**ISYS3888 - Information Systems Project**

**Project: Web Platform for Client-Supplier Business Management -Supplier Part**

**FINAL REPORT**

PREPARED BY:

Supervisor: Ms.Farnaz Farid

Team Name: Group 3 DayDream

Team members:

|  |  |
| --- | --- |
| Xiaohan Li | 470011746 |
| Trung Kien Hoang | 470020243 |
| Yinan Chen | 470041259 |
| Tiansheng Liu | 470342488 |
| Kewen Su | 470503539 |
| Fengjun Li | 470342112 |
| Haozhe Zhang | 460335162 |

**TABLE OF CONTENTS**

1. Executive Summary..................................................................4

2. Project Objectives......................................................................................6

3. Proposed Methodology.........................................................................9

3.1 Front-end Methods.........................................................................10

3.2 Back-end Methods.........................................................................10

3.3 Teamwork methods.........................................................................10

3.4 Auxiliary methods.........................................................................10

3.5 Testing methods.........................................................................11

4. System Architecture.........................................................................11

4.1 4-Tier System Architecture.........................................................................11

4.2 React Native-Native connection................................................................ 12

4.3 States Management Architecture...................................................................12

4.4 Database Structure...................................................................14

5. Software Used for The System.........................................19

5.1 Front-end Software...................................................................19

5.2 Back-end Software...................................................................19

6. Prototype Development......................................................20

6.1 Required Output...................................................................20

6.2 System Documentation ...................................................................20

6.3 Issues Arising During the Prototype Development......................................23

6.4 Modified Prototype...................................................................24

6.5 Prototype Testing and Confirmation of The Requirement.......................... 30

7. Test Management...................................................................32

7.1 Test Plan for Prototype...................................................................32

7.2 Scoping of Testing...................................................................32

7.3 Stakeholders...................................................................33

7.4 Platform for Testing Purpose..................................................................33

7.5 Test Procedures...................................................................33

7.6 Non-functional Testing...................................................................34

7.7 Quality Control ...................................................................35

7.8 Test Result...................................................................36

8. Summary...................................................................39

9. Individual Contribution...................................................................40

10.Appendix...................................................................41

11.References...................................................................42

**1. Executive Summary**

"Wouldn’t it be easier if I could simply outsource the tasks?" is some feeling that almost everyone will experience when working under pressure. Based on such a huge user demand, we are very honored to participate in the Client-vendor Business Management Web Platform project, our ultimate goal is to build a trusted community marketplace for people to outsource tasks, find local services or complete flexible jobs to earn money.

In the joint development of this project with Group 7, the part we are mainly responsible for is the "Supplier", that is start the design, completion and test of functions from scratch such as registration, login, posting employment posts and so on. During the development of the project, we formulated a project plan and checked with our client Dr. Bassem Suleiman and hold a weekly meeting to present our progress. In the design and iteration of the prototype, we chose to use Balsamiq to complete the 1.0 lo-fi version of the demo and our final hi-fi version after several iterations, which include 34 webpages and 9 main functions of supplier users. We use Vue.js in the front-end framework development and Django in the back-end development, meanwhile we deployed the back-end server on Amazon Web Services. In the current status, we are working on the completion of the code test and docking of the front and back ends.

The purpose of this report is to document the lifecycle of the development of the project, using user-center design principles, from conceptualization to the final Web Application which runs successfully. It summarizes the findings from the project proposal where objectives and competitive product research was conducted, and the methodology we used for the final hi-fi prototype and product development. The other section of this document sets out the details of product development and test management.

**2. Project Objectives**

In the “Web Platform for Client-Supplier Business Management” project, our object is to meet the needs of customers for this project and provide customers with a web application that can be used in practice. Since we are the supplier side of the web, we hope that all suppliers on the web can make the most suitable choice by flexible filtering and searching of jobs and real-time communication with customers.

Based on the above requirements, we hope to divide this project into the following small objectives for realisation, so that customers can have a clearer understanding of the progress of each stage and judge whether our project is completed

**Stage 1: Project plan**

In the infancy stage of the project, a project plan should be drawn up to set out the background, goals, scope, deliverables of the project based upon currently gathered information. The plan aims to give the customer a rough idea of current project planning and to have a preliminary understanding of the project lifecycle and the expected output of the project.

**Stage 2: Low-fidelity prototype**

At this stage we need to design a low-fidelity prototype which aims to present the general UI design of the web application to the client and ensure that the follow-up development is in the right direction.

**Stage 3: Project proposal**

The project proposal elaborates on every single aspect of the project, which is written up based upon feedback from the project plan, low-fidelity prototype and interviews with the client. The purpose of this deliverable is to outline the value proposition of the project and to set out what value could the project bring to the client. At this stage, the client can put forward more detailed comments and suggestions for our proposal

**Stage 4 : High-fidelity prototype**

The High-fidelity prototype provides an interactive interface with possibly the highest similarity to what will be delivered as the final product. It provides an intuitive approach for both client and testers to navigate through interfaces and check whether the design has met the expectations of the client.

**Stage 5 : Web Platform for Client-Supplier Business Management**

The key objective of the project is to build a web application with the intuitive UI design and reliable back-end database system underlying to perform business management between client and supplier. As the final product, customers can rate our projects.

### Stage 6 : System testing

The system testing aims to validate the integrity of the system, by which each individual component can interact with each other correctly, so that the client can have a comprehensive view of the product.

**4. Proposed Methodology**

Throughout the development process of the project, our team has followed the agile methodology, because at the early stage of the project we thought that agile development is more suitable than the waterfall model or other models. The specific reasons are as follows: the progress of our project needs to be timed. In order to communicate with clients, how can we make real-time improvements to meet changing needs. Secondly, members of our group are willing to conduct close discussions on the design of the project. Even if face-to-face communication is hard due to the impact of the epidemic, this does not affect our cooperation through social software and zoom. The experience over the past few months has also proved that our initial choice of agile method has far more benefits than disadvantages. And then we will describe our front-end and back-end development and testing methods in detail.

**4.1 Front-end Methods**

HTML5, CSS and JavaScript comprise the foundation of building the user interface of the information system, upon which front-end frameworks can be implemented to help facilitate the efficiency of development. In terms of framework, Vue.js is chosen amongst currently prevailing frameworks as it is approachable, and therefore, enabling the development team to be familiar with the techniques in a short time.

**4.2 Back-end Methods**

SQLite will be chosen as the database service of the information system as it is a mature relational database service and has been implemented by many of the world’s largest organizations, which ensures the reliability, maturity and security of the storage system. The back-end development will be implemented upon Django, a high-level Python Web framework that enables faster development with less code, compared to frameworks based upon other programming languages. The usage of Django could shorten the project lifecycle and accelerate the process of the system entering into the market.

**4.3 Teamwork Methods**

Considering the size of the development team and that team members are collaborating remotely; the Scrum method will be implemented as a guidance of teamwork. Due to the feature of being light-weight and flexible, Scrum allows the development team to collaborate efficiently and ensure the possibility of responding to changes.

**4.4 Auxiliary Methods**

GitHub is the current prevailing solution for developers to collaborate remotely in terms of code storage, and therefore, is chosen to be the method to store files and codes that we created throughout the project. Zoom is the main approach to hold formal group meetings and client meetings throughout the project due to the pandemic situation. Apart from formal meetings, daily communications through texts or images will be performed via WeChat. In terms of Low-Fi prototype, Balsamiq is chosen as the tool to build the prototype as it is easy to set up the environment and thus enables developers to convert ideas and concepts into the prototype with less effort. With the restriction of the short project lifecycle, The Hi-fi prototype will be built with static web pages without using particular prototyping tools. The prototype aims to improve flaws that were reflecting on the Low-Fi prototype and help the client to get a better understanding of the final look of the user interface.

**4.5 Testing Methods**

Although the project schedule is tight, we still pursue the robustness of our deliverables. Therefore, we use unit testing and system testing to ensure the foundation of the software is solid. Furthermore, as we are implementing the Agile approach throughout the project lifecycle, we would also perform acceptance testing to ensure that the client is satisfied with the deliverables at the end of the project.

**5. System Architecture**

**5.1 4 - Tier System Architecture**

**5.1.1 Presentation Tier**

Presentation tier is the interface of our product, this interface built through the Vue framework(HTML, CSS, JavaScript) and EL-UI Library, which can realize interaction between user and product. The main purpose of interface design is user-friendly interaction and satisfies all requirements of the user using our web application. The interface can collect user POST or GET requests and transform them to the backend through the delivery tier.

