London South Bank University

GIT repository: https://github.com/irfanessa2/SFE_Crab

Video Demo: https://www.youtube.com/watch?v=SXQOmqiR5mg

A3 Poster: https://ibb.co/VN4RcHq

MODULE CODE: CSI_5_SFE

TEAM NUMBER: K17

Student Numbers: 4008609, 4024763, 4001576, 3920479

Scenario: Crab Express LTD

DATE OF SUBMISSION: 12/05/2023

WORD COUNT EXCLUDING TOC, REFRENCES and Appendix: 5921 Words

Contents

1. Introduction

The following report outlines the documentation, evaluation, analysis and implementation of a system designed to help Express Firm LTD improve their business operations through efficiency and oversight.

2. Project Overview

2.1. Project problem:

Express Firm Ltd initiated a crab enterprise in the southwestern area of Bangladesh. Upon collection, the crabs exhibit a hard shell which becomes soft following a shell change (mud). Soft shell crabs are valued at a up to ten times higher compares hard shells. The company wishes to streamline their operations via an online platform/portal.

2.2. Aim

The aim of this project is to have a working online portal for Express Firm LTD allowing the employer to keep track of their business and employees, along with the ability for employees to view/manage their details and input data for the employer.

2.3. Objectives

The objectives associated with this project are as follows:

1. Select a suitable methodology
2. Consider legal, social, ethical economic and comercial impacts
3. Select a suitable framework/library
4. Provide the client with a UI prototype, and alter it based on their needs
5. Clearly outline and discuss requriments in documentation and with the client. These include but are not limited to:
    - An interactive and responsive application which calculates the revnue for each month and year
    - Record the buying and selling prices
    - Record the crab category
    - Maintain employee details for both temporary and permanent employees
    - Display the best fisherman and crab death rate

3. Software Development Methodology

3.1. Criteria for Software Development Methodology

The following were taken into account when researching and deciding the most appropriate software development methodology:

1. Suitable for smaller teams (3-4 members)

2. Flexibility with changing requirements based on client needs

3. Complexity, How difficult the system is to use

4. Budget, The time and costs involved with production

5. Risk, How predictable is the production process

3.2. Research

3.2.1. Research into Waterfall methodology including bespoke pros and cons

The waterfall methodology, also called the plan-driven model, is a project management approach where all the requirements are gathered upfront (Adobe communications team, 2023). Furthermore, (Otoyo, 2023) states that a phase must be completed before moving on to the next stage. This results in the project manager and developers needing to be extremely thorough with their requirements gathering (I.e. user stories) and setting deadlines for each phase/checkpoint. The waterfall method is usually used in large system engineering projects spanning several sites. The phases within the waterfall methodology consist of (in order) requirements, design, implementation, verification and maintenance. It is worth noting that these phases vary slightly depending on the literature, however, they follow the same general idea regarding progression.

The advantage of using this methodology in this scenario is that it allows developers who join midway through the project to quickly catch up, as all the information they'll require will be contained within the requirements document (Adobe communications team, 2023). A further advantage is that because the requirements are provided in a textual case study and are unlikely to change, the group would benefit from the edge of having a well-defined structure where progress can easily be measured, this, however, could also be a massive risk as the team will be conducting weekly meetings with the client where ambiguities and further clarification could arise not previously accounted for. It is also worth noting that the cost is relatively fixed as the requirements are not likely to change.

A disadvantage to following the waterfall methodology in this case, as stated above, is that because the requirements are fixed, it could exclude the client (Lucid Content Team, 2023), resulting in the weekly meetings being deemed useless and not impactful on the quality of the end product (online portal). Another disadvantage is that the testing phase is left towards the end, and considering this model follows a sequential methodology where previous steps are not revisited, this is very risky if certain functionality is missed, or fundamental bugs are detected.

3.2.2. Research into RAD development including bespoke pros and cons

Rapid Application Development (RAD) is a quick methodology that benefits from continuous client feedback and prioritizes swift prototyping over meticulous planning, allowing for iterative updates unlike the inflexible waterfall model (Chien, 2020). RAD is best suited for small teams (3-9 members).

Its advantage lies in providing an initial prototype to the client for review, facilitating swift changes, and reusing components for faster development (Egeonu, 2022).

However, RAD's drawbacks include its reliance on a team of developers and designers for frequent changes and difficulties in onboarding new team members due to insufficient documentation, potentially slowing development.

3.2.3. Research into Agile methodology including bespoke pros and cons

Agile takes an adaptive approach where planning is incremental, meaning it is relatively easy to change the software based on client requirements (Otoyo, 2023). Agile takes into account Scrum, where a team would work on a short sprint spanning 1-4 weeks, kanban, where work is visually represented on a physical board (Atlassian, 2023) and extreme programming (XP).

The advantage to an agile approach, in this case, is that because changes can be made frequently due to its adaptability, it will take advantage of weekly client meetings resulting in a more bespoke final software that satisfies the client, this is also means bugs can be attended to after the development stage has passed. This differs from the waterfall methodology, where all the phases are rigid. Another advantage is that deployment time will be quicker when compared to waterfall as development can be split into short sprints where features are released based on priority rather than all at once. This allows the system to be usable earlier and would be beneficial in this scenario as the client is keen on seeing progress as early as possible.

A disadvantage to this approach is that it is challenging to set rigid timelines and fixed costs due to its limited predictability. This will require clear communication with the client that the timeline and costs will vary depending on what they wish to change/add. Another disadvantage is that, similar to RAD, if a member were to join the project mid-way through, they would struggle to catch up due to the lack of documentation

3.2.4. Research into Integration and Configuration including bespoke pros and cons

Integration and configuration is used when an existing system’s modules are being modified and reused. This may, however, require the customisation of components to be compatible with the new system. This approach is usually only used in large enterprises as they have the capabilities to deal with the high complexity of module modification. Integration and configuration consist of the following stages; Requirements specification, Software discovery and evaluation, Requirements refinement, Application system configuration and Component adaptation and integration (collegenote, 2023).

An advantage to this approach is that it can be cost-effective since you avoid having to develop software from scratch, this leads to another advantage where development time is significantly shorter since modules are reused.

A disadvantage is that an early prototype is impossible (teamques10, 2016). This is because, in bottom-up integration, multiple modules would be required to work together and would not be functional independently. A further disadvantage is that testing and troubleshooting are complex as the modules implemented may use logic unfamiliar to the current developers. This would be even more difficult if documentation were limited.

3.2.5. Research into Spiral including bespoke pros and cons

The spiral software development model, introduced by Boehm (1988), is an iterative and incremental approach. Each development cycle yields a more complete software version. The model involves four phases: planning, risk analysis, engineering, and evaluation.

Planning identifies goals, objectives, and project risks. The risk analysis phase prioritizes and minimizes these risks. The engineering phase designs, codes, and tests the software, and requirements are reassessed. The evaluation phase ensures the program meets client requirements, refining goals for future iterations.

The Spiral Model manages complex, evolving projects but incurs higher costs due to its complexity. It emphasizes risk management, potentially preventing costly delays (Alshamrani et al, 2015). The model's flexibility allows for significant adaptability and client collaboration, with changes implemented in subsequent iterations.

However, its complexity can lead to vast, often unpredictable costs (Rana, 2021). Extensive documentation, while facilitating new team member integration, may be hard to interpret, increasing complexity (Boehm, 1988).

3.3. Comparisons, final decision and justification

Model	**Strengths**	**Weaknesses**
Waterfall	• Predictable (costs anddeadlines) • Well documented – more manageable for a new teammember to catch up • Simple and easy to use • Good for smaller projects	• Changing the scope would mean starting an entirely new project • Working software is only produced towards the end of the software • Minimal risk analysis • It does not work for complex system models
Agile (Scrum/Kanban /RAD/XP)	• Allows for change in requirements (helpful in thiscase as client communication is regular) • Reduces risk as deploymentis fragmented • Allows for stakeholder/client engagement	• Cost is not defined until the end product is developed • Delivery of the product is fragmented (deployed bit by bit), this can also be an advantage • Documentation can get confusing, and new team members may struggle to catch up
Integration and Configuration	• Reduces cost as somemodules are already developed • It should only be used if the old system is your own	• Upgrading, testing, and troubleshooting may be difficult as developers may not understand the code
Spiral	• The first iteration of the software is produced early • Risk is analysed with everyiteration leading to risk reduction	• Documentation can get highly complex • Recommended for only larger scale products • Time management is complex • as the number of iterations is unknown

Figure 1: Comparing SD methodologies

Based on the comparisons above, the waterfall model was discarded due to its rigidness and the lack of risk analysis since it would cause the final project to be delivered all at once at the end, rather than in fragments which would take advantage of client interaction. After team and client discussions, the final decision was between Kanban and Spiral due to its flexibility and risk reduction. The team agreed on Kanban as it would lead to cleaner documentation, the ability to set and meet strict deadlines and its suitability for smaller scaled projects. Kanban would also be suitable for Trello (project management tool). (Written post development) The methodology was followed successfully using Trello boards and regular client meetings.

4. Requirements

4.1. Functional and Non-functional requirements

Functional Requirements	**Non-functional requirements**
Maintain employee details and salary, depending on the two types of employees: permanent employees (paid monthly) and temporary employees (paid daily).	The portal should be accessible from different devices (i.e., phones) and web browsers (i.e. Firefox) and support multiple users simultaneously.
The buying and selling price of crabs according to their categories (A, B, C, and D) and calculate the revenue for each depending on the circumstance.	The portal should be user-friendly and easy to navigate, with clear instructions and labels.
Identify the fisherman delivering good quality crab to increase profit	The portal should have backup and recovery. functions if the system fails or data is lost.
Record the death rate of crabs while making hard-shell crabs to soft-shell crabs.	It should be scalable for the growing. business needs of the company.
Calculate the revenue each month/year, depending on the buying and selling price of hard- and soft-shell crabs.	The portal should be secure and only accessible to authorised users.
	The online portal should have good performance and response time, even when handling large amounts of data, to ensure a smooth user experience for the company's owner and employees.

Figure 2: Functional and non-functional requirements

4.2. Use Case diagrams

The case diagrams provided below outline all the features of the application respective to the appropriate user.

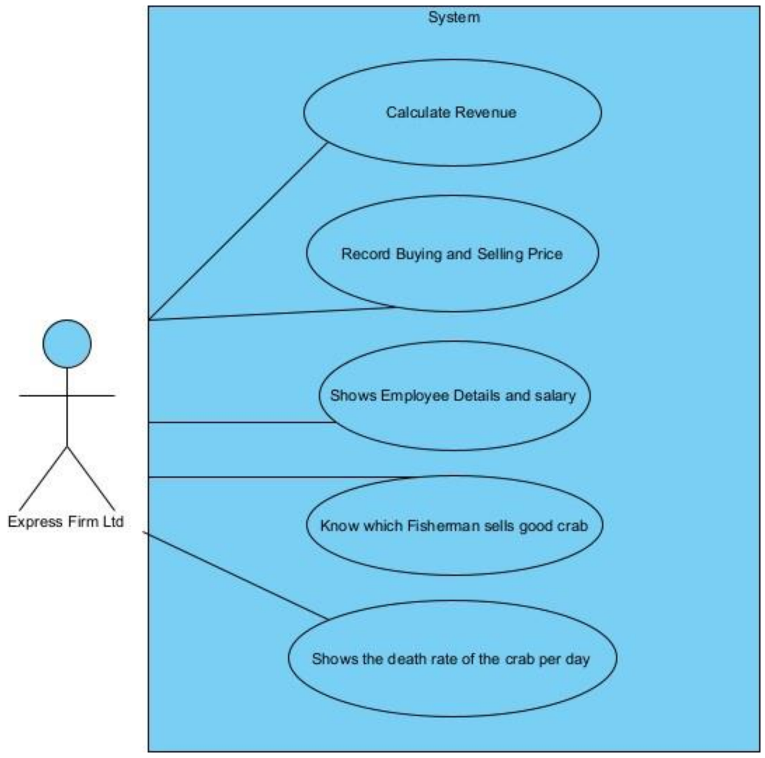

System
Calculate Revenue
Record Buying and Selling Price
Shows Employee Details and salary
Know which Fisherman sells good crab
Shows the death rate of the crab per day
Express Firm Ltd

Figure 3: First iteration use case diagram

The first use case diagram was not the right one for the application. We created this diagram and asked the client for feedback; he then specifically told us that the actor, in this case, was not thecompany itself. So in order to create valuable and correct use case diagrams, each actor will have its own diagram, and then a 3rd diagram will be designed, including both of these actors and their interactions on the system.

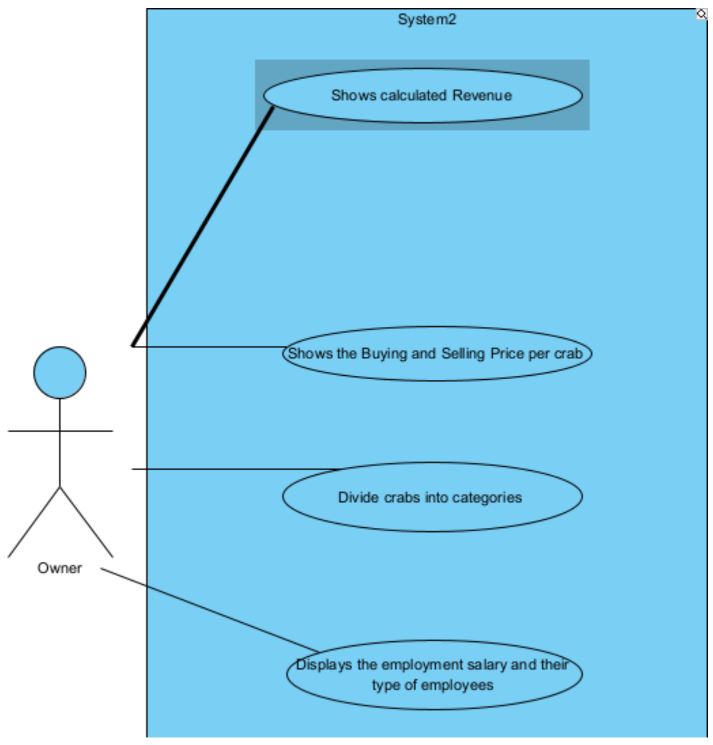

System2
Shows calculated Revenue
Shows the Buying and Selling Price per crab
Divide crabs into categories
Displays the employment salary and their type of employees
Owner

Figure 4: Second iteration use case diagram

The 2nd use case diagram was created using owner as an actor and explaining his functions in the company. The most important actions of the owner in the company are showing calculated Revenue, Showing the buying and selling price for crab, dividing them into categories and displaying the employee salary and the type of employers to him.

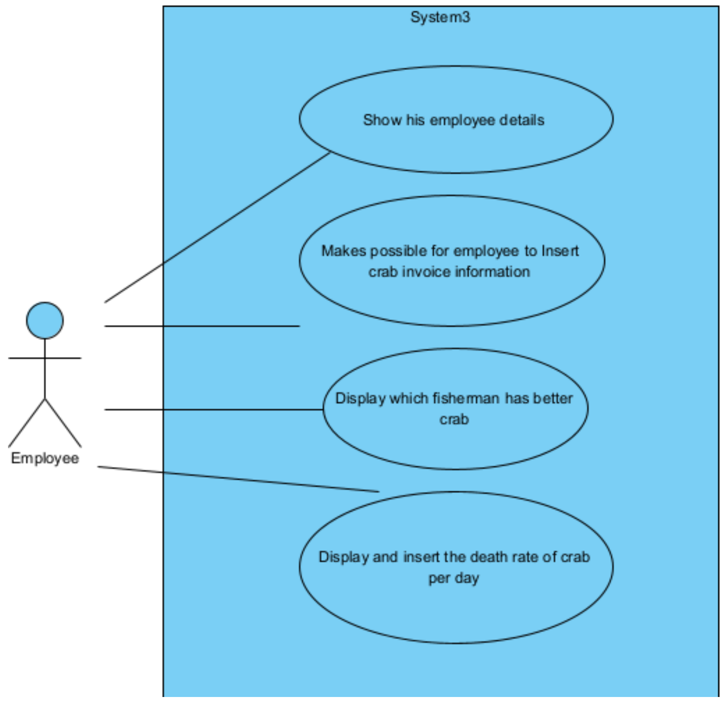

System3
Show his employee details
Makes possible for employee to Insert crab invoice information
Display which fisherman has better crab
Display and insert the death rate of crab per day
Employee

Figure 5: Third iteration use case diagram

The 3rd use case diagram (see figure 5) shows the employee and his functions like displaying his employee details, make possible for him to insert crab invoice information, display which fisherman has a better crab quality and the crab death rate per day.

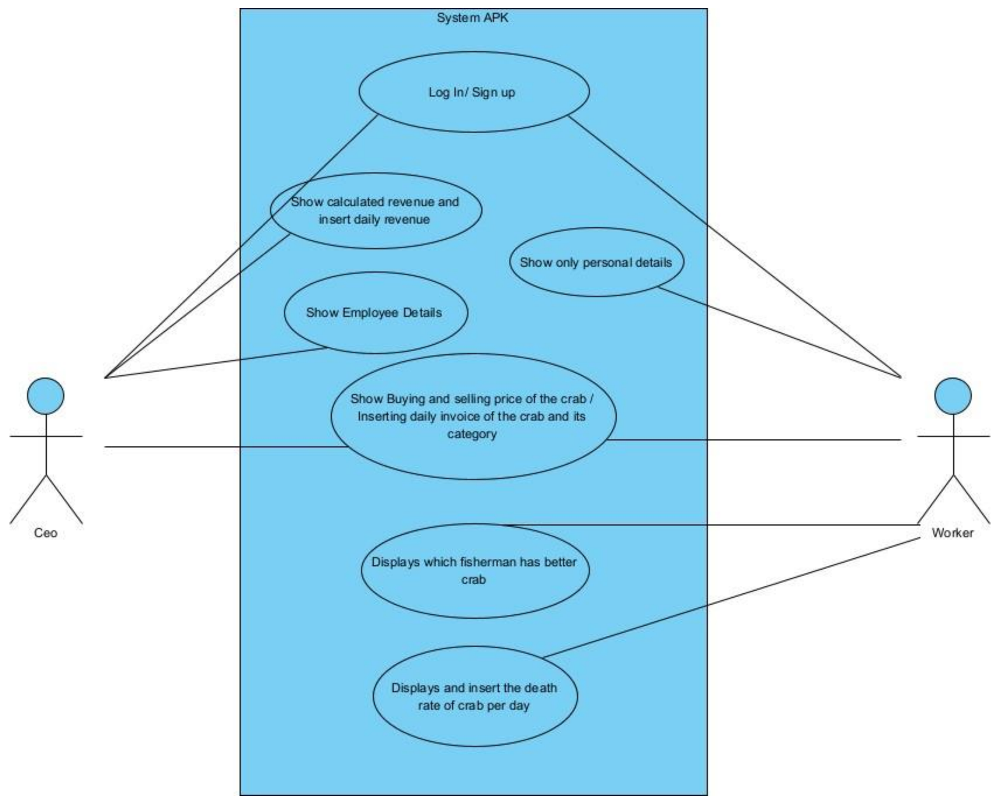

System APK
Log In/ Sign up
Show calculated revenue and insert daily revenue
Show only personal details
Show Employee Details
Show Buying and selling price of the crab / Inserting daily invoice of the crab and its category
Displays which fisherman has better crab
Displays and insert the death rate of crab per day
Ceo
Worker

Figure 6: Fourth iteration use case diagram

The 4th use case diagram (see figure 6) shows the employee and his functions like displayinghis employee details, make possible for him to insert crab invoice information, display which fisherman has a better crab quality and the crab quality death rate per day.

4.3. User Personas / user stories

As an Owner, I want to see the calculated revenue of each month and each year 1

As an Owner, I want to record the buying and selling price of each crab, also divided into the category of the crab quality from A to D 2

As an Owner, I want to see all employees details including their email, password, first name, last name, status and salary 3

As an Employee, I want to see my personal details on the application 4

As an Employee, I want to record the buying and selling price of each crab, and create invoice for that 5

As an employee, I want to know the fisherman ID, the quality of each crab bought by the fishermen and the death rate of the crab per day 6

Figure 7: User Stories sorted by priority (1 = HIGH)

5. Legal, social, ethical, economic and commercial considerations

5.1 Research

5.1.1. Research into ACM/IEEE code of ethics

The Software code of ethics as defined by (Gotterbarn, Miller and Rogerson, 1997) states eight key principles:

1. PUBLIC – Professionals in software engineering should consistently operate in alignment with societal benefit.

2. CLIENT AND EMPLOYER – Software professionals should conduct themselves in ways that serve their client and employer's interests, while ensuring it aligns with societal welfare.

3. PRODUCT – Software professionals should strive for their products, including any changes made, to reflect the pinnacle of professional excellence.

4. JUDGMENT – Software professionals should uphold their professional judgment's integrity and autonomy.

5. MANAGEMENT – Individuals managing or leading software engineering endeavors should endorse and advocate for an ethical management style in software development and upkeep.

6. PROFESSION – Professionals in the software engineering field should contribute to upholding the profession's reputation and integrity, always considering societal benefit.

7. COLLEAGUES – In their interactions with peers, software professionals should demonstrate fairness and offer their support.

8. SELF – Software professionals should engage in continuous professional learning and endorse an ethical approach to practicing their profession.

5.1.2 Research into BSC code of conduct

The BSC code of conduct as defines by (BCS, 2022) states four key principles:

1. You make IT for everyone
2. Show what you know, lean what you don't
3. Respect the organization or individual you work for
4. Keep IT real, Keep IT professional. Pass IT on

5.1.3 Research into DPA 2018

The UK Data Protection Act 2018, encompassing the General Data Protection Regulation (GDPR), regulates personal data use and protection. It grants individuals rights such as access, rectification, erasure, and portability of their data. Controllers and processors must handle data lawfully and transparently, with robust security measures. The Act establishes the Information Commissioner's Office (ICO) for enforcement and fines. It allows data processing for law enforcement and intelligence, balancing privacy and national security. It emphasizes accountability, with organizations required to prove compliance.

