University of Puerto Rico

Mayagüez Campus

Electrical and Computer Engineering Department

BIT BUSTERS

Capstone Project Progress Report
Prepared for:

Dra. Nayda G. Santiago
Dr. Isidoro Couvertier

Prepared by Bit Busters:

Jeremy J. Márquez González (Project Manager)
Giomar Santiago Galloza
Kevin O. Valentín Avilés
Juan D. Pérez Sepúlveda

Zeta MinusOne

Executive Summary (Written by: Kevin O. Valentín)

Roof Rx, a roofing contractor with a wide variety of clients in California, Texas, and the southern United States, is leading a project to modernize inspection procedures through the integration of drone technology and machine learning algorithms. This initiative comes in response to the challenges posed by the global pandemic, with the important goal of ensuring precise and cost-effective assessments. The primary objective of the project is to develop a web application powered by machine learning algorithms to augment inspection precision and efficiency.

The design phase of the project involved careful consideration of client-specified design choices, leading to the generation of multiple variations to align with project objectives. The system architecture, designed to capitalize on AWS services such as Amplify, API Gateway, Lambda, S3, RDS, SageMaker, and Cognito, underscores the project's technological foundation. As of the latest update, task progress stands at 51%, with implementation and testing poised to commence on March 19, 2024.

Significant milestones have been reached, including the completion of research and design diagrams encompassing ER and Table Diagrams, Cloud Software Architecture Diagram, and System Architecture Diagram, Sequence Diagrams, Use Case Diagrams and Flowchart Diagrams. These achievements mark key progress points, with adjustments made to throughput to ensure timely completion. These adjustments arose from delays in different phases making the change, of the throughput time, to six tasks per week one of these.

Delays in the research and design phases were encountered due to miscalculated time estimations. An expenditure analysis revealed the need for cost reductions, prompting adjustments to expenses, which currently represent 22% of the initial budget which is consumed only in human resources, fringe benefits and equipment. Also, because there was a reduction of cost in supplies, the actual budget of the project is $87,105.19. The expenditure analysis is calculated from February 13, 2024 to March 12, 2024.

The forthcoming phase entails the start of implementation and testing on March 19, 2024. The team will concentrate on the implementation of various front-end, back-end, and ML model training features while adhering to uniform coding standards and documentation practices.

In conclusion, Roof Rx remains steadfast in its commitment to delivering a bespoke and cost-effective solution that not only meets client expectations but also enhances inspection processes for improved efficacy and precision.

Table of Contents

1. Introduction (Written by: Kevin O. Valentín)

Roof Rx, a seasoned roofing contractor serving clients across California, Texas, and the southern United States since 1998, has continuously adapted to industry challenges. However, the onset of the global pandemic introduced unprecedented obstacles, disrupting traditional inspection methods, and necessitating the exploration of innovative solutions. Recognizing the limitations posed by lockdowns and safety measures, our team swiftly embraced drone technology for aerial imagery capture, revolutionizing the assessment process by eliminating the need for on-site personnel.

Yet, with the adoption of drone-generated imagery came a new challenge: the efficient analysis and identification of potential issues, particularly broken pipes. In response, we embarked on the development of a web application empowered by cutting-edge machine learning algorithms. This transformative approach not only accelerates inspections but also ensures unparalleled accuracy and cost-effectiveness compared to conventional methods.

Our primary objective is to achieve a minimum precision rate of 80% in identifying broken pipes, thereby streamlining the inspection workflow, and facilitating the rapid generation of reports. Leveraging the robust infrastructure provided by AWS services such as Amazon RDS, MySQL, and AWS Lambda, coupled with versatile tools like React and TensorFlow, forms the cornerstone of our technological ecosystem, guaranteeing scalability, reliability, and optimal resource utilization.

Our comprehensive testing process encompasses functional tests for upload functionality, accuracy tests for machine learning models, and user interface tests to ensure ease of use and responsiveness. We also conduct tests for API integration, database integrity, and user authentication mechanisms to verify secure access control and data management. Additionally, we implement contingency strategies to address any test failures promptly, including debugging and fixing issues, optimizing models, and enhancing user interfaces based on feedback. Through meticulous testing and continuous refinement, we aim to deliver a robust and user-friendly web application that meets the highest standards of quality and reliability.

The budget for the project underwent adjustments due to increased usage of Amazon Web Services (AWS) and the need to reduce costs from February 9, 2024, to March 8, 2024, resulting in a total expenditure. Recognizing the necessity for more AWS resources, the team revised the supplies budget initially outlined and updated it to reflect current requirements. Consequently, the estimated budget was reduced by 3%, with the current budget standing at $87,105.19. Notably, recurring supply costs are incurred due to AWS usage, amounting to an annual expense of $1,842.00, unless additional AWS services are required for performance enhancements or additional features. While supplies, miscellaneous, additional benefits, and indirect costs remain unused, they will be allocated during the implementation phase.

Looking ahead, the team is committed to adhering to revised timelines and meeting client expectations. With a focus on client satisfaction, we are committed to deliver a tailored and cost-effective solution that meets the distinctive needs of our client.

2. Technical Progress (Written by: Giomar Santiago)

The design process for our project generated a few design variations that the team considered and discussed with our client. Some of the design choices were specified by the client, such as the front-end library and the cloud services that would be made available to the team. The team had a couple of design alternatives that will be explained next. Our system consists of a relational database that requires the elaboration of an ER diagram and a Table Diagram. Bit Busters developed the ER diagram with one main difference, which consisted of a report entity that encompassed all the attributes needed for the generation of a report. The final alternative was chosen based on further evaluation by the team and feedback from the client.

Bit Busters developed the UI design considering the features and functionalities established by the client. To design the appearance of our system, the team chose to follow Google's Material Design, while also considering feedback from our client. This produced some variations and changes to our system which were presented to our client. The team followed the client's feedback and recommendations to produce a design that meets the client's expectations and provides better functionality. Given that the client required some specific technologies to develop the solution, Bit Busters adjusted the design to said technologies. The constraints in our development involved the selection of the front-end framework and the cloud services provider. For the front-end, the team will be using ReactJS [1], for which the team has previous experience. For more information on the front-end, see **Appendix D.1**.

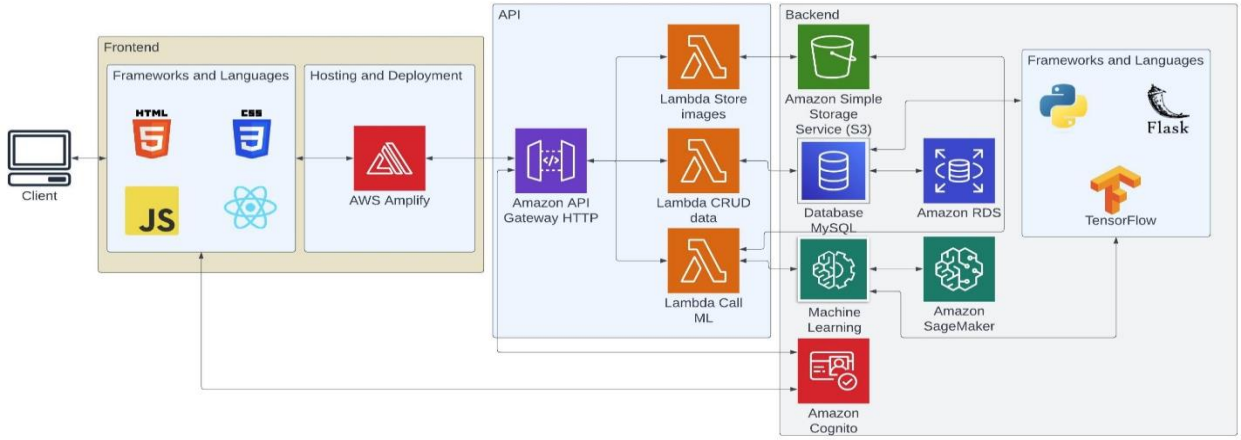

Figure 1. *System Architecture Diagram*

The system design of the team's web application is shown in **Figure 1** as the overall system architecture. The web application's front-end, which will be deployed and hosted on AWS Amplify, is what the client interacts with and sees, and it's where the tables, text boxes, icons, and pop-ups are displayed. The different display elements will collect user input, which will modify the displays correspondingly. Depending on the operation performed by the user, the client will receive the corresponding output from the API and back-end, which is where other components are contained and work collectively. The API component is broken down into the Amazon API Gateway, along with the Lambda functions. The back-end component is broken down into the storage service, which will be Amazon S3, the database, hosted on Amazon RDS, the Machine Learning model, which will be developed and trained with TensorFlow and deployed on Amazon SageMaker, and the user authentication, which will be managed with Amazon Cognito. The team will use Python as the main programming language for the back-end and the Flask framework. For more information on the back-end, see **Appendix D.2**. For detailed information on the system architecture and interfaces, see **Appendix E**.

3. Task Progress (Written by: Jeremy J. Márquez)

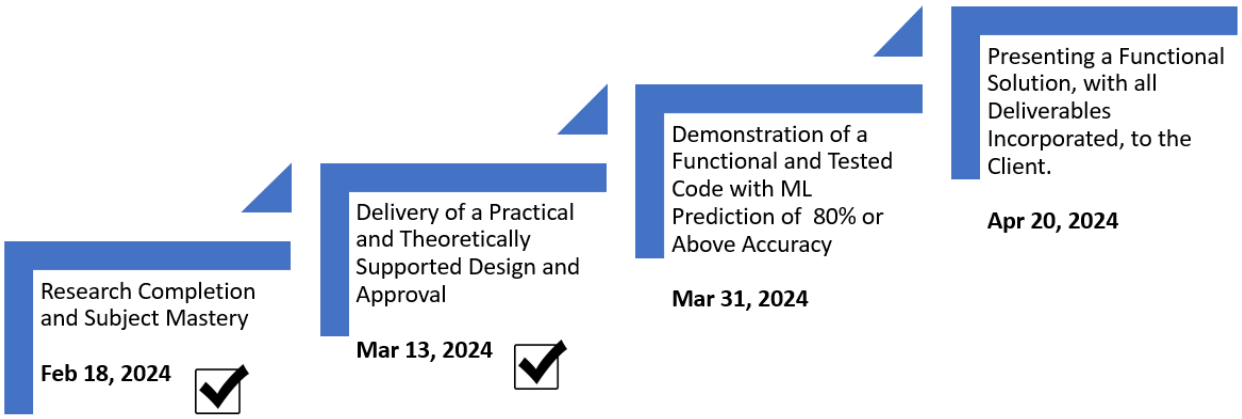

Figure 2. *Milestones*

Figure 2 contains all the completed and left to do milestones for Bit Busters team. Research completion was achieved by providing references and demonstrating acquired knowledge about the topics covered in the tasks. Design diagrams and a prototype were part of the completion of the second milestone. The research phase (first milestone) was delayed by two days and the design phase (second milestone) by 17 days. This was due to a failure in calculating the time required for research, for the first milestone, and the elaboration of the SRS document and all diagrams in the second milestone. To amend these delays the expected throughput changed from five tasks per week to six tasks per week. This leads to an estimated completion time of four weeks leaving one week remaining to handle unexpected events.