**5.1.2 Delivery Tier**

The delivery layer that transmits data from the front end to the back end. This tier is built through JavaScript, JSON, AXIOS. The delivery layer will collect the request information of the interface layer and convert it into the corresponding JSON format and then transmit it to the backend for processing. This tier also can receive JSON data transmitted back from the functional logic layer.

**5.1.3 Functional logic Tier**

The functional logic layer built by Python and Django framework can receive the JSON data transmitted by the delivery layer, process the data in the database accordingly, and then return JSON data to the delivery layer. This layer is the main area for realizing the entire product function. All functional logic will be stored in this layer through python, and only this layer can interact with the database.

**5.1.4 Database Tier**

An SQLite database built through Django framework and python, the main function of the database is to store data, and can realize the interaction with the functional module. Database layer only can interact with the functional logic layer.

**5.2 React Native-Native connection**

Our products are divided into two sections: front-end and back-end. The front-end uses Vue framework, HTML language, CSS, and JavaScript to develop the interface of the webpage. In the process of front-end development, the front-end development team uses GitHub to integrate the code, all members upload the sections they are responsible for, and finally the debugging connection of the front-end web page is unified. The back-end development team also used GitHub for code integration and debugging at the beginning. After everyone created the database together, Each member developed at a local server and uploaded the data to GitHub. However, due to the impact of COVID-19, the entire development process was carried out online, and the team members were scattered in different countries, resulting in many difficulties in front-end and back-end docking. Finally, we decided to put the back-end module on the AWS cloud server for Integration and debugging. The front end is also convenient to use the HTTP port to connect the front-end interface to the back-end server. Our front-end and back-end API uses the AXIOS package(Under VUE framework) and JSON package. The front-end transmits JSON data to the back-end through the HTTP port. The back-end receives the JSON data and analyses the request. It performs different database operations according to different requests, and finally returns one JSON data to the front end. Finally, our front-end runs locally, and the back-end module is deployed on AWS cloud server. Now you can open our front-end interface anywhere to experience our whole web application. This definitely reduces the time and cost required for the integration work, making the entire project more rapid and effective.

**5.3 States Management Architecture**

State management is a rather vague concept, and it does not have an accurate or formal definition in computer science. In this project, state management refers to “The management of the state of user interface controls which may determine the change of the state of data or information in the application including the buttons, text field, filter etc.” (Flutter, 2019). In addition, the state can be divided into ephemeral state and app state.

**5.3.1 Ephemeral State**

Ephemeral state refers to the state which can be nested in one single widget (Flutter, 2019). In this project, filter, sort and search functions provide an ephemeral state in their own widget which means the states out of this and other widgets would not be changed once the user leaves this widget.

For example, the filter function is used to filter quotes or orders for the supplier by several parameters including dates, budget, category etc. Once a filter is chosen, the view of the user would be changed to the filtered attributes. However, if the user refreshes or goes into other functionalities which contain a list of quotes or orders, the view would remain as default. This means the state of the widget only changes when the user clicks the filter and reset it to default when refreshing or leaving. The sort and search functions are similar with the previous one.

**5.3.2 App State**

Application state refers to the state that should be shared across several widgets in the system or between user sessions (Flutter, 2019). In this project, sign up/ log in, messenger, update profile and push quote functions provide an app state which means the states would maintain changes in this and other widgets even users leave the widget.

For example, the sign up/ log in functions would provide a certificate which means the state of the certificate is modified once the user succeeds to sign up/ log in. Other widgets such as dashboard would check for the user's state first when the user tries to enter in so that only signed up/ logged in with the modified state can get into their dashboard. This means the state of the widgets would maintain changes in other widgets which require the state. The update profile change quote status and push quote functions are similar with the previous one.

The My message function is a bit different. Because if a user tries to refresh or leave the function, the messages he/she sends would be maintained but only in this widget as history. However, it has no influences on other widgets.

**5.4 Database Structure**

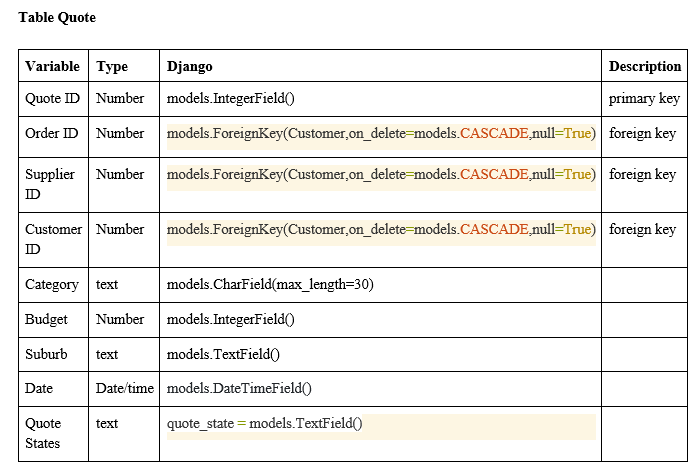
**5.4.1 Database Table Structure**

Following table is our data structure for both customer and supplier sides. *Customer* and *Supplier* are main users for the website including their detailed information recorded. *Order* and *Quote* are main objects which contain main functionalities and transaction mode of the website. They carry the task details including description, category, date, status etc. *Discuss* is the main different ways that a customer can communicate with a supplier.



****

****

****

**5.4.2 Relationship Diagram**

The following relationship diagram implies the relationship between different tables.

A supplier can sign up or log in with their username and password in the supplier homepage.

A supplier can update their profile in the dashboard with the following attributes of supplier.

The salt and iteration are obtained once the supplier signed in successfully.

# They are used to encrypt the password.

A Quote can be generated once a supplier bid on with an order raised by a customer.

# On the other hand, an order can have several quotes since several suppliers can bid on the same order.

The quote states would be changed if a customer accepts or rejects it finally.

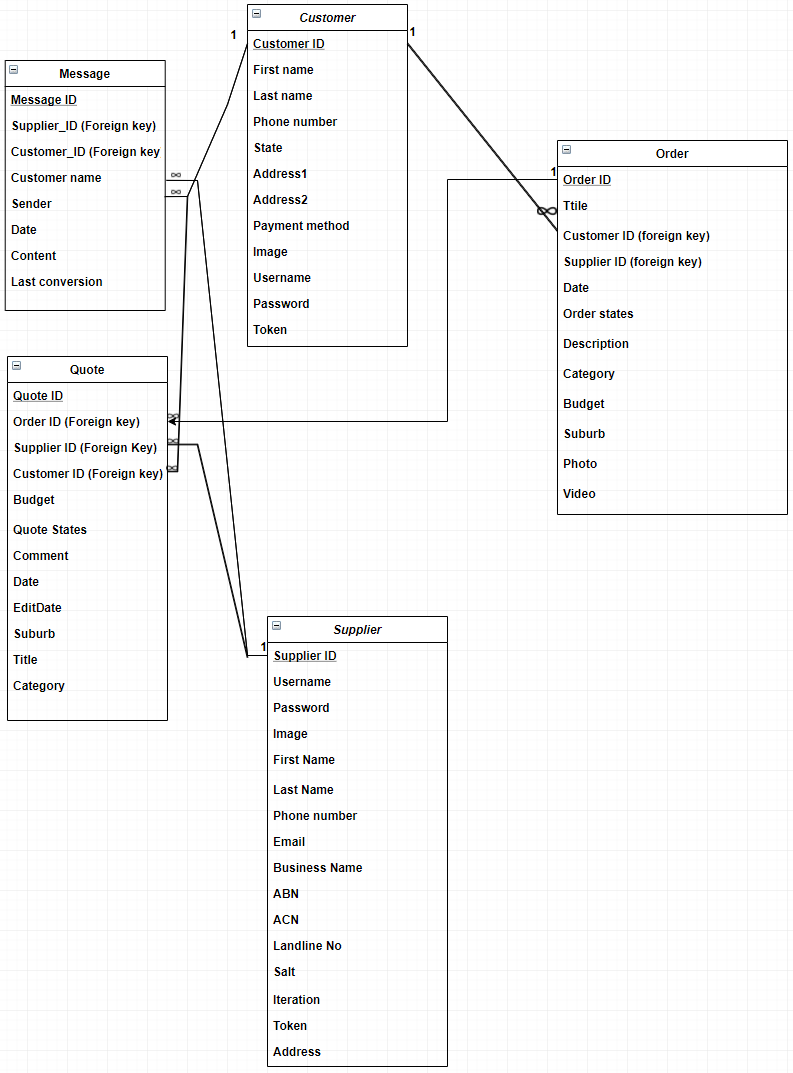
The budget in Order would be decided when one of the Quote is accepted, and the other Quotes would be rejected.