5.1.4 Research into Computer Misuse Act 1990

The Computer Misuse Act of 1990 as defined by *(legislation.gov.uk, 1990),* criminalises three main actions: unauthorised access to computer material, unauthorised access with intent to commit further offences, and unauthorised modification of computer material. These provisions cover acts such as hacking, creating or spreading computer viruses, and cybercrimes that involve stealing sensitive data or damaging systems. Penalties depend on the severity of the offence, ranging from fines to imprisonment. The Act has been critiqued for its broad language, potentially ensnaring benign activities like penetration testing. However, it remains a key legal instrument in the UK's fight against cybercrime. The Act was notably updated in 2015 under the Serious Crime Act to include unauthorised acts causing serious damage.

5.1.5 Research into Copyright, Designs and Patents Act 1988

The Copyright, Designs and Patents Act of 1988 *(Legislation.gov.uk, 1988)* provides legal protection for creators, granting them exclusive rights over their original works, designs, and inventions. It covers literary, dramatic, musical, artistic works, as well as sound recordings, films, broadcasts, and typographical arrangements. The Act also provides exceptions, such as fair dealing, allowing limited use of copyrighted material without permission. It further establishes the concept of moral rights, allowing creators to be acknowledged and protect their works from distortion. Lastly, it outlines civil and criminal penalties for copyright infringement.

5.1.6 Research into GDPR

The European Union's General Data Protection Regulation (GDPR), enacted in 2018, strengthens data protection for individuals within the EU. It mandates transparency about data collection, use, and storage, and gives individuals control over their personal data. Key provisions include the right to access, rectify, and erase one's data, and to object to or limit its processing. Organisations must also obtain clear consent for data collection, report data breaches promptly, and appoint a Data Protection Officer in certain cases. Non-compliance can result in significant fines. Although an EU regulation, GDPR has worldwide impact due to its applicability to any organisation handling EU citizens' data.

5.2 Evaluations and Considerations (including ethical)

The system is relatively compliant with the Data Protection Act of 2018. One reason for this is that employees are required to and have permissions to keep their details such as salary up to date. Another factor includes the disclosure of data, the system designed ensures the user’s password is not reveled, though it is saved in the database (which may also be a concern as though it is encrypted, a master password can reveal all passwords) it is not shown to any actors/users.
The system is also compliant with the Copyright, Designs and Patents Act of 1988. This is because all content, for example the logo, has been designed by our team and does not infringe any copyright theft.

Creating an online portal for employers and employees to track the buying and selling of crab involves several societal, commercial, and economic considerations:

1. Societal: The portal should promote fair trade practices, sustainable fishing, and ethical treatment of workers. It could educate users about the importance of sustainable crab fishing, the impact of overfishing on ecosystems, and the rights of maritime workers.

2. Commercial: It should be user-friendly, secure, and efficient, enabling accurate tracking of transactions, inventory, and pricing. Integration with other systems like CRM, accounting, and supply chain management could enhance its value to businesses.

3. Economic: The portal could influence market dynamics by providing real-time data on supply and demand, potentially affecting prices. Furthermore, by improving efficiency, it could reduce costs and increase profitability for businesses.

4. Regulatory Compliance: Considerations should be given to data privacy and protection laws, such as GDPR if operating within or dealing with the EU, to ensure that personal data of employers and employees is handled appropriately.

5. Global Trade: If the platform is to be used internationally, it should take into account different currencies, tax laws, and import/export regulations.

6. Digital Divide: The portal's accessibility to those with limited internet access or digital literacy skills should also be considered.

6. Technical review

6.1 Evaluation criteria

The following was taken into account when researching and deciding the most appropriate technology stack (framework/library/database (backend):

1) Learnability

2) Ease of use

3) Scalability

4) Performance

5) Documentation and community support

6.2 Research into technology stacks

6.2.1 Research into React

React is a JavaScript library used for building UIs. It is generally used for single-paged projects. It also uses Document Object Model (DOM) to help manage UI updates and structure webpages as trees (Adwaith, 2022). This, in turn, improves performance through reducing the number of DOM manipulations required. React is also module-based, meaning UI features can be split up into smaller reusable components (Miller, 2021).

An advantage to using the React library is that it is relatively easy to learn, assuming an intermediate background in JavaScript (criteria 1). Another advantage is that community support is strong due to its popularity (criteria 5). Its performance is also good due to the use of DOM (criteria 4).

A disadvantage to using React is that though community support is high, it severely lacks documentation, this is because the library is constantly changing, hence leading to outdated documentation (Sureka, 2023). A further disadvantage is that though it is based on JavaScript, new developers may find it challenging to use it in combination with HTML.

6.2.2 Research into Flutter

Flutter is an open-source framework typically used for mobile applications (IOS, Android, Linux, etc.). Flutter uses Dart, initially developed by Google. It also provides prebuilt widgets allowing for easier UI development (Amadeo.R, 2017). It is worth noting that flutter allows for real-time updates when code is updated, which allows for more manageable iterations.

The main advantage to flutter is that because it's cross-platform, tests only need to be run on one platform rather than all. Another advantage is that because flutter is coded in Dart, the performance is seen to be high due to its effective compiler.

A disadvantage to using flutter is the steep learning curve in using Dart because of its low popularity and community support (Montaño, 2023). A further disadvantage is that it has limited libraries because it is a new language, and developers have had little time to produce such libraries.

6.2.3 Research into flask

Flask is a framework based on python used for web development, it can be used for a range of cases ranging from simple e-portfolios to APIs. (PythonBasics) states that flask is often called a microframework as it is developed to keep applications simplistic and minimalistic. It also provides vital tools such as URL routing and template rendering.

The main advantage of using flask is that it is based on python, meaning most developers will find it easy to code. This is also partly due to its extensive documentation (Metwalli, 2022) and the variety of online tutorials. Another advantage is that it is lightweight. This is due to the fact that it only provides essential tools and has a small code base. It is also highly modular.

A disadvantage to flask is that it may not have all the tools the developer may require. This means they would have to download such libraries separately. Another disadvantage is that (Panchal, 2022) developers are required to write database queries themselves using SQL. This is due to the lack of Object-Relational Mapping (OMR). This could lead to errors in queries which is essential to take into account in this case, as the online portal being developed will require at least two database tables.

6.3 Evaluation and final justifications

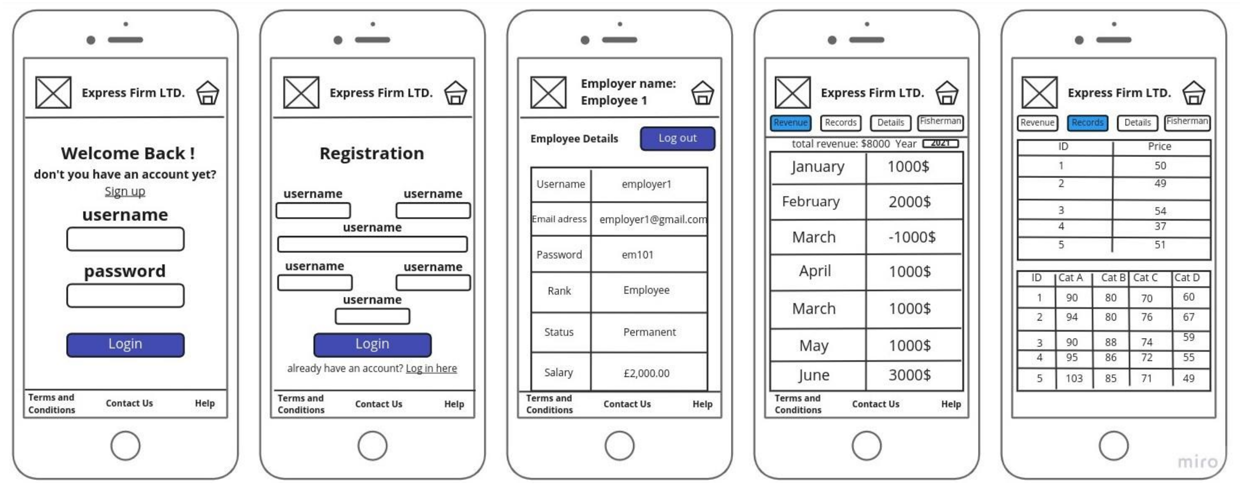
Framework	**Strengths**	**Weaknesses**
React	• Based on JavaScript – easy to code for developers who are already familiar • Large and extensive community • Modular-based – meaning code and UI features can be deployed bit by bit. This also results in reusable components	• If a developer is not familiar with JavaScript, it can be a steep learning curve • It is a library, not a framework, meaning there is a lack of guidelines on how to structure code • It requires additional tools working along it, for example, Webpack
Flutter	• High performance due to dart compiler • Strong community support • Cross-platform – same code works on all devices. This also saves time when testing	• Limited libraries due to it being a relatively new framework • Larger app size as it does not use web technologies • Though it has a strong community, it is relatively small when compared to React and Flask
Flask	• Lightweight as it does not come with many built-in features • It has a focus on modularity, allowing for the scalability and reusability of components • Easy to test due to its lightweight nature	• It may not include all features required by the developer due to its lightweight nature • It does not provide any security features • Not suitable for larger and more complex tasks/programs

Figure 8: Comparing framework/libraries

After extensive team and client discussion, we settled on react for the front end and MySQL for the database. This is mainly because react is better suited for small and medium-sized projects and its large and robust community support. The literature for the library is also extensive. Another reason for this decision is that it strongly focuses on modularity, meaning we could reuse components rather than duplicating code. The decision was also backed by the fact that react has declarative programming, which would significantly reduce bugs and testing intensity.

7. Design

7.1. View 1 – Low Fidelity



Username	employer1
Email adress	employer1@gmail.com
Password	em101
Rank	Employee
Status	Permanent
Salary	£2,000.00

January	1000$
February	2000$
March	-1000$
April	1000$
March	1000$
May	1000$
June	3000$

ID	Price
1	50
2	49
3	54
4	37
5	51

ID	Cat A	Cat B	Cat C	Cat D
1	90	80	70	60
2	94	80	76	67
3	90	88	74	59
4	95	86	72	55
5	103	85	71	49

Figure 9: Low-fidelity view of system

The low-fidelity prototype shown above (see figure 9) was created as an initial prototype to discuss with the client. This was changed multiple times following client meetings as the permissions for the special permissions for the owner were clarified.

7.2. View 2 – High Fidelity

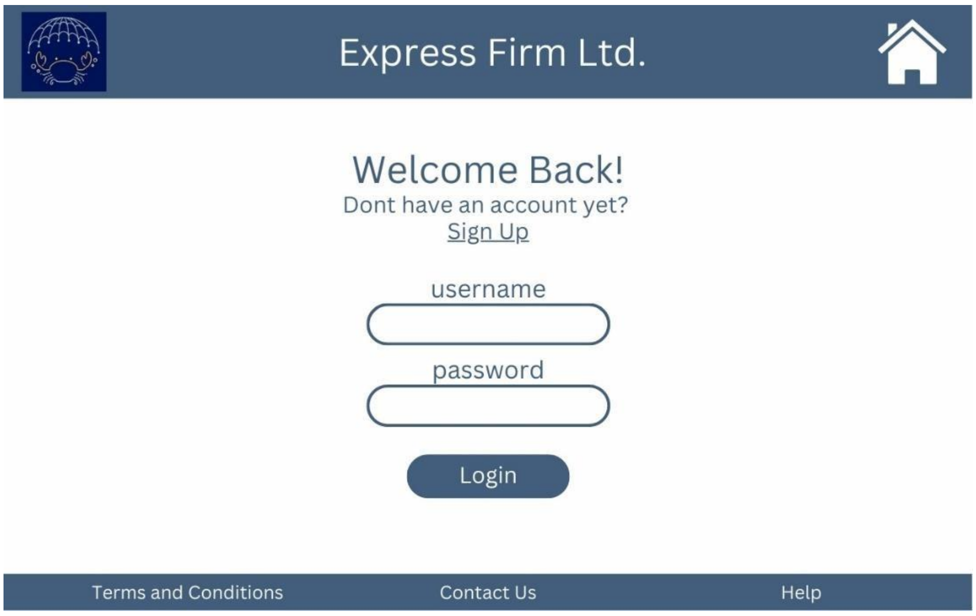


Figure 10: Login Page Prototype

The High-fidelity prototype created using figma, outlines the UI that all actors will interact with when initially logging in to the system.

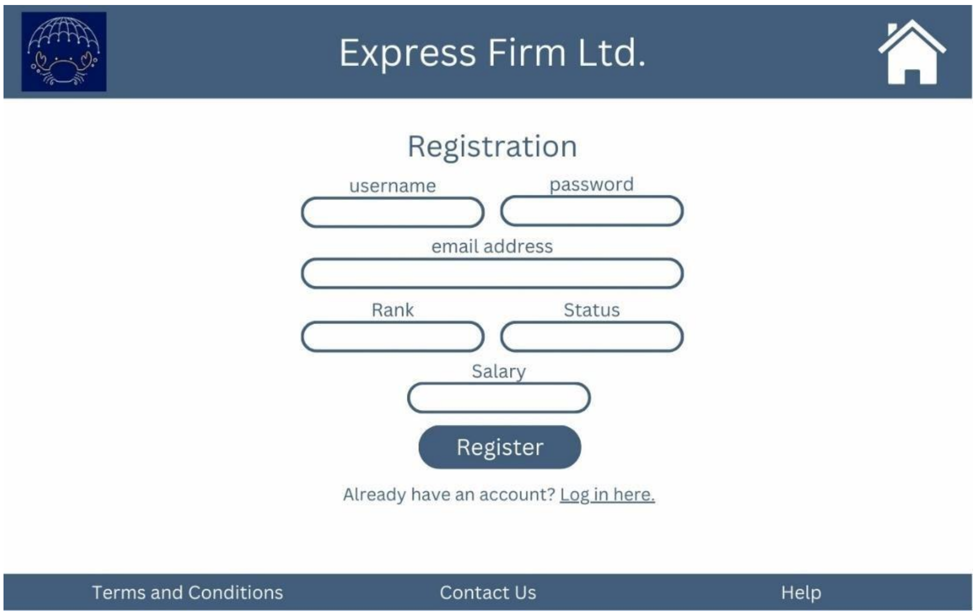
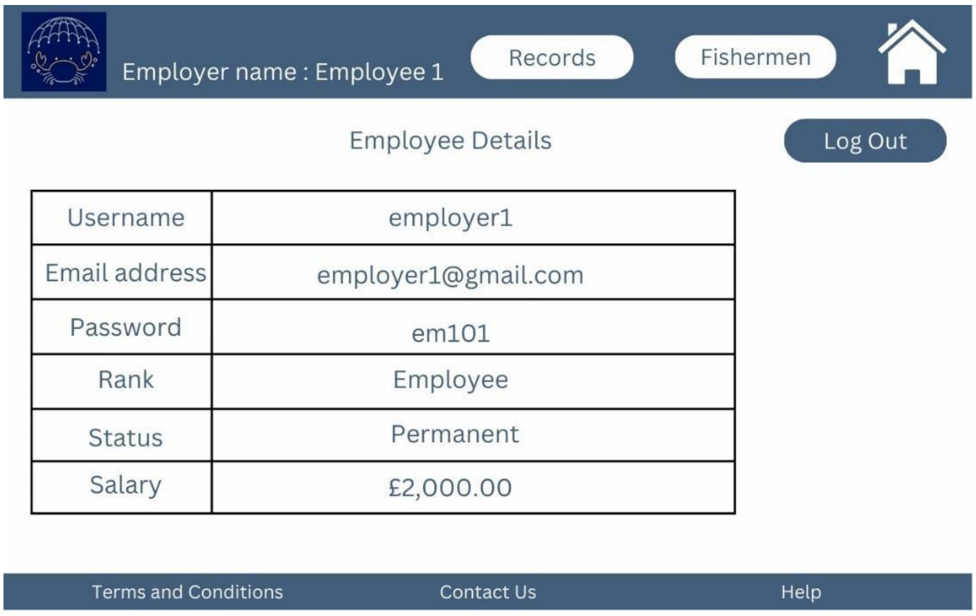


Figure 11: Sign Up Page Prototype.

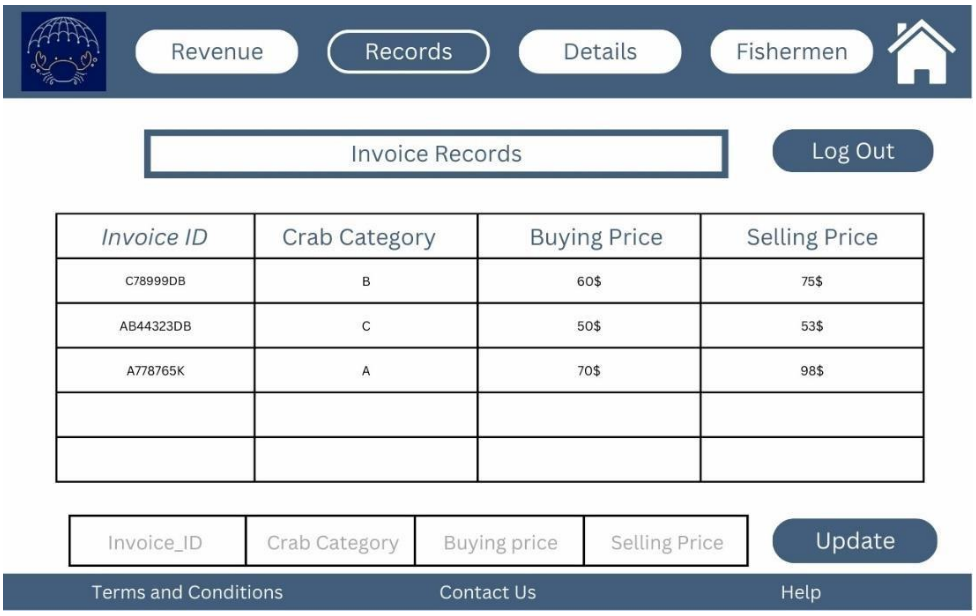
If the user click on the Sign Up button shown in figure 11, they will be directed to this page where they can specify their details as well as rank (employee and owner) which will then set their specific permissions.



Username	employer1
Email address	employer1@gmail.com
Password	em101
Rank	Employee
Status	Permanent
Salary	£2,000.00

Figure 12: Employee Detail Page Prototype.

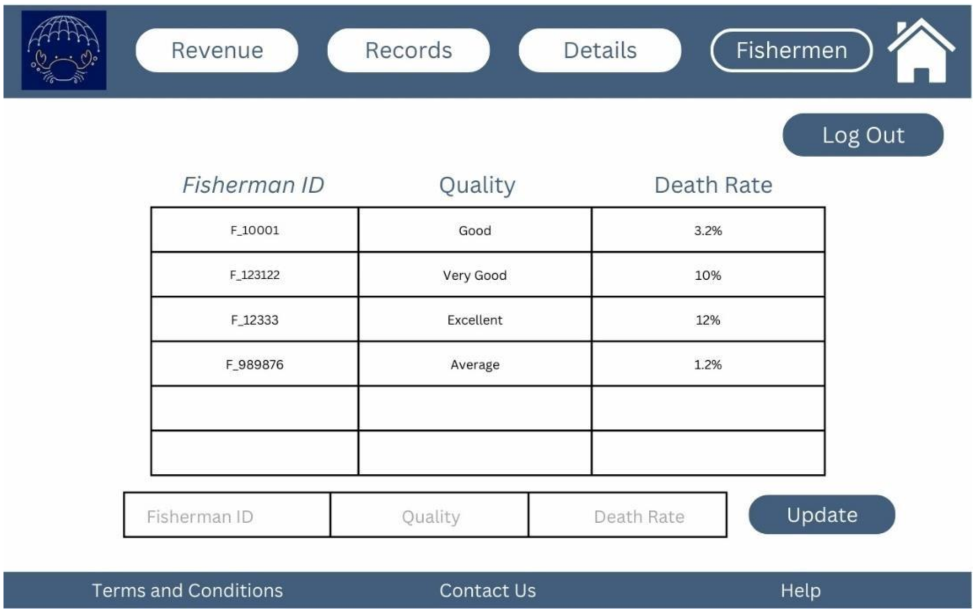
As per the client requirements, the employee is able to view their own details, which include, their salary, employment status and rank. These details can also been seen by the owner (see figure 16).



Invoice ID	Crab Category	Buying Price	Selling Price
C78999DB	B	60$	75$
AB44323DB	C	50$	53$
A778765K	A	70$	98$

Figure 13: Invoice Records Page Prototype.

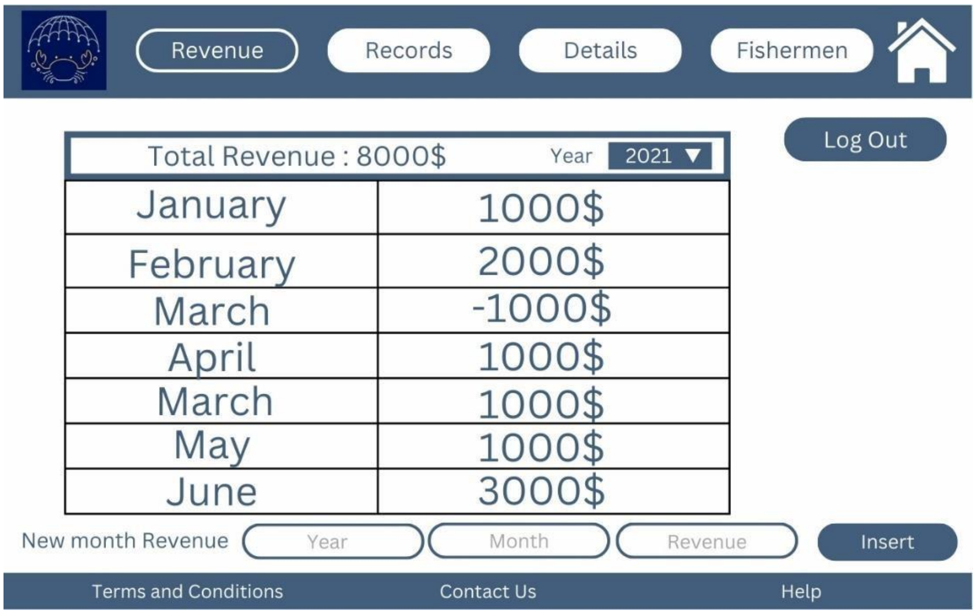
All organisation members are able to view invoices for the buying and selling prices, they are also able to insert and update queries as needed.



