 488 111
Venue Information
Halls
Gross Are
Main Auditorium
5000
Conference Room A
4000
* Hall
Conference Room B
* Gross Area(Sqm)
2000
Cancel
Save

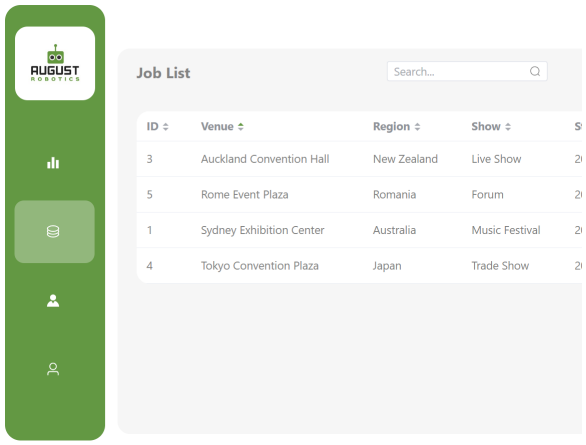
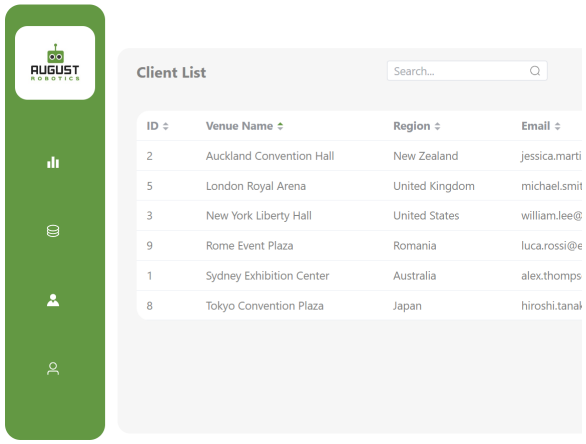

Client Detail
* Name
Sydney Exhibition Center
* Region
Australia
* Email
alex.thompson@gmail.co
* Phone Number
+61 488 111 222
Venue Information
Halls
Gross Are
Exhibition Hall B
7000
Conference Room B
2000


Marking Jobs
Hall
Show
Colour
Pre. # Corners
Pre. # Numbers
Pre. # Others
Pre. Net Are
Meeting Room C
Live Show
Green
40
60
40
4000
Festival Hall
White
20
0
0
0
Meeting Room C
White
30
30
100
0
Meeting Room C
Blue
0
0
0
0
Outdoor Pavilion
Blue
0
0
0
0
Rows to be added
2
Meeting Room B
Meeting Room C

Import and export data	As an administrator/ data maintainer, I want to be able to import the job duration, number of marks, and other relevant information, so that I can import existing data into the system easily	19.1	Given I am a logged in as an administrator /data maintainer and selecting the selected job import page, when a job import page is opened, then I am able to input the job data in different data through excel sheet.	User selects 'Job import page'	Job import page will be opened	NOT IMPLEMENTED	*Not required as an editable table allows for seamless copy and paste of all data e
				User inputs the required data	Data is added through excel sheet	NOT IMPLEMENTED	*Not required as an editable table allows for seamless copy and paste of all data e
		19.2	Given I am a logged in as an administrator /data maintainer and have imported a correct job excel sheet, when the save button is clicked, then the job data is imported correctly and has been saved in the database.	User navigates to the imported required job excel sheet	Displays the Job excel sheet	NOT IMPLEMENTED	*Not required as an editable table allows for seamless copy and paste of all data e
				User clicks the save button	Job data should be imported and saved in the database	NOT IMPLEMENTED	*Not required as an editable table allows for seamless copy and paste of all data e
		19.3	Given I am a logged in as an administrator /data maintainer and have imported a job excel sheet that lacks key data or contains incorrect data (such as incorrect data type), when The job Excel sheet has been verified by the system, then the job data can't be saved in the database and an error notification will be sent to users	User navigates to the imported job excel sheet that is inconsistent	Displays the Job excel sheet	NOT IMPLEMENTED	*Not required as an editable table allows for seamless copy and paste of all data e
				User clicks the save button	Job data should not be saved and an error message should be displayed	NOT IMPLEMENTED	*Not required as an editable table allows for seamless copy and paste of all data e
	As a Database Platform user, I want to be able to export the data in Excel spreadsheet format, so that I can easily share	20.1	Given I am logged in as an administrator /data maintainer and viewing the dashboard, when the export Excel button is clicked, then I am able to see the browser is downloading the spreadsheet	User navigates to the dashboard	Dashboard is displayed	NOT IMPLEMENTED	

	the information with others and analyse it further.			User clicks export excel button	Spreadsheet should be downloaded	NOT IMPLEMENTED	
Searching	As a Database Platform user, I want to be able to filter, sort and search the data based on various criteria such as client, region, or date, so that I can quickly find the data I need	21.1	Given I am logged in as an administrator /data maintainer and viewing the dashboard, when I select keywords such as client name and venue name or time range, and click the filter button, then I am able to see and select the results filtered based on the keywords I select	User navigates to the dashboard	Dashboard is displayed	PASS	
				User selects keyword filter such as 'client name', 'venue', etc	Data based on the selected keyword should be displayed	PASS	*Filtering done through search
				User clicks filter button	Data based on the selected keyword should be displayed	PASS	*Filtering done through search
		21.2	Given I am logged in as an administrator /data maintainer and viewing the dashboard, when I type keywords such as client name and venue name, and click the search button, then I am able to see and select the results searched based on the keywords I type	User navigates to the dashboard	Dashboard is displayed	PASS	
				User types keywords such as 'client name', 'venue', etc	words displayed	PASS	