Currently, the team has completed 51% of all tasks based on the amount of completed tasks over the number of total tasks. The team is implementing the design, which consists of 38% of all tasks, planning to reach 89% completion when finished with the implementation according to the Kanban Board. For a more detailed breakdown of the task progress and graphics refer to **Appendix I.**

4. Expenditure Analysis (Written by: Juan D. Pérez)

Figure 3 represents the expenses of the team until March 12, 2024. In human resources and Fringe benefits the team spent the established amount that it was assigned in the initial budget. On Additional benefits, the team don't use any yet because none of our team or consultants did overtime work or took vacation during the beginning of the project until March 12, 2024. In equipment, the team already use everything they had due they needed laptops to work on the project. In case the team need another equipment, they will use indirect cost to solve any inconvenience. About the supplies, the team have not used any money yet, but in the implementation phase they will do it. Also, because the team need to reduce the cost of services, they updated some of them and now the cost of the supply has been reduced from $2,392.80 to $460.50, reducing the cost by 81%. The miscellaneous and indirect costs have not been touched on yet because the team didn't have any circumstance to use it. For now, the team has only used $19,889.85 of our total. This amount represents 22% of the team's initial budget. If the team updates the total using the new budget for the supplies, they have a total of $87,105.19 and based on this new budget, they spent around 23% of the new budget. The amount spent now is based on the 4 of 10 weeks the team have worked. For more detailed information, refer to **Appendix H**.

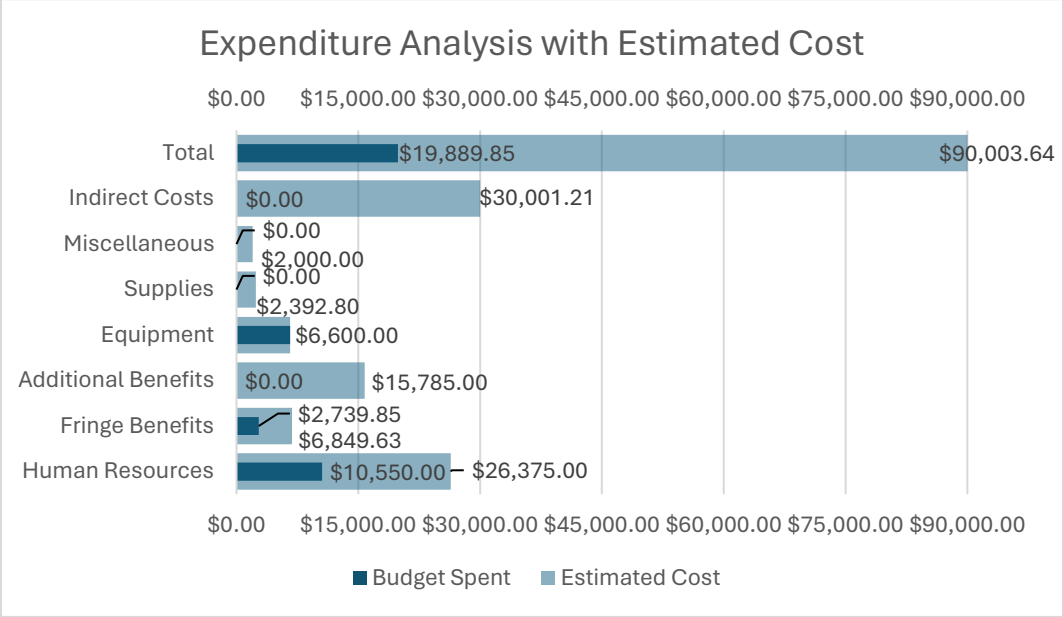

Figure 3: Expenditure Analysis of the initial budget from February 13, 2024 to March 12, 2024

5. **Next Steps** (Written by: Jeremy J. Márquez)

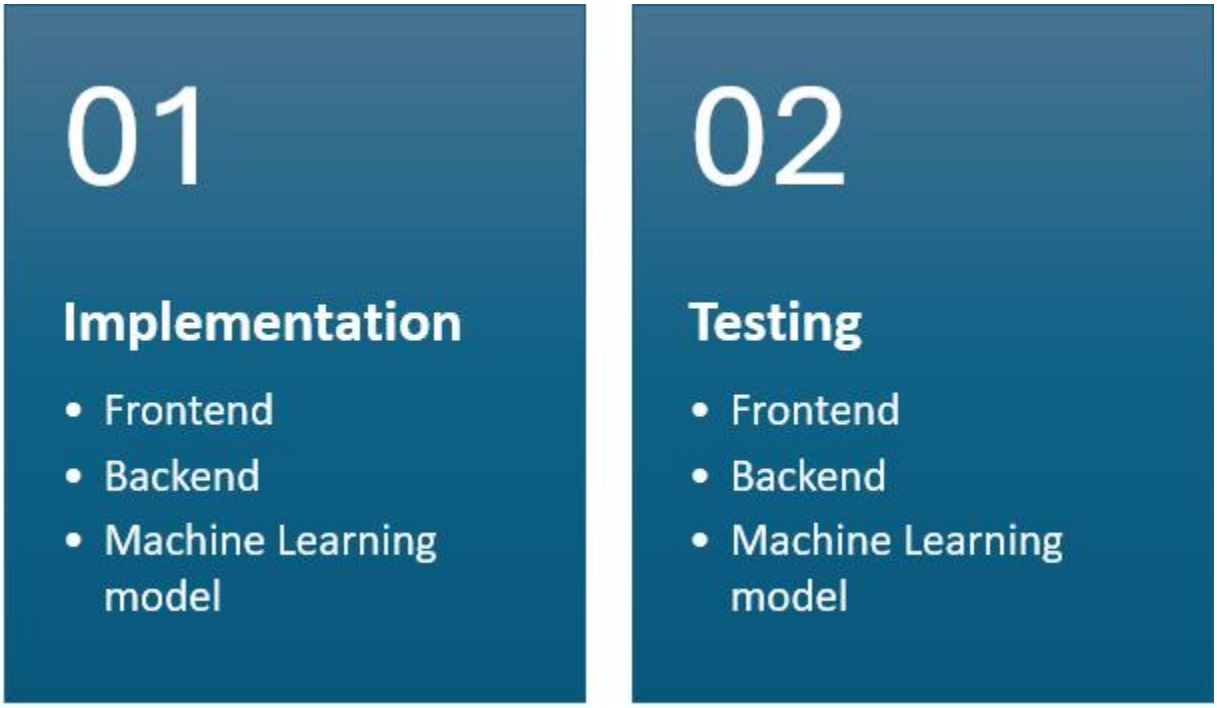

Figure 4. *Next Steps*

The team will start the implementation and testing established in **Figure 4** on Mar 19, 2024. Before entering the implementation phase, the team finished the required research for basic understanding of all topics involved in the project and elaborated all the design diagrams for the database (ER and Table Diagrams), front end web pages (Prototype, Use Case Diagram etc.), services (Cloud Software Architecture Diagram), and the whole system (System Architecture Diagram, Sequence Diagram etc.). The Kanban Board, with an expected throughput of five tasks per week, was the tool and measurement metric the team relied on, but now an expected throughput of six tasks per week will be used to measure the progress for these next steps.

For these ongoing and upcoming tasks, the team will be focusing on implementing different shared and individual tasks related to the different features of the front end, back-end, or ML model training. The coding style adhered to will be uniform and endorsed by the team, ensuring consistent documentation across all members to enhance code comprehension.

6. **Bibliography** (All)

[1] *"React – a JavaScript library for building user interfaces," – A JavaScript library for building user interfaces. [Online]. Available: https://reactjs.org/. [Accessed: 5-Mar-2023].*

[2] *"Angular - One framework. Mobile & desktop." [Online]. Available: https://angular.io/. [Accessed: 5-Mar-2023].*

[3] *"PostgreSQL vs MySQL: Which One is Better for Your Use Case?" [Online]. Available: https://www.integrate.io/blog/postgresql-vs-mysql-which-one-is-better-for-your-use-case/#:~:text=PostgreSQL%20is%20preferred%20for%20managing,comes%20to%20read%2Donly%20queries. [Accessed: 5-Mar-2023].*

[4] *"About TensorFlow," TensorFlow. [Online]. Available: https://www.tensorflow.org/about. [Accessed: 1-Mar-2023].*

[5] *"AWS Pricing Calculator," calculator.aws. https://calculator.aws/#/createCalculator/Amplify (accessed Mar. 16, 2024).*

[6] *"Precios de AWS Amplify | Servicios de frontend web y móviles | Amazon Web Services," Amazon Web Services, Inc. https://aws.amazon.com/es/amplify/pricing/ (accessed Mar. 16, 2024).*

[7] *"What is server-side rendering - AWS Amplify Hosting," docs.aws.amazon.com. https://docs.aws.amazon.com/amplify/latest/userguide/What-is-server-side-rendering.html (accessed Mar. 16, 2024).*

[8] *"AWS Pricing Calculator," calculator.aws. https://calculator.aws/#/createCalculator/Cognito (accessed Mar. 16, 2024).*

[9] *"AWS Pricing Calculator," calculator.aws. https://calculator.aws/#/createCalculator/APIGateway (accessed Mar. 16, 2024).*

[10] *"AWS Pricing Calculator," calculator.aws. https://calculator.aws/#/createCalculator/Lambda*

[11] *"Informática sin servidor – Precios de AWS Lambda – Amazon Web Services," Amazon Web Services, Inc. https://aws.amazon.com/es/lambda/pricing/#AWS_Lambda_Pricing (accessed Mar. 16, 2024).*

[12] *"AWS Pricing Calculator," calculator.aws. https://calculator.aws/#/createCalculator/S3*

[13] *"AWS Pricing Calculator," calculator.aws. https://calculator.aws/#/createCalculator/RDSMySQL*

[14] *"DB instance classes - Amazon Relational Database Service," docs.aws.amazon.com. https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.DBInstanceClass.html*

[15] *"AWS Pricing Calculator," calculator.aws. https://calculator.aws/#/createCalculator/SageMaker (accessed Mar. 16, 2024).*

Appendix

A. **Glossary** (Kevin O. Valentín)

A.1. Acronyms

1. ML- Machine Learning
2. AWS- Amazon Web Services
3. DBMS- Database Management System
4. SQL- Structured Query Language
5. RDS- Relational Database Service
6. DB- Database
7. GB- Gigabyte
8. UI- User Interface
9. PDF- Portable Document Format
10. API- Application Programming Interface
11. CMS- Content management systems
12. AWS S3- Amazon Simple Storage Service

A.2. Terminology

1. Amazon RDS- Amazon Relational Database Service, is a fully managed database service offered by AWS. It simplifies the setup, operation, and scaling of relational databases in the cloud, providing users with a cost-effective and scalable solution for storing and accessing their data.
2. MySQL- is an open-source relational database management system (RDBMS) known for its ease of use, reliability, and scalability. It's widely used for building web applications, powering content CMS, and managing data in various industries.
3. React- is a JavaScript library for building user interfaces, known for its component-based architecture. It efficiently updates and renders components as data changes, thanks to its virtual DOM. With React, developers can create reusable UI components and manage state effectively, enabling the creation of dynamic and interactive web applications.
4. TensorFlow- is an open-source machine learning framework developed by Google Brain. It's designed to facilitate the development and deployment of machine learning models, particularly deep learning models, for a wide range of tasks such as image and speech recognition, natural language processing, and reinforcement learning.
5. AWS Amplify- is a set of tools and services provided by Amazon Web Services (AWS) to streamline the development of cloud-powered applications. It offers a range of features including authentication, storage, APIs, and hosting, allowing developers to quickly build scalable and secure applications. With Amplify, developers can easily integrate backend services into their frontend applications,

enabling faster development cycles and reducing the complexity of managing infrastructure.