A supplier can use my message to have communication with several customers and the history messages would be stored.

# Vice versa

The order state would be changed to “in progress” once the Quote is accepted and changed to “completed” once the Order is done and changed to “closed” once the Order is passed.

What’s more, a customer can communicate with several suppliers to bid on the time, budgets and other details in the order



**6. Software Used for The System**

As mentioned above, the team divides the workload to two sub-teams, front-end team and back-end team, to realise the separate development of the system. In this part, what software is used in this project would be discussed.

**6.1 Front-end Software**

**6.1.1 VUE Framework and Related Packages**

VUE is a progressive framework for building user interfaces which combines *HTML*, *CSS* and *JavaScript* together. It focuses on the view layer and adopts a bottom-up incremental development design. In this project, VUE is used for implementing the whole user interface and several functionalities such as filter.

**6.2 Back-end Software**

**6.2.1 Django Framework and Related Packages**

Django is an open source web application framework, written in Python, which provides pragmatic, clean design and fast development. In this project, Django is used for creating databases, realising the interaction between front-end HTML and database in python. The team uses *Django model* to build an SQLite database, uses *rest framework package* and *other python packages* to realise the JSON API and related functionalities such as posting or getting data to or from between front-end and databases.

**6.2.2 AWS Cloud Server**

Amazon Web Services provides reliable, scalable and low-cost cloud computing services. In this project, AWS Cloud Server is used for maintaining the database on the cloud server so that every team member can adjust their own coding part on their own computers. Because during the joint commissioning of front-end and back-end teams, there should be an intermediary so either of the teams can get the request from another. Otherwise, both sub-teams work can only be put in one local host and be adjusted by one member which is inefficient.

# 7. Prototype Development

## 7.1 Required Output

The project aims to deliver a web platform for client-supplier business management, where customers are able to post jobs on the website according to personal demand and suppliers are able to bid on jobs based upon personal skills. As the project is undertaken by two developing teams and our team is in charge of the supplier side of the project. Therefore, in the following sections, we will only set out the detailed information of the prototype on the supplier side.

## 7.2 System Documentation

### 7.2.1 Supplier Homepage

* The new supplier can visit the home page to view key features and the value proposition of the website
* The new supplier can click on “Become a Supplier” button to register and create a business account

### 7.2.2 Registration/Sign up

* New suppliers can register and create a business profile/account
* New supplier fills in the form with business name, email (username), password (twice)
* Creates a new supplier account and store it in the database
* Password strength based on security industry standards
* Proper (standard) data validation
* Upon successful sign-up, supplier sees the dashboard with a list of all most recent jobs

### 7.2.3 Login

* Username/password login
* Secure and fast login
* Forget password (based on web standards/practices)
* Upon successful login, the supplier sees the dashboard page

### 7.2.4 Dashboard

* Upon successful login, the supplier can see the latest five jobs and five latest notifications in the dashboard interface.
* The supplier will be redirected to the login page in the case where the system does not detect login status.

### 7.2.5 Update Profile

* Upon successful supplier login, the supplier can edit profile details
  + Mandatory (valid values must be provided): business name, email, password (current password, new password twice), ABN and ACN
  + Optional: business owner’s name (first and last), mobile number, landline number, address

### 7.2.6 Check quote/Job status

* The supplier can check the status of their jobs or quotes through the jobs/quotes page
* The supplier can check quote status by selecting the “Bid on” option in the dropdown menu. Each quote is displayed as one of the following three statuses: pending, accepted or rejected
  + Pending: the customer has not decided whether to accept the quote or not
  + Accepted: customer accepts the quote
  + Rejected: customer declines the quote
* The supplier can check current job status by selecting the “Current” option in the dropdown menu. Each current job is displayed as one of the following two statuses: working in progress or completed
  + Working in progress: The work has just begun or has been partially completed
  + Completed: the job has been done by the supplier, waiting for confirmation from the customer side.
* The supplier can check history job status by selecting the “History” option in the dropdown menu. Each history job is displayed as closed status after the completed job has received confirmation from the customer.
* The supplier can navigate through different pages by using the pagination bar, which provides several flexible approaches for the supplier to jump to the expected page number.

### 7.2.7 Search

* Suppliers can type in search keywords in the search bar to search for jobs.
* The search bar is available in Check quote/job status page and Browse tasks page
  + Check quote/job status page: search jobs from supplier’s own jobs
  + Browse tasks page: search jobs from all jobs posted by customers

### 7.2.8 My Messages

* The supplier can see a list of users on the left side of the message interface. Each user in the list is displayed with the avatar, nickname, and latest chat date
* The supplier can send a text message by typing in the chat box and click send button and see recent chat messages on the chat window
* The supplier can view quote request detail sent by the customer and respond the quote request with all the necessary details

### 7.2.9 Browse Tasks

* The supplier can see list of all jobs, clicks on a job card to get into the job detail page
* The supplier can filter jobs based upon job category and job budget

### 7.2.10 Job Details

* The supplier can bid on a job posted by a customer in Job details page
* Supplier starts to bid with price, message, skill, duration, start time, end time, preferred time plot. Price information should be placed in the quote section, and the remaining information will be placed in the comment section.
* The supplier can switch jobs listed on the left side to view job details
* the supplier can click on the user icon to start a conversation with the customer.

### 7.2.11 Supplier Web System Architecture

## 

**The Interaction of web pages**

Suppliers can jump to the login and signup pages from the homepage. When the supplier login and signup successfully, the page will jump to their dashboard automatically. In their dashboard, they can check the job status, improve the profile and have communication with customers. For choosing the job to get paid, in the browse jobs page, suppliers can filter the job and click the job card to get the details of the job to make an offer. When the database gets the data successfully, the page will jump to the dashboard automatically and also can click the icon of the customer to have a communication with the customer to ask some details about the job in the job detail page.

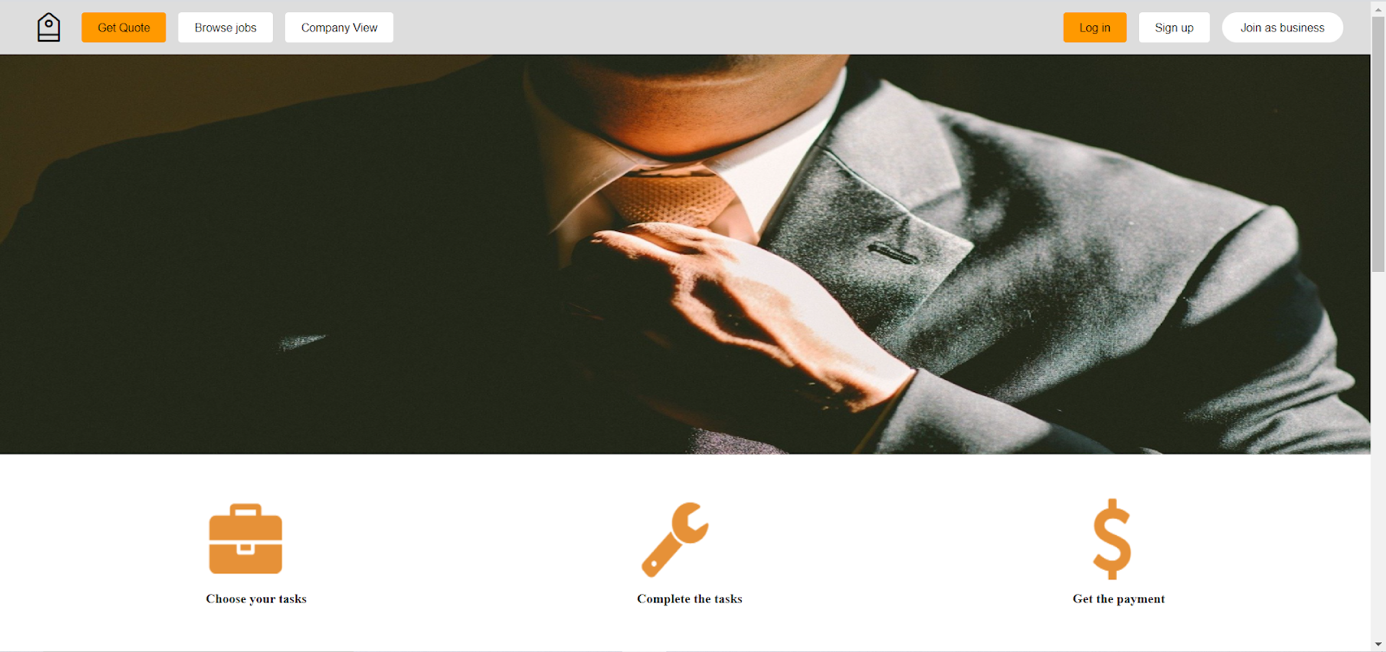
## 7.3 Working Prototype

Throughout the lifecycle, we have improved our prototype from a Hi-fi prototype into a working prototype. As there are many pages for our prototype, a link will be provided in the appendix section.