			User clicks search option	Data will be displayed based on the keywords typed	PASS	
	21.3	Given I am logged in as an administrator /data maintainer and viewing the dashboard, when Click one of the sorting buttons(sort by date, sort by alphabet), then I am able to see the results sorted based on the criteria I chose	User navigates to the dashboard	Dashboard is displayed	PASS	
			User clicks sort by data option	Data is displayed based on date	PASS	

				User clicks sort by alphabet	Data is displayed based on alphabet	PASS	 
Admin User Management	As an administrator, I want to be able to create a new account for a user, so that a new employee can use the system	22.1	Given I am logged in as an administrator, when the user management page is opened, then I can create a new user.	User navigates to user management page	user management page is opened	NOT IMPLEMENTED	
				User clicks create new user option	New user is created	NOT IMPLEMENTED	
		22.2	Given I am creating a new user, when the user management page is opened, then I can set the user name, email address, password, and user (privilege) type for new users.	User navigates to user management page	User management page is opened	NOT IMPLEMENTED	
				User can set the user name for new user created	User name data can be added	NOT IMPLEMENTED	
				User can set the email address new user created	Email address data can be added	NOT IMPLEMENTED	
				User can set the password new user created	Password data can be added	NOT IMPLEMENTED	
				User can set the privilege new user created	Privileges option can be added	NOT IMPLEMENTED	
		22.3	Given I am creating a new user, when user name or mailbox already exists, then the user can not be created.	User adds existing email address while creating new account	User cannot be created and show mail address already exists	NOT IMPLEMENTED	

	As an administrator, I want to be able to delete an existing account, so that in active accounts can be removed from the system	23.1	Given I am logged in as an administrator, when the user management page is open, then I can delete users whose type of privilege is job data maintainer or viewer.	User navigates to user management page	User management page is opened	NOT IMPLEMENTED	
				User clicks delete option for users with maintainer or viewer privileges	User will be deleted	NOT IMPLEMENTED	
	As an administrator, I want to be able to edit an existing user, so that a user's details can be changed.	24.1	Given I am viewing an account in the user management page, when I click edit, then I can change the user's permission level.	User navigates to the user management page	User management page is displayed	NOT IMPLEMENTED	
				User clicks edit option	Display user permission level change options	NOT IMPLEMENTED	
				User changes the permission level	Permission level will be updated	NOT IMPLEMENTED	

Unit Tests Plan

Unit test - Each individual piece of code such as methods, and classes should be tested.

The programmer will write a unit test to test small pieces of code to test their functionality and check if it is working as intended to do.

(Sample outline)

Goal

- To define the correct behavior of the system so that later development won't affect previous components.
- To regulate the process of CI.
- To limit the ability of logical bugs from merging into the deployed code

Scope

- For each feature that we implement, create a corresponding unit test.
- Each unit test starts from an empty scene and dedicatedly runs through the feature to assert desirable outcomes.

Schedule

Whenever someone finishes implementing a new feature, he/she would write the unit test and confirm that the test passes before starting a pull request. This decision was made to:

- Simplify the planning process.
- Individual assignments for testing tasks are not required, which helps shorten the length of Sprint planning sessions.
- Ensure that tests are written correctly as specified. The code author should have a much better understanding of the requirements and the developed feature, making them a suitable candidate to produce test cases that can detect any deviation from the specification in the future.
- Should the test cases be insufficient or deviate from the intended specifications, the reviewer can raise the issue with the author to rectify it.

Personnel Involved

- 10 x Developers
    - All developers are expected to produce test cases for the features that they have developed.

Writing Unit Tests

Generally, the steps to follow would be:

1. Write a test for each method.
2. Check for data flow faults.
3. If the method has if/else statements, write a test for each block of the statement.

    a. Write a test with preconditions that satisfy the first if.
    b. Write a test with preconditions that satisfies each else if.
    c. Write a test that satisfies the else condition.
4. Write tests to check for boundary conditions. (I.e. null parameters, index out-of-bounds, exceptions etc.).

Tool

(TO BE ADDED)

Unit Test Coverage Report

(TO BE ADDED)

Sample Unit Test Summary

(DATA TO BE ADDED)

Unit Test ID		**Tester(s)**	
Tested Unit		**Test Date**	
Description		**Test File**	
Jira Issue		**Test Function**	
Pre-requisite		**Post-requisite**	

Test Cases

(DATA TO BE ADDED)

Test Case ID	Test Case Description	Input Values	Expected Outputs	Actual Outputs	Test Result	Comments

Unit Tests

Tests were carried out according to the Unit Tests Plan. Tests were written using pytest, and can be found on the under relevant `/tests` folders.

User Registration

Test Case ID	Test Case Description	Input Values	Precondition	Expected Outputs	Actual Outputs	Test Result	Comments	Date Tested	Tester Name
1.1	Registering a new user with valid username and password.	{'username': 'test', 'password': 'test'}	N/A	User is successfully registered	User is successfully registered	PASS	-	08 Sep 2023	Andrew Liu
1.2	Registering a new user with invalid username (length > 32)	{'username': 'longusername1111\ 1111111111111111111111111111\ 1111111111111111111111111111\ 111111111', 'password': 'test'}	N/A	Request is denied User is not registered	Request is denied User is not registered	PASS	-	08 Sep 2023	Andrew Liu
1.3	Registering a username which already exists.	{'username': 'test', 'password': 'test'}	1. 1	Request is denied User is not registered	Request is denied User is not registered	PASS	-	08 Sep 2023	Andrew Liu
1.4	Registering with no password	{'username': 'test', 'password': ''}	N/A	Request is denied User is not registered	User is registered successfully	FAIL	Password should be checked in the back-end.	08 Sep 2023	Andrew Liu
1.5	Registering with no username	{'username': '', 'password': 'test'}	N/A	Request is denied User is not registered	User is registered successfully	FAIL	Username should be checked in the back-end.	08 Sep 2023	Andrew Liu