6. AWS API Gateway- is a fully managed service that enables developers to create, publish, and manage APIs at scale. It simplifies the process of building RESTful APIs and WebSocket APIs, providing features for authentication, authorization, and monitoring. With AWS API Gateway, developers can easily connect their applications with backend services or other AWS services, reducing the complexity of API management and accelerating development cycles.
7. AWS Lambda- is a serverless compute service on AWS, allowing developers to run code in response to events without managing servers. It scales automatically and charges only for the compute time consumed, simplifying development and reducing operational overhead. With Lambda, developers can focus on writing code rather than managing infrastructure, enabling rapid innovation.
8. AWS S3- is a scalable cloud storage service provided by Amazon Web Services. It allows users to store and retrieve any amount of data from anywhere on the web. With features like high availability, durability, and security, S3 is commonly used for data backup, static website hosting, content distribution, and application data storage. Developers can seamlessly integrate S3 into their applications using AWS SDKs or APIs, making it a versatile and reliable storage solution for a wide range of use cases.
9. AWS SageMaker-is a managed service by Amazon for building, training, and deploying machine learning models at scale. It simplifies the entire machine learning workflow, from data labeling and model training to deployment and monitoring. With SageMaker, developers and data scientists can experiment with algorithms, optimize models, and deploy them into production seamlessly.
10. AWS Cognito- is a fully managed identity service provided by Amazon Web Services, enabling developers to add authentication, authorization, and user management to their applications easily. It offers features like user sign-up and sign-in, multi-factor authentication, and user directory management. With Cognito, developers can securely authenticate users across devices and platforms while maintaining control over user data and access permissions. It simplifies the implementation of authentication and authorization workflows, allowing developers to focus on building their applications.
11. Flask- is a lightweight Python web framework, known for its simplicity and flexibility, ideal for building web applications quickly and efficiently. It provides the essential tools for web development without imposing strict architectural patterns, allowing developers to structure their applications as needed. With Flask, developers can focus on writing clean and concise code, making it a popular choice for a variety of web projects.

B. User Requirements (Kevin O. Valentín)

A. User Requirements

A.1 Functional

A.1.1 All users will be able to create an account for accessing the application's features.

A.1.2 All users will be able to create a password for their account.

A.1.3 All users will be able to provide their email for account creation.

A.1.4 All users will be able to select which type of account they want to create.

A.1.5 All users will be able to provide the web application with a key to create an account.

A.1.6 All users will be able to login to the web application using their credentials (email and password).

A.1.7 All users will be able to update their password.

A.1.8 All users will be able to recover their password.

A.1.9 All users will be able to update their first name to keep their information up to date.

A.1.10 All users will be able to update their last name to keep their information up to date.

A.1.11 All users will be able to update their phone number to keep their information up to date.

A.1.12 All users will be able to upload images for the ML model to analyze.

A.1.13 All users will be able to type the client's first name for the generation of a report.

A.1.14 All users will be able to type the client's last name for the generation of a report.

A.1.15 All users will be able to type the client's phone number for the generation of a report.

A.1.16 All users will be able to type the client's email for the generation of a report.

A.1.17 All users will be able to type the client's address (country, state, city, street, door number, zip code) for the generation of a report.

A.1.18 All users will be able to add comments to the report.

A.1.19 Supervisor Inspectors will be able to access all reports for information.

A.1.20 Supervisor inspector (admin account) will be able to delete user accounts.

A.1.21 Inspectors will only be able to access their generated reports for information.

A.1.22 All users will be able to electronically sign the reports for virtual management.

A.1.23 All users will be able to export reports in PDF format.

A.1.24 All users will be able to edit the reports before signing them.

A.1.25 All users will be able to log out of the web application.

A.1.26 All users will be presented with a slide of images with the machine learning results.

A.1.27 All users can delete images that were not correctly analyzed by the machine learning model selected.

A.2 Non-functional

A.2.1 The user interface will be intuitive and easy to navigate, requiring minimal knowledge for users to operate effectively.

A.2.2 The web application will respond promptly to user interactions, under typical load conditions.

A.2.3 The web application will provide informative error messages to users in case of input validation errors.

B. System Requirements

B.1 Functional

B.1.1 The web application will allow two types of accounts to be created: Supervisor Inspector and Inspector.

B.1.2 The web application will require generated keys for the user to create an account.

B.1.3 The web application generated keys, for account creation, will be different for Supervisor Inspector and Inspector accounts.

B.1.4 All web applications generated keys, for account creation, would expire after 30 minutes.

B.1.5 The web application will notify the user if the account was successfully created or not when creating the account.

B.1.6 The web application will let the user sign-in if the account is successfully created.

B.1.7 The web application will allow users with an account to login using their email and password.

B.1.8 The web application will provide the user with the ability to recover access to the account.

B.1.9 The web page will allow users to reset their password when requesting account recovery.

B.1.10 The web application will ensure the new password provided by the user matches in both fields used to change it.

B.1.11 The web application will be able to identify two types of user roles: general user (Inspector) and administrator (Supervisor Inspector).

B.1.12 The web application will have a tab to display the user's account information.

B.1.13 The web application will allow users to navigate the different pages.

B.1.14 The web application will allow the user to change their account information.

B.1.15 The web application will ask the user for confirmation when attempting to change information.

B.1.16 The web application will allow new reports to be created.

B.1.17 The web application will allow the user to input the client's first name for report generation.

B.1.18 The web application will allow the user to input the client's last name for report generation.

B.1.19 The web application will allow the user to input the client's phone number for report generation.

B.1.20 The web application will allow the user to input the client's email for report generation.

B.1.21 The web application will allow the user to input the client's address (country, state, city, street, door number, zip code) for report generation.

B.1.22 The web application will allow the user to input comments for report generation.

B.1.23 The web application will have an interface to allow the user to upload images to the ML model to analyze.

B.1.24 The web application will allow the user to edit a report before signing it.

B.1.25 The web application will be able to generate reports of roof inspections.

B.1.26 The web application will include in the reports any comments written by the user.

B.1.27 The web application will provide the ability to view the information saved on the reports.

B.1.28 The web application will have a tab to display a history of generated reports.

B.1.29 The web application will automatically sort the generated reports from newest to oldest.

B.1.30 The web application will allow the user to filter the reports using date as criteria.

B.1.31 The web application will allow the user to filter the reports using the client's name as criteria.

B.1.32 The web application will only allow users access to the features that are associated with their account's role (Supervisor Inspector and Inspector).

B.1.33 The web application will allow Supervisor Inspector accounts to delete user accounts.

B.1.34 The web application will allow Supervisor Inspector accounts to access all stored reports.

B.1.35 The web application will limit inspector accounts to access only the reports generated by them.

B.1.36 The ML model will detect and identify broken pipes in the roof images for report inclusion.

B.1.37 The web application will ask for the user's old password for verification when the user attempts to update their password.

B.1.38 The web application will include the inspector's information and the images when generating a report.

B.1.39 The web application will autofill the inspection date when generating a report.

B.1.40 The web application will provide an estimate of costs when generating a report.

B.1.41 The web application will indicate how many broken pipes were identified by the ML when generating a report.

B.2 Non-Functional

B.2.1 The system will implement a role-based access control system to grant or restrict data access based on user roles and assigned permissions.

B.2.2 The web application will align with Google's Material Design web guidelines.

B.2.3 The ML model will have a minimum precision of 80% in identifying broken pipes.

B.2.4 The ML model will be able to analyze an image in 15 seconds or less.

B.2.5 The web application will store all reports and their corresponding data for future reference.

B.2.6 The web application will store all data related to the user's account and information for future reference or updates.

B.2.7 The web application will store URLs which redirect to the images used for the corresponding generated report.

B.2.8 The format of images needs to be in JPEG for analysis by the ML model.

B.2.9 Each report will adhere to an established design format to maintain uniformity across reports.

B.2.10 The webpage will be accessible by the following browsers: Chrome, Safari, Firefox, and Microsoft Edge.

B.2.11 The webpage will be able to display errors in uploading images and image scanning failure.

C. System Specifications (Written by: Jeremy J. Márquez)

TensorFlow

Specification	Description
Framework Name	TensorFlow
Purpose	Machine learning framework for building and training models
Version	2.15.0
Programming Language	Python (primary language for TensorFlow)
Supported Operative Systems	Linux, macOS, Windows
Model Compatibility	**Convolutional Neural Networks (CNNs)**: For image processing tasks.
	Transfer Learning with Pretrained Models: Facilitates transfer learning, enabling fine-tuning of pretrained models like MobileNet, Inception, ResNet, EfficientNet.
	Data Augmentation: Supports data augmentation techniques to increase diversity of training data and improve model generalization.
Model Deployment	Options for deploying trained models (e.g., **AWS SageMaker**, TensorFlow Serving)

Table 1. *TensorFlow Framework Specifications*

React

Specification	Description
Framework Name	React
Purpose	JavaScript library for building user interfaces
Version	18.0.0+
Programming Language	JavaScript (ES6+)
Supported Platforms	Web

Table 2. *React Library Specifications*

AWS Amplify

Specification	Description
Service Name	AWS Amplify
Purpose	Full-stack cloud solution for building and deploying web and mobile applications
Features	Key features provided by AWS Amplify (e.g., **hosting**, authentication, API)
Supported Platforms	Web, mobile (iOS, Android), desktop

Specification	Description
Supported Frameworks	**React**, Angular, Vue.js, React Native, Ionic, Flutter
Authentication	Authentication and authorization services (e.g., **Amazon Cognito**)
API	Managed API services (e.g., REST APIs, **AWS API Gateway**)
Hosting	Hosting solutions for static and dynamic web applications
Continuous Deployment	Automated deployment pipelines with Git-based workflows
Functions (Lambda)	Serverless functions (**AWS Lambda**) for backend logic

Table 3. *AWS Amplify Cloud Service Specifications*

AWS API Gateway

Specification	Description
Service Name	AWS API Gateway
Purpose	Fully managed service for creating, deploying, and managing APIs
API Types	**RESTful APIs**, WebSocket APIs
Integration	Integration with AWS services (e.g., **AWS Lambda**)
HTTP Methods	Supported HTTP methods (e.g., **GET, POST, PUT, DELETE**)
Authentication	Authentication options (e.g., **Amazon Cognito**, AWS IAM, OAuth)