As stated in the system documentation section, our modified prototype comprises the following pages:

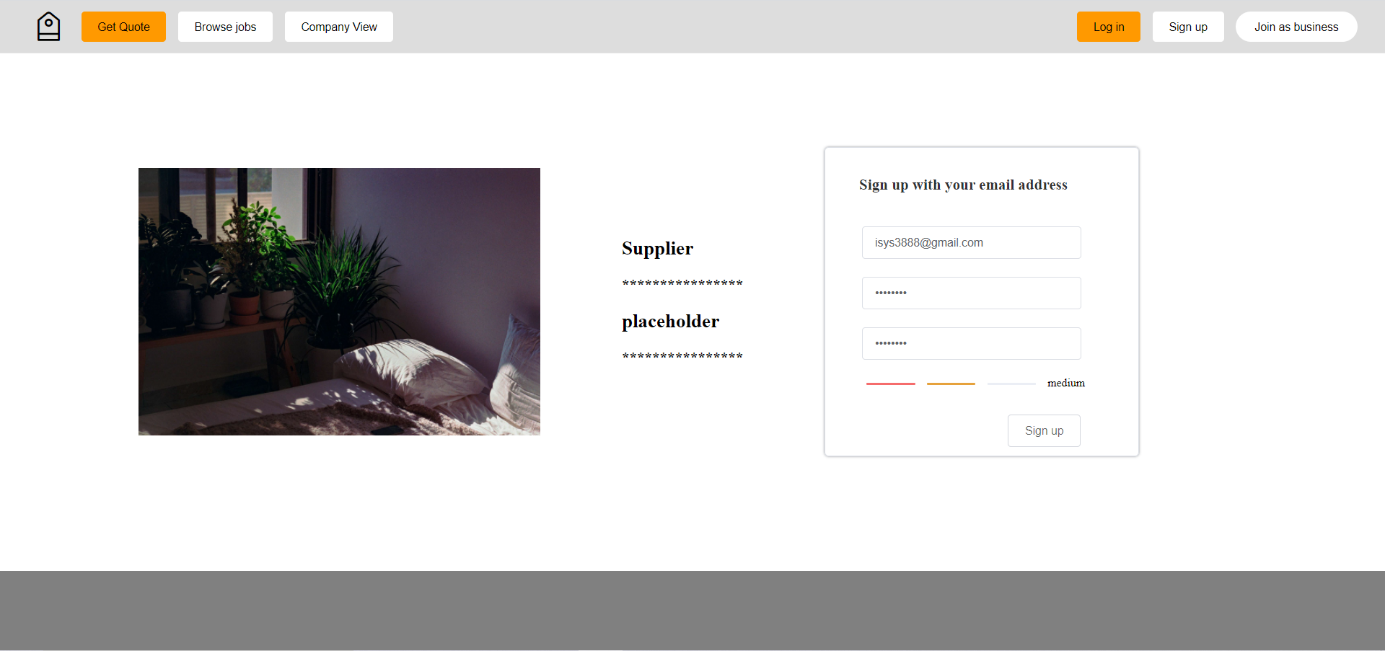
* Supplier home page
* Login page
* Sign-up page
* Dashboard page
* Update profile page
* Check quote/job status page
* My messages page
* Browse jobs page
* Job details page

### 7.4.1 Supplier home page



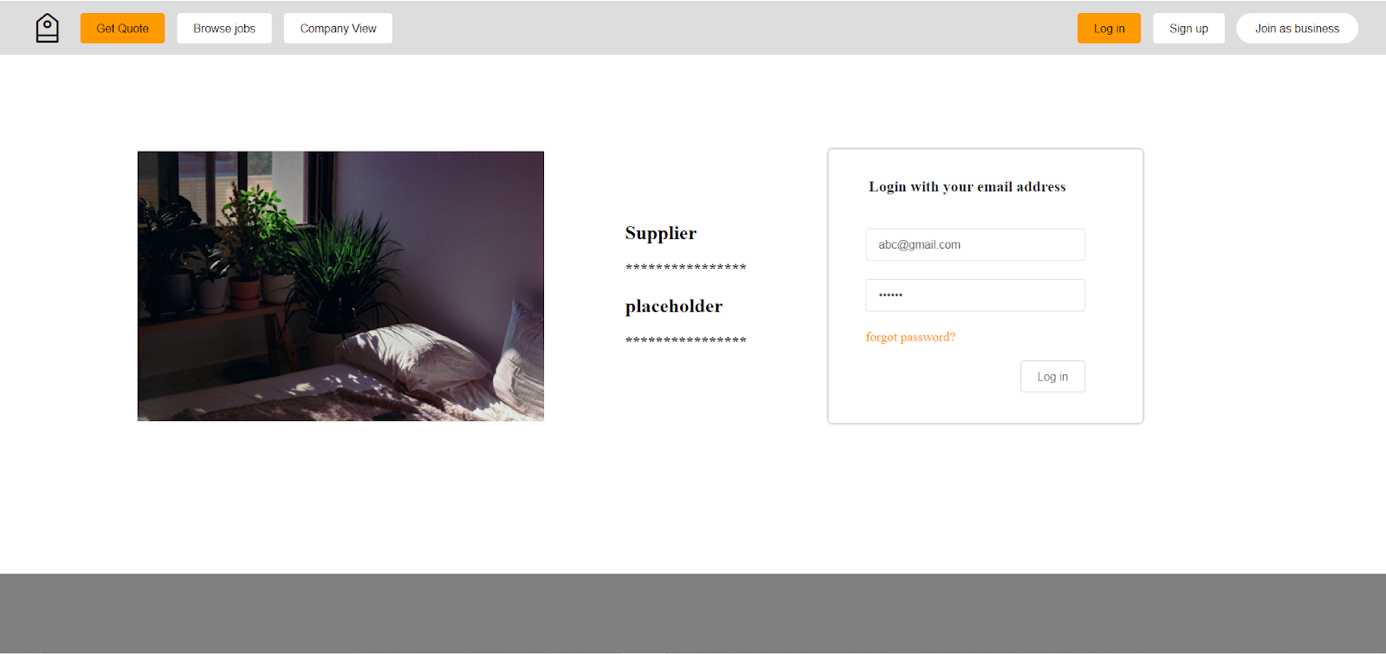
* Displaying key features and possibly recent events for supplier
* Entries for login, sign-up pages

### 7.4.2 Supplier sign-up page



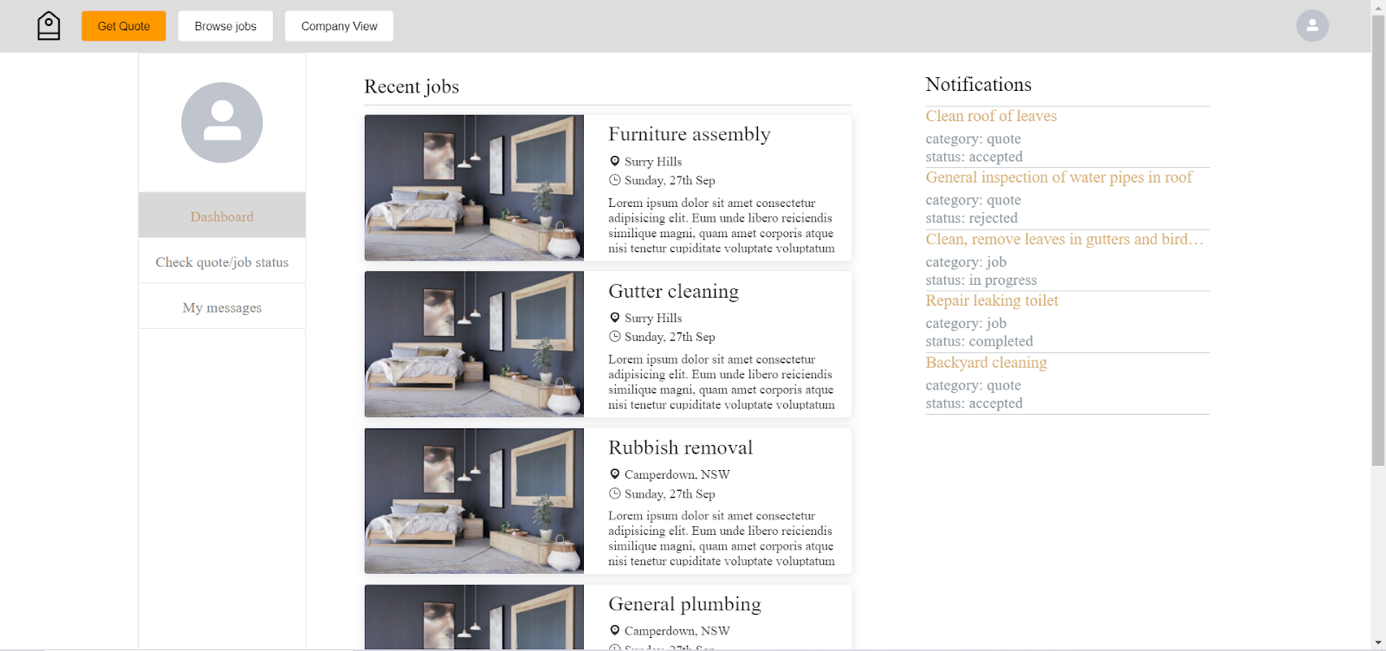
* Providing sign-up function for business users to register user accounts
* Implementing encryption processes to ensure users’ account security
* Email format revalidation, password length revalidation
* Password strength based on security industry standards