User Login

Test Case ID	Test Case Description	Input Values	Precondition	Expected Outputs	Actual Outputs	Test Result	Comments	Date Tested	Tester Name
2.1	Logging in with an invalid username	{'username': 'incorrect_test', 'password': 'test'}	1.1	Request is denied User is not logged in	Request is denied User is not logged in	PASS	-	07 Sep 2023	Andrew Liu
2.2	Logging in with invalid password	{'username': 'test', 'password': 'incorrect_test'}	1.1	Request is denied User is not logged in	Request is denied User is not logged in	PASS	-	07 Sep 2023	Andrew Liu
2.3	Logging in with valid username and password	{'username': 'test', 'password': 'test'}	1.1	Request is successful, A valid token is sent back in the response.	Request is successful, A valid token is sent back in the response.	PASS	-	07 Sep 2023	Andrew Liu

Client

Test Case ID	Test Case Description	Input Values	Precondition	Expected Outputs	Actual Outputs	Test Result	Comments	Date Tested	Test Name
3.1	Adding a valid new client	"client_name": "client1", "email": "client1@email.com", "phone": "1234567890"	2.3	Client is added to the database.	Client is added to the database.	PASS	-	08 Sep 2023	Andrew Liu
3.2	Adding an empty client	{}	2.3	Request is denied. Client is not added to database.	Request is denied. Client is not added to database.	PASS	-	08 Sep 2023	Andrew Liu

3.3	Updating a valid client	"client_name": "client1", "email": "client2@email.com", "phone": "0987654321"	2.3	Client details are updated.	Client details are updated.	PASS	-	08 Sep 2023	Andrew Liu
3.4	Updating a client which doesn't exist	{ "client_name": "client2", "email": "client2@email.com", "phone": "0987654321" }	2.3	Request is denied.	Request is denied.	PASS	-	08 Sep 2023	Andrew Liu
3.5	Updating a client with no fields	{}	2.3	Request is denied.	Request is denied.	PASS	-	08 Sep 2023	Andrew Liu
3.6	Getting a valid client	/client/update/1	2.3	Client details are returned.	Client details are returned.	PASS	-	08 Sep 2023	Andrew Liu
3.7	Getting an client which doesn't exist	/client/update/5	2.3	Request is denied.	Request is denied.	PASS	-	08 Sep 2023	Andrew Liu
3.8	Deleting a valid client	{id: "1"}	2.3	Client is successfully deleted.	Client is successfully deleted.	PASS	-	08 Sep 2023	Andrew Liu
3.9	Deleting a client which doesn't exist	{}	2.3	Request is denied.	Request is denied.	PASS	-	08 Sep 2023	Andrew Liu

Job

Test Case ID	Test Case Description	Input Values	Precondition	Expected Outputs	Actual Outputs	Test Result	Comments	Date Tested	Tester Name
4.1	Listing jobs	{ "j_client":client_obj.id, "start_date":"2023-09-15", "end_date":"2023-09-18", "marking_days":"3", "status":"In Progress" }	3.1	Job is listed in response.	Job is listed in response.	PASS	-	14 Sep 2023	Andrew Liu
4.2	Adding a job	{ "j_client":client_obj.id, "start_date":"2023-09-15", "end_date":"2023-09-18", "marking_days":"3", "status":"In Progress" }	3.1	Job is successfully added.	Job is successfully added.	PASS	-	14 Sep 2023	Andrew Liu

Hall

Introduction

This test plan covers the functionalities related to Hall management in the system. It includes CRUD operations and some essential setup fixtures to ensure prerequisites are met.

Because the Django Backend using JWT Token Authentication, we need to set up auth fixtures and assign a JWT token to the mockup client's request header.

Test Environment

- **Framework**: pytest, Django, DRF (Django Rest Framework)
- **Database**: Default database configured with Django.

Test Data Setup

Test Data Setup ID	Test Case Description	Input Values	Precondition	Expected Outputs	Actual Outputs	Test Result	Comments	Date Tested	Tester Name
TDS01	Admin Group Setup	None	Django and DRF are configured properly	'admin' group exists in the database	'admin' group gets created in the database	PASS	None	15 Sep 2023	Wei Zhao
TDS02	Admin Registration	username: "admin", password: "123456", privilege:" admin"	'admin' group exists	The admin user is registered	The admin user is registered	PASS	None	15 Sep 2023	Wei Zhao
TDS03	Admin Authentication	username: "admin", password: "123456"	Admin user exists	The bearer token is received	The bearer token is received and set on the api_client's request header	PASS	None	15 Sep 2023	Wei Zhao

Test Case ID	Test Case Description	Input Values	Precondition	Expected Outputs	Actual Outputs	Test Result	Comments	Date Tested	Tester Name
5.1	Adding a Hall to a specific client	hall="Test Hall," area=500, h_client=client_obj	client_obj = Client.objects.create(venue_name="Test Venue", region="Test Region", email="test@example.com", phone="1234567890")	hall_object -> the hall of the created hall status.HTTP_201_CREATED	status.HTTP_201_CREATED	PASS	None	15 Sep 2023	Wei Zhao
5.2	Get Hall Detail	hall_id of the created hall	Successfully created a hall in the previous step, 5.1	Details of the hall (name, area) are fetched correctly	Details of the hall (name, area) are fetched correctly	PASS	None	15 Sep 2023	Wei Zhao
5.3	Updating a Hall	Updated hall name: 'Updated Expo Hall'	Hall with the given hall_id exists	The hall's name in the database is updated to 'Updated Expo Hall'	The hall's name in the database is updated to 'Updated Expo Hall'	PASS	None	15 Sep 2023	Wei Zhao
5.4	Updating an invalid Hall	hall_id that does not exist in the database		get 404 error hall not found	get 404 error hall not found	PASS	None	15 Sep 2023	Wei Zhao
5.5	Delete Hall	hall_id of the created hall	Hall with the given hall_id exists	The hall is deleted successfully	The hall is deleted successfully	PASS	None	15 Sep 2023	Wei Zhao
5.6	Deleting an invalid job	hall_id that does not exist in the database		get 404 error hall not found	get 404 error hall not found	PASS	None	15 Sep 2023	Wei Zhao

Development & Deployment

This section includes Team R Development and Deployment guides, processes and procedures.