Table 4. *AWS API Gateway Cloud Service Specifications*

AWS Lambda

Specification	Description
Service Name	AWS Lambda
Purpose	Serverless compute service for running code without provisioning or managing servers
Supported Languages	Programming languages supported (e.g., Node.js, **Python**, Java)
Runtime Environment	Environment configurations for running code (e.g., Node.js 14.x, **Python 3.8+)**
Function Triggers	Event sources that trigger function execution (e.g., **API Gateway**, **AWS S3**)
Execution Timeout	15 sec
Integration	Integration with other AWS services (e.g., **AWS S3**, **API Gateway**)

Table 5. *AWS Lambda Cloud Service Specifications*

AWS S3

Specification	Description
Service Name	AWS S3 (Simple Storage Service)
Purpose	Object storage service for storing and retrieving data
Data Types	Supported data types (**Images**, **Binary Data**)
Storage Classes	Standard
Object Management	**Upload**, download, copy, and **delete** objects in buckets
Event Notifications	Event-driven notifications for S3 object actions (**Trigger Lambda Functions**)

Table 6. *AWS S3 Cloud Service Specifications*

AWS RDS

Specification	Description
Service Name	AWS RDS (Amazon Relational Database Service)
Purpose	Managed relational database service for deploying, operating, and scaling databases
Database Engines	Supported database engines (e.g., **MySQL**, PostgreSQL, Oracle)
Deployment Options	Single-AZ
Automated Backups	Automated backups with configurable retention periods

Table 7. *AWS RDS Cloud Service Specifications*

AWS SageMaker

Specification	Description
Service Name	AWS SageMaker
Purpose	Fully managed service for building, training, and deploying machine learning models
Supported Frameworks	Machine learning frameworks supported (e.g., **TensorFlow**, PyTorch, MXNet)
Model Deployment	Deployment of trained models to production environments (e.g., **TensorFlow Model**)
Inference	Real-time endpoints for model predictions
Auto Scaling	Auto-scaling of deployed endpoints based on workload

Table 8. *AWS SageMaker Cloud Service Specifications*

AWS Cognito

Specification	Description
Service Name	AWS Cognito

Specification	Description
Purpose	Fully managed service for user identity and access management
User Pools	User directories for managing user sign-up, sign-in and password reset
Authentication	Authentication methods (e.g., **username/password**, **generated tokens**)
Security	AWS Cognito is compliant with industry standards such as ISO 27001, SOC 2, and GDPR, demonstrating its commitment to security and data privacy.
User Data Protection	Compliance with GDPR and data privacy regulations

Table 9. *AWS Cognito Cloud Service Specifications*

MySQL

Specification	Description
Database Engine	MySQL
Purpose	Relational database management system (RDBMS)
Version	8.0.0+
Data Types	Supported data types (e.g., **INTEGER**, **VARCHAR**, **DATE**, **NUMERIC**)

Table 10. *MySQL DBMS Specifications*

Flask

Specification	Description
Framework Name	Flask
Purpose	Micro web framework for building web applications in Python
Version	2.2.0+
Routing	URL routing for defining endpoints and handling requests
Views	Views and view functions for processing HTTP requests
Request Handling	Request handling mechanisms (e.g., request object, request methods)
Response Handling	Response handling mechanisms (e.g., response object, response methods)
Database Integration	Integration with databases (e.g**., SQLAlchemy**, Flask-SQLAlchemy)

Table 11. *Flask Framework Specifications*

D. Analysis of Alternatives (Written by: Giomar Santiago)

In the team's first meeting with the client, the client came up with some specific technologies they wanted the team to use for the development of the solution. The constraints presented by the client encompassed the use of the ReactJS library and the AWS cloud services.

D.1 Front-End

For the development of the front-end, the client established ReactJS as a constraint. This is one of the two technology constraints the client established to Bit Busters for the development of the solution. ReactJS is considered to have a medium difficulty compared to other frameworks considered to have a higher difficulty, such as Angular [2]. Another advantage to consider is that React has big community support, which is advantageous in the case any questions arise in development. Furthermore, all team members have previous experience with ReactJS, which is helpful to optimize development time.

	ReactJS	**Angular**
Language	JavaScript	TypeScript
Cost	$0 (open source)	$0 (open source)
Learning Curve	Medium	Hard

Table 12. *Front-End Information*

D.2 Back-End

For the development of the back-end, the team considered the use of Python or JavaScript. These two programming languages were suggested by the client, but the team had the freedom to take the decision. The team decided to develop the back-end with Python, since it offers a wide range of libraries for data management which is essential for manipulating datasets and facilitates the integration of ML into the back-end. Python is also known for its simplicity and reliability, which can significantly speed up the development process. Python also supports popular web frameworks like Flask, which simplify backend development for web applications.

	Python	**JavaScript**
Frameworks	Flask, Django	Express.js, Node.js
ML libraries availability	Extensive	Good, but not as mature
Simplicity	Simpler/Cleaner	More verbose

Table 13. *Back-end information*

D.3 Machine Learning

For the development of the machine learning capabilities of our web-application, the team investigated the different machine learning libraries available for Python. The team

considered options such as TensorFlow and PyTorch, and ultimately decided on using TensorFlow [4] for its broad model availability for image classification and vast architecture support. It also offers a good balance between speed and accuracy, while also providing different techniques to handle limited data sets such as Data Augmentation and Transfer Learning. TensorFlow has been widely adopted, leading to a strong community, extensive documentation, and a wealth of resources including tutorials, case studies, and best practices. It's important to note that the team also considered using Amazon SageMaker to develop, train, and deploy the ML capabilities of our project, but through meetings with the client it was decided the cost would be too high. For this reason, Amazon SageMaker will only be used to deploy our ML model.

	TensorFlow	**PyTorch**
Model availability and architecture support	Vast	Not as extensive
Image classification support	Comprehensive	Strong
Community adoption	Greater	Growing

Table 14. *Machine learning information*

D.4 Database

For the database, the team was only able to consider the database services available on AWS, since one of the constraints established by the client was the use of Amazon Web Services. Taking this into consideration, the team reviewed Amazon RDS and Amazon Aurora. In conversations with the client, the use of Amazon RDS was decided, taking into consideration the higher base cost per GB of storage of Amazon Aurora. The next decision the team had to make was regarding the DBMS. MySQL and PostgreSQL were the two main DBMS that the team reviewed. MySQL was chosen due to reliability and general faster performance [3]. MySQL has been around for a longer time and has a larger user base. As a result, there are more resources, tutorials, and community support available for MySQL.

Database	**Storage Cost**	**Pricing Model**
Amazon RDS	Lower base cost per GB	DB instance type and storage allocated
Amazon Aurora	Higher base cost per GB	DB instances and storage usage
DBMS	**Performance/Reliability**	**Community Support**
MySQL	Faster query execution speed	More extensive, has been around for longer
PostgreSQL	Better when performing complex queries	Good

Table 15. *Database information*

E. System Architecture and Interfaces (Written by: Jeremy J. Márquez)

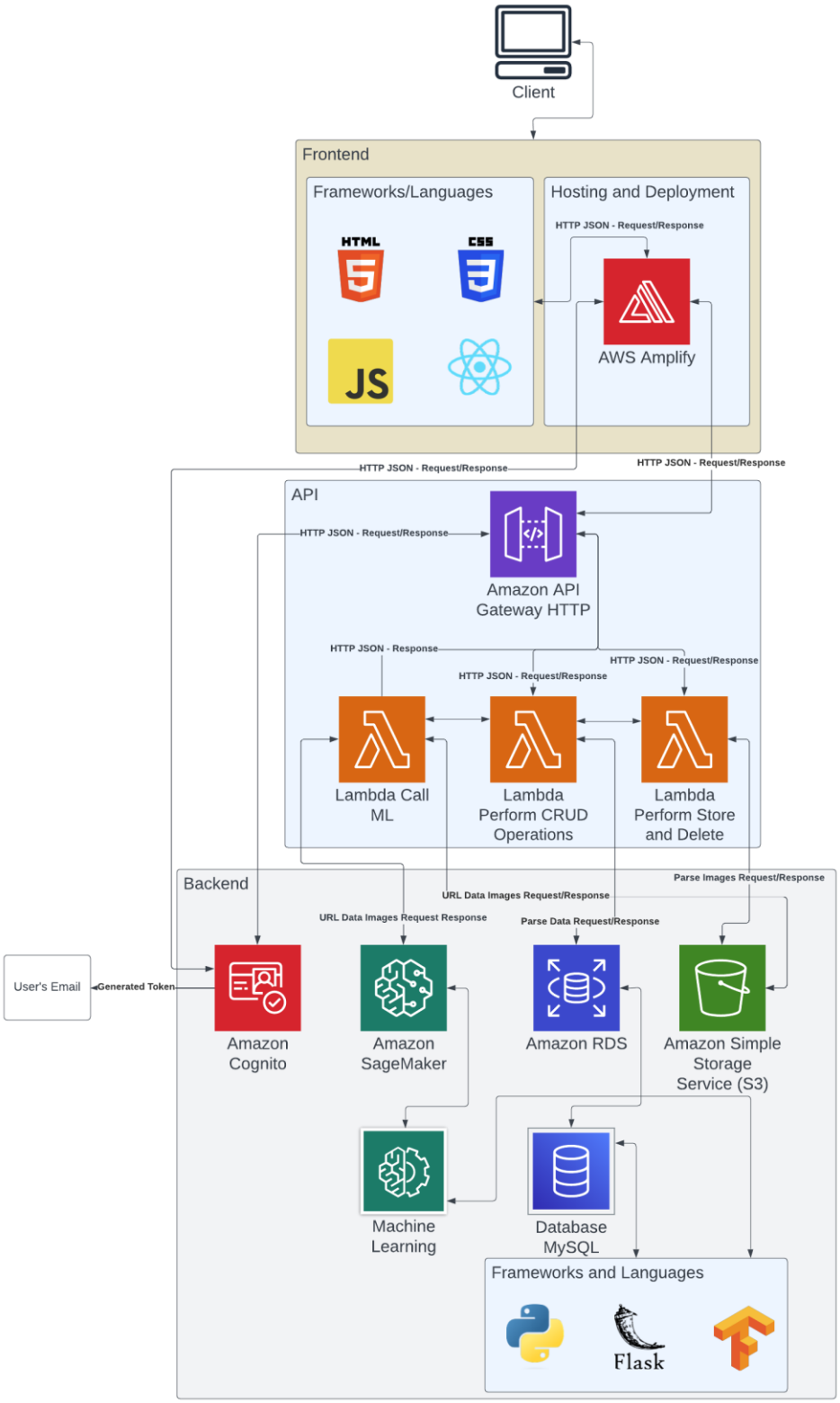

Figure 5. *Detailed System Architecture Diagram*

Bit Buster's Solution consists of a Front End, API and Back End with all communications occurring with JSON as its data interchange format (**Figure 5**). The main components of each of these main elements are the following:

- **Front End**
  - **Frameworks and Languages**
    - HTML
      - Creates and structures the web pages
    - CSS
      - Styles and formats the appearance of HTML elements
    - JavaScript
      - Creates dynamic and interactive behavior on the web pages
    - React (JavaScript Library)
      - Builds the user interface for the web application
  - **Hosting and Deployment**
    - AWS Amplify
      - Deploys and hosts the Front – End code of the web application
      - Send Requests to:
        - AWS API Gateway
        - AWS Cognito
- **API**
  - **AWS API Gateway**
    - Routes the different requests from the following sources:
      - AWS Amplify
      - AWS Cognito
    - The types of requests that can be routed are the following:
      - CRUD Operations to the Database
      - Delete or Store operations to AWS S3 Bucket
    - Receives responses from all AWS Lambda functions
  - **AWS Lambda**
    - Call ML Model
      - Calls the ML Model, for prediction, by providing the URLs of the images stored in the S3 bucket.
        - These URLs are provided as inputs by the S3 Bucket when the user upload images to generate reports, and they are stored in the S3 Bucket.
      - Receives the images URLs and predictions to send them to AWS API Gateway for further interactions in the system.
        - These URLs and predictions are provided as inputs by the ML Model.