Fisherman ID	Quality	Death Rate
F_10001	Good	3.2%
F_123122	Very Good	10%
F_12333	Excellent	12%
F_989876	Average	1.2%

Figure 14: Fisherman and death rate of crab details Page Prototype.

The employees are also able to view the best fishermen based on the quality of crab and death rate. These records can be updated as needed and are available for all organisation members to view.



Revenue Records Details Fishermen

Log Out

Total Revenue : 8000$ Year 2021 ▼

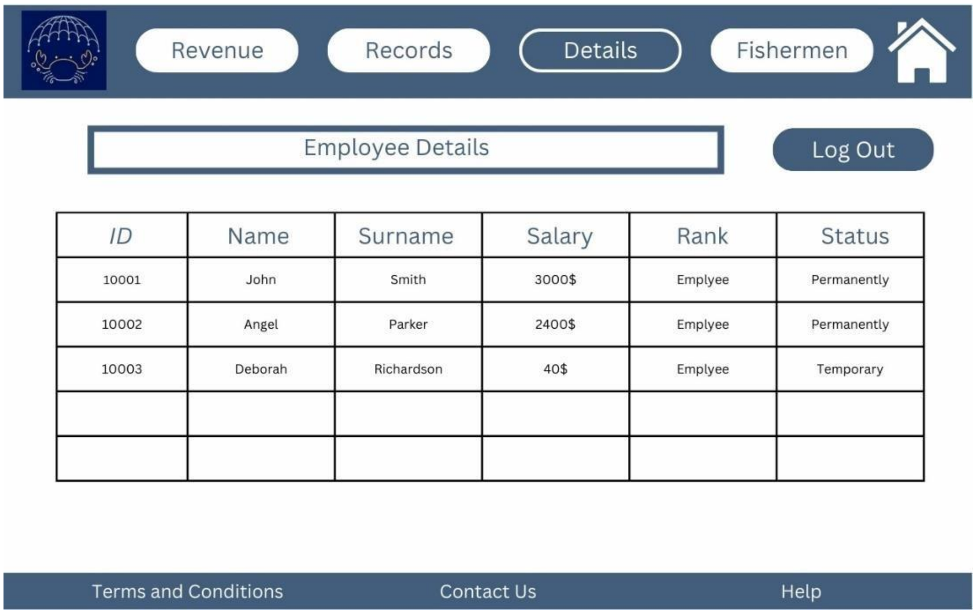
January	1000$
February	2000$
March	-1000$
April	1000$
March	1000$
May	1000$
June	3000$

New month Revenue Year Month Revenue Insert

Terms and Conditions Contact Us Help

Figure 15: Revenue Page Prototype (Only for owner).

The page shown above (see figure 15) is only accessible by the owner who is able to view financial records, as well as filter the year/ month. They can also insert records as needed.



Revenue Records Details Fishermen

Employee Details

Log Out

ID	Name	Surname	Salary	Rank	Status
10001	John	Smith	3000$	Emplyee	Permanently
10002	Angel	Parker	2400$	Emplyee	Permanently
10003	Deborah	Richardson	40$	Emplyee	Temporary

Terms and Conditions Contact Us Help

Figure 16: All employers details Page Prototype (Only for owner).

The owner also has special permissions which provides him with a page displaying details for all employees (see figure 16), these details include, employee ID (PK), rank, status and full name.

7.3. View 3 – Database design

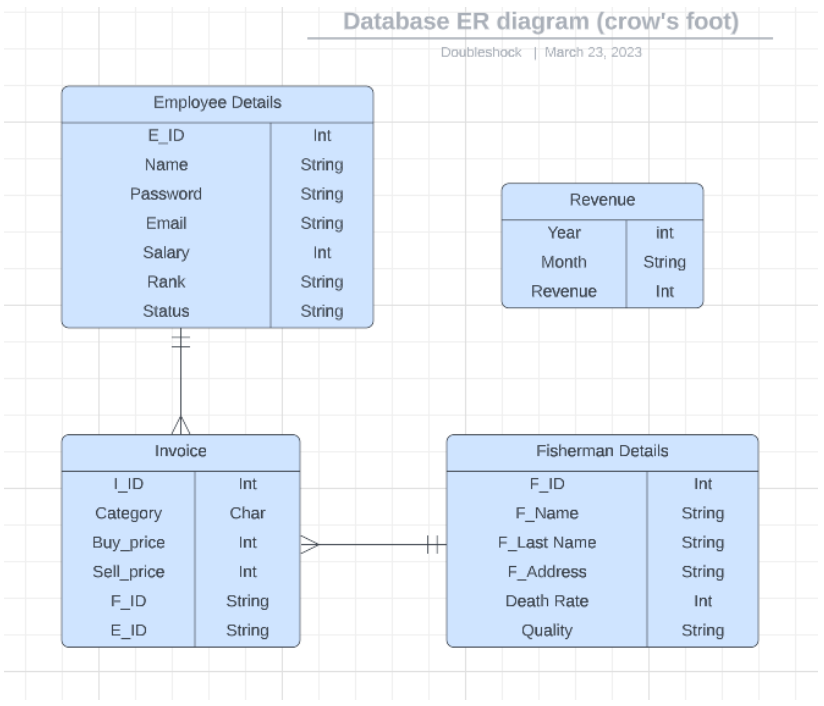
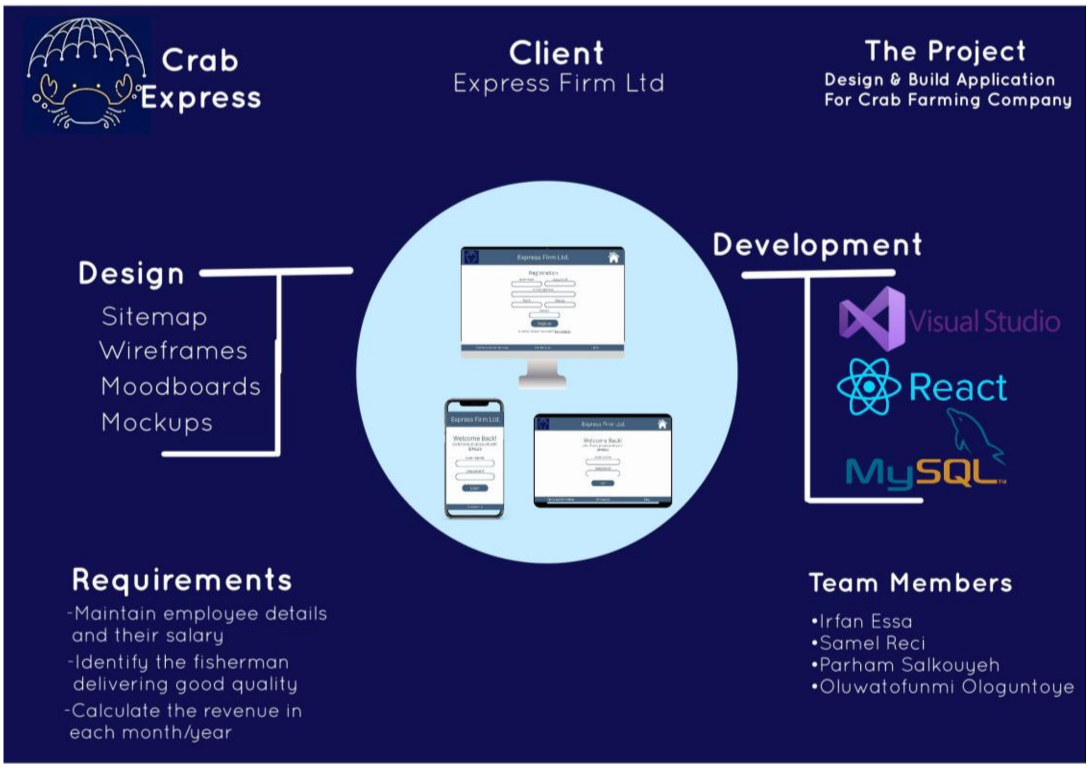


Figure 17: Database ER relation diagram.

Database design is critical for understanding an application's backend. It includes table creation based on a relationship diagram. The Employee details table is essential for login and signup pages, storing employee data and linking to the Invoice table via its primary key, E_ID. The Invoice table, updated with each transaction, ties to Fisherman Details and Employee tables, showing invoice ID, crab details, and fisherman ID. Fisherman Details table stores information about fishermen and crab details, linked to the Invoice table through its primary key, F_ID. The Revenue table, updated only by the CEO, presents monthly and yearly revenue.

7.4 Poster



Crab Express

Client
Express Firm Ltd

The Project
Design & Build Application
For Crab Farming Company

Design
Sitemap
Wireframes
Moodboards
Mockups

Development
Visual Studio
React
MySQL

Requirements
-Maintain employee details and their salary
-Identify the fisherman delivering good quality
-Calculate the revenue in each month/year

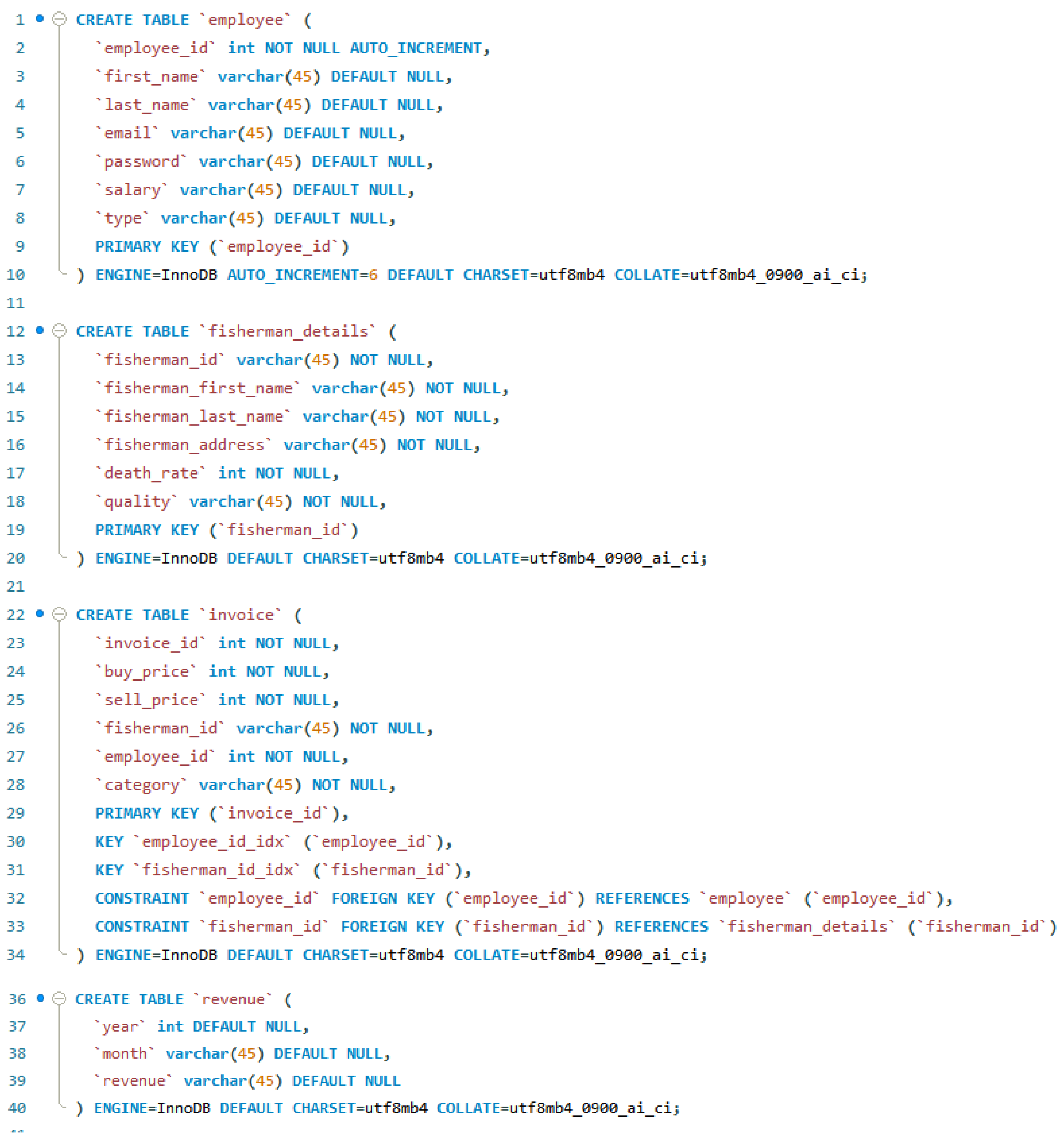
Team Members
•Irfan Essa
•Samel Reci
•Parham Salkouyeh
•Oluwatofunmi Ologuntoye

Figure 18: Poster

https://ibb.co/VN4RcHq

8. Development & Testing

8.1 Database Tables



```sql
CREATE TABLE `employee` (
  `employee_id` int NOT NULL AUTO_INCREMENT,
  `first_name` varchar(45) DEFAULT NULL,
  `last_name` varchar(45) DEFAULT NULL,
  `email` varchar(45) DEFAULT NULL,
  `password` varchar(45) DEFAULT NULL,
  `salary` varchar(45) DEFAULT NULL,
  `type` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`employee_id`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `fisherman_details` (
  `fisherman_id` varchar(45) NOT NULL,
  `fisherman_first_name` varchar(45) NOT NULL,
  `fisherman_last_name` varchar(45) NOT NULL,
  `fisherman_address` varchar(45) NOT NULL,
  `death_rate` int NOT NULL,
  `quality` varchar(45) NOT NULL,
  PRIMARY KEY (`fisherman_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `invoice` (
  `invoice_id` int NOT NULL,
  `buy_price` int NOT NULL,
  `sell_price` int NOT NULL,
  `fisherman_id` varchar(45) NOT NULL,
  `employee_id` int NOT NULL,
  `category` varchar(45) NOT NULL,
  PRIMARY KEY (`invoice_id`),
  KEY `employee_id_idx` (`employee_id`),
  KEY `fisherman_id_idx` (`fisherman_id`),
  CONSTRAINT `employee_id` FOREIGN KEY (`employee_id`) REFERENCES `employee` (`employee_id`),
  CONSTRAINT `fisherman_id` FOREIGN KEY (`fisherman_id`) REFERENCES `fisherman_details` (`fisherman_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `revenue` (
  `year` int DEFAULT NULL,
  `month` varchar(45) DEFAULT NULL,
  `revenue` varchar(45) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figure 19: Database table queries

These are SQL queries to create four different tables: employee, fisherman_details, invoice, and revenue in a database.

The employee table has seven columns, which include employee_id (primary key), first_name, last_name, email, password, salary, and type. All fields except the employee_id can have null values.

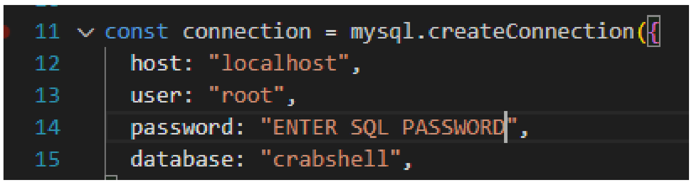
The fisherman_details table has six columns: fisherman_id (primary key), fisherman_first_name, fisherman_last_name, fisherman_address, death_rate, and quality. All fields are required.

The invoice table has six columns: invoice_id (primary key), buy_price, sell_price, fisherman_id, employee_id, and category. This table also has foreign keys that link to the employee and fisherman_details tables using the respective id fields, ensuring relational integrity between these tables.

Finally, the revenue table is simpler with only three columns: year, month, and revenue. All these fields can have null values.

8.2. Code for key features

8.2.1. SQL connection

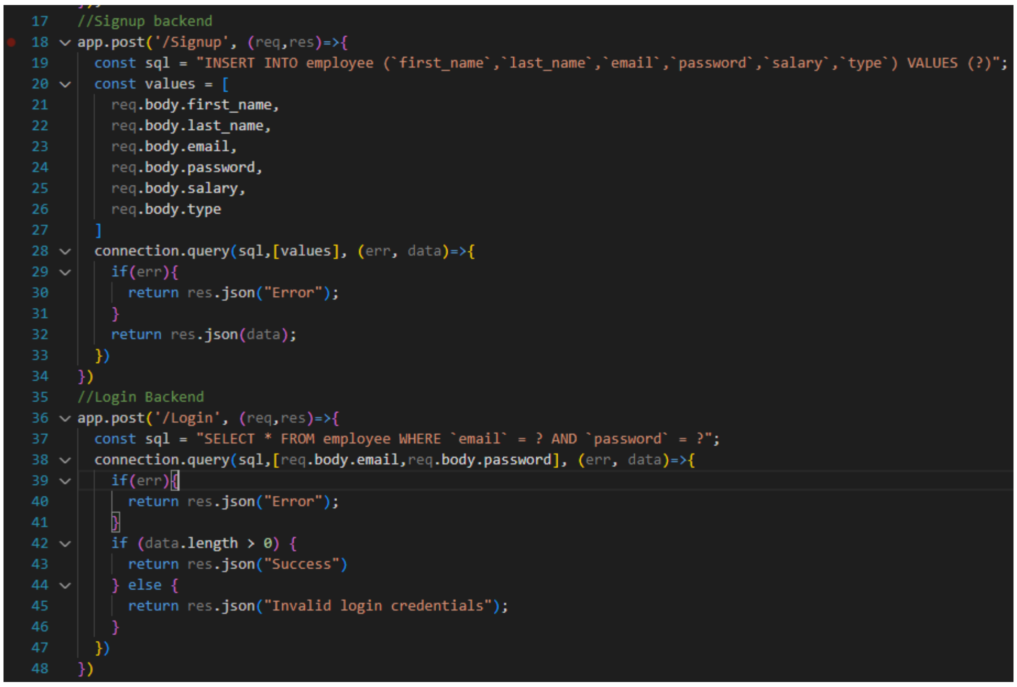


```
const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "ENTER SQL PASSWORD",
  database: "crabshell",
```

Figure 20: Code for SQL connection

This JavaScript code creates a connection to a MySQL database. It uses the mysql.createConnection method with an object specifying connection details: host (localhost), user (root), password, and database (crabshell). This connection can be used to run SQL queries on the 'crabshell' database. Replace "INSERT YOUR PASSWORD HERE" with the actual password.

8.2.2. Login and Registration entries



```
//Signup backend
app.post('/Signup', (req,res)=>{
  const sql = "INSERT INTO employee (`first_name`,`last_name`,`email`,`password`,`salary`,`type`) VALUES (?)";
  const values = [
    req.body.first_name,
    req.body.last_name,
    req.body.email,
    req.body.password,
    req.body.salary,
    req.body.type
  ]
  connection.query(sql,[values], (err, data)=>{
    if(err){
      return res.json("Error");
    }
    return res.json(data);
  })
})
//Login Backend
app.post('/Login', (req,res)=>{
  const sql = "SELECT * FROM employee WHERE `email` = ? AND `password` = ?";
  connection.query(sql,[req.body.email,req.body.password], (err, data)=>{
    if(err){
      return res.json("Error");
    }
    if (data.length > 0) {
      return res.json("Success")
    } else {
      return res.json("Invalid login credentials");
    }
  })
})
```

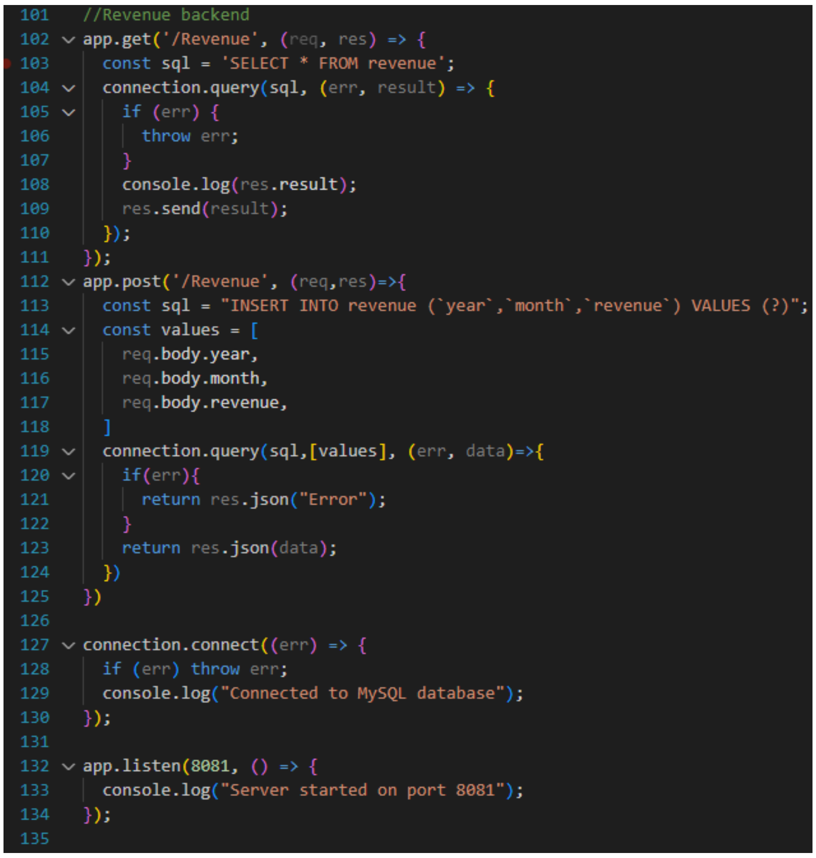
Figure 21: Code for Sign Up and Log In

The /Signup endpoint takes user-provided information from the request body, like first name, last name, email, password, salary, and type, and inserts these values as a new record into the 'employee' database table. The /Login endpoint checks the 'employee' table for a record matching the provided email and password, and responds with a success message for a match, or an invalid credentials message otherwise.

The /Employee_Details endpoint retrieves every record from the 'employee' table and sends this data back to the client. Similarly, the /Records GET endpoint fetches all records from the 'invoice' table. The /Records POST endpoint, on the other hand, creates a new record in the 'invoice' table using data provided in the request body, such as invoice_id, buy_price, sell_price, fisherman_id, employee_id, and category.

Each endpoint includes error handling, where if a database query error occurs, the server responds with an error message. The code represents a typical RESTful API structure for managing employee and invoice data.

8.2.3. Revenue inserts

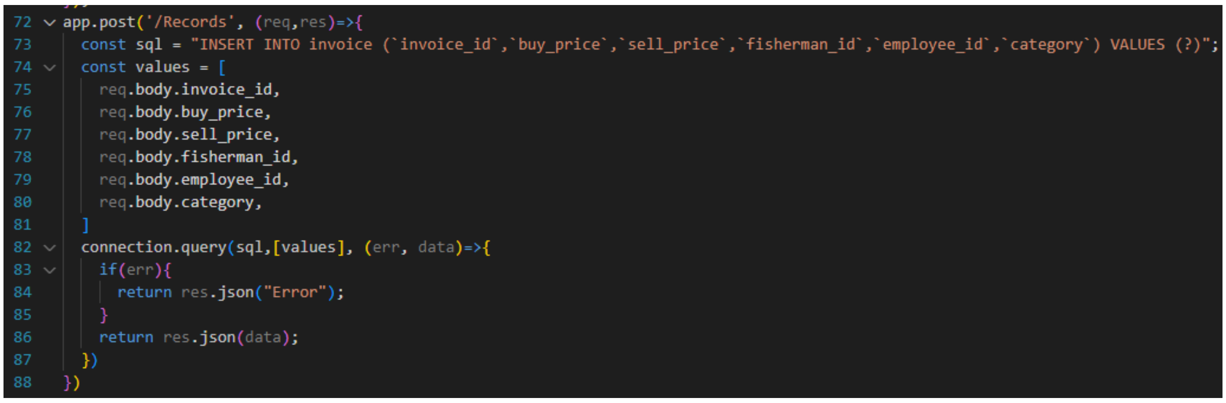