Environment Setup

Operating System Requirement

Minimum requirements :

- **Memory:** 4 GB
- **Graphics Card:** AMD Radeon R5 M230
- **CPU:** Intel Core i3-2340UE
- **File Size:** 3 GB
- **OS:** Windows 7,8,8.1

recommended specs :

- **Memory:** 8 GB
- **Graphics Card:** AMD Radeon R7 A10-7850K
- **CPU:** Intel Core i5-4400E
- **File Size:** 3 GB
- **OS:** Windows 10

Install Python :

- **Python version:** 3.8, 3.9, 3.10, 3.11
- **Install pip**

Install Vue:

- **Step 1:** Download the node.js from the official website: https://nodejs.org/en/download
- **Step 2:** Configure environment variables for node.js
- **Step 3:** Download the npm by the command:

```
npm install -g npm
```

- **Step 4:** Use Vite to create a Vue project. Input the following command:

```
npm create vite@latest
```

After that, you should input y to accept proceed, input the project name and select Vue and JavaScript.

- **Step 6:** Run this new project by the command:

```
cd project_name
npm run dev
```

Install Django :

Installation instructions are slightly different depending on whether you're installing a distribution-specific package, downloading the latest official release, or fetching the latest development version.

Installing an official release with pip:

```
$ python -m pip install Django
```

Installing a distribution-specific package:

Check the distribution specific notes to see if your platform/distribution provides official Django packages/installers. Distribution-provided packages will typically allow for the automatic installation of dependencies and supported upgrade paths; however, these packages will rarely contain the latest release of Django.

Many third-party distributors are now providing versions of Django integrated with their package-management systems. These can make installation and upgrading much easier for users of Django since the integration includes the ability to automatically install dependencies (like database adapters) that Django requires.

Install PostgreSQL:

- **Step 1 :** Download the latest PostgreSQL Version for Windows x86-64
- **Step 2 :** Setup PostgreSQL
- **Step 3 :** Select Components: PostgreSQL Server, pgAdmin4, Stack Builder, Command Line Tool
- **Step 4 :** Select Data Directory
- **Step 5 :** Set Password and Port Number
- **Step 6 :** Finish Installation

Setting Up A Database Server:

Step 1 : Create a database and define the database name

Step 2 : Find database setting in django back-end

- Navigate to settings.py
- Find a dictionary variable called “DATABASES”

Step 3 : Change items in DATABASES to corresponding parameters :

- **ENGINE** 'django.db.backends.postgresql_psycopg2'
- **NAME** Database name e.g. dbtest previously created in pgAdmin
- **USER** Database username (default is **postgres**)
- **PASSWORD** Database password
- **HOST** Database host (In development stage, use **localhost** or IP Address **127.0.0.1** also available)
- **PORT** The port that used to run the database (Default port is **5432**)

Start the Django Server:

To serve Django application over the local network, run this command :

```
python manage.py runserver 0.0.0.0:8000
```

Alternatively, you can use the IP address of the host machine on the Public network.

Copy of README.md on GitHub

Front end Setup:

Recommended IDE Setup

VSCode + Volar (and disable Vetur) + TypeScript Vue Plugin (Volar).

Type Support for `.vue` Imports in TS

TypeScript cannot handle type information for `.vue` imports by default, so we replace the `tsc` CLI with `vue-tsc` for type checking. In editors, we need TypeScript Vue Plugin (Volar) to make the TypeScript language service aware of `.vue` types.

If the standalone TypeScript plugin doesn't feel fast enough to you, Volar has also implemented a Take Over Mode that is more performant. You can enable it by the following steps:

1. Disable the built-in TypeScript Extension
    a. Run `Extensions: Show Built-in Extensions` from VSCode's command palette
    b. Find `TypeScript and JavaScript Language Features`, right click and select `Disable (Workspace)`
2. Reload the VSCode window by running `Developer: Reload Window` from the command palette.

Customize configuration

See Vite Configuration Reference.

Project Setup

```
npm install
```

Compile and Hot-Reload for Development

```
npm run dev
```

Type-Check, Compile and Minify for Production

```
npm run build
```

Lint with ESLint

npm run lint

Back end Setup

Option 1 : Run the django program on localhost

Install Django and related dependencies

```
## make sure you are in the working directory Django-backend/

python -m venv venv

## Create virtual environment

## for windows
source venv/Scripts/activate

## for mac os
source venv/bin/activate

pip install -r requirements.txt
```

Setting Up A Database Server

Step 1 : Open the settings

Go to the following directory AugustRoboticsBackend/settings.py

Step 2 : Edit settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',

        ## Optional, you can change the database name to whatever you want
        'NAME': 'db_augustRobotics',
        'USER': 'weizhao1',
        'PASSWORD': 'pass4teamR!',
        'HOST': 'backend-data.postgres.database.azure.com',
        'PORT': 5432,

    }
}
```

Step 3 : If you set the database name, please create a database with the same name using pgAdmin4 or psql command line tool

Step 4 : Migrage the database to localhost

```
python manage.py migrate
```

Start the Django Server directly on localhost

Note:

Change the 'HOST' to 'localhost' to start the Django Server

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'db_augustRobotics',
        'USER': 'weizhao1',
        'PASSWORD': 'pass4teamR!',
        'HOST': 'backend-data.postgres.database.azure.com',
        'PORT': 5432,

    }
}
```

```
python manage.py runserver
```

Option 2 : Start the Django app with Docker

If you are not familiar with docker or have not installed docker yet, feel free to check the WiKi for Docker

You can use this Docker file to build a Docker image for your Django project. To build the image, navigate to the directory that contains the Docker file and enter the following command:

Note:

when using Docker Compose, the services are running in separate containers, and you should use the service name as the hostname for connecting to the database. In this case, the service name is db.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'db_augustRobotics',
        'USER': 'weizhao1',
        'PASSWORD': 'pass4teamR!',
        'HOST': 'backend-data.postgres.database.azure.com',
        'PORT': 5432,

    }
}
```

```
cd Django-backend

# This will create a Docker image named django-backend.
docker compose up -d

# Then run the docker script
./build-project-docker.sh
```

That's it! Now we have a Docker container running Django app.