        - Perform CRUD Operations
            - Calls AWS RDS to perform one of the following CRUD operations on the Database:
                - Get, Post, Update or Delete
            - Receives and returns to AWS API Gateway the response status code from AWS RDS.
        - Perform Store and Delete in S3
            - Interacts with the S3 Bucket to:
                - Store
                    - Occurs when images are uploaded for report generation
                - Delete
                    - Occurs when the report generation is canceled after the images where uploaded and stored in the S3

- **Back End**
    - **AWS Cognito**
        - Receives request from AWS Amplify to:
            - Validate user when signing in
            - Generate tokens for signing up or resetting password
        - Calls AWS API Gateway to:
            - Request user information for validation when signing in
        - Sends generated tokens to the user's email to:
            - Provide access to password reset
            - Enable account creation (sign up)
        - Receives response from API Gateway regarding the user's information
        - Sends response to AWS Amplify to:
            - Permit the users access to their accounts (sign in)
    - **AWS SageMaker**
        - Handle requests and responses between the Lambda "Call ML Model" and the ML model.
    - **AWS RDS**
        - Handle requests and responses between the Lambda "Perform CRUD Operations" and the Database.
    - **AWS S3**
        - Receives request from Lambda "Perform Store and Delete" to:
            - Store information when:
                - generating report
            - Delete information when

        - Canceling report
        - Denying Images
    - Sends requests with images URLs to Lambda “Call ML Model” after storing images
- **Machine Learning Model**
    - Performs predictions based on the input received through AWS SageMaker
- **Database**
    - Relational Database Management System (RDBMS)
        - MySQL
    - Stores data based on the input received through AWS RDS
- **Frameworks and Languages**
    - Python
        - Manages data and routes request and responses
    - Flask
        - provides tools and libraries to simplify the handling of HTTP requests and responses, URL routing, and interaction with databases
    - TensorFlow
        - Provides tools to build and train the ML Model

F. Design Documentation (All)

i. Cloud Architecture (Juan D. Pérez)

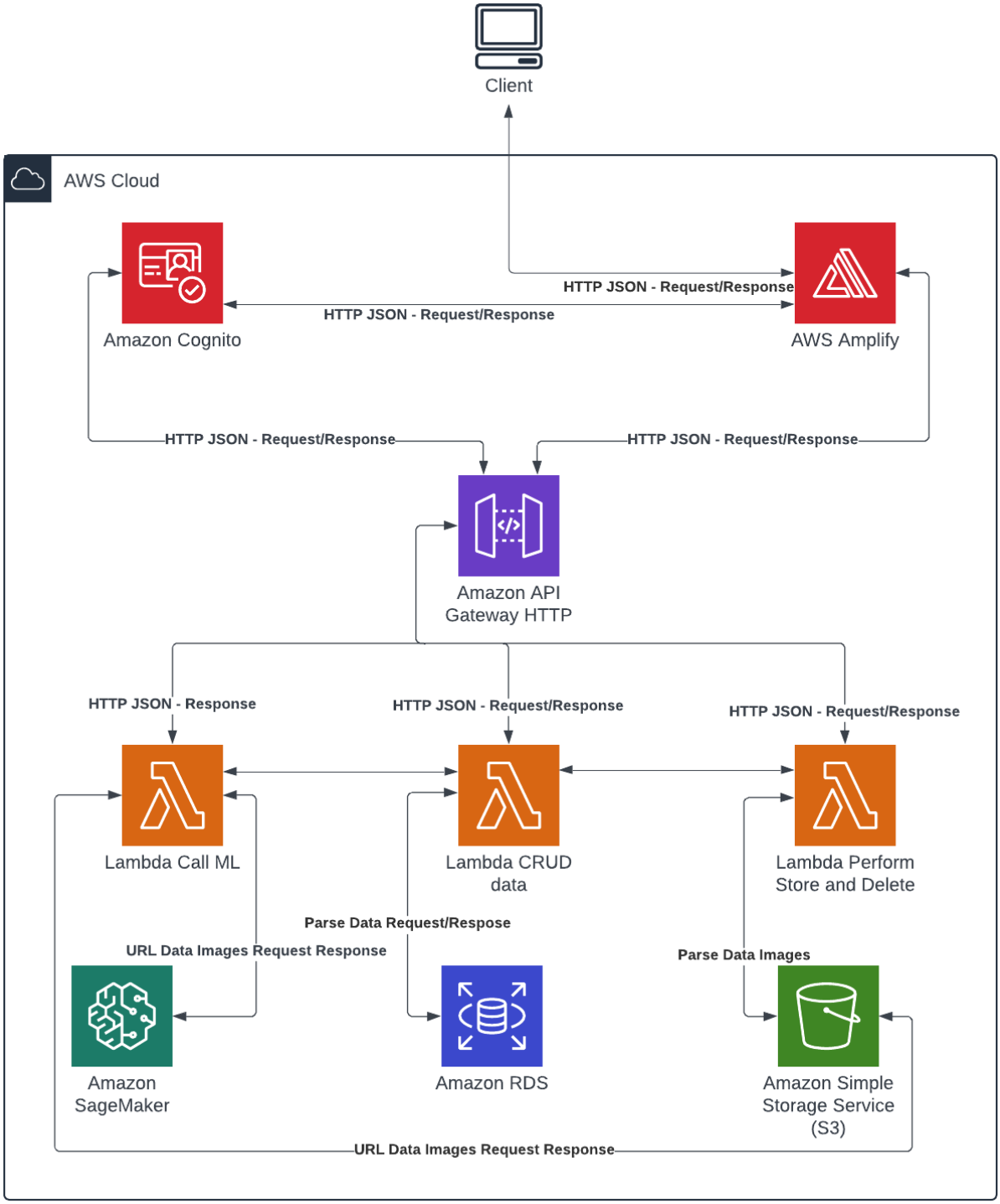

Figure 6: *Cloud Architecture Services*

ii. Flowchart Diagrams (Written by: Juan D. Pérez)

The blue boxes in each Figure represent the pages of the web application. The darker colors represent processes, and the light colors represent conditions that some processes have. Every page has a different color for process and conditions to represent that are different pages. The Sign-in page is the page our web applications begin. The user can go from the sign-in page to the sign-up to create an account, to the home page to begin using the functions of the app or to the reset password page in case you forgot your password. If the user sign-out from the home page, Edit Personal information page, History of reports page, Visual report page, report generation page, admin home page, upload images page, inspectors table page, Report Generated page or Inspect Photo page, the user will be redirected to the Sign-in page. If the user is on the Home page, Administrator home page (in case the user is Admin), History of Reports page or Edit Personal Information page it can navigate between these pages. If the user is on the Visual Report Page, Report Generated page, Inspect Photos Page or Report Generated page it can go to Home page, Administrator Home (in case the user is Admin), History of Reports page or Edit Personal Information page. The light-yellow conditions in *Figure 19* and *Figure 20* are the conditions to go to Home page, Sign-in page, Edit Personal Information page or History of Report page. If the user is in the sign-up page, it can navigate back to the sign-in page if they already have an account, or it will be redirected if the account were created. After the user upload images and presses start report, it will go to Report Generation page. When the user finishes the process in the report generation page it will go to the inspect photo page, and when the user finishes the process in the inspect photo page, it will be redirected to Report Generated page. In case the user is on the History of report page, and it wants to view a report, after the user press's view, it will redirected to the Visual Report page.