**7.4.3 Supplier login page**



* Providing login function for business users to log into user accounts
* Implementing encryption processes to ensure users’ account security
* Email format revalidation, password length pre-validation

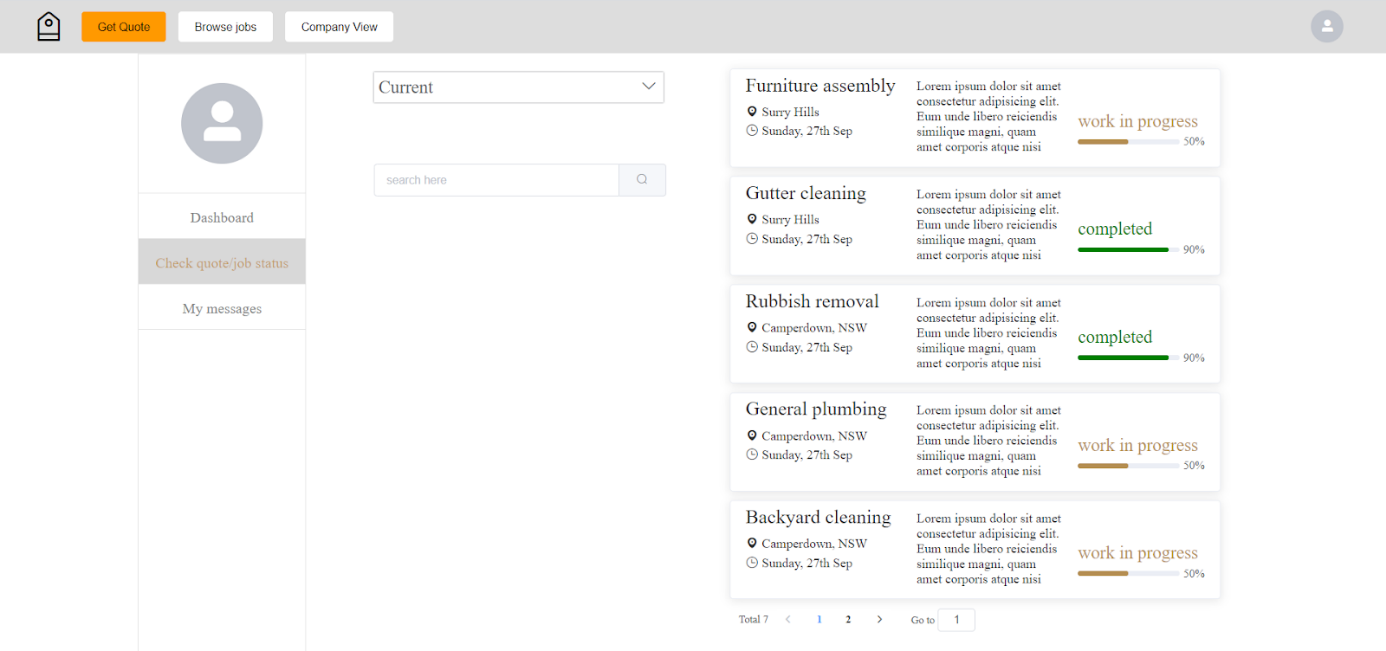
### 7.4.4 Supplier dashboard page



* Displaying five most recent job cards as the entry to corresponding job description pages
* Displaying five most recent notifications based on updates of quote/job status

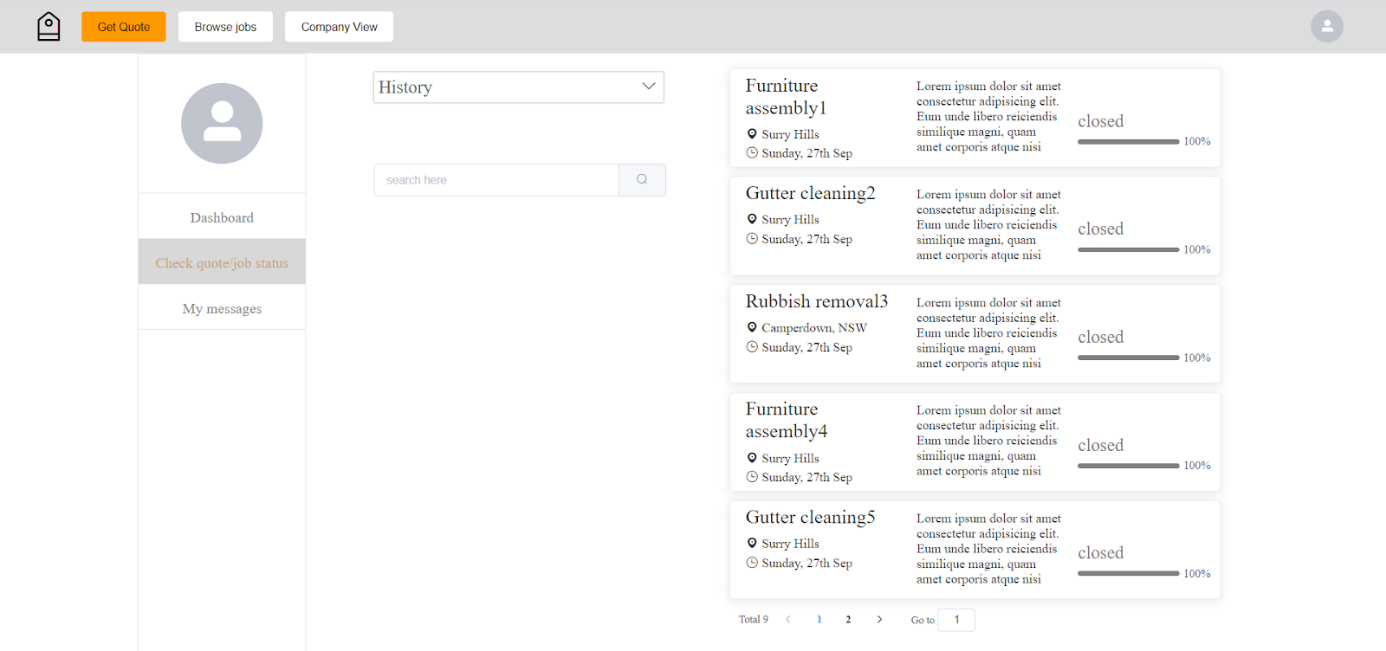
### 7.4.5 Supplier quote/job status page

#### **current jobs**



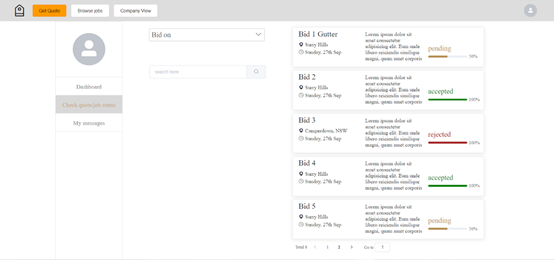
•       Displaying all current jobs for the user