```
//Revenue backend
app.get('/Revenue', (req, res) => {
  const sql = 'SELECT * FROM revenue';
  connection.query(sql, (err, result) => {
    if (err) {
      throw err;
    }
    console.log(res.result);
    res.send(result);
  });
});
app.post('/Revenue', (req,res)=>{
  const sql = "INSERT INTO revenue (`year`,`month`,`revenue`) VALUES (?)";
  const values = [
    req.body.year,
    req.body.month,
    req.body.revenue,
  ]
  connection.query(sql,[values], (err, data)=>{
    if(err){
      return res.json("Error");
    }
    return res.json(data);
  })
})

connection.connect((err) => {
  if (err) throw err;
  console.log("Connected to MySQL database");
});

app.listen(8081, () => {
  console.log("Server started on port 8081");
});
```

Figure 22: Code for revenue inserts

This Node.js code provides endpoints to fetch and insert data into a 'revenue' table in a MySQL database. The GET '/Revenue' endpoint retrieves all revenue records, while the POST '/Revenue' endpoint inserts a new record. If a database connection error occurs, it throws an error. Finally, the server starts listening on port 8081, logging a successful connection to the console.

8.2.4. Invoice Inserts

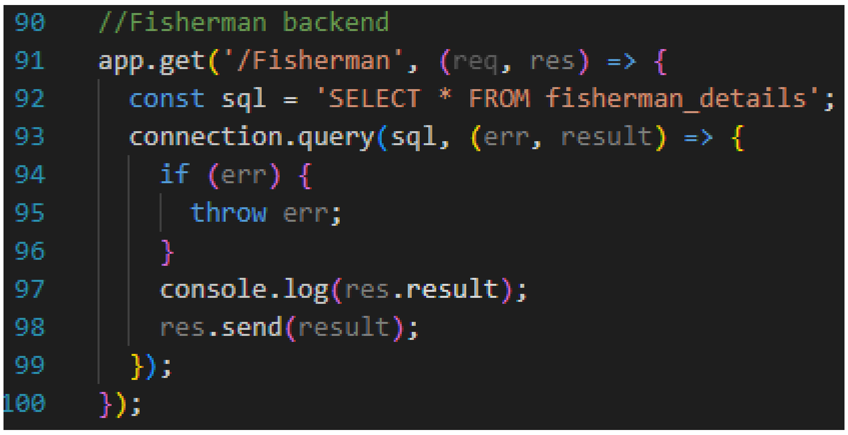


```
app.post('/Records', (req,res)=>{
  const sql = "INSERT INTO invoice (`invoice_id`,`buy_price`,`sell_price`,`fisherman_id`,`employee_id`,`category`) VALUES (?)";
  const values = [
    req.body.invoice_id,
    req.body.buy_price,
    req.body.sell_price,
    req.body.fisherman_id,
    req.body.employee_id,
    req.body.category,
  ]
  connection.query(sql,[values], (err, data)=>{
    if(err){
      return res.json("Error");
    }
    return res.json(data);
  })
})
```

Figure 23: Code for invoice inserts

The code shown in figure 23 inserts a new invoice record into the MySQL database, using data from the request body. Error responses are also handled.

8.2.5. Fisherman details inserts