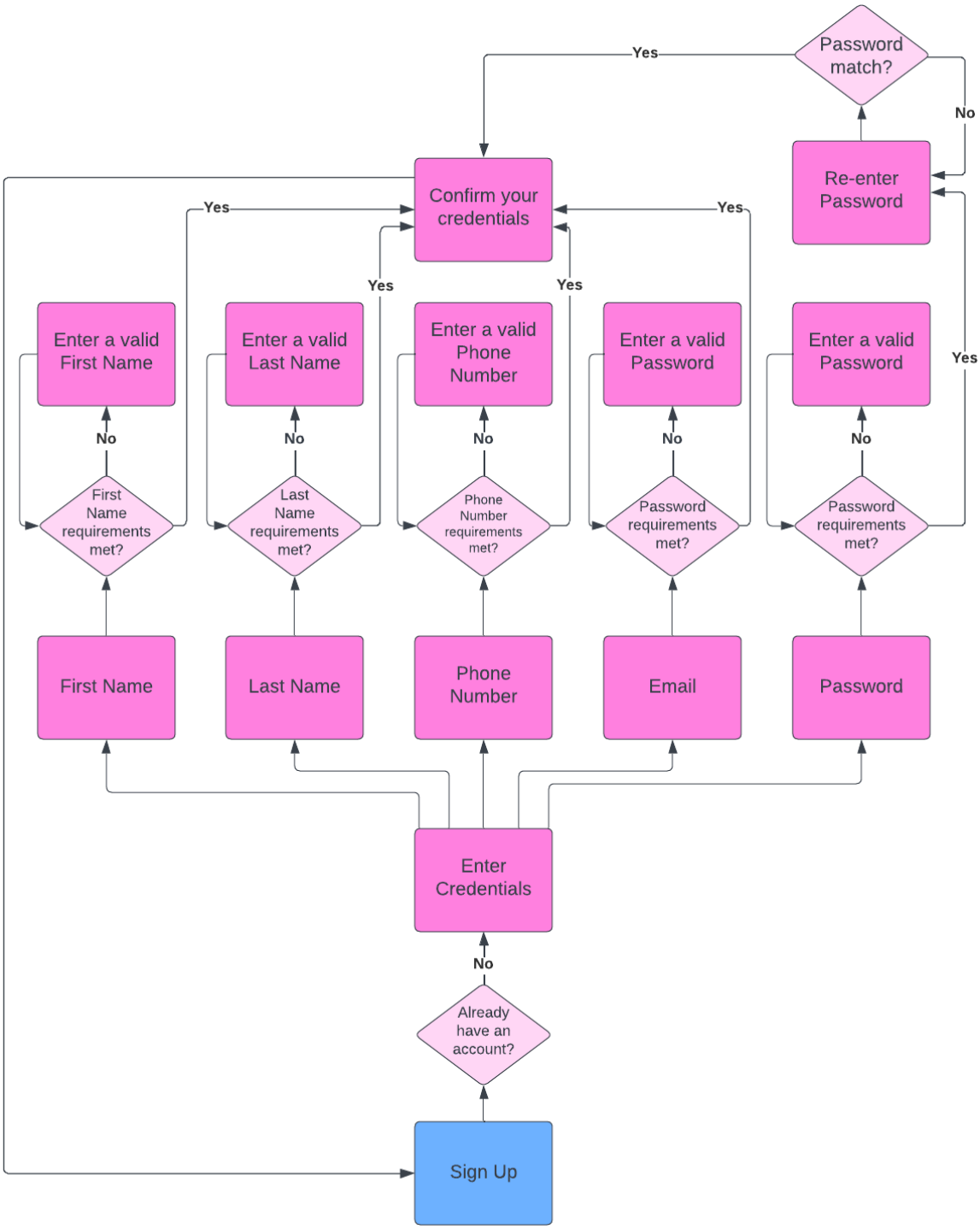

Figure 7: *Sign-Up page Flowchart Diagram*

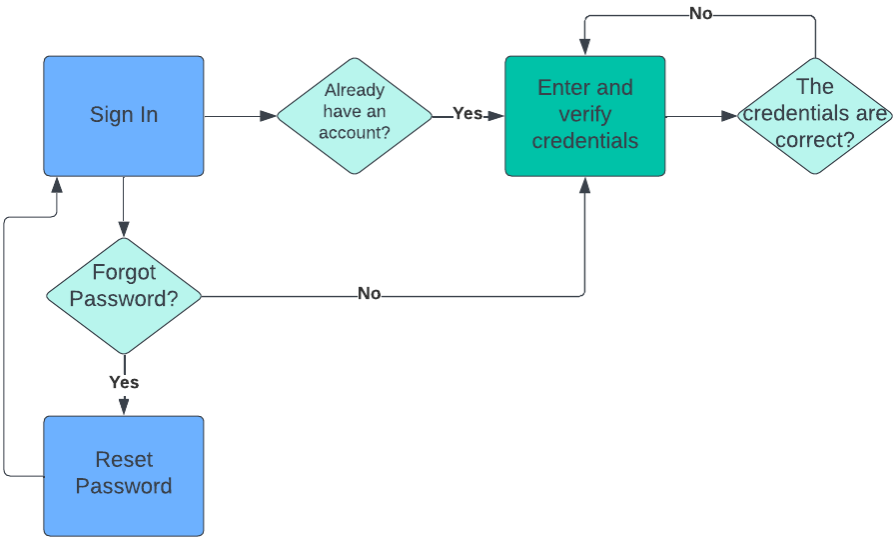

Figure 8: *Sign-in page Flowchart Diagram*

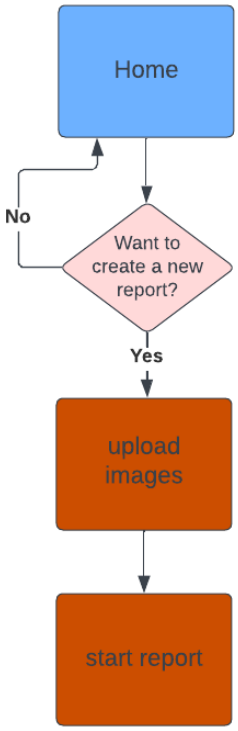

Figure 9: *Home page Flowchart Diagram*

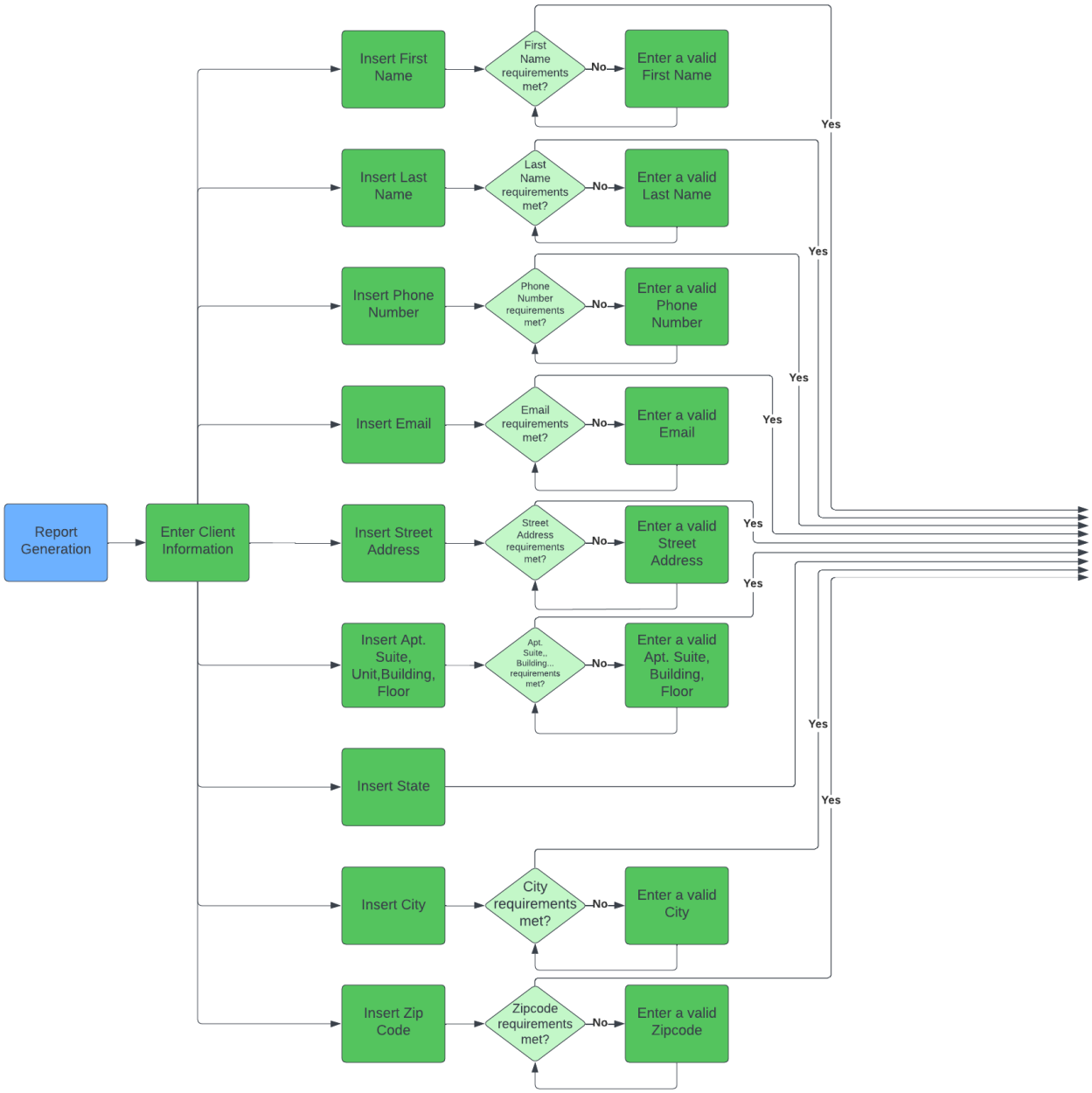

Figure 10: *Report generation page Flowchart Diagram*

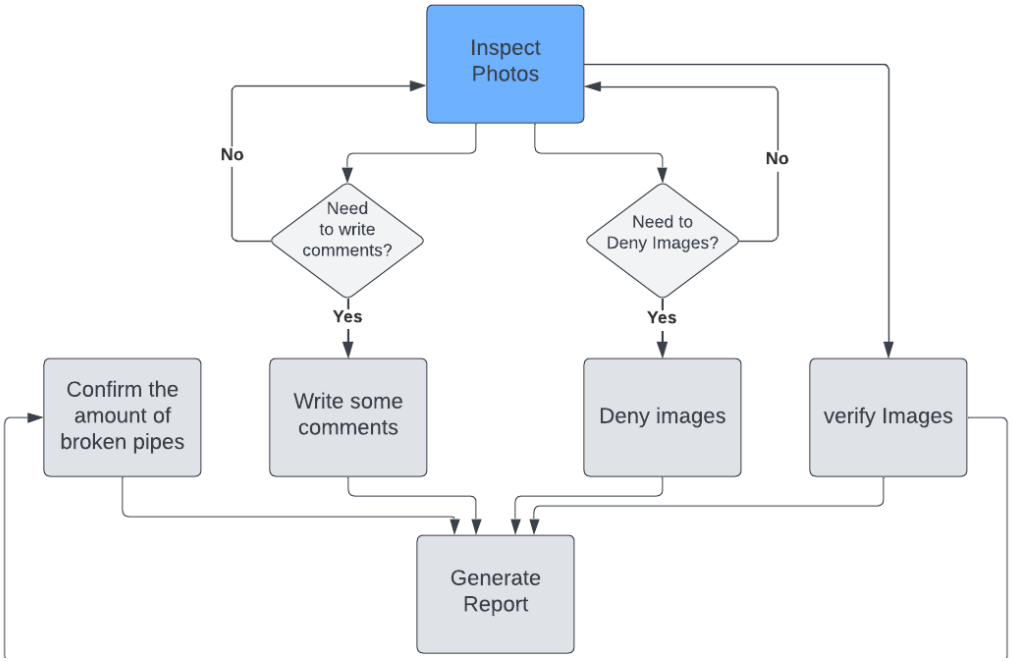

Figure 11: *Inspect Photos page Flowchart Diagram*

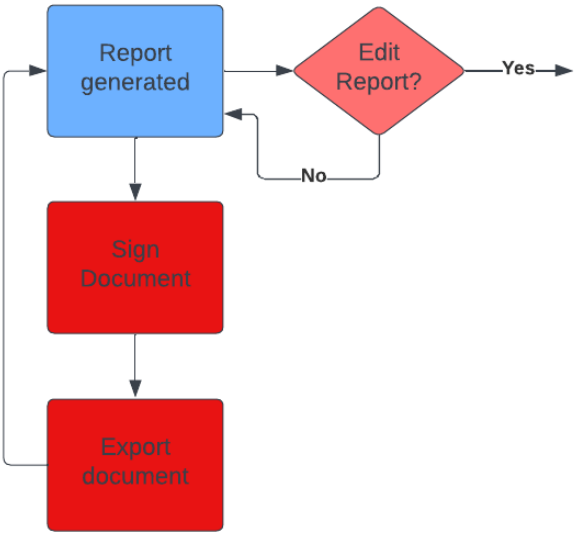