#### **history jobs**



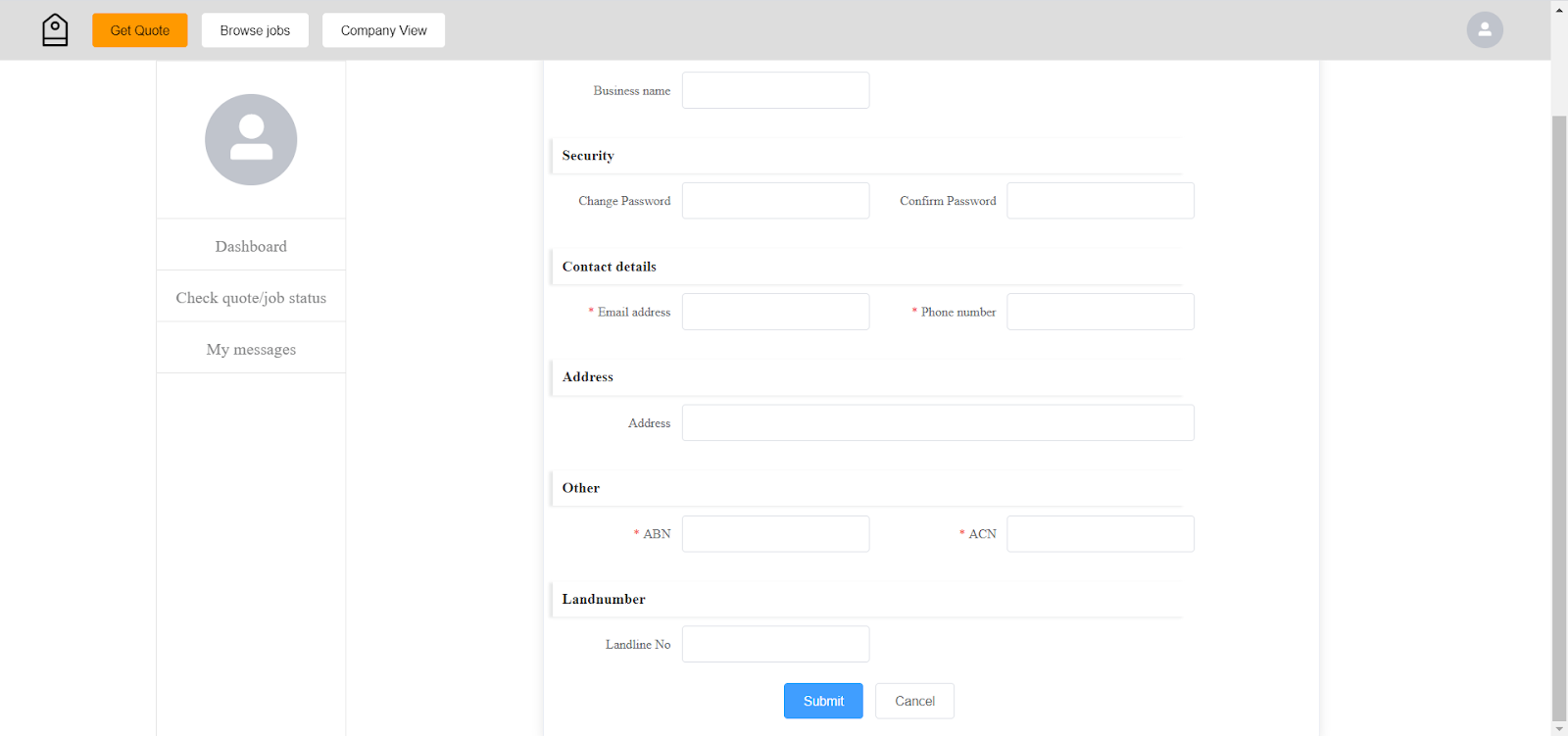
•       Displaying all history jobs that user has taken in the past

#### **bidding items**



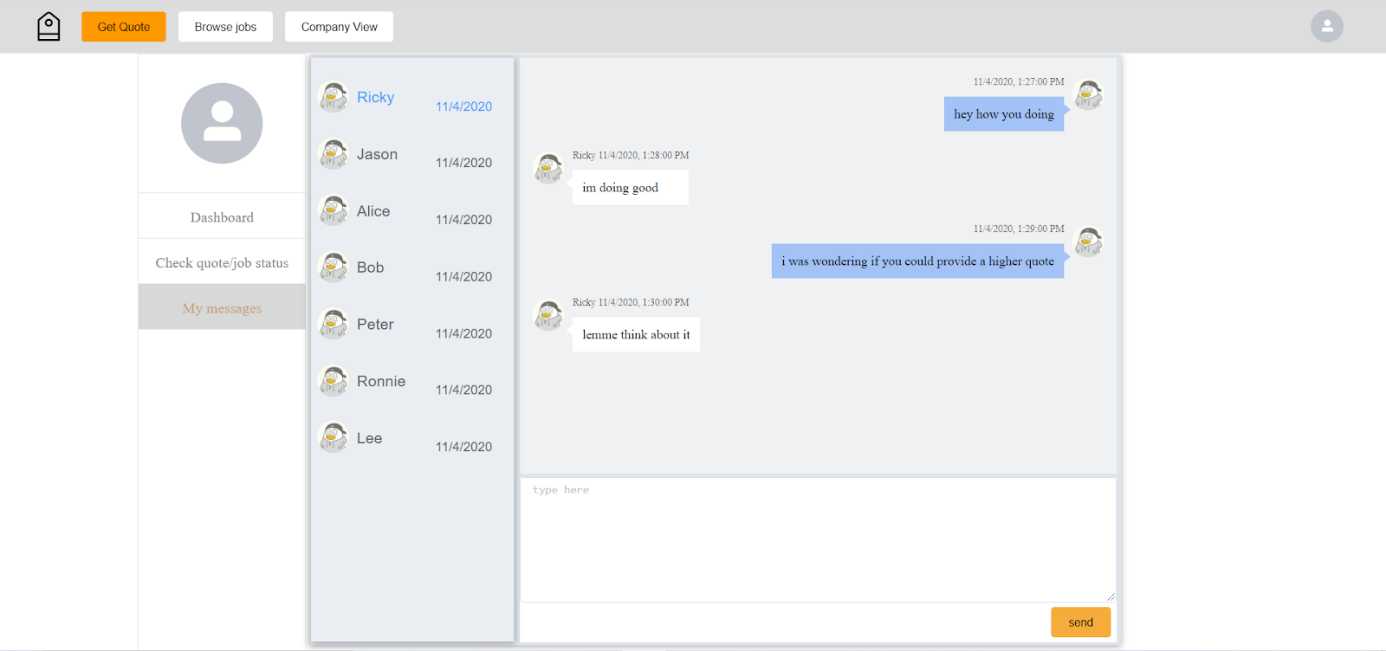
* Displaying all jobs that the user is currently bidding on

### 7.4.6 Update profile page



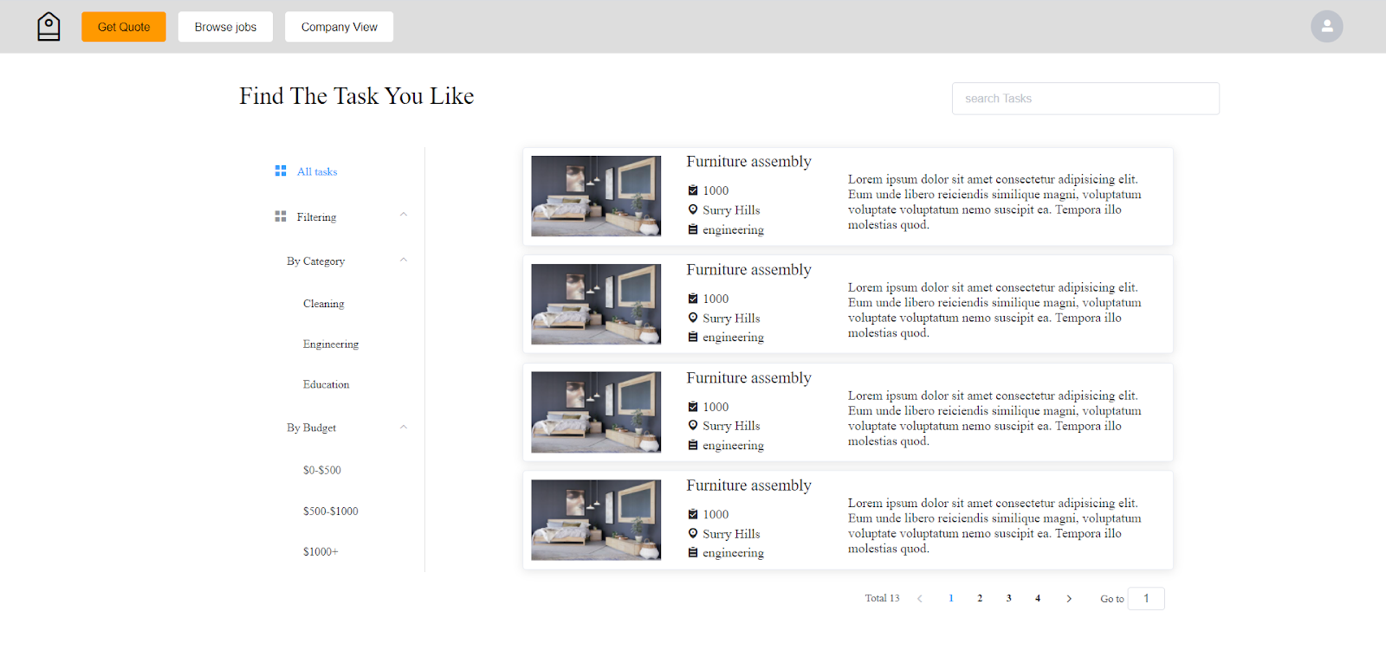
* Supplier can update profile in this page