```
//Fisherman backend
app.get('/Fisherman', (req, res) => {
  const sql = 'SELECT * FROM fisherman_details';
  connection.query(sql, (err, result) => {
    if (err) {
      throw err;
    }
    console.log(res.result);
    res.send(result);
  });
});
```

Figure 24: Code for Fisherman details

This code retrieves all records from the 'fisherman_details' table in a MySQL database. If an error occurs during the operation, it throws the error, otherwise, it sends the results back.

8.3. Final Online Portal Snippets

8.3.1. Login Page

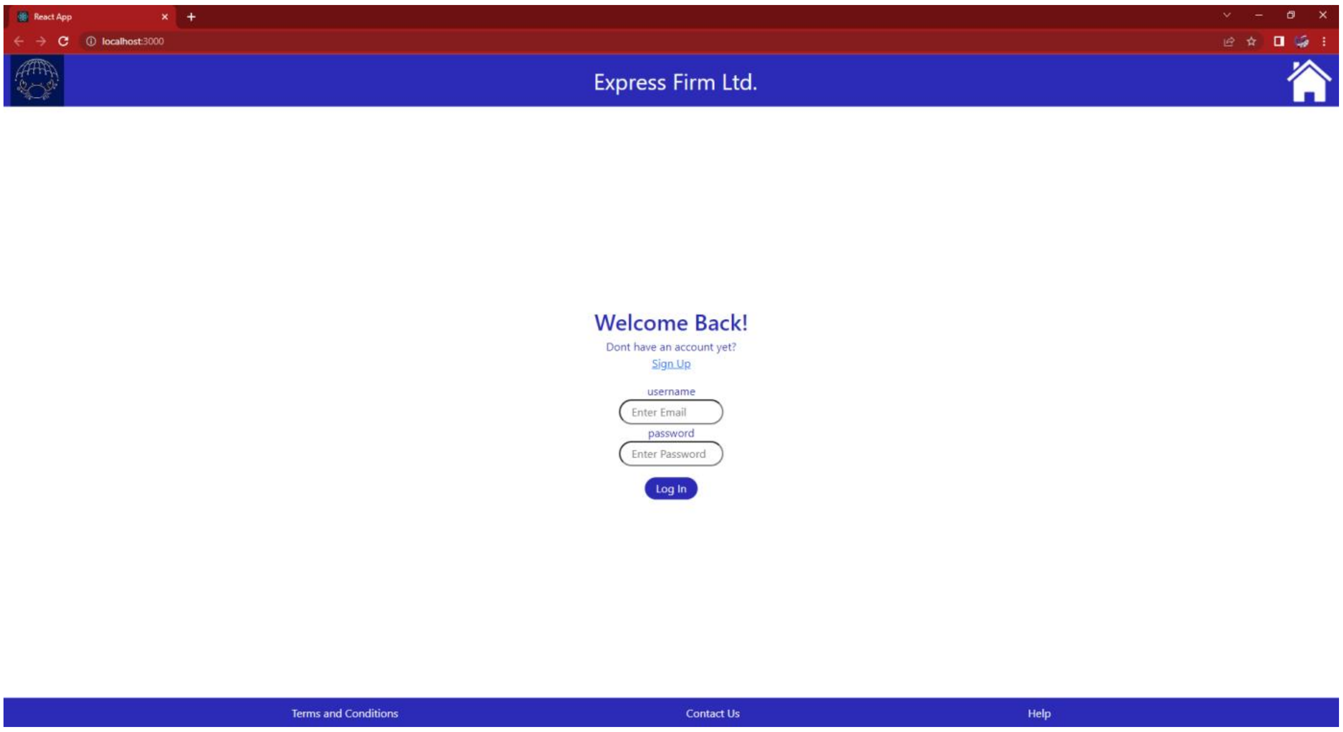


Figure 25: Final Log In page

When the user first loads the portal, they will be shown the log in page (see figure 25). They can click the SignUp button if they have not yet registered

8.3.2. Signup Page

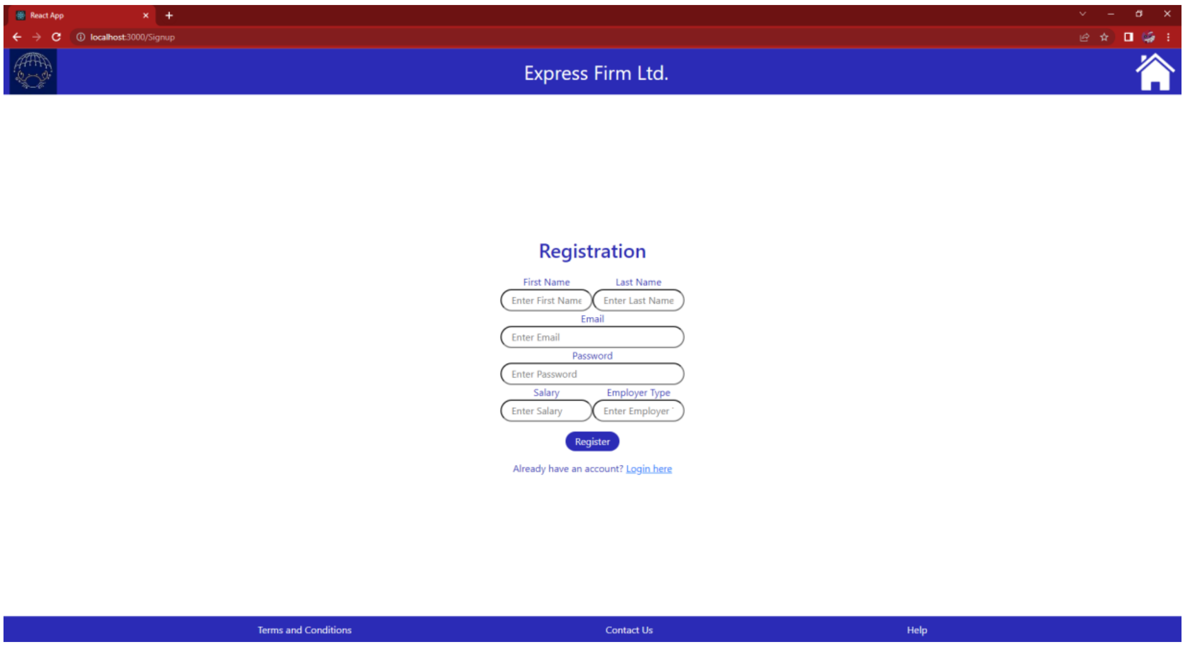
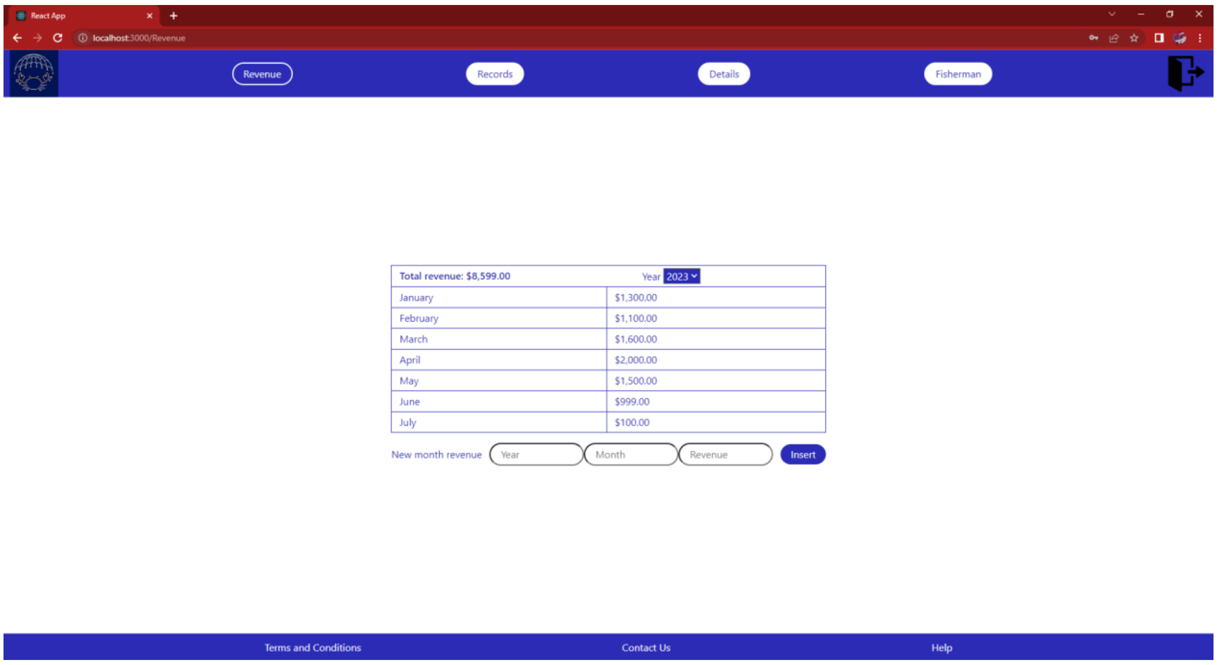


Figure 26: Final Sign up page

The Sign up page prompts the employee to enter the full name, email (which will be used as a user name), password, which is encrypted, their salary and type of employment (temporary or permanent)

8.3.3. Revenue Page

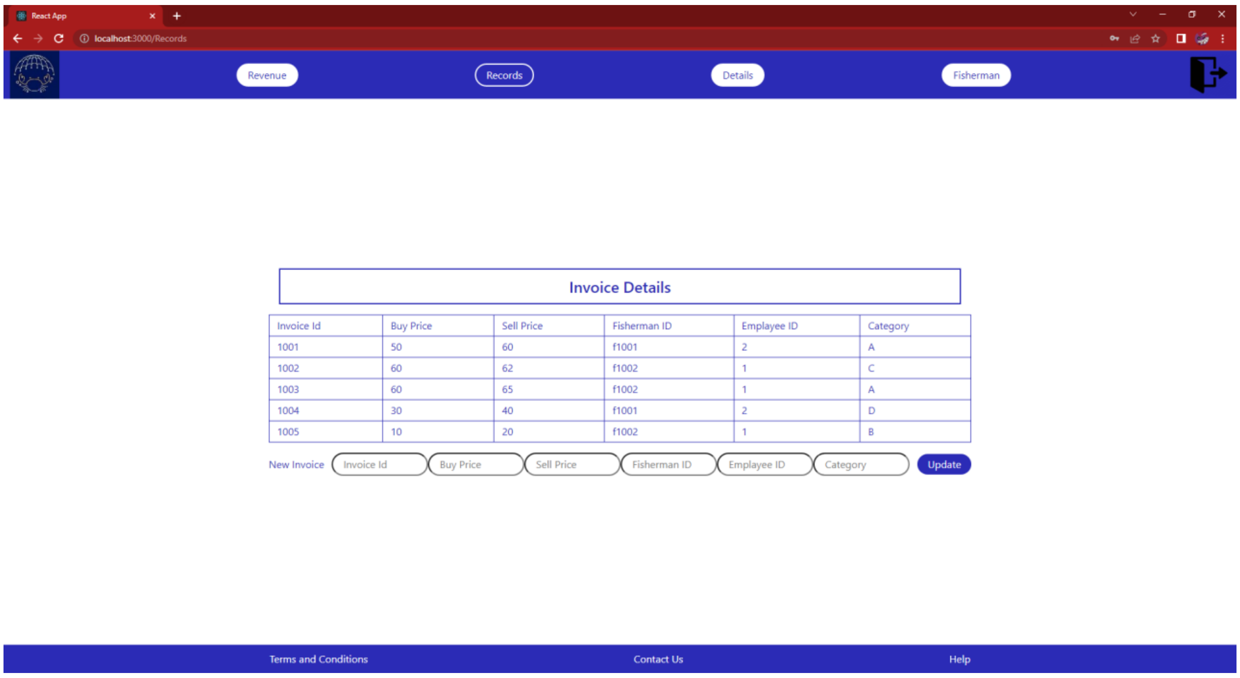


Total revenue: $8,599.00	Year 2023
January	$1,300.00
February	$1,100.00
March	$1,600.00
April	$2,000.00
May	$1,500.00
June	$999.00
July	$100.00

Figure 27: Final Revenue page

The employees can then track the business's revenue using the page shown in figure 27

8.3.4. Records/Invoice Page



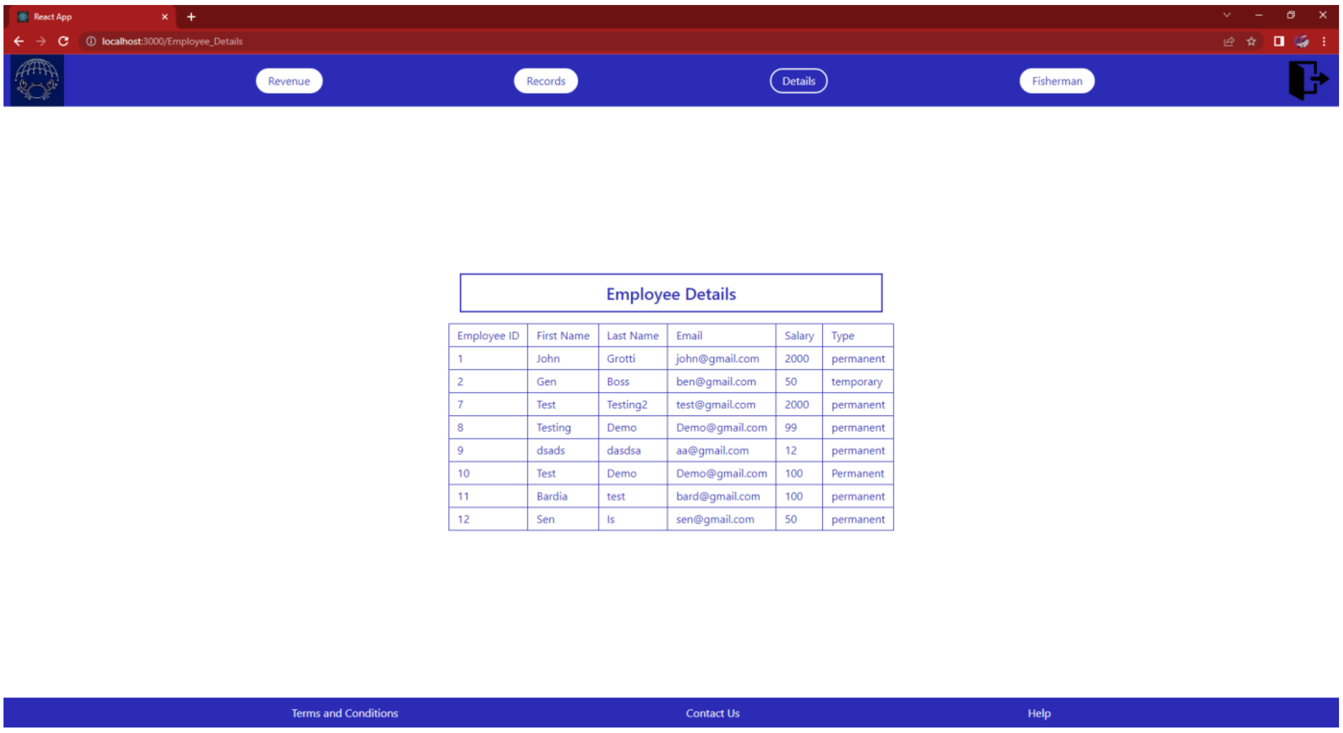
Invoice Details

Invoice Id	Buy Price	Sell Price	Fisherman ID	Emplayee ID	Category
1001	50	60	f1001	2	A
1002	60	62	f1002	1	C
1003	60	65	f1002	1	A
1004	30	40	f1001	2	D
1005	10	20	f1002	1	B

Figure 28: Final Invoice page

They can also insert and view all invoice records

8.3.5. Employee Details Page



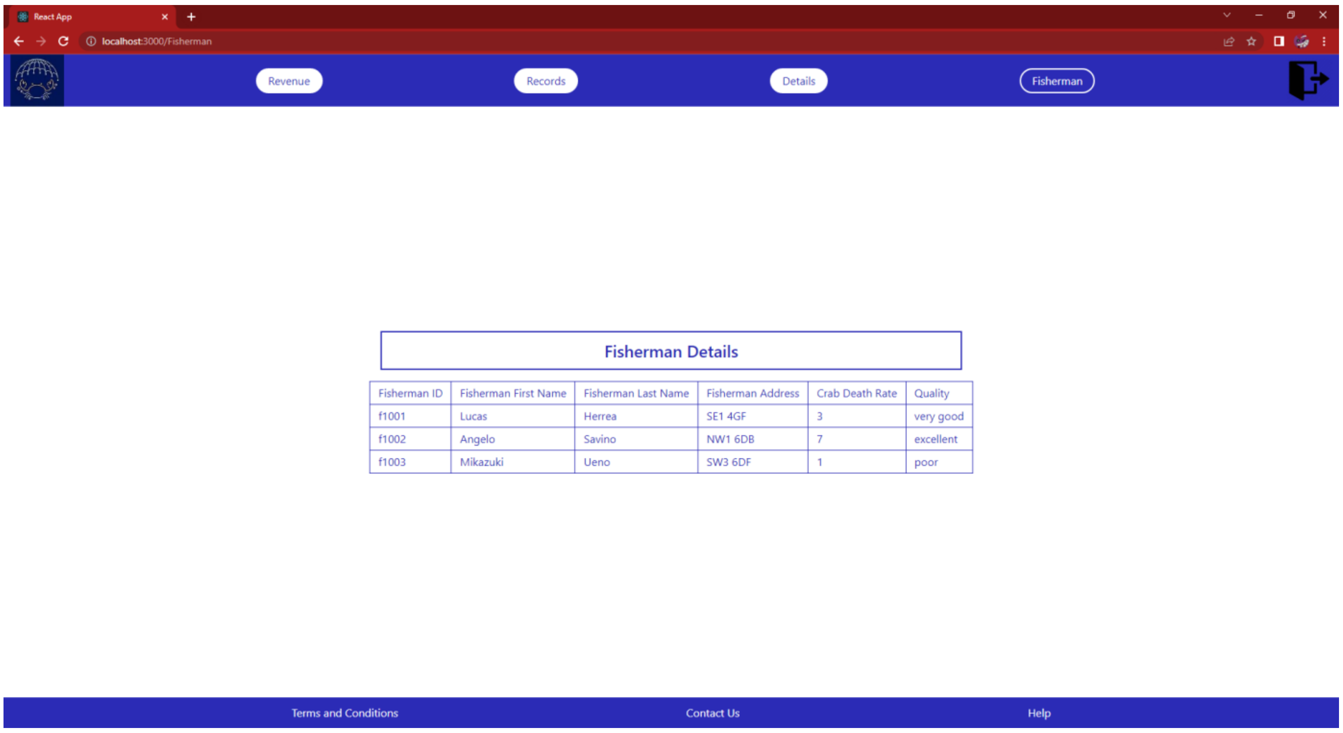
Employee Details

Employee ID	First Name	Last Name	Email	Salary	Type
1	John	Grotti	john@gmail.com	2000	permanent
2	Gen	Boss	ben@gmail.com	50	temporary
7	Test	Testing2	test@gmail.com	2000	permanent
8	Testing	Demo	Demo@gmail.com	99	permanent
9	dsads	dasdsa	aa@gmail.com	12	permanent
10	Test	Demo	Demo@gmail.com	100	Permanent
11	Bardia	test	bard@gmail.com	100	permanent
12	Sen	Is	sen@gmail.com	50	permanent

Figure 29: Final employee details page

As per the client requirements, the employee can view all their details. This also helps with being compliant with the data protection act as it ensures details are up to date.

8.3.6. Fisherman Details Page



Fisherman Details

Fisherman ID	Fisherman First Name	Fisherman Last Name	Fisherman Address	Crab Death Rate	Quality
f1001	Lucas	Herrea	SE1 4GF	3	very good
f1002	Angelo	Savino	NW1 6DB	7	excellent
f1003	Mikazuki	Ueno	SW3 6DF	1	poor

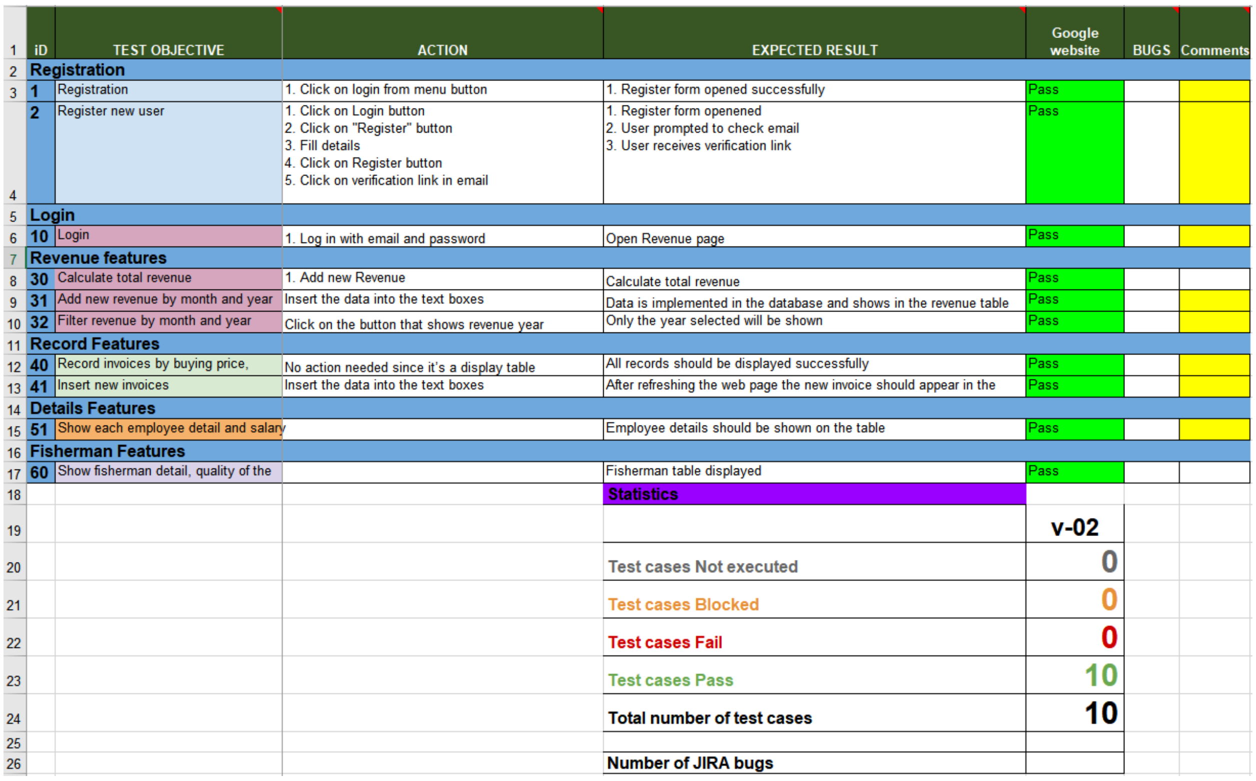
Figure 30: Final Fisherman details page

The employees and owner can also view details of the fisherman they have purchased crab from. This allows them to determine the best supplier.

8.4. Client feedback

Post-client feedback, we swiftly addressed the UI discrepancies to make it align more closely with the original prototype. The navigation bar was redesigned for seamless redirection, enhancing user experience significantly. We also introduced a logout function, an essential yet previously overlooked feature. The client's critique was invaluable, prompting us to refine details and foster a more intuitive and user-friendly platform. This process illuminated the importance of iterative development and open client communication, which will continue to shape our approach in future projects.

8.5 Regression Testing



iD	TEST OBJECTIVE	ACTION	EXPECTED RESULT	Google website	BUGS	Comments
Registration						
1	Registration	1. Click on login from menu button	1. Register form opened successfully	Pass		
2	Register new user	1. Click on Login button 2. Click on "Register" button 3. Fill details 4. Click on Register button 5. Click on verification link in email	1. Register form openened 2. User prompted to check email 3. User receives verification link	Pass		
Login						
10	Login	1. Log in with email and password	Open Revenue page	Pass		
Revenue features						
30	Calculate total revenue	1. Add new Revenue	Calculate total revenue	Pass		
31	Add new revenue by month and year	Insert the data into the text boxes	Data is implemented in the database and shows in the revenue table	Pass		
32	Filter revenue by month and year	Click on the button that shows revenue year	Only the year selected will be shown	Pass		
Record Features						
40	Record invoices by buying price,	No action needed since it's a display table	All records should be displayed successfully	Pass		
41	Insert new invoices	Insert the data into the text boxes	After refreshing the web page the new invoice should appear in the	Pass		
Details Features						
51	Show each employee detail and salary		Employee details should be shown on the table	Pass		
Fisherman Features						
60	Show fisherman detail, quality of the		Fisherman table displayed	Pass		
			Statistics			
				v-02		
			Test cases Not executed	0		
			Test cases Blocked	0		
			Test cases Fail	0		
			Test cases Pass	10		
			Total number of test cases	10		
			Number of JIRA bugs			

Figure 31: Regression log

A regression log was created to keep track of all system feature testing. All tests were executed on a windows desktop as well as IOS. All test cases were passes, ensuring the system was functional as per user and client requriments

8.6 Video Demonstration and GIT repo

https://www.youtube.com/watch?v=SXQOmqiR5mg

https://github.com/irfanessa2/SFE_Crab

9. Team Reflection

Reflecting on our journey, our team efficiently executed the online crab portal project, with each member contributing to some extent. We achieved our main objective of creating an intuitive and engaging portal for fishermen, which is now brimming with useful information, high-quality visuals, and interactive elements, for example the ability to insert records.

In retrospect, however, we acknowledge certain areas for improvement. Time management and task delegation could have been streamlined more effectively. Some phases of the project saw bottlenecks due to overlapping tasks. If we were to undertake this project again, we would invest more time in meticulous planning and establish clearer roles and responsibilities from the outset.

In addition, our user interface, while functional and visually pleasing, could benefit from further user testing and refinement. This would ensure optimal user experience, particularly for those less tech-savvy.

Given more time, we would love to expand our project further. We envision developing a community feature for users to exchange insights and experiences. This would foster a more engaging and interactive platform. Additionally, we could incorporate an AI-based identification system to help users identify categories of crab.

Overall, this project has been a valuable learning experience, providing insights not only about project management but also about the fascinating world of crab processing. We are eager to apply these software engineering learnings to future endeavors.

10. References

[1] Adobe communications team (2023). https://business.adobe.com/blog/basics/waterfall#:~:text=The%20Waterfall%20methodology%20%E2%80%94%20also%20known,before%20the%20next%20phase%20begins. [accessed 21 February 2023]

[2] Otoyo, L. (2023) *Software Processes.* [Software Engineering lecture notes, CSI_5_SFE]. London South Bank University, Week 2, 2023.

[3] Lucid Content Team (2023). https://www.lucidchart.com/blog/pros-and-cons-of-waterfall-methodology [accessed 21 February 2023)

[4] Chien, C (2020) https://codebots.com/app-development/what-is-rapid-application-development-rad [accessed 21 February 2023]

[5] Egeonu.E (2022) https://distantjob.com/blog/rad-model-advantages-and-disadvantages/ [accessed 21 February 2023]

[6] https://www.atlassian.com/agile/kanban#:~:text=What%20is%20kanban%3F,of%20work%20at%20any%20time [accessed 21 February 2023]

[7] https://www.collegenote.net/curriculum/software-engineering-csit/52/299/ [accessed 22 February 2023]

[8] https://www.ques10.com/p/2200/discuss-the-advantage-and-disadvantage-of-integr-1/ [accessed 23 February 2023]

[9] B. W. Boehm, "A spiral model of software development and enhancement," in Computer, vol. 21, no. 5, pp. 61-72, May 1988, doi: 10.1109/2.59.

[10] Alshamrani, A. and Bahattab, A., 2015. A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model. International Journal of Computer Science Issues (IJCSI), 12(1), p.106.,

[11] https://artoftesting.com/spiral-model [accessed 27th February 2023]

[12] https://www.freecodecamp.org/news/reactjs-basics-dom-components-declarative-views/#:~:text=What%20is%20the%20DOM%3F,%2C%20attributes%2C%20and%20so%20on. [accessed 20th March 2023]

[13] What Is React? (codecademy.com) [accessed 20th March 2023]

[14] https://www.pangea.ai/dev-reactjs-development-resources/disadvantages-of-react-native-reactjs-which-is-better/#:~:text=React%20Native%20suffers%20from%20poor,figuring%20out%20how%20things%20work. [Accessed 21st March 2023]

[15] https://waverleysoftware.com/blog/why-use-flutter-pros-and-cons/ [accessed 24th March 2023]

[16] https://builtin.com/software-engineering-perspectives/flask [accessed 24th March 2023]

[17] https://techifysolutions.com/flask-vs-django/ [accessed 24th March 2023]