```
# API Documentation available at the localhost:8000 once the backend is running
http://localhost:8000/
```

How to write the pytest and run the tests?

go to the directory Django-backend/AugustRoboticsBackend/tests/

write the tests for your endpoint

for example, we have a endpoint on 'localhost:8000/' we could test the endpoint by the following code

```
@pytest.mark.django_db
def test_home_page(client):

    ## you could use reverse or replace with the whole url:
    ## url = '/'
    url = reverse('hello_world')
    response = client.get(url)
    assert response.status_code == 200
```

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.hello_world, name='hello_world'),  # The name of the URL pattern is 'home_view'
]
```

Run the pytest

```
## run on your local machine
pytest


## docker
docker compose exec -T backend pytest
```

GitHub Branching and PR

GitHub Flow Branch Strategy

1. **Main branch (master or main):** This branch represents the stable production version of the project. It should always contain production-ready code.

2. **Development branch (e.g., develop):** This branch serves as the integration branch for ongoing development. It is where **Feature branches** are merged when they are ready for testing and integration.

3. **Feature branches:** Each new feature or task should have its own branch created off the **Development branch**. These branches are prefixed with a feature/ or task/ prefix., team member creating the branches should also have their name embedded after the feature/ and the purpose of this branch should have a descriptive name (By also adding what part the code is changing frontend/backend). For example:

- `feature/member1/front-end-user-authentication`
- `task/member2/back-end-implement-payment-gateway`

Feature branches allow individual team members to work on their specific features or tasks without disrupting others.

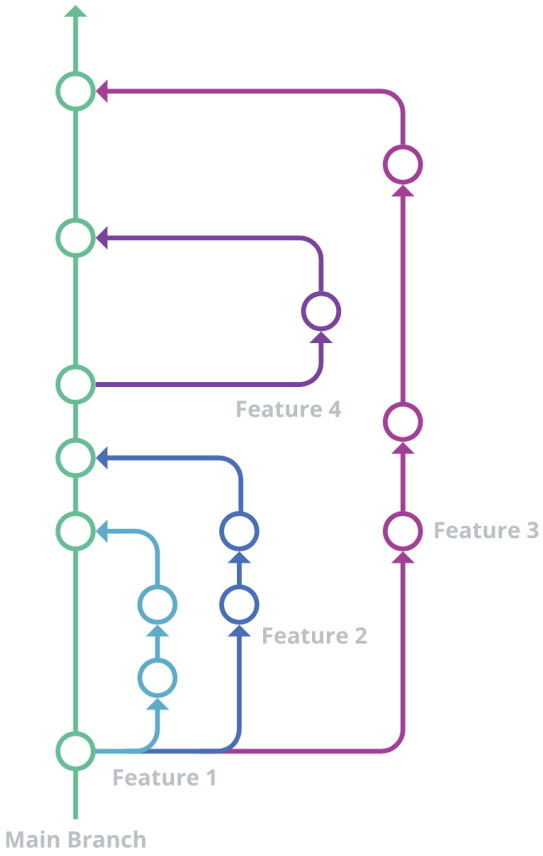
4. **Pull Requests:** Once a feature is completed or a task is finished, the developer opens a pull request to merge their branch into the **Development branch**. The pull request allows for code review, discussion, and testing before merging.

5. **Code Review and Testing:** Team members review the code changes in the pull request, provide feedback, and perform testing to ensure the quality and functionality of the code.

6. **Merge to Development:** After the code has been reviewed and approved, the **Feature branch** is merged into the **Development branch**.

7. **Release Branches (optional):** If you want to maintain separate branches for specific releases, you can create release branches off the **Development branch**. This allows for stabilization, testing, and preparing the code for deployment.

8. **Hotfix Branches (optional):** If a critical bug is discovered in the production code, a hotfix branch can be created off the **Main branch**. This branch allows for quick fixes that can be merged back into both the main and development branches.



Custom image inspired by the GitHub Flow Guide.

Pros of GitHub Flow

There are some benefits to using the GitHub flow:

- Compare with other Git branch strategies, GitHub is the most simple one.
- Because of the simplicity of the workflow, this Git branching strategy allows for Continuous Delivery and Continuous Integration.
- This Git branch strategy works great for small teams and web applications.

Deployment Guide

Heroku Deployment Guide for August Robotics Job Database Management Software Platform

1. Introduction:

This guide provides step-by-step instructions for deploying the August Robotics Job Database Management Software Platform on Heroku using a GitHub repository.

2. Pre-requisites:

- A Heroku account.
- A GitHub account with the project repository.
- Heroku CLI is installed on your local machine.

3. Configuration:

- **Environment Variables**: Set up environment variables in Heroku by navigating to the "Settings" tab in your Heroku app dashboard, and under the "Config Vars" section, add your environment variables.
- **Domain and SSL**: Configure your domain and SSL under the "Settings" tab in your Heroku app dashboard to ensure secure HTTP connections.
- **Database Configuration**: Ensure your `DATABASE_URL` environment variable is set up with the correct database credentials.