### 7.4.7 Supplier Message Page



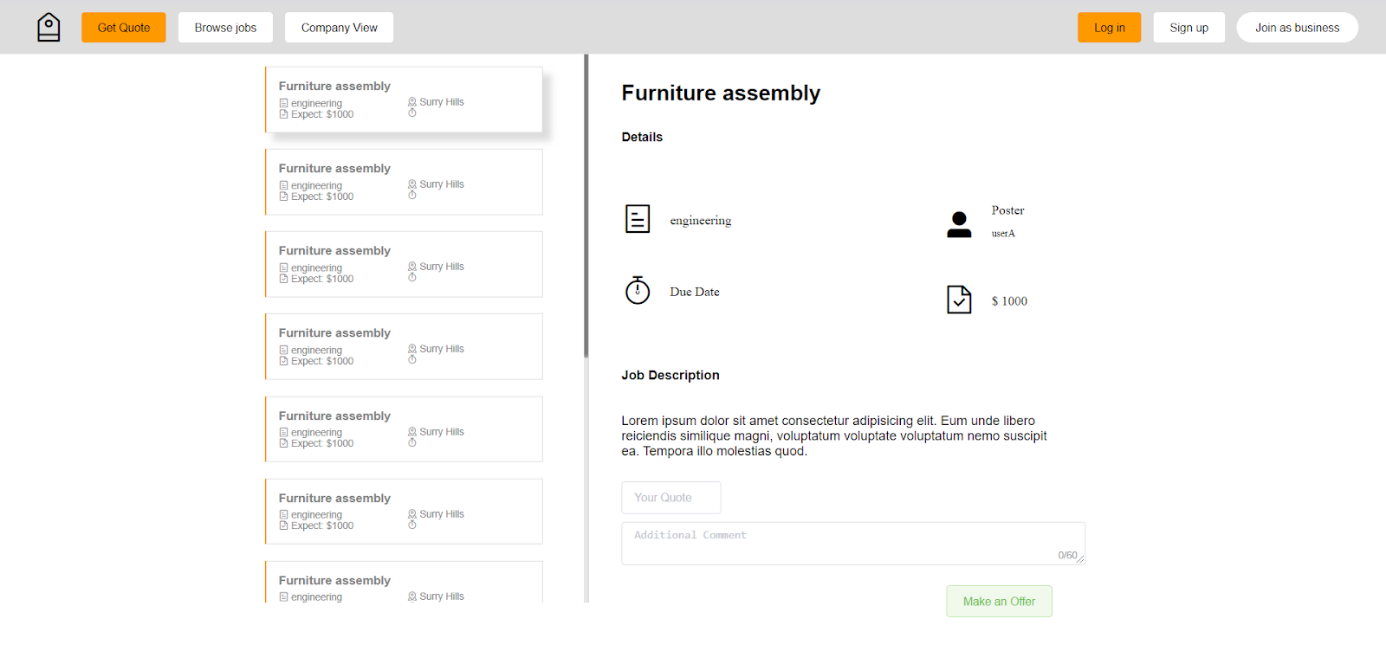
* Displaying a list of customers that the user has contacted before.
* Chat with a user by clicking the user in the list
* Type in the chat box and chat with the customer with details of jobs

### 7.4.8 Browse Tasks Page



* Providing filtering function for suppliers to find suitable jobs
* Providing search function for suppliers to search jobs
* Clicking on a job card to navigate to the job details page

### 7.4.9 Job Details Page



* Displaying detailed information about a job
* Supplier types in the quote and relevant information in the quote and comment section respectively
* Clicking on “Make an Offer” button to send a quote request
* The supplier can click the "poster icon" to have a chat with customer, it will jump to the message page

## 7.5 Prototype Testing and Confirmation of The Requirement

### 7.5.1 Prototype Testing

For the project, we need to test the logical and functional correctness. It includes the links between each page and the correct page click order design. It also includes the data transmission between front and back end. Data transmission is particularly important for the operation of the whole system. Therefore, we use hypothetical data to have the testing to make sure the front and back ends can interact with each other. In the prototype, the username and password of suppliers can be correctly stored in the database and when other users use the same email address to try to sign up a new account, the system will alert this user the email has been used. Or if the registration is not successful due to server or network reasons, the user will get relevant prompts. Meanwhile, except for the suppliers’ information, the job list data also occupies an important position during the whole process. The back end needs to correctly filter the working data according to the requirement of the front-end. Like, in the browse task page, the web page needs to be correctly filtered by the category and budget of the job, and then show the new job list for users. We need to test that the back end can filter out the correct data, and then transfer the data to the front end.

### 7.5.2 Confirmation of the requirement

Role: Supplier

|  |  |
| --- | --- |
| Website Pages | Status |
| Sign up page:  1. Register and create a business account  2. Sign up with email as username and password needs twice input.  3. Creates a new supplier account and stores it in the database.  4. Password encryption. | Completed |
| Log in page:  1.      Username/password login  2.      Secure and fast login | Completed |
| Supplier’s dashboard page:  1.      Upon successful sign up, the supplier sees the dashboard with a list of all most recent jobs.  2.      Upon successful login, the supplier can edit profile details.  3.      Suppliers can check the status of their jobs or quotes through the jobs/quotes page.  4.      Supplier has a simple message centre to communicate with customers.  5.      User-friendly UI. | Completed |
| Browse Task page:  1.      Upon successful login, the supplier can see the new job list from customers.  2.      Suppliers can filter the job in budget and categories.  3.      Suppliers can search jobs to get similar matching results.  4.      Supplier can see a list of all jobs with filters, clicks on a view the job detail button which takes them to the job detail page.  5.      Suppliers can bid on a job posted by a customer after successful login and additional comment in the job detail page.  6.      Supplier to respond to a quote request after successful login in the job detail page. | Completed |

# 7. Test Management

This test describes how to test the “Web Platform for Client-Supplier Business Management”. It provides detailed information such as scope of testing, testing purposes, test procedures, non-functional testing, quality control and test result.

## 7.1 Test plan for prototype

Plan 1: Start the front-end and back-end services on the local side to show the client the entire process of the system and the functions required by the system, and let the client score and give feedback on each page and function from the user's perspective.

Plan 2: Package the front-end and back-end code and send it to customers, allowing the client to test and operate at their own extremes, and score and give feedback on the user experience of each page and function.

## 7.2 Scoping of testing

* The testing phase includes unit testing, system testing and acceptance testing. System testing includes the most basic functional testing and non-functional testing.
* The test type mentioned in this plan is the test in the demand phase, that is, the test process for functional verification of the Web Platform for Client-Supplier Business Management.

## 7.3 Stakeholders

* Clients: Clients are those who log in as employers in order to find out the tardy who are suitable with their specific jobs or tasks. They need to write a short job description which sums up the job’s information so that the suitable supplier can apply applications.
* Suppliers: Suppliers are job seekers who arrange a flow of information in the platform to sort out what relates to their capabilities. They can apply for the job and contact the clients to offer themselves.
* Organization: a consulting company, our organization, through tactical and strategic approaches, is aiming to empower small to medium size businesses.

**7.4 Platform For Testing Purpose**

A test automation platform takes an important role in the modern Web developer. Our web application is tested by Django’s test-execution framework which is an extremely useful tool for fixing bugs. Furthermore, this tool is helpful in simulating requests, inspecting outputs, verifying code and inserting test data.