[18] Gotterbarn, D., Miller, K. and Rogerson, S. (1997). Software engineering code of ethics. Communications of the ACM, 40(11), pp.110–118. doi:https://doi.org/10.1145/265684.265699.

[19] BCS (2022). BCS Code of Conduct | BCS. [online] www.bcs.org. Available at: https://www.bcs.org/membership-and-registrations/become-a-member/bcs-code-of-conduct/. [Accessed 10th May 2023]

[20] legislation.gov.uk (1990). *Computer Misuse Act 1990*. [online] Legislation.gov.uk. Available at: https://www.legislation.gov.uk/ukpga/1990/18/contents. [Accessed 10th May 2023]

[21] Legislation.gov.uk (1988). *Copyright, Designs and Patents Act 1988*. [online] Legislation.gov.uk. Available at: https://www.legislation.gov.uk/ukpga/1988/48/contents. [Accessed 11th May 2023]

11. Appendix

Model	**Strengths**	**Weaknesses**
Waterfall	• Predictable (costs anddeadlines) • Well documented – more manageable for a new teammember to catch up • Simple and easy to use • Good for smaller projects	• Changing the scope would mean starting an entirely new project • Working software is only produced towards the end of the software • Minimal risk analysis • It does not work for complex system models
Agile (Scrum/Kanban /RAD/XP)	• Allows for change in requirements (helpful in thiscase as client communication is regular) • Reduces risk as deploymentis fragmented • Allows for stakeholder/client engagement	• Cost is not defined until the end product is developed • Delivery of the product is fragmented (deployed bit by bit), this can also be an advantage • Documentation can get confusing, and new team members may struggle to catch up
Integration and Configuration	• Reduces cost as somemodules are already developed • It should only be used if the old system is your own	• Upgrading, testing, and troubleshooting may be difficult as developers may not understand the code
Spiral	• The first iteration of the software is produced early • Risk is analysed with everyiteration leading to risk reduction	• Documentation can get highly complex • Recommended for only larger scale products • Time management is complex • as the number of iterations is unknown

Figure 1: Comparing SD methodologies

Functional Requirements	**Non-functional requirements**
Maintain employee details and salary, depending on the two types of employees: permanent employees (paid monthly) and temporary employees (paid daily).	The portal should be accessible from different devices (i.e., phones) and web browsers (i.e. Firefox) and support multiple users simultaneously.
The buying and selling price of crabs according to their categories (A, B, C, and D) and calculate the revenue for each depending on the circumstance.	The portal should be user-friendly and easy to navigate, with clear instructions and labels.
Identify the fisherman delivering good quality crab to increase profit	The portal should have backup and recovery. functions if the system fails or data is lost.
Record the death rate of crabs while making hard-shell crabs to soft-shell crabs.	It should be scalable for the growing. business needs of the company.
Calculate the revenue each month/year, depending on the buying and selling price of hard- and soft-shell crabs.	The portal should be secure and only accessible to authorised users.
	The online portal should have good performance and response time, even when handling large amounts of data, to ensure a smooth user experience for the company's owner and employees.

Figure 2: Functional and non-functional requirements

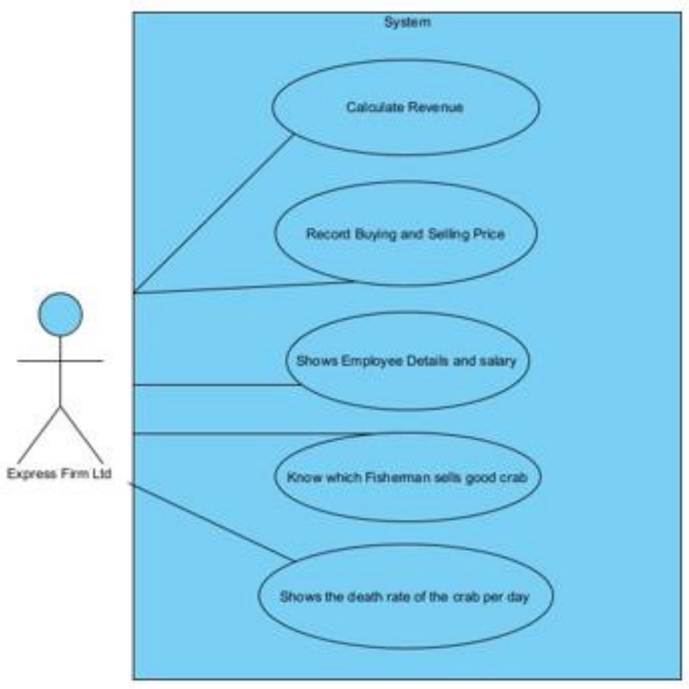

System
Calculate Revenue
Record Buying and Selling Price
Shows Employee Details and salary
Know which Fisherman sells good crab
Shows the death rate of the crab per day
Express Firm Ltd

Figure 3: First iteration use case diagram

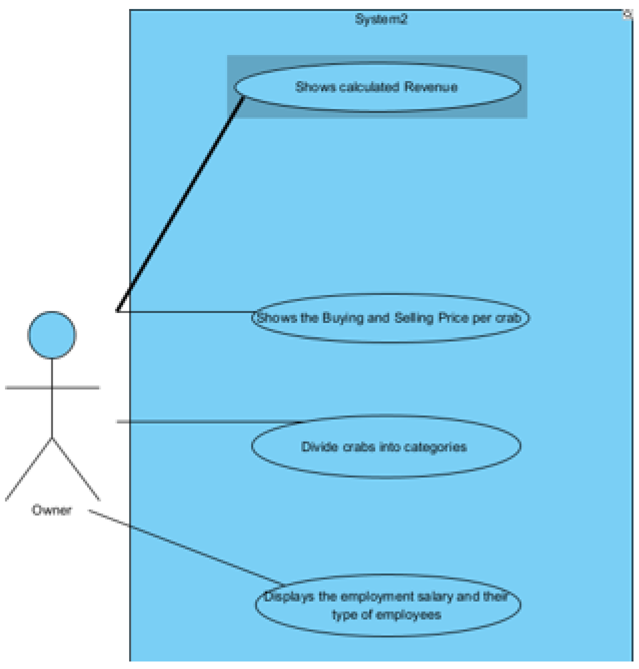

System2
Shows calculated Revenue
Shows the Buying and Selling Price per crab
Divide crabs into categories
Displays the employment salary and their type of employees
Owner

Figure 4: Second iteration use case diagram

The 2nd use case diagram was created using owner as an actor and explaining his functions in the company. The most important actions of the owner in the company are showing calculated Revenue, Showing the buying and selling price for crab, dividing them into categories and displaying the employee salary and the type of employers to him.

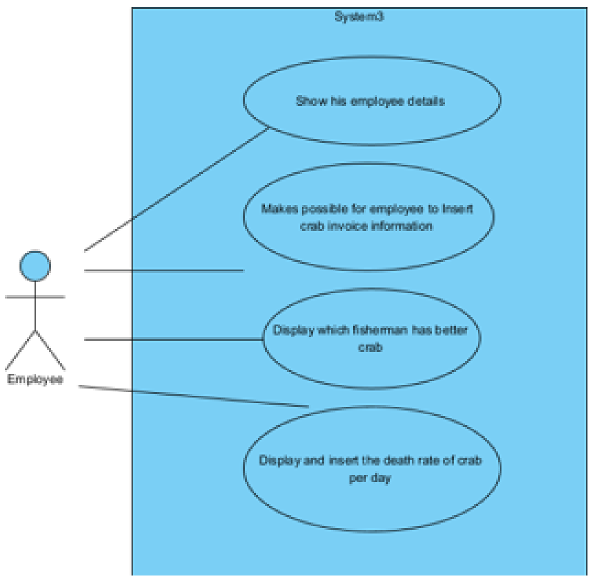

System3
Show his employee details
Makes possible for employee to Insert crab invoice information
Display which fisherman has better crab
Display and insert the death rate of crab per day
Employee

Figure 5: Third iteration use case diagram

The 3rd use case diagram (see figure 5) shows the employee and his functions like displaying his employee details, make possible for him to insert crab invoice information, display which fisherman has a better crab quality and the crab death rate per day.

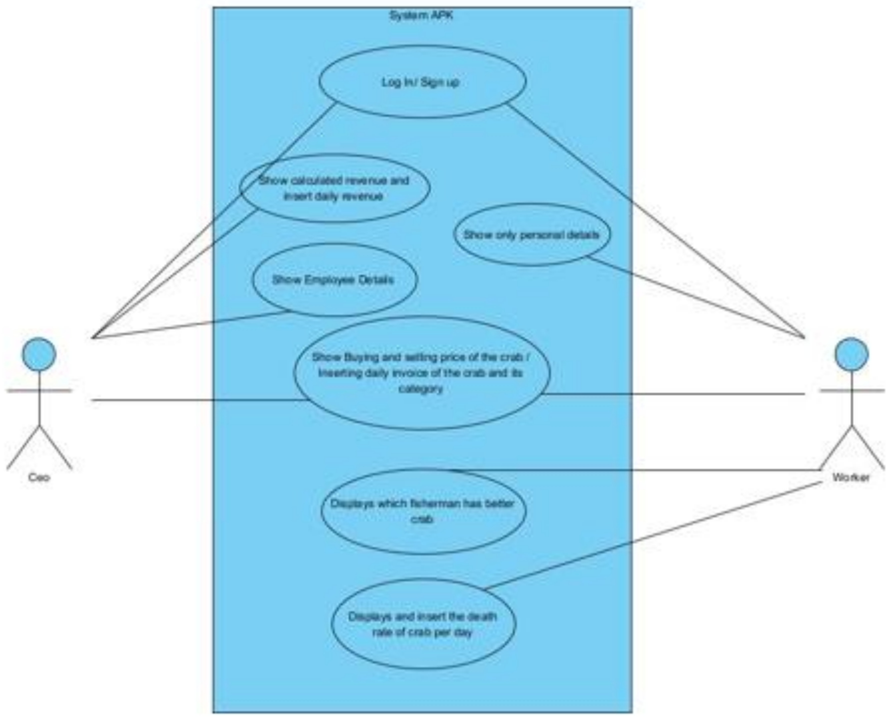

System APK
Log In/ Sign up
Show calculated revenue and insert daily revenue
Show only personal details
Show Employee Details
Show Buying and selling price of the crab / Inserting daily invoice of the crab and its category
Ceo
Worker
Displays which fisherman has better crab
Displays and insert the death rate of crab per day

Figure 6: Fourth iteration use case diagram

As an Owner, I want to see the calculated revenue of each month and each year 1

As an Owner, I want to record the buying and selling price of each crab, also divided into the category of the crab quality from A to D 2

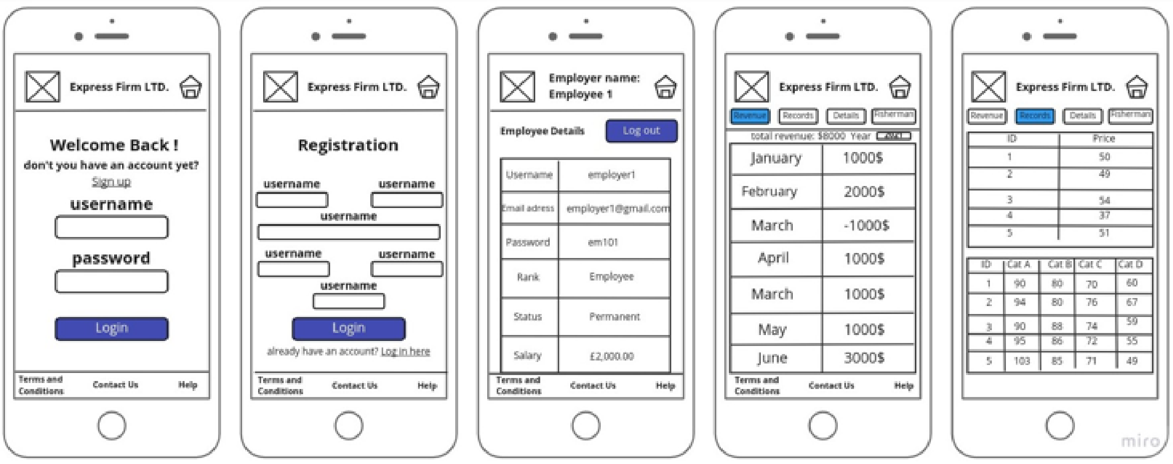
As an Owner, I want to see all employees details including their email, password, first name, last name, status and salary 3

As an Employee, I want to see my personal details on the application 4

As an Employee, I want to record the buying and selling price of each crab, and create invoice for that 5

As an employee, I want to know the fisherman ID, the quality of each crab bought by the fishermen and the death rate of the crab per day 6

Figure 7: User Stories sorted by priority (1 = HIGH)



Username	employer1
Email adress	employer1@gmail.com
Password	em101
Rank	Employee
Status	Permanent
Salary	£2,000.00

January	1000$
February	2000$
March	-1000$
April	1000$
March	1000$
May	1000$
June	3000$

ID	Price
1	50
2	49
3	54
4	37
5	51

ID	Cat A	Cat B	Cat C	Cat D
1	90	80	70	60
2	94	80	76	67
3	90	88	74	59
4	95	86	72	55
5	103	85	71	49

Figure 9: Low-fidelity view of system

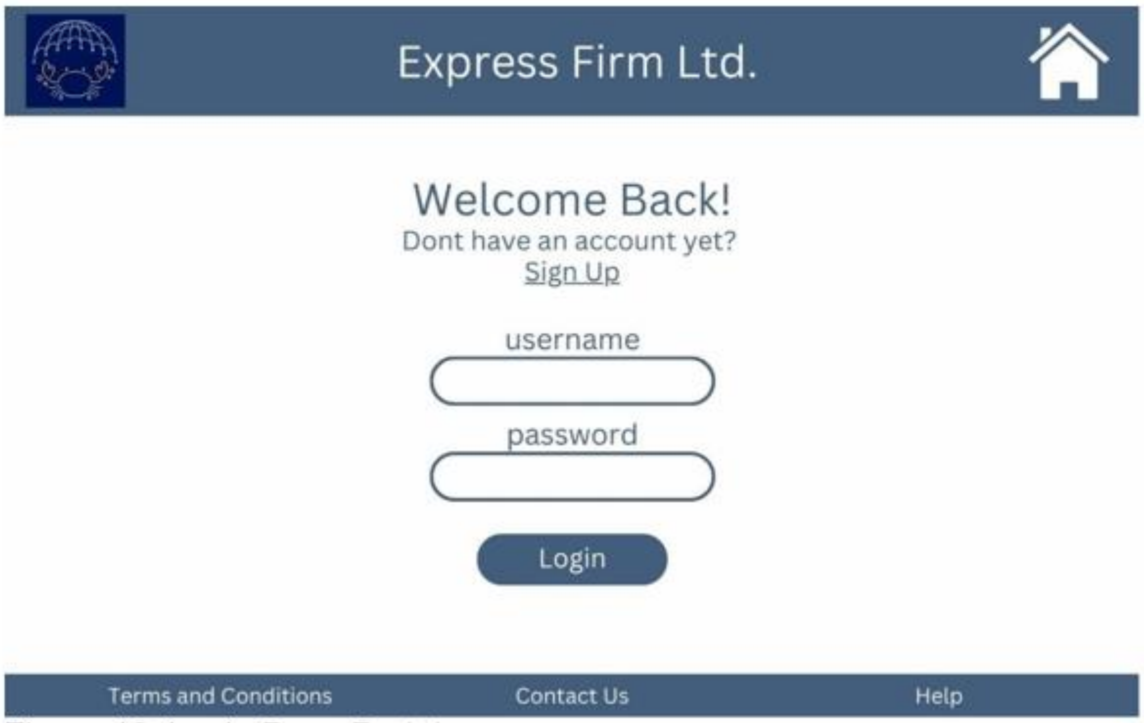
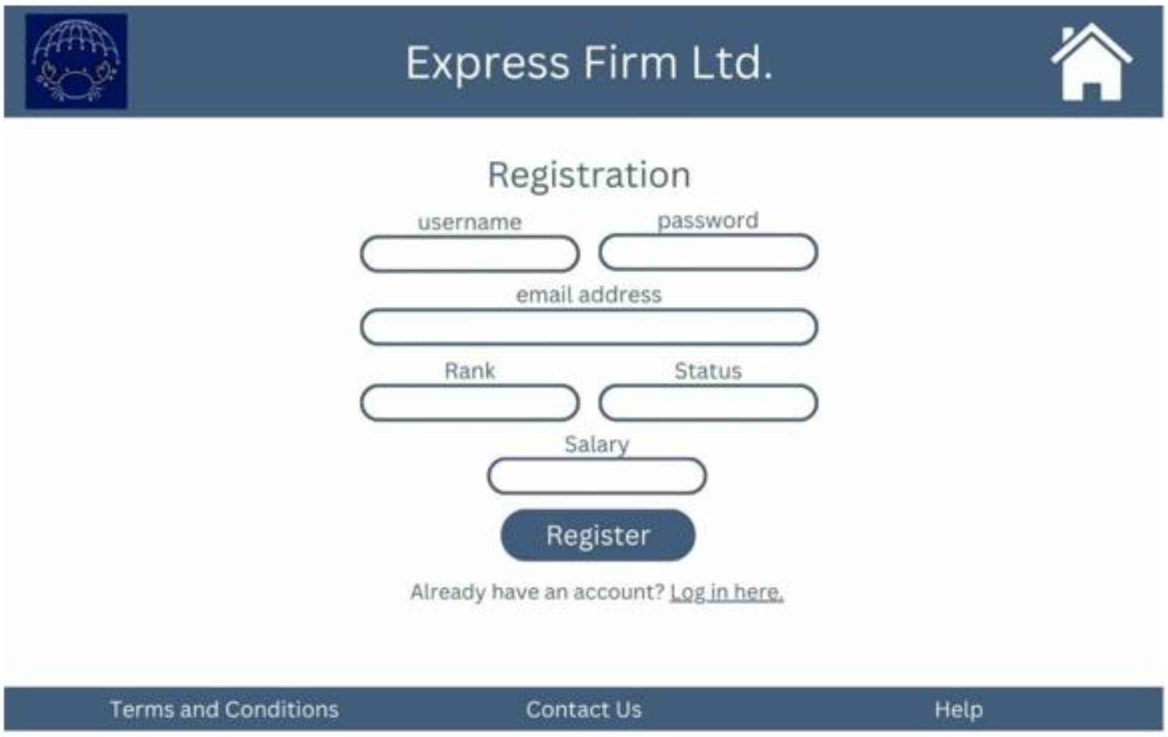


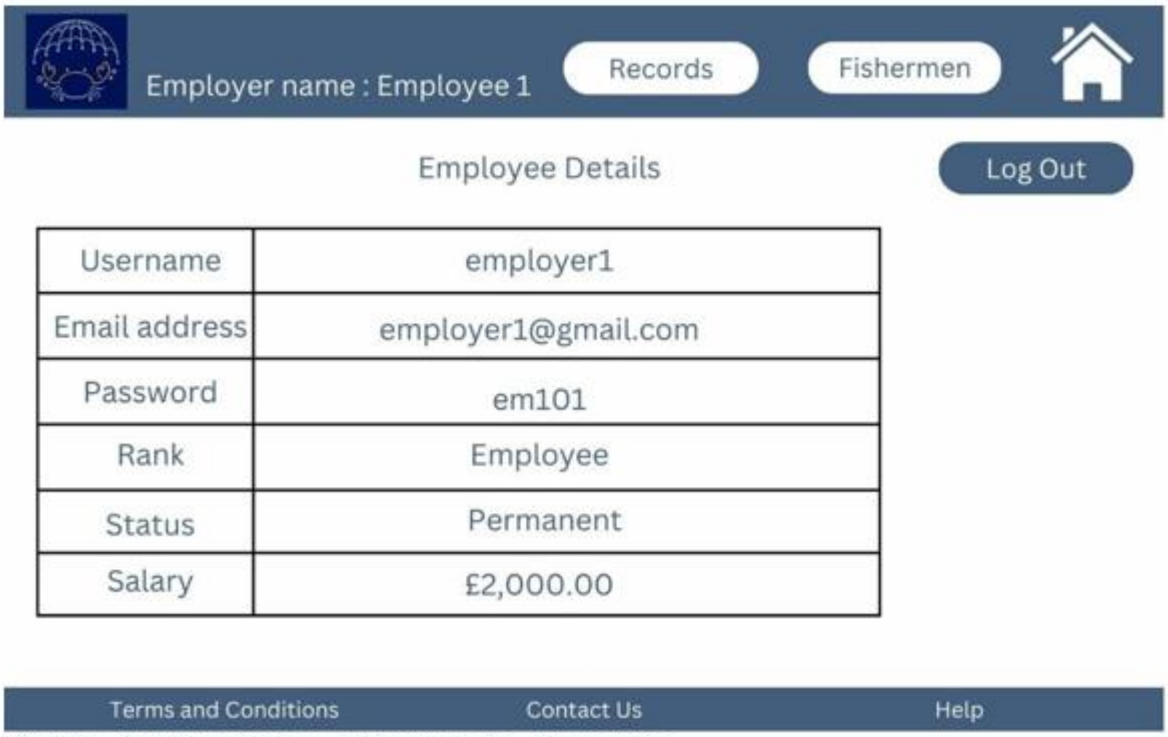
Figure 10: Login Page Prototype


Express Firm Ltd.

Registration

username	password
email address	
Rank	Status
Salary	

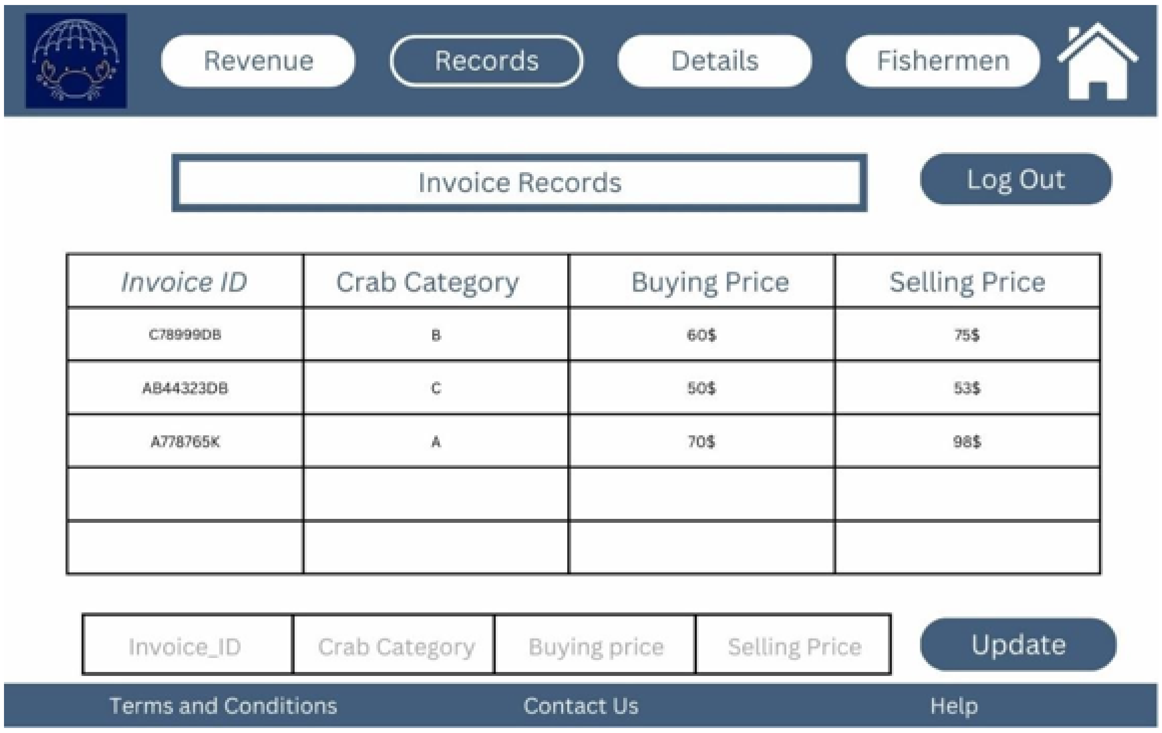
Figure 11: Sign Up Page Prototype.



Employee Details

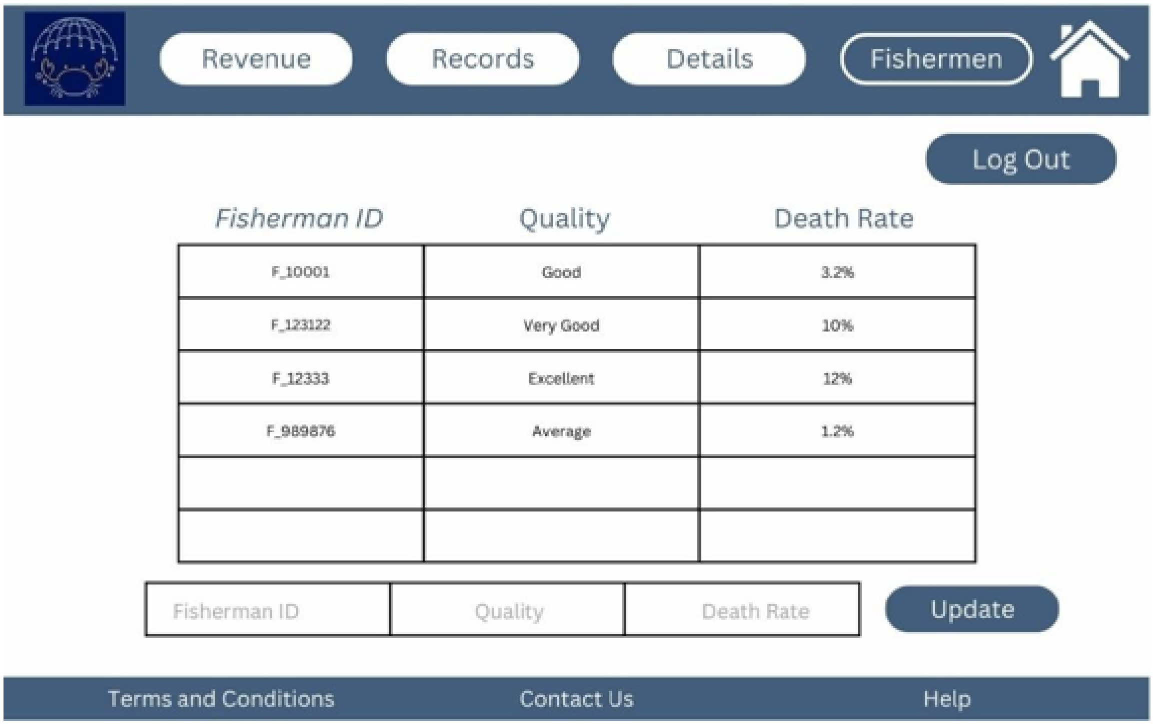
Username	employer1
Email address	employer1@gmail.com
Password	em101
Rank	Employee
Status	Permanent
Salary	£2,000.00

Figure 12: Employee Detail Page Prototype.



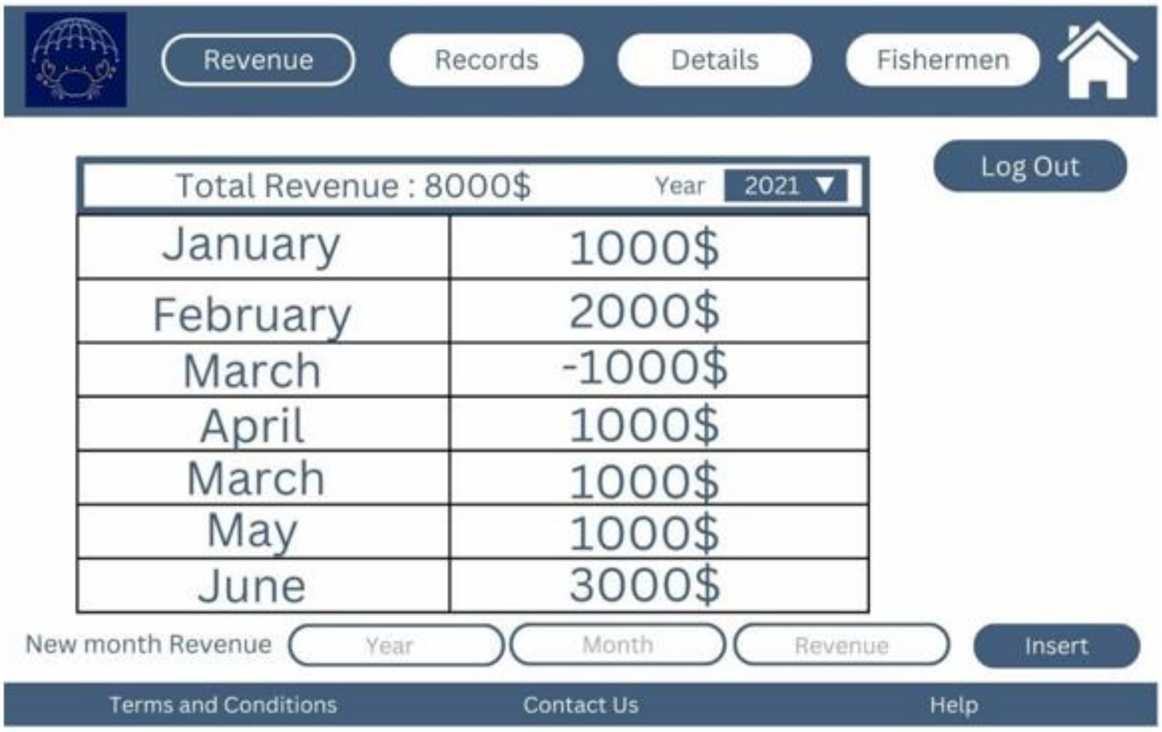
Invoice ID	Crab Category	Buying Price	Selling Price
C78999DB	B	60$	75$
AB44323DB	C	50$	53$
A778765K	A	70$	98$

Figure 13: Invoice Records Page Prototype.



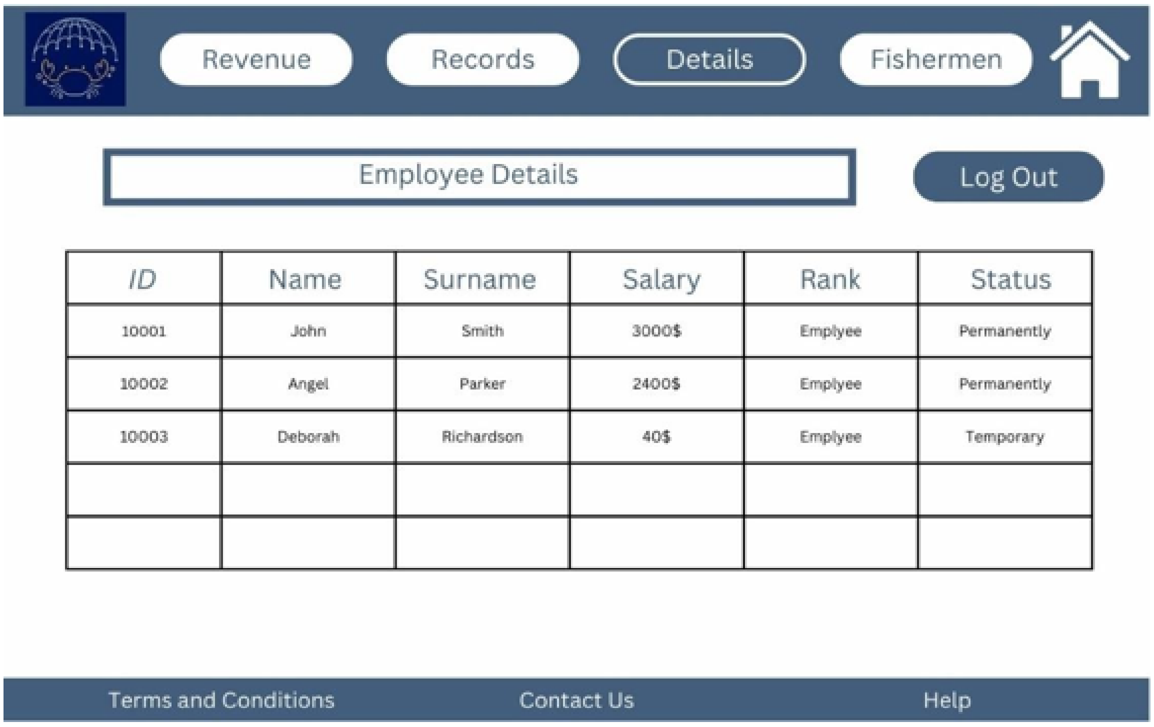
Fisherman ID	Quality	Death Rate
F_10001	Good	3.2%
F_123122	Very Good	10%
F_12333	Excellent	12%
F_989876	Average	1.2%

Figure 14: Fisherman and death rate of crab details Page Prototype.



Total Revenue : 8000$	Year 2021 ▼
January	1000$
February	2000$
March	-1000$
April	1000$
March	1000$
May	1000$
June	3000$

Figure 15: Revenue Page Prototype (Only for owner).



ID	Name	Surname	Salary	Rank	Status
10001	John	Smith	3000$	Emplyee	Permanently
10002	Angel	Parker	2400$	Emplyee	Permanently
10003	Deborah	Richardson	40$	Emplyee	Temporary

Figure 16: All employers details Page Prototype (Only for owner).

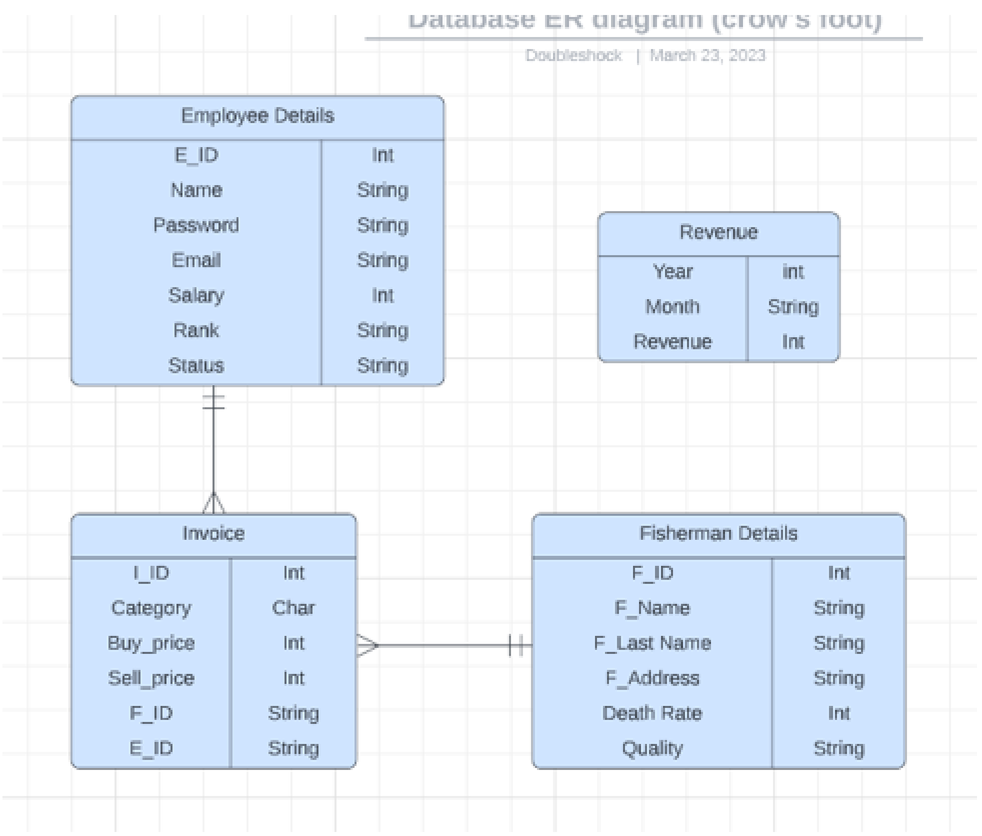


Figure 17: Database ER relation diagram.

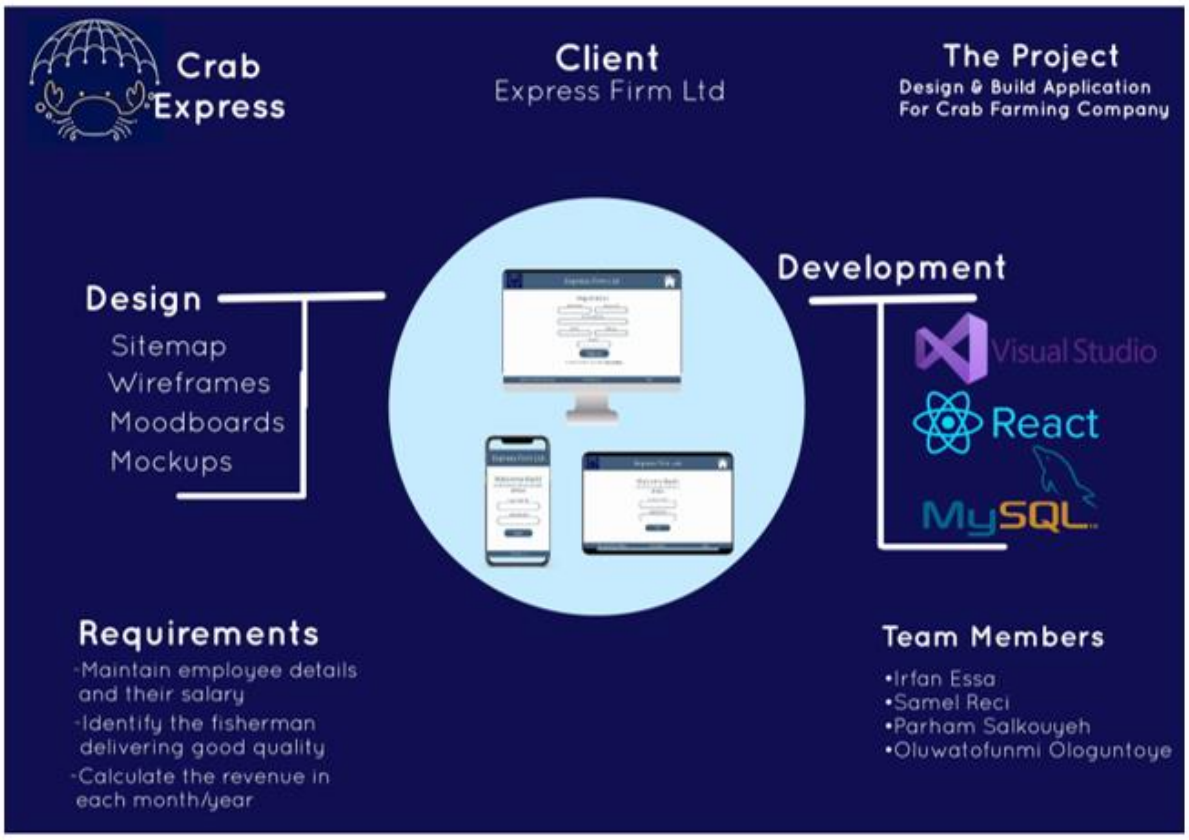
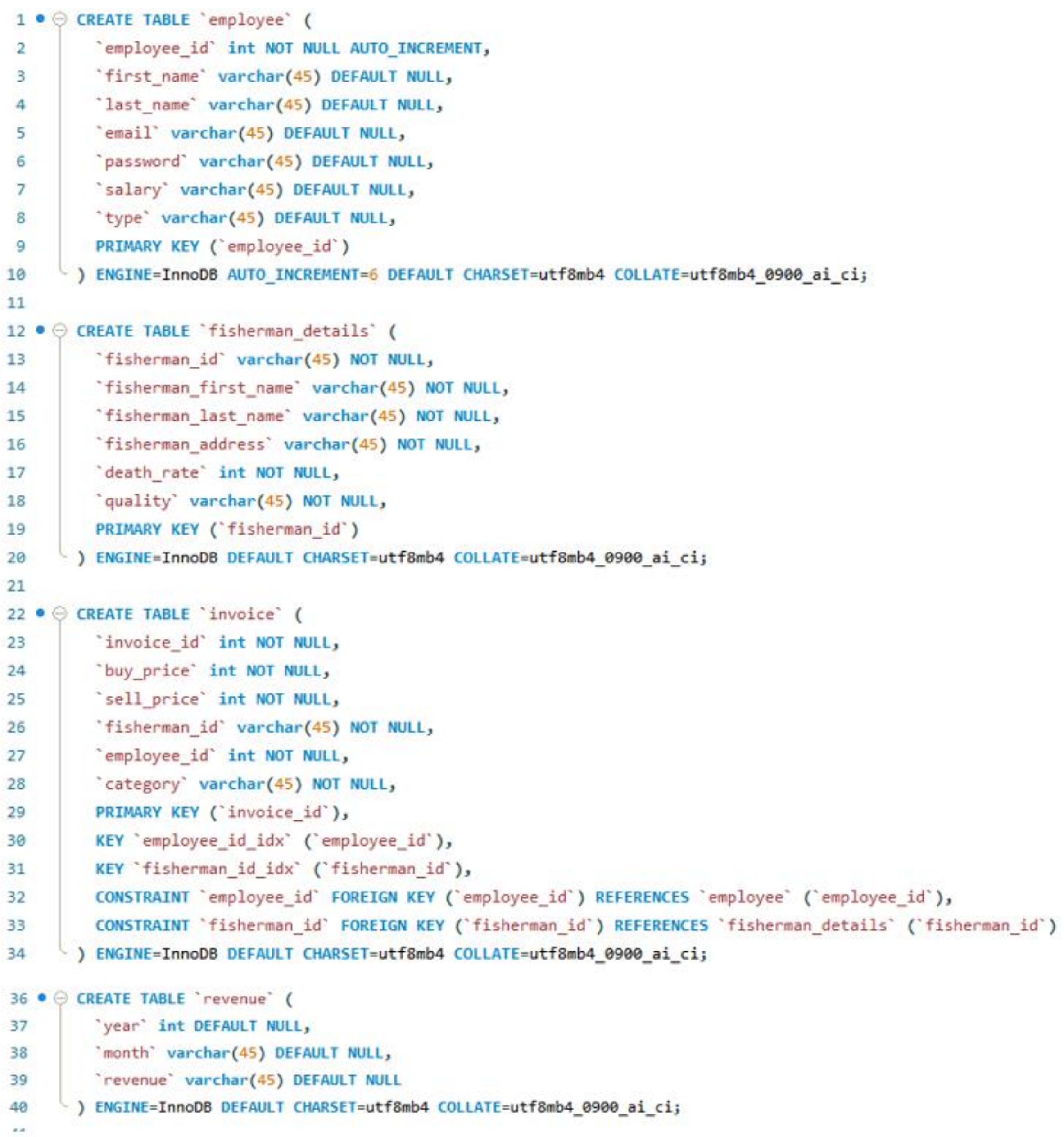
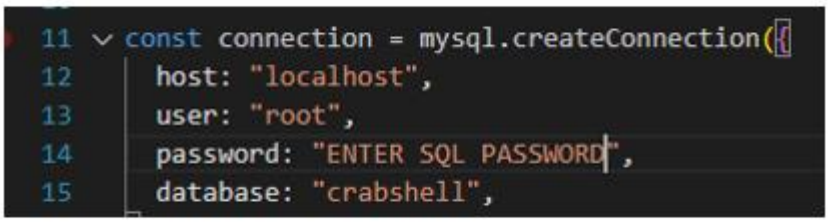


Figure 18: Poster



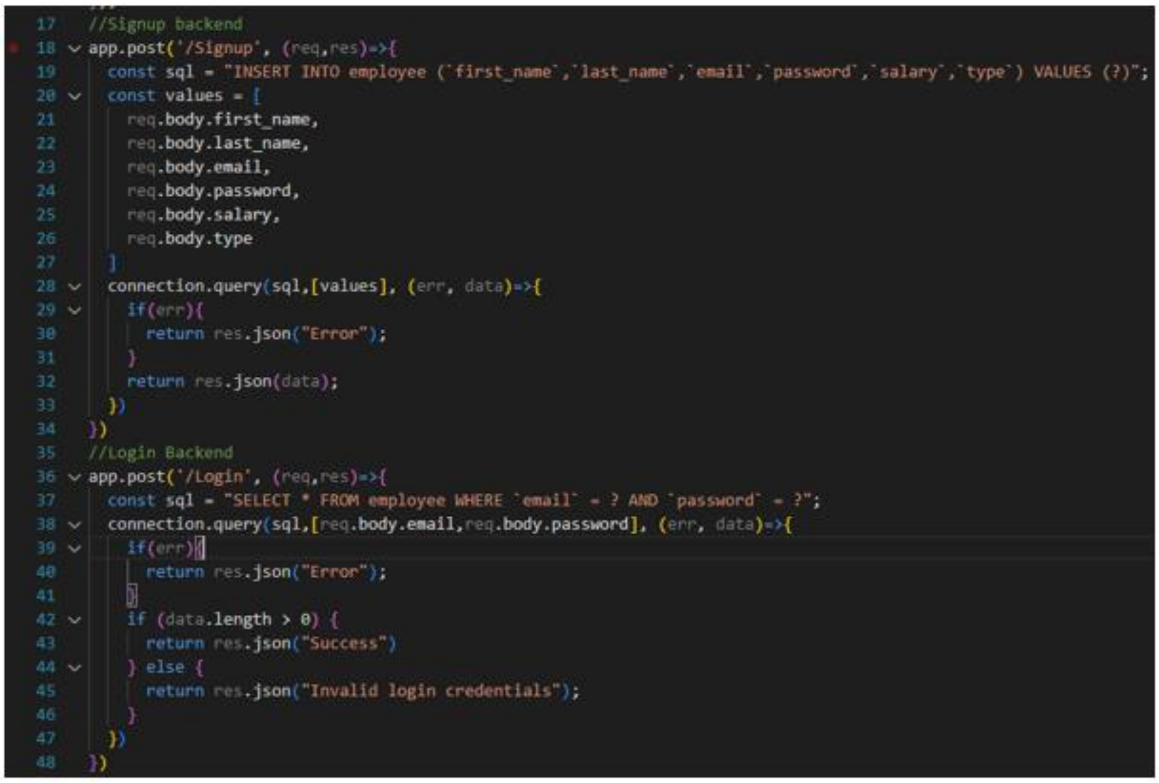
```sql
CREATE TABLE `employee` (
  `employee_id` int NOT NULL AUTO_INCREMENT,
  `first_name` varchar(45) DEFAULT NULL,
  `last_name` varchar(45) DEFAULT NULL,
  `email` varchar(45) DEFAULT NULL,
  `password` varchar(45) DEFAULT NULL,
  `salary` varchar(45) DEFAULT NULL,
  `type` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`employee_id`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `fisherman_details` (
  `fisherman_id` varchar(45) NOT NULL,
  `fisherman_first_name` varchar(45) NOT NULL,
  `fisherman_last_name` varchar(45) NOT NULL,
  `fisherman_address` varchar(45) NOT NULL,
  `death_rate` int NOT NULL,
  `quality` varchar(45) NOT NULL,
  PRIMARY KEY (`fisherman_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `invoice` (
  `invoice_id` int NOT NULL,
  `buy_price` int NOT NULL,
  `sell_price` int NOT NULL,
  `fisherman_id` varchar(45) NOT NULL,
  `employee_id` int NOT NULL,
  `category` varchar(45) NOT NULL,
  PRIMARY KEY (`invoice_id`),
  KEY `employee_id_idx` (`employee_id`),
  KEY `fisherman_id_idx` (`fisherman_id`),
  CONSTRAINT `employee_id` FOREIGN KEY (`employee_id`) REFERENCES `employee` (`employee_id`),
  CONSTRAINT `fisherman_id` FOREIGN KEY (`fisherman_id`) REFERENCES `fisherman_details` (`fisherman_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