Figure 12: *Report Generated page Flowchart Diagram*

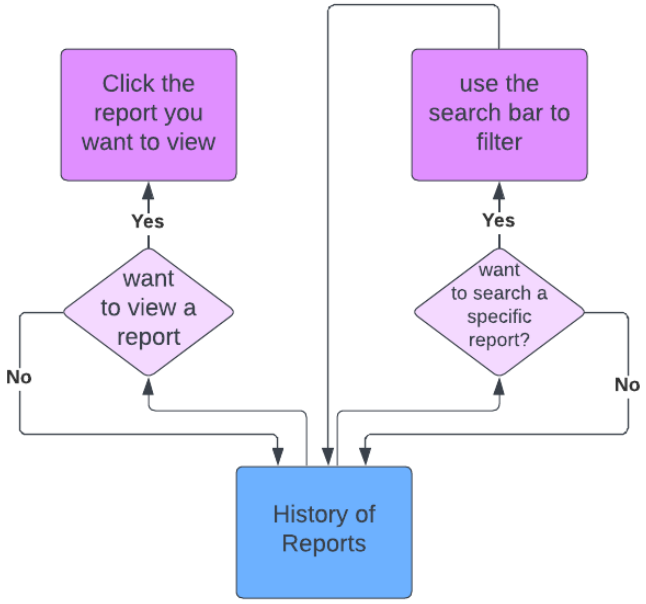

Figure 13: *History of Reports page Flowchart Diagram*

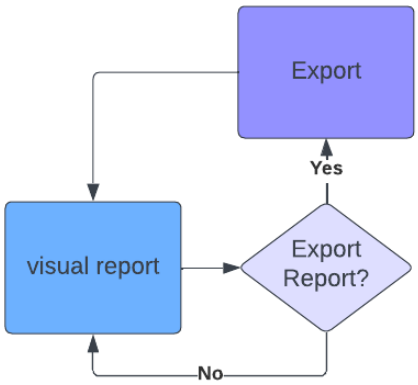

Figure 14: *Visual Report page Flowchart Diagram*

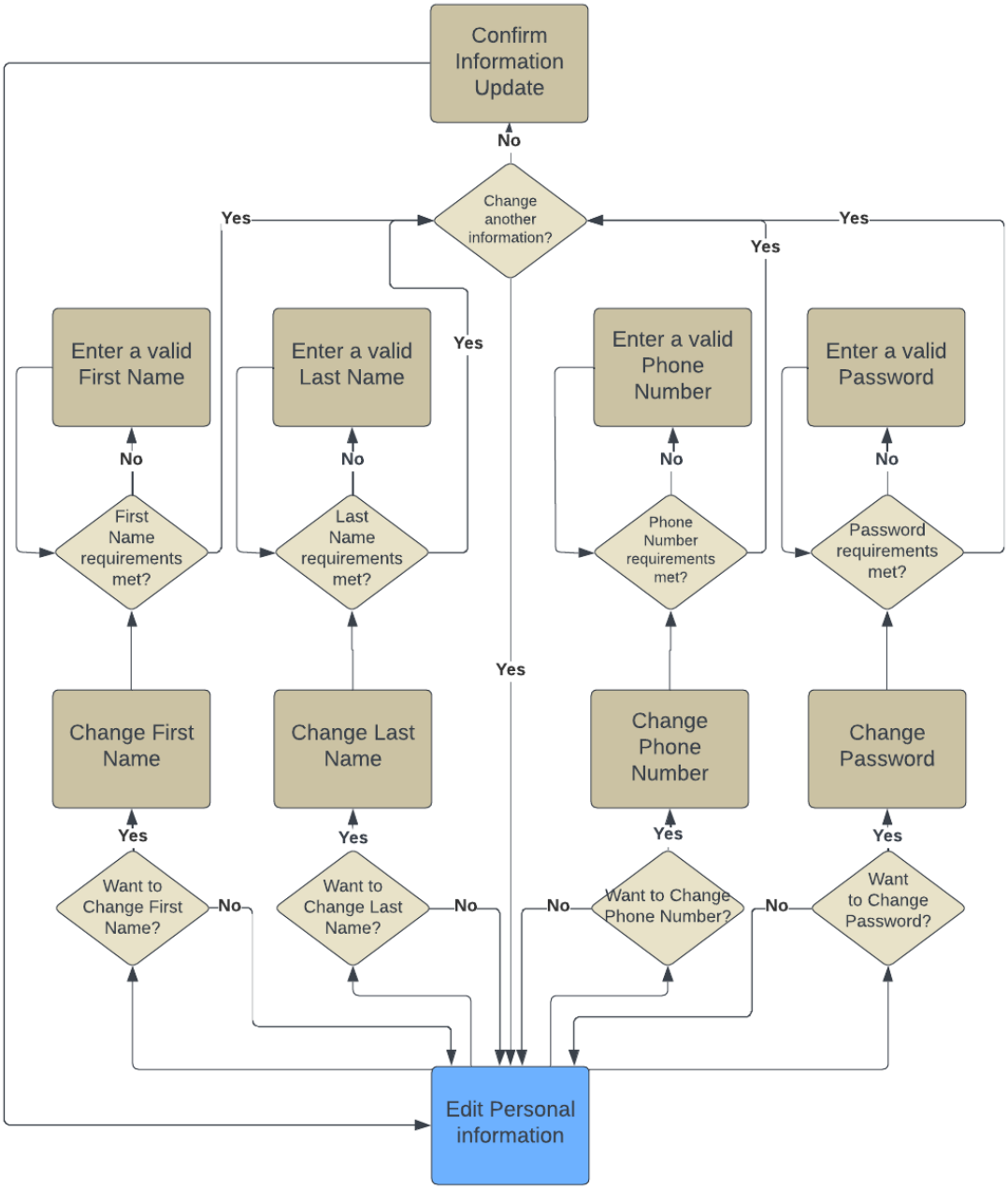

Figure 15: *Edit Personal information page Flowchart Diagram*

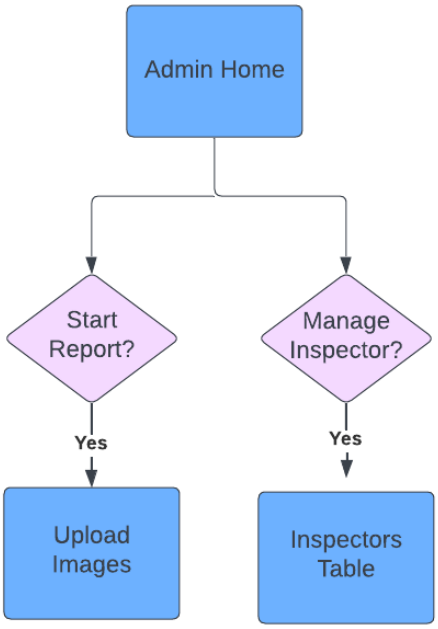

Figure 16: *Administrator Home page Flowchart Diagram*

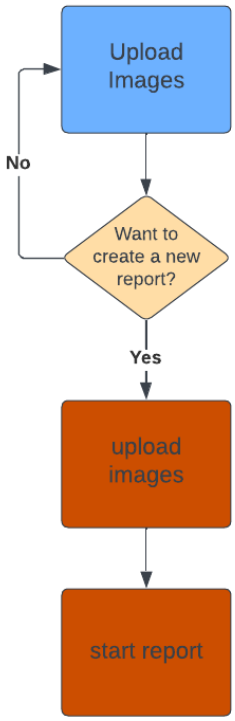

Figure 17: *Administrator Upload images page Flowchart Diagram*