**7.5 Test Procedures**

Registration/Sign up

1. The test to check the register with an existing account.
2. The test to check the register with a new account.
3. The test to check if the application redirects to the “dashboard” page after successful signing up.

Login

1. The test to check username/password login.
2. The test to check if the application redirects to the “Post a Job” page after successful signing up.

Update

1. The test to check a successful login.
2. The test to check editing profile details.

Check quote/job status

1. The test to check the job status or quotes.

Response to quote request

1. The test to check a successful login.
2. The test to check a quote request

Search

1. The test to check searching for a job
2. The test to check if the application redirects to the “Dashboard” page after starting the app.

**7.6 Non-functional Testing**

### 7.6.1 User Interface

* User Interface Testing is used to verify the interaction between the user and the software, and to check whether the objects in the user interface behave as expected.
* Overall Interface Test: Standardization, Consistency and Rationality of the user interface
* Interface Element Test: Focus on icon, words, picture, etc. element in interface.

### 7.6.2 Usability

Usability testing is used to verify the user can use the product to complete his task, how efficiently, and how subjectively he feels. Find other users to use the website, and record and analyses their experience feelings. To summaries their suggestions to improve the UI designing.

### 7.6.3 Performance and Reliability

The website has certain ability to judge and correct errors and prompt. For incorrect input, such as when the required option in the profile is empty, the system will automatically judge and remind the user with red letters. And in the search function, automatic similar matching can be performed regardless of whether the letter entered by the user is uppercase or lowercase. The test includes input fields as a user under related functions to test whether the system can be judged normally.

### 7.6.4 Speed

Website speed is to ensure the smoothness of users using the system. Speed includes the speed at which the web page loads and the speed at which the database returns user demand data. We will use postman to test and simulate the speed of back-end transmission data and related tools to test the speed of front-end web page loading.

### 7.6.5 Compatibility

Compatibility testing includes the cross-platform deployment of the website and the correct operation of a user in different browsers such as IE and Google. Further, tests include strong compatibility of the webpage, no line breaks after changing the webpage size, and the normal use of functions.

**7.7 Quality Control**

Activities:

* 1. Review:
     1. Requirements:

We have identified all requirements for the web application from the client.

* 1. Design:

We have implemented a low-fide prototype and a high-prototype that our client satisfies with them. The templates for application now are designed same as the final prototype.

* 1. Code:

We have used HTML5, CSS, JavaScript, Angular JS for the front-end technologies; Python, Django and PostgreSQL/MongoDB for the back-end technologies.

* 1. Test Plan:

We have planned to do all functional and non-functional tests in the application.

* 1. Test Case:

We have passed all the test cases that we have claimed from requirements.

* 1. Testing levels:
     1. Unit Testing:

We have used unit testing methods for our test cases to ensure that our code correctly returns outputs, detects and protects against bugs in the future.

* 1. System Testing

Software system testing is a comprehensive test of the final software system to ensure that the final software system meets product requirements and follows system design.

System testing includes functional testing and non-functional testing; therefore, we try to use the test data and import the database to test whether the data and functions are in accordance with the expected requirement.  For non-functional testing, we try to get the test in different testing environments such as testing system compatibility on different browser platforms.

* 1. Acceptance Testing:

The task of acceptance testing is to verify whether the software functions and performance and other characteristics of the acceptance test are consistent with business requirements.

Looking for different testing users to help test and provide acceptance feedback. It includes the correctness, completeness and uniformity of various data, and whether the function meets the requirements of the client provided.

**7.8 Test Result**

We have used the unit test to check for each individual function:

1. Registration/sign up:
   1. The request is “200” after successful signing up.
   2. The request is “404” if it fails to sign up.
2. Login:

a. The request is “200” after successful logging in.

b. The request is “404” if it fails to log in.

1. Update Profile:

a. The request is “200” after a successful editing profile.

b. The request is “404” if it fails to edit the profile.

1. Check quote/job status:

a. All the outputs that return from check\_quote() function match with the expected outputs.

b. The request is “200” if clicking to “Check Quote” page.

1. Check profile:

a. All the outputs that return from check\_profile() function match with the expected outputs.

b. The request is “200” if clicking a photo icon.

1. Search:

a. All the outputs that return from search() function match with the expected outputs.

After the unit tests are all passed, we ran the interaction tests for the following functions:

1. Login after successful sign up.
2. Update Profile after successful login.
3. Check quote after successful login.
4. Check profile after successful login.
5. Log in and use the search function.

The final step of testing is that we implemented a system test for the application. We ran the server and tested if the system correctly ran for all pages and functions.

文本

描述已自动生成

This is two examples of unit test for pages and check profile function:

The first unit test checks if the page exists or not and the second unit test checks the profile of the supplier if the details match with what we expect. In general, we will use unittest methods for checking existence for all pages and test cases in test procedures. More specifically, for page-existence testing, we suppose to see the status code responding to “200” which represents that the request has succeeded. In addition, those functions, which return an output, should be matched with our expected outputs.

**8. Summary**

In this project, we aim to develop a web platform to provide job opportunities including house cleaning, moving, transportation, gardening and other different types of work for both legal individual workers and companies. We use Agile development method and separate the product to front-end and back-end sub-teams so that every member of the team can contribute to their own tasks. Firstly, we work together and successfully developed appropriate low-fidelity, high-fidelity prototypes as well as making agreement on project proposals with the client to define a basic version of the project. Then, we determine the methodologies and system architectures to build the framework for the entire system. In the development stage, the Front-end team uses Vue framework including HTML, CSS and JavaScript to develop the user interface including several functions such as filer, sort and search. The Back-end team uses the Django framework with Python language to construct the database, implement related functionality and deploy the database to the AWS cloud server. The sub-team firstly conducts their code separately to pass the basic requirements. Then, the team integrates the sub-team’s work together and runs an integration test to ensure most of the functional requirements are tested and proved to be completed in the stage. After the implementation, the team conducts a pilot test on the product to test on different user cases with different sceneries for both functional and non-functional requirements. The outcomes show that most of the requirements can be met in the product, but there is still some detailed method or additional requirements that needs to be improved. For example, we cannot upload the image or complete my message functions as the client appended.

As a result, the project gives us lots of experience on a real scene of the project including team management, schedule management, risk management, client communication and separate construction and integration of front and back ends. Meanwhile, for the members who started the project with zero development basis, during this project development process, they gradually became familiar with the development process and mastered the use of development tools and languages. In the future, the team can be more familiar with such web application projects to have a higher degree of completion and have a clearer scope and recognition on team management. What’s more, we believe the existence of our project is an attempt that the freelancer marketspace application software can be more popular

**9. Individual Contribution**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Member name | SID | Sub team | Contribution | Code contribution | Meeting contribution |
| Xiaohan Li | 470011746 | Back-end team | Full contribution | Database construction,  Order/quote related functionalities, Database cloud deployment, API functionalities | Full attendence |
| Trung Kien Hoang | 470020243 | Back-end team | Full contribution | Database construction, Profile related functionalities | Full attendence |
| Yinan Chen | 470041259 | Front-end  team | Full contribution | Front-end developed,  profile updated page. | Full attendence |
| Tiansheng Liu | 470342488 | Front-end  team | Full contribution | Front-end developed, homepage, browse task page and part of job description page. | Full attendence |
| Kewen Su | 470503539 | Front-end  team | Full contribution | Front-end developed, login page, signup page, job status, my message | Full attendence |
| Fengjun Li | 470342112 | Front-end  team | Full contribution | Front-end developed the sign up and job description page. | Full attendence |
| Haozhe Zhang | 460335162 | Back-end team | Full contribution | Database construction,  log in/sign up related functionalities, Database cloud deployment, API functionalities | Full attendence |

see appendix *database address*

**10.Appendix**

**Hi-Fi prototype**

https://balsamiq-wireframes.appspot.com/?state={%22action%22:%22open%22,%22ids%22:[%221Nq4zULF0wvrdiqrxQQ-LtkrPbq0wbhvC%22],%22userId%22:%22105314720530682293366%22}

**Database address**

<http://3.135.198.81:8081/>

**11.References**

Freelance Marketplace 2.0: Where Future of Work Happens - .Austin Yang(2020, August 23).  Retrieved November 12, 2020, Retrieved from:

<https://austinyang.co/freelance-marketplace-2-0/>

Growjo. (n.d.). Retrieved November 12, 2020, Retrieved from: https://growjo.com/company/Airtasker

Flutter. (2019) *Statemanagement*. Retrieved from:

https://flutter.dev/docs/development/data-and-backend/state-mgmt/ephemeral-vs-app