CREATE TABLE `revenue` (
  `year` int DEFAULT NULL,
  `month` varchar(45) DEFAULT NULL,
  `revenue` varchar(45) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

Figure 19: Database table queries



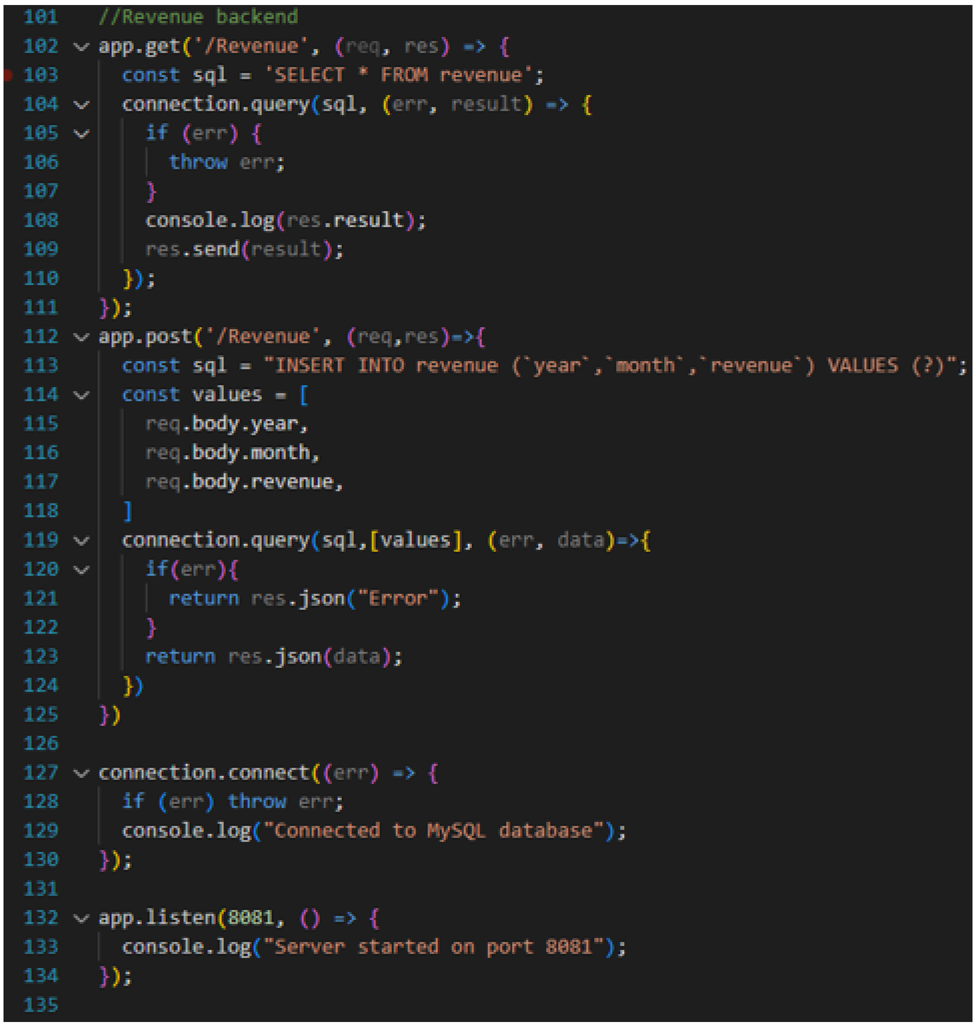
```javascript
const connection = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "ENTER SQL PASSWORD",
    database: "crabshell",
```

Figure 20: Code for SQL connection



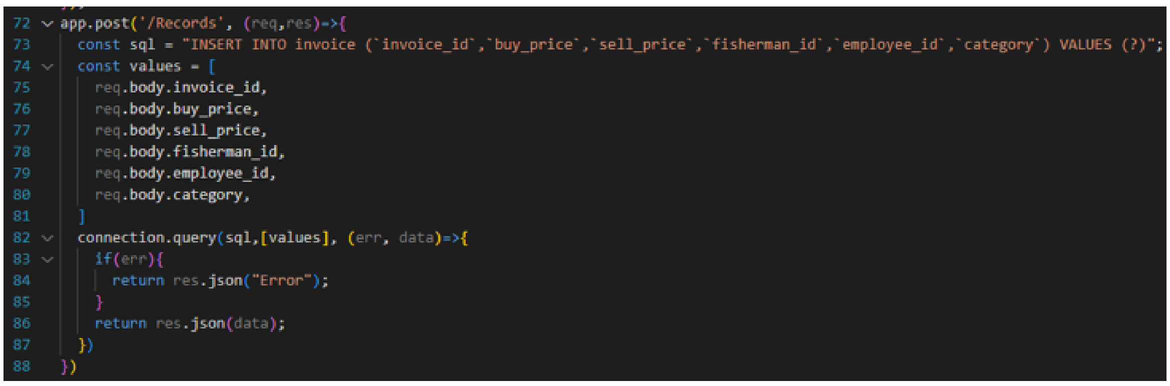
```
//Signup backend
app.post('/Signup', (req,res)=>{
  const sql = "INSERT INTO employee (`first_name`,`last_name`,`email`,`password`,`salary`,`type`) VALUES (?)";
  const values = [
    req.body.first_name,
    req.body.last_name,
    req.body.email,
    req.body.password,
    req.body.salary,
    req.body.type
  ]
  connection.query(sql,[values], (err, data)=>{
    if(err){
      return res.json("Error");
    }
    return res.json(data);
  })
})
//Login Backend
app.post('/Login', (req,res)=>{
  const sql = "SELECT * FROM employee WHERE `email` = ? AND `password` = ?";
  connection.query(sql,[req.body.email,req.body.password], (err, data)=>{
    if(err){
      return res.json("Error");
    }
    if (data.length > 0) {
      return res.json("Success")
    } else {
      return res.json("Invalid login credentials");
    }
  })
})
```

Figure 21: Code for Sign Up and Log In



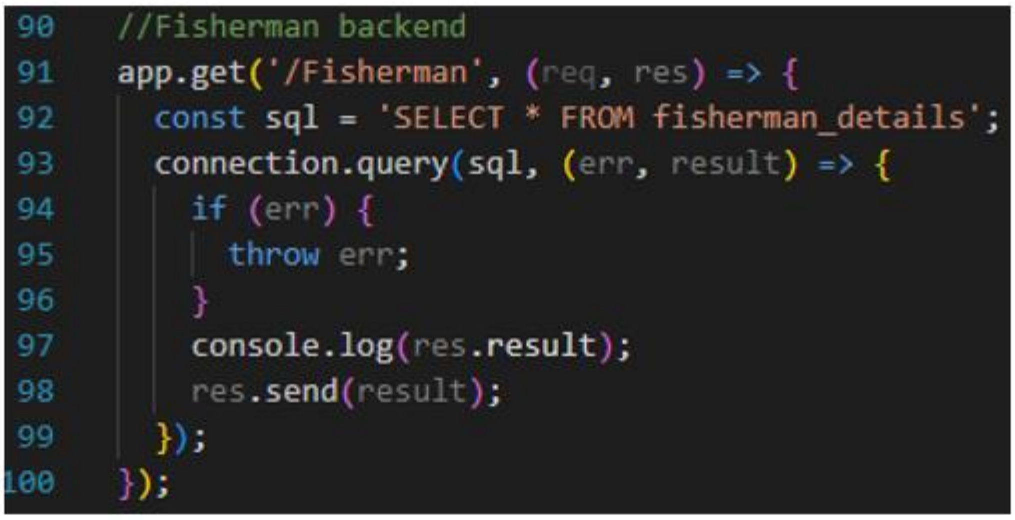
```
//Revenue backend
app.get('/Revenue', (req, res) => {
  const sql = 'SELECT * FROM revenue';
  connection.query(sql, (err, result) => {
    if (err) {
      throw err;
    }
    console.log(res.result);
    res.send(result);
  });
});
app.post('/Revenue', (req,res)=>{
  const sql = "INSERT INTO revenue (`year`,`month`,`revenue`) VALUES (?)";
  const values = [
    req.body.year,
    req.body.month,
    req.body.revenue,
  ]
  connection.query(sql,[values], (err, data)=>{
    if(err){
      return res.json("Error");
    }
    return res.json(data);
  })
})

connection.connect((err) => {
  if (err) throw err;
  console.log("Connected to MySQL database");
});

app.listen(8081, () => {
  console.log("Server started on port 8081");
});