4. Deployment Strategy for Frontend and Backend on Heroku

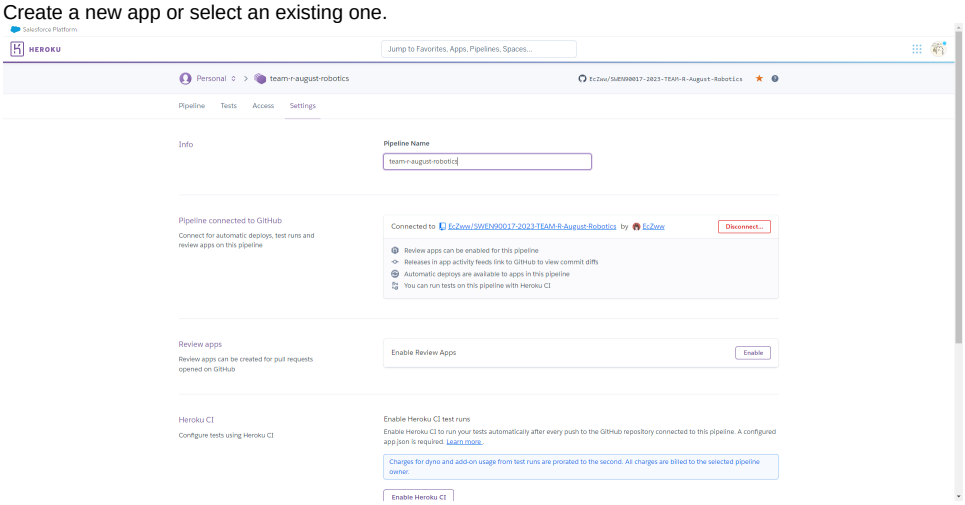
Given the constraint that Heroku does not support deploying both frontend and backend applications from a single branch with a `Procfile`, a strategic approach is adopted to facilitate the deployment of both applications. The strategy is outlined as follows:

1. **Branch Segregation**:
    - The project repository is structured such that it houses both the frontend and backend code. However, to conform to Heroku's deployment mechanism, the deployment process is segregated based on branches.
    - The `main` branch is designated for the frontend application, while a separate branch named `backend-deployment` is designated for the backend application.
2. **Deployment Process**:
    - **Frontend Deployment**:
        - The frontend application is deployed directly from the `main` branch to Heroku. Whenever changes are pushed to the `main` branch, Heroku triggers a deployment process for the frontend application, ensuring that the live frontend application is always in sync with the latest code in the `main` branch.
    - **Backend Deployment**:
        - The backend application is deployed from the `backend-deployment` branch to Heroku. Similar to the frontend, any pushes to the `backend-deployment` branch trigger a deployment process for the backend application on Heroku, ensuring the live backend application reflects the latest backend code.
3. **Procfile Utilization**:
    - Separate `Procfile` entries are utilized for each application to specify the commands needed to start the frontend and backend applications on Heroku.
    - For the frontend, the `Procfile` in the `main` branch contains the necessary command to serve the Vue.js application.
    - For the backend, the `Procfile` in the `backend-deployment` branch contains the necessary commands to handle Django migrations and to start the Django application using Gunicorn.

5. Deployment Process:

- **Connect GitHub Repository**:
    1. Log in to your Heroku account and navigate to the dashboard.

2. Create a new app or select an existing one.



3. Go to the "Deploy" tab, and under "Deployment method", choose "GitHub".
4. Connect GitHub account (if not already connected) and search for the repository containing your project.
5. Click "Connect" to connect the repository to your Heroku app. https://github.com/EcZww/SWEN90017-2023-TEAM-R-August-Robotics

- **Create New App in Pipeline:**

  In Heroku, a pipeline is a group of Heroku apps that share the same codebase. Each app in a pipeline represents one of the following stages in a continuous delivery (CD) workflow:

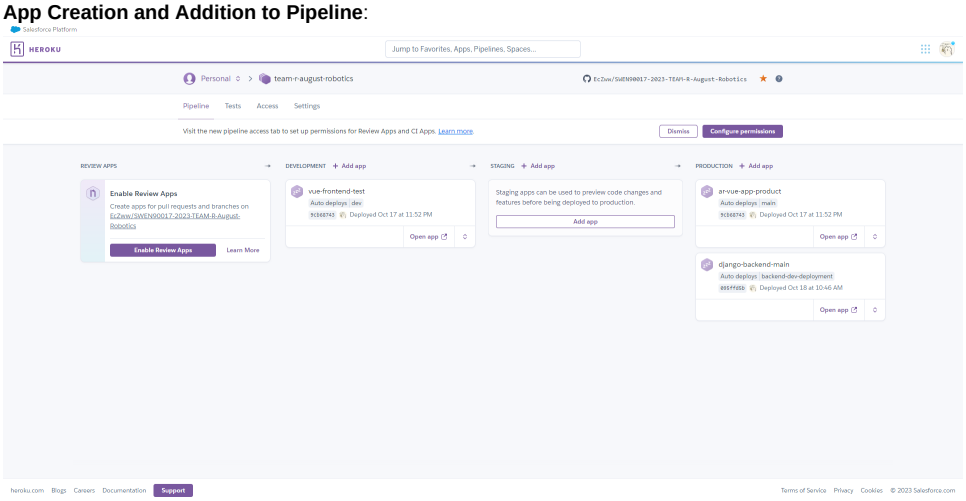
  1. **Review**
  2. **Development (dev)**
  3. **Staging**
  4. **Production (main)**

Here's how to create a new app within a pipeline and deploy different branches to different stages:

i. **Pipeline Creation**:

- Log into Heroku dashboard and click "New" at the top right, then select "Create new pipeline."
- Enter a name for your pipeline and connect it to the corresponding GitHub repository.

ii. **App Creation and Addition to Pipeline**:

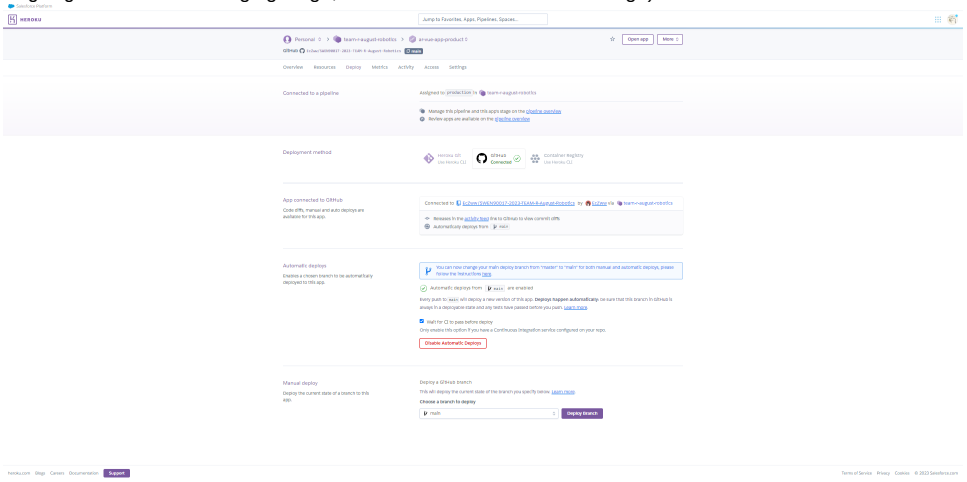


- Under your pipeline, click "Add app" for each stage (dev, staging, main).
- If you already have Heroku apps created, you can choose to add them to the pipeline. Otherwise, create new apps for each stage.

2. **Deploying Different Branches**:

- Under the "Deploy" tab of each app, connect to GitHub (if not already connected).

- Choose the GitHub branch that corresponds to the stage of the app (e.g., a `dev` branch for the Development stage, a s`taging` branch for the Staging stage, and `main` for the Production stage).



- Enable "Automatic Deploys" so that any push to the specified branch will trigger a deployment to the corresponding stage in the pipeline.

- **Deploy the App**:

    1. Under the "Manual deploy" section, choose the branch you want to deploy and click "Deploy Branch."
    2. Optionally, enable "Automatic Deploys" for your chosen branch to have Heroku automatically deploy your app whenever changes are pushed to that branch on GitHub.

6. Post-Deployment Checks:

- Verify successful deployment by checking the activity feed in your Heroku dashboard.
- Test your app's functionality to ensure it operates as expected on Heroku.

7. Maintenance and Management:

- Monitor application performance using Heroku's built-in monitoring tools.
- Rotate credentials and manage user access through the Heroku dashboard.

8. Troubleshooting:

- Check the logs in Heroku dashboard or use the command `heroku logs --tail` in the terminal to troubleshoot any issues.

9. Appendix:

The `Procfile` is crucial for instructing Heroku on how to run applications on its platform. It allows for a straightforward definition of various process types and commands necessary to start Vue.js and Django applications, aiding in the automated deployment and management of apps on Heroku.

- Backend Django Heroku Procfile
    - ```
      release: python Django-backend/manage.py migrate

      web: gunicorn --chdir Django-backend AugustRoboticsBackend.wsgi:application
      ```
- Frontend Vue Heroku Procfile
    - `web: npx serve -s Vue-frontend/dist -l ${PORT:-5000}`

10. References:

- Heroku Documentation
- GitHub Documentation

Once deployed, the '**Open app**' button at the top right corner can be used to view the app.
      ```

Backend API Documentation

API Documentation Enhancement Using `drf-yasg`

The static html API documentation for Django backend applications is available on localhost:8000/redocs/ when running the backend via docker or locally.

`drf-yasg` is a great tool to generate real-time OpenAPI/Swagger specifications for Django Rest Framework applications. It provides a user-friendly interface to navigate and test your API endpoints, making it easier for frontend developers and other stakeholders to understand and interact with your backend services.

- Automate document generation via docstring
- provide a unified document style
- improve document maintainability and readability, and provide frontend developers with a better document browsing experience.
- Increase project understandability and maintainability.

docstring Example

```
def add_client(request):
    """
    This function is used to add a new client to the system.

    **Endpoint:** `/client/add`

    **Method:** `POST`

    :param request: Django HttpRequest object
    :type request: HttpRequest
    :return: JsonResponse containing client creation status and client details
    :rtype: JsonResponse

    **Example**::

        sample request:

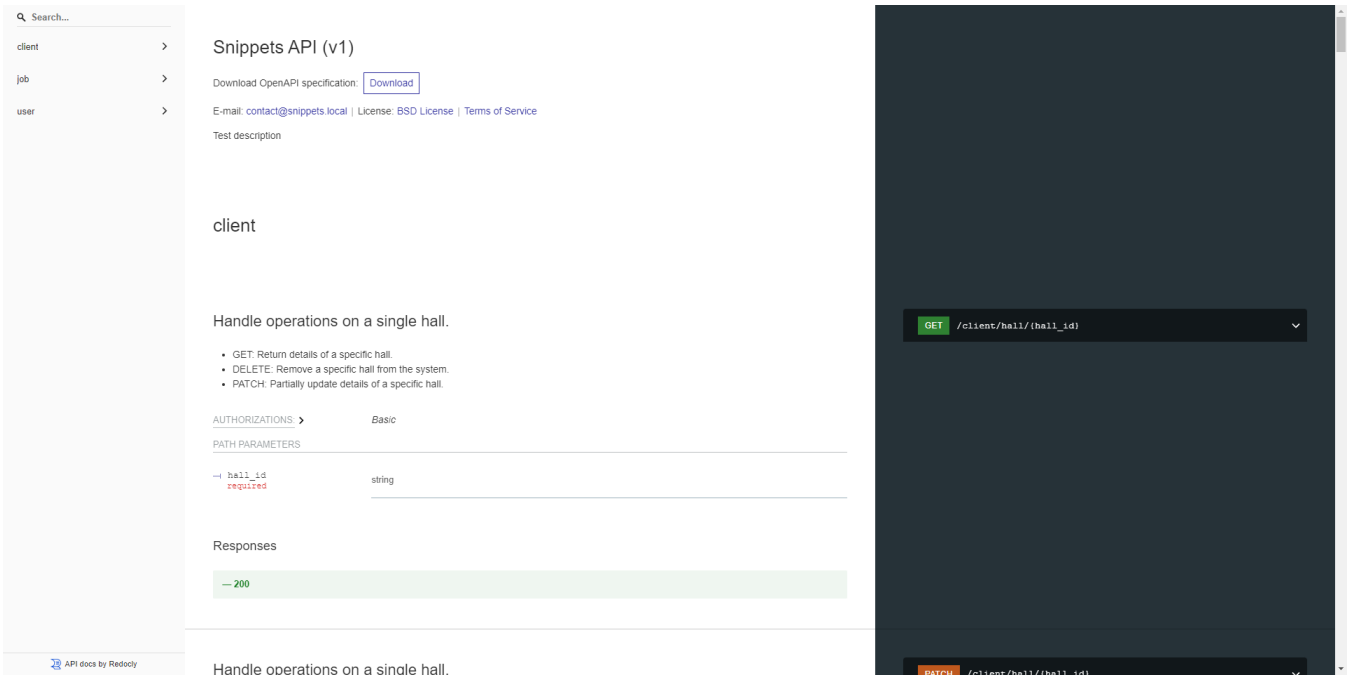
        .. code-block:: json

            {
                "client_name": "client1",
                "email": "client1@email.com",
                "phone": "1234567890"
            }

        if the function is successful, it will return something like this:

        .. code-block:: json

            {
                "success": "Client created successfully.",
                "client": {
                    "client_name": "client1",
                    "email": "client1@email.com",
                    "phone": "1234567890"
                }
            }
    """
...The implementation of the function...
```


Search...
client
job
user
Snippets API (v1)
Download OpenAPI specification:
Download
E-mail: contact@snippets.local | License: BSD License | Terms of Service
Test description
client
Handle operations on a single hall.
GET: Return details of a specific hall.
DELETE: Remove a specific hall from the system.
PATCH: Partially update details of a specific hall.
AUTHORIZATIONS:
Basic
PATH PARAMETERS
hall_id
required
string
Responses
200
API docs by Redocly
GET /client/hall/{hall_id}
Handle operations on a single hall.

Backend database connection

In this project, we use the PostgreSQL database service provided by Microsoft Azure.

Connect to the database in backend via settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'test-dev', ## the name of database
        'USER': 'weizhao1', ## change 'postgres' to your PostgreSQL username
        'PASSWORD': 'pass4teamR!',  ## change 'postgres' to your PostgreSQL password
        'HOST': 'backend-data.postgres.database.azure.com', ## change 'db' to 'localhost' if you are not using
docker
        'PORT': 5432,
    }
}
```

View the database through pgadmin4

Please follow these steps to connect to an Azure Database for PostgreSQL - Flexible Server using pgAdmin 4:

1. **Open pgAdmin 4**: Launch the pgAdmin 4 application on your computer.
2. **Register a new server**: In the pgAdmin 4 interface, right-click on "Servers" in the left-side browser tree, and select "Register" -> “Server"
3. **Configure server details**: In the "Register - Server" window, you will see multiple tabs - "General", "Connection", "SSL", and others. Fill in the following details:
    a. General tab
        i. **Name**: Provide a name for the connection (e.g., "myAzureFlexInstance").
    b. Connection tab
        i. **Hostname/address**: Enter **backend-data.postgres.database.azure.com**
        ii. **Port**: Leave the port number as is (default is 5432) if you don't want to connect through pgBouncer. If you are using PgBouncer for connection pooling, change the port number to 6432.
        iii. **Maintenance database**: `test-dev`
        iv. **Username**: Enter **weizhao1**
        v. **Password**:

            pass4teamR!
4. **Save the configuration**: Click the "Save" button to save the server registration. pgAdmin 4 will now establish a connection to your Azure Database for PostgreSQL Flexible Server.
5. **Access the database**: Once connected, you can expand the server in the browser tree to view databases, schemas, and tables. You can also interact with the server using the built-in query tool and manage your database objects.

Handover Plan

Date

We will be conducting our handover on the **13th November, 2023** because that is the only time our client is available.

Process

We will allow questions through out this meeting.

1. We will set up a zoom meeting with our client and provide an overview of what this meeting entails. We will mention the technologies used in our product as a reminder.
2. We will then go very briefly go over some diagrams we have made, so that they can have a higher level understanding of our product.
    a. This will include our Architecture Diagram, Entity Relationship Diagram, and Component Diagram.
    b. Only the briefest overview is to be given, just so they can have an idea of how we structured the app.
3. We will provide the link to our app via Zoom chat.
4. We will first be going over demo over our product to show what has been completed throughout the course of the year.
5. We will mention bugs that are still present in the app.
6. We will go over our Deployment Guide
    a. We intend to give a brief summary, we will encourage them to read it thoroughly later.
7. We will provide our Github repository to them. We will also be giving them access and ownership over it.
8. Next, an email will be sent containing all our relevant documents to them and we will make sure they received it.
    a. Documents Include:
        i. Architecture Diagram
        ii. Entity Relationship Diagram
        iii. Component Diagram.
        iv. Domain Diagram
        v. Use Case Diagram
        vi. Sequence Diagram
        vii. Communication Diagram
        viii. Deployment Guide
9. We will have Q&A now for any lingering questions.