```

Figure 22: Code for revenue inserts



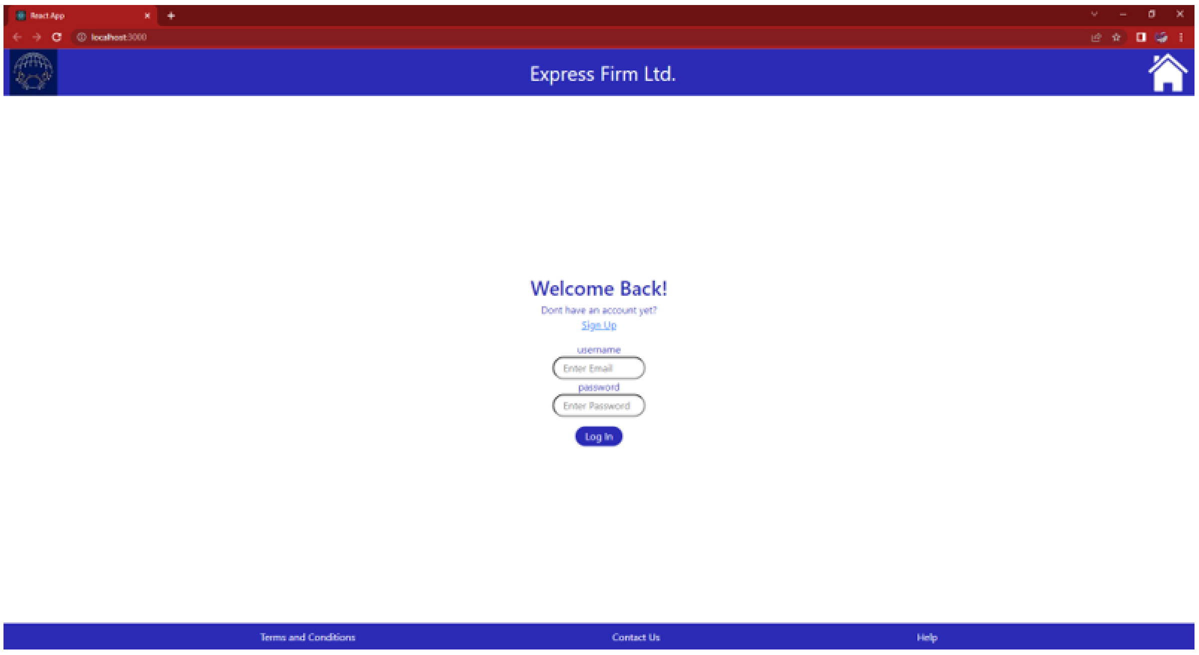
```
app.post('/Records', (req,res)=>{
  const sql = "INSERT INTO invoice (`invoice_id`,`buy_price`,`sell_price`,`fisherman_id`,`employee_id`,`category`) VALUES (?)";
  const values = [
    req.body.invoice_id,
    req.body.buy_price,
    req.body.sell_price,
    req.body.fisherman_id,
    req.body.employee_id,
    req.body.category,
  ]
  connection.query(sql,[values], (err, data)=>{
    if(err){
      return res.json("Error");
    }
    return res.json(data);
  })
})
```

Figure 23: Code for invoice inserts



```
//Fisherman backend
app.get('/Fisherman', (req, res) => {
  const sql = 'SELECT * FROM fisherman_details';
  connection.query(sql, (err, result) => {
    if (err) {
      throw err;
    }
    console.log(res.result);
    res.send(result);
  });
});
```

Figure 24: Code for Fisherman details



React App

localhost:3000

Express Firm Ltd.

Welcome Back!

Dont have an account yet?

Sign Up

username

Enter Email

password

Enter Password

Log In

Terms and Conditions Contact Us Help

Figure 25: Final Log In page

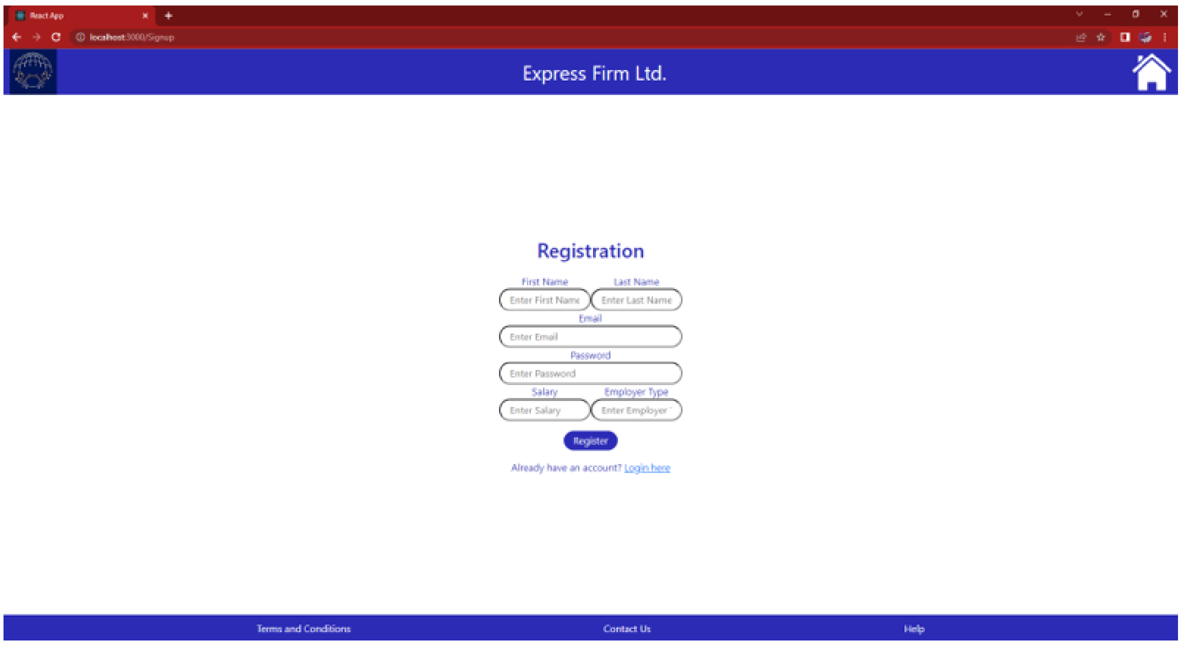
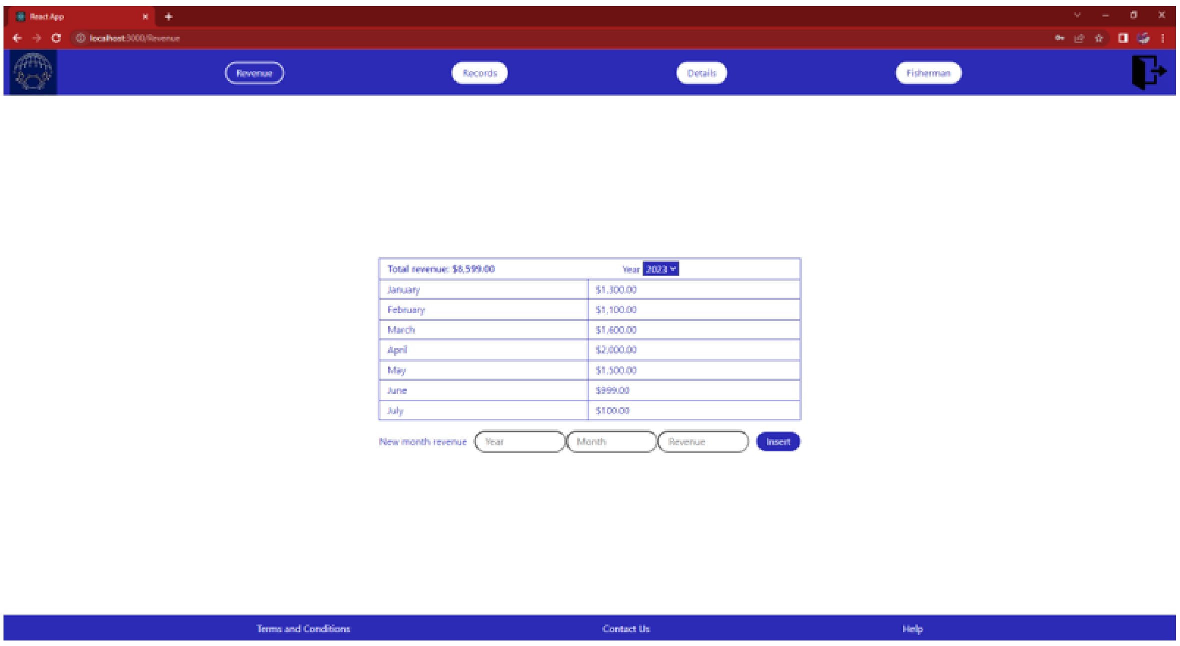


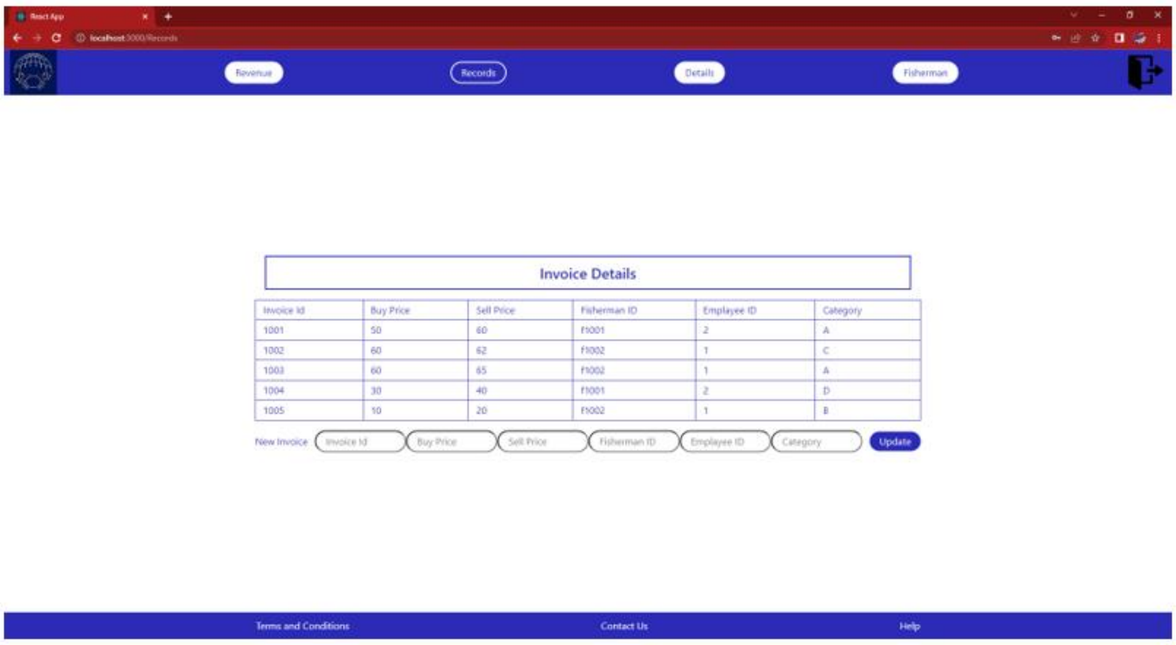
Figure 26: Final Sign up page



Total revenue: $8,599.00	Year 2023
January	$1,300.00
February	$1,100.00
March	$1,600.00
April	$2,000.00
May	$1,500.00
June	$999.00
July	$100.00

New month revenue Year Month Revenue Insert

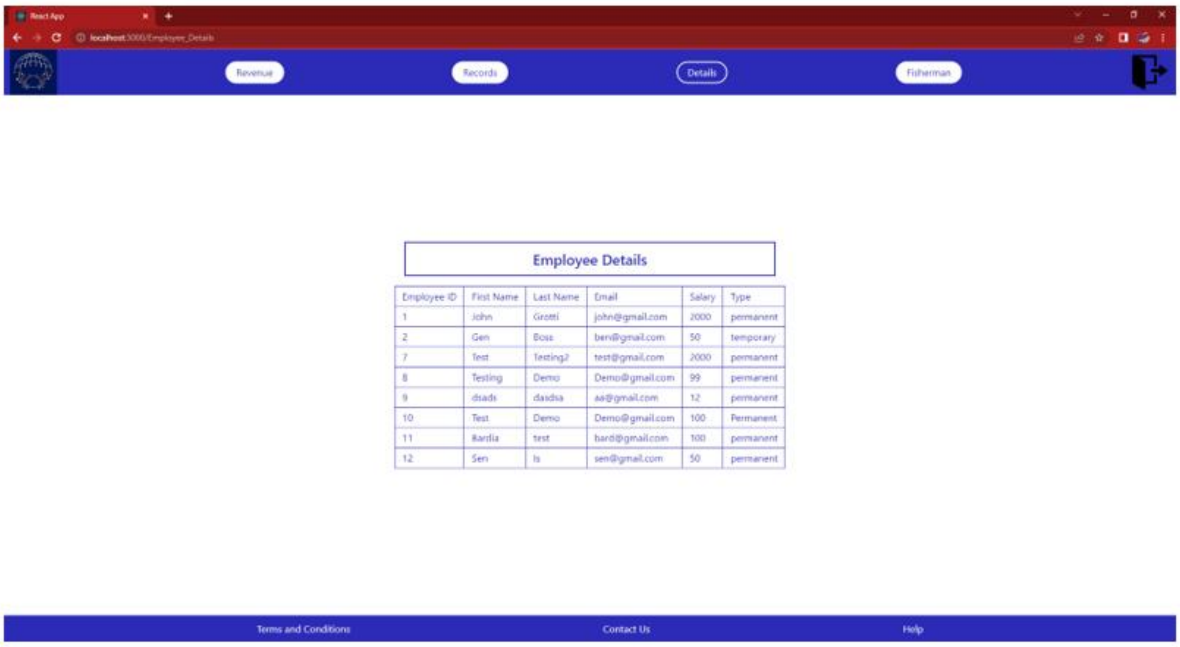
Figure 27: Final Revenue page



Invoice Details

Invoice Id	Buy Price	Sell Price	Fisherman ID	Emplayee ID	Category
1001	50	60	F1001	2	A
1002	60	62	F1002	1	C
1003	60	65	F1002	1	A
1004	30	40	F1001	2	D
1005	10	20	F1002	1	B

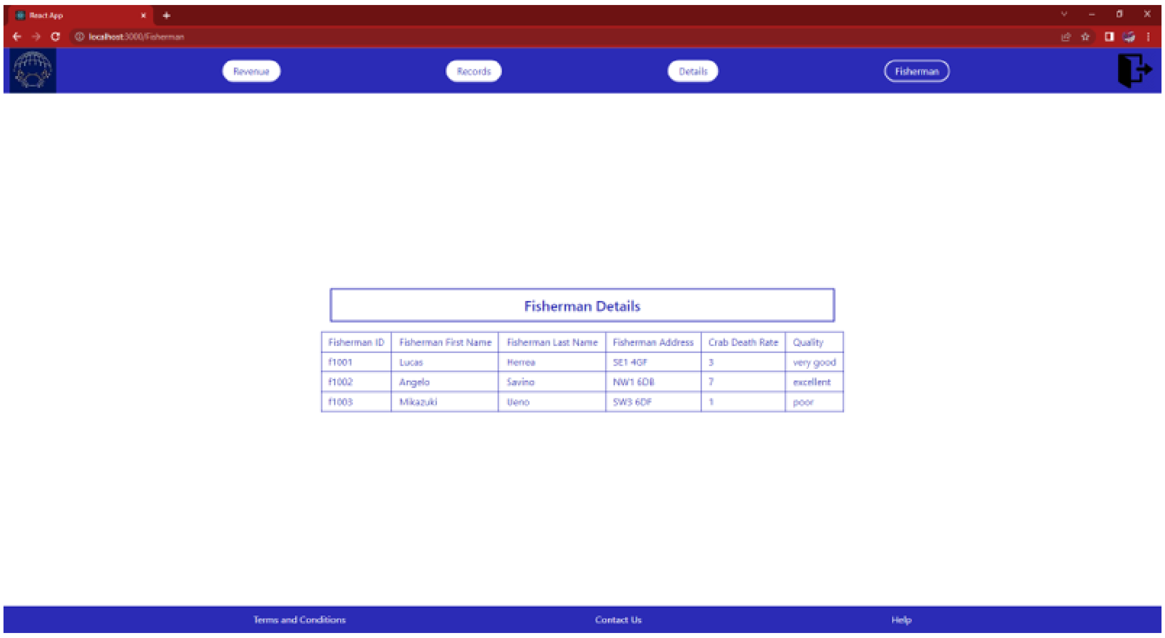
Figure 28: Final Invoice page



Employee Details

Employee ID	First Name	Last Name	Email	Salary	Type
1	John	Grotti	john@gmail.com	2000	permanent
2	Gen	Boss	ben@gmail.com	50	temporary
7	Test	Testing2	test@gmail.com	2000	permanent
8	Testing	Demo	Demo@gmail.com	99	permanent
9	dsads	dasdsa	aa@gmail.com	12	permanent
10	Test	Demo	Demo@gmail.com	100	Permanent
11	Bardia	test	bard@gmail.com	100	permanent
12	Sen	Is	sen@gmail.com	50	permanent

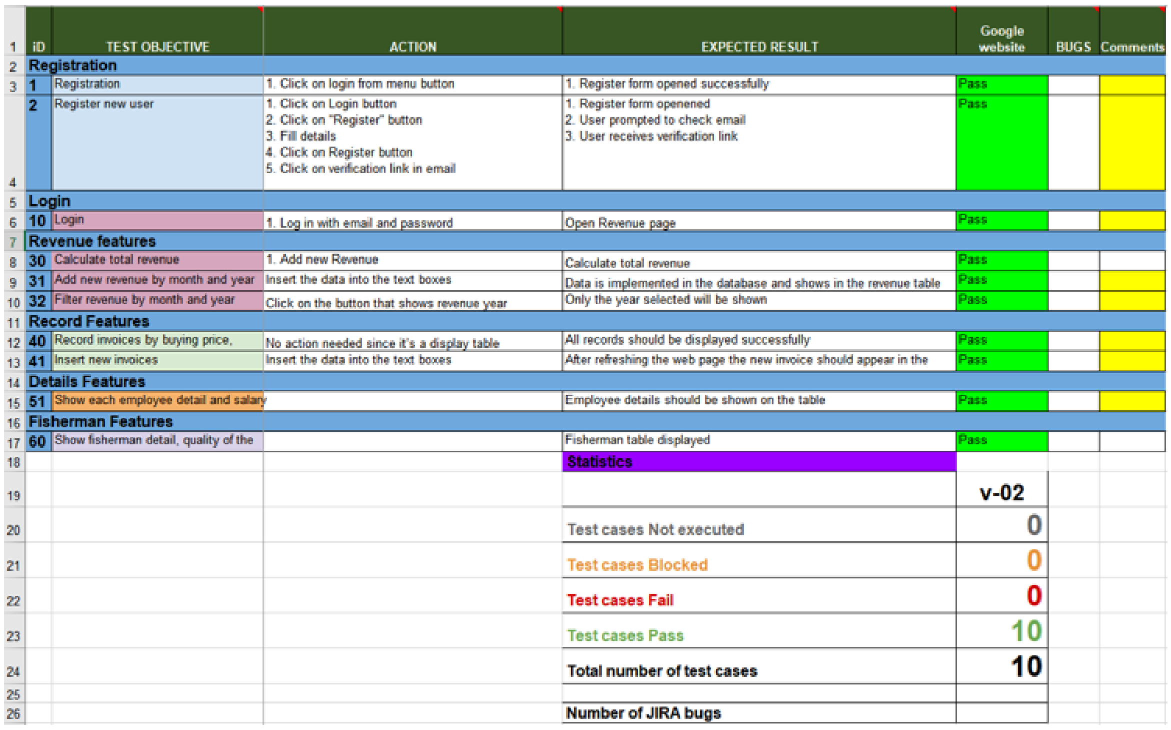
Figure 29: Final employee details page



Fisherman Details

Fisherman ID	Fisherman First Name	Fisherman Last Name	Fisherman Address	Crab Death Rate	Quality
f1001	Lucas	Herrea	SE1 4GF	3	very good
f1002	Angelo	Savino	NW1 6DB	7	excellent
f1003	Mikazuki	Ueno	SW3 6DF	1	poor

Figure 30: Final Fisherman details page



iD	TEST OBJECTIVE	ACTION	EXPECTED RESULT	Google website	BUGS	Comments
Registration						
1	Registration	1. Click on login from menu button	1. Register form opened successfully	Pass		
2	Register new user	1. Click on Login button 2. Click on "Register" button 3. Fill details 4. Click on Register button 5. Click on verification link in email	1. Register form openened 2. User prompted to check email 3. User receives verification link	Pass		
Login						
10	Login	1. Log in with email and password	Open Revenue page	Pass		
Revenue features						
30	Calculate total revenue	1. Add new Revenue	Calculate total revenue	Pass		
31	Add new revenue by month and year	Insert the data into the text boxes	Data is implemented in the database and shows in the revenue table	Pass		
32	Filter revenue by month and year	Click on the button that shows revenue year	Only the year selected will be shown	Pass		
Record Features						
40	Record invoices by buying price,	No action needed since it's a display table	All records should be displayed successfully	Pass		
41	Insert new invoices	Insert the data into the text boxes	After refreshing the web page the new invoice should appear in the	Pass		
Details Features						
51	Show each employee detail and salary		Employee details should be shown on the table	Pass		
Fisherman Features						
60	Show fisherman detail, quality of the		Fisherman table displayed	Pass		
			Statistics			
				v-02		
			Test cases Not executed	0		
			Test cases Blocked	0		
			Test cases Fail	0		
			Test cases Pass	10		
			Total number of test cases	**10**		
			Number of JIRA bugs			

Figure 31: Regression log