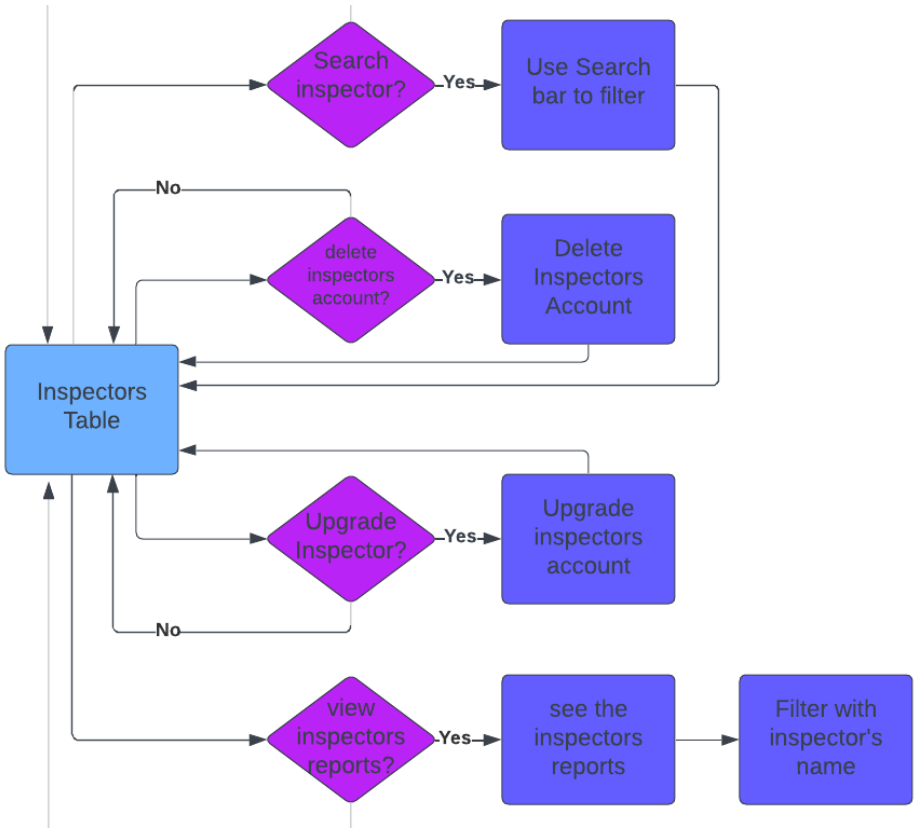

Figure 18: *Inspectors Table page Flowchart Diagram*

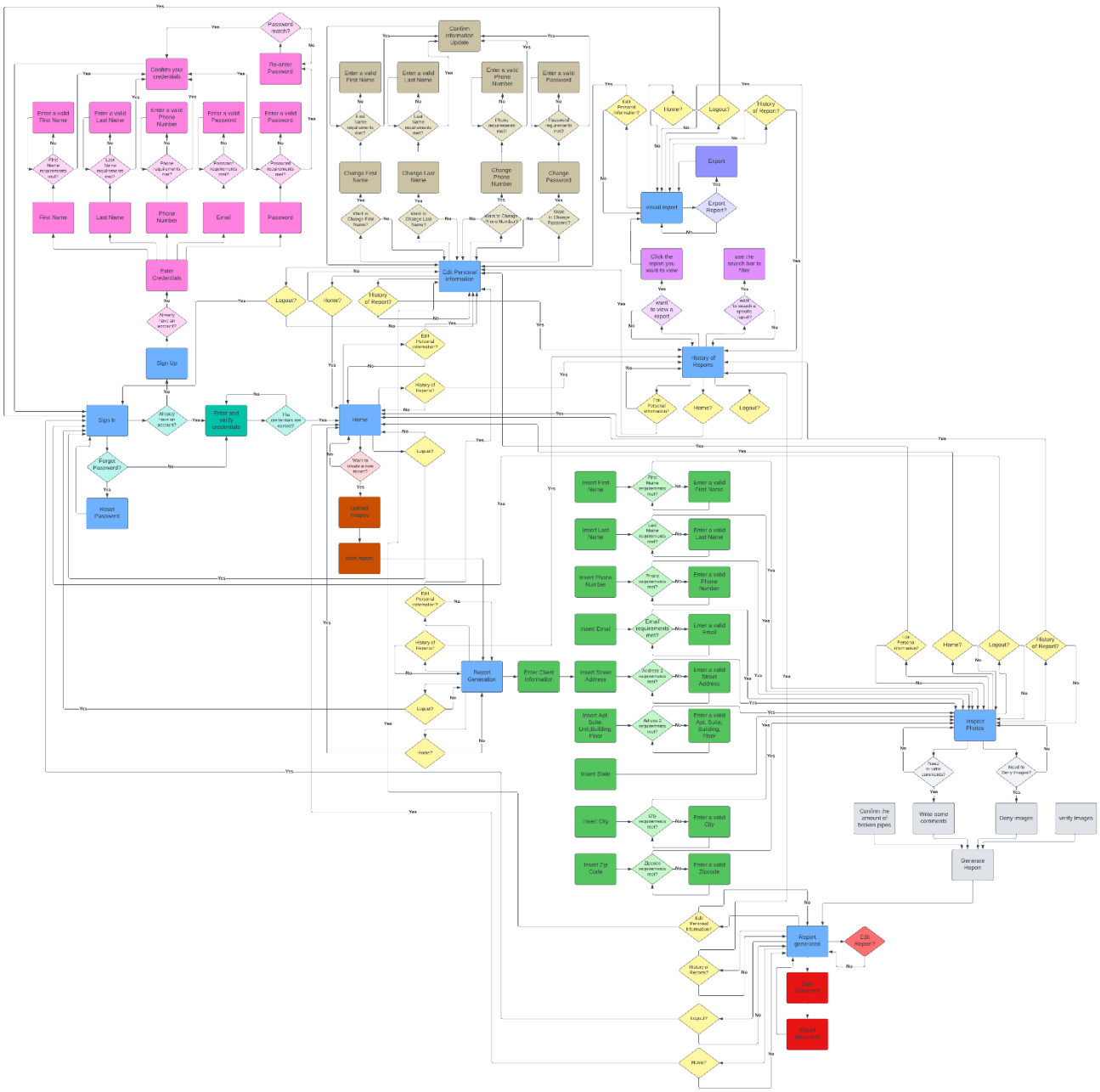

Figure 19: *User Flowchart Diagram*

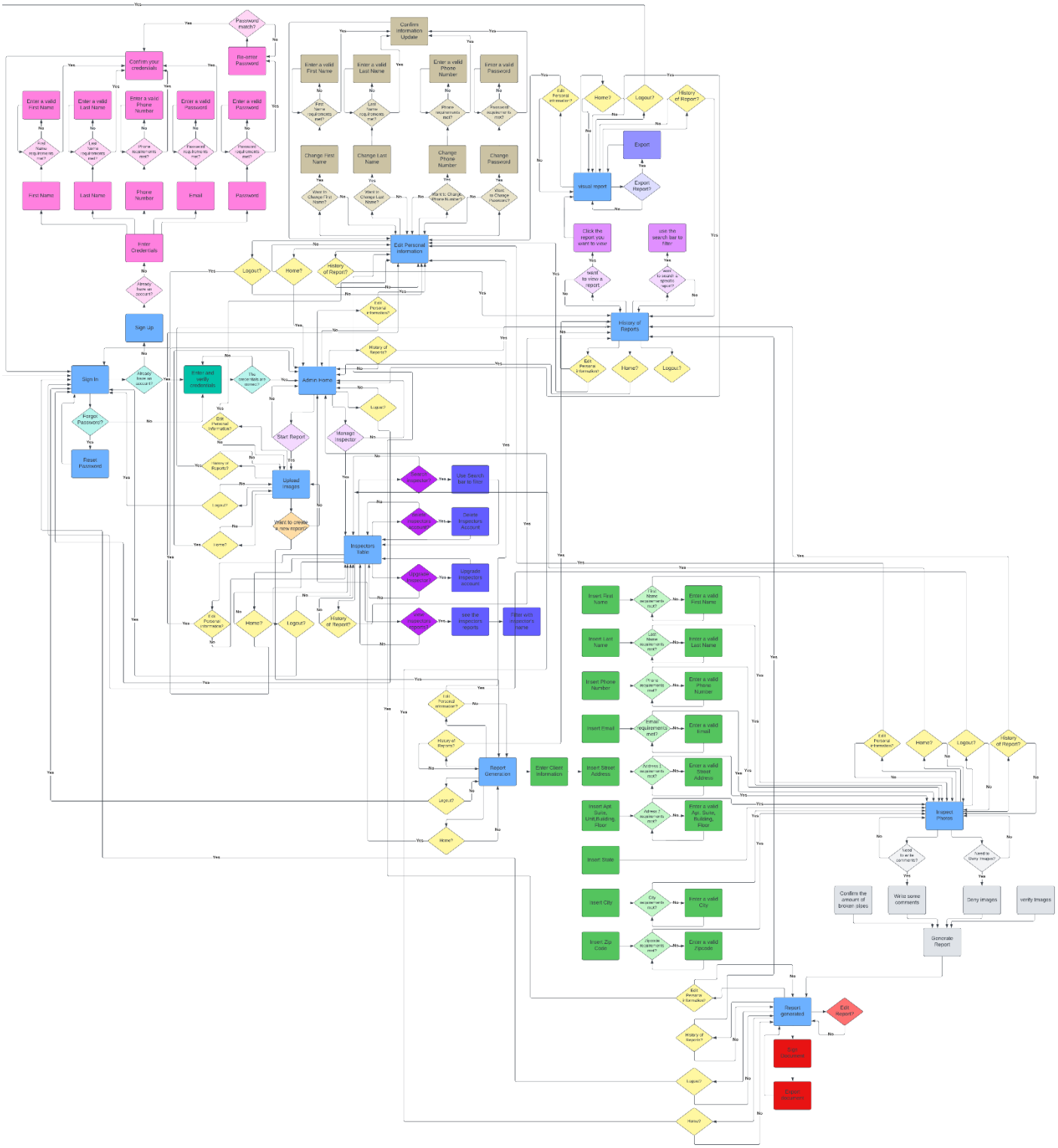

Figure 20: *Admin Flowchart Diagram*

iii. Sequence Diagrams (Giomar Santiago, Juan D. Pérez)

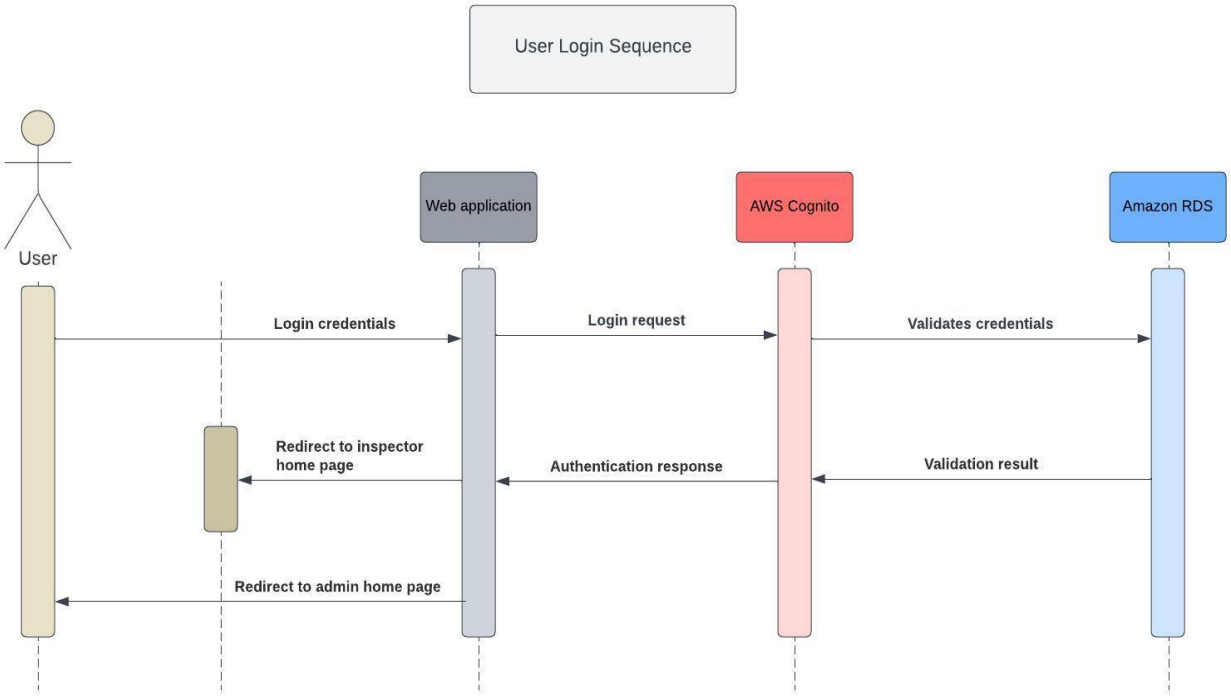

Figure 21: *User Login Sequence Diagram*

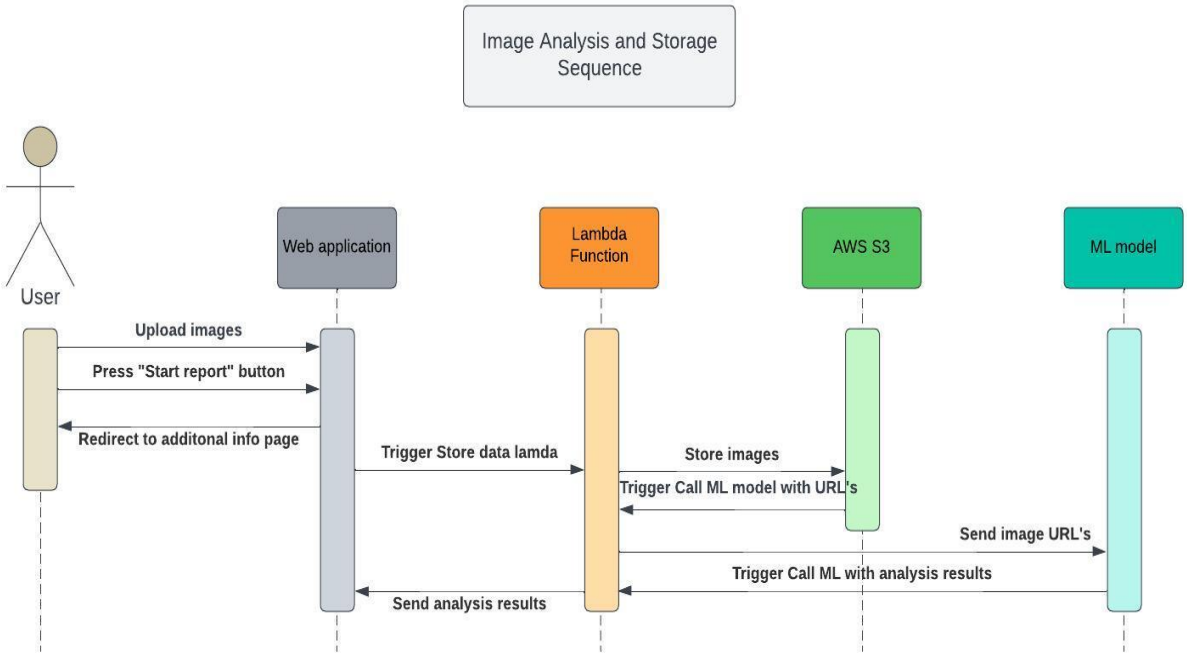

Figure 22: *Image Analysis and Storage Sequence Diagram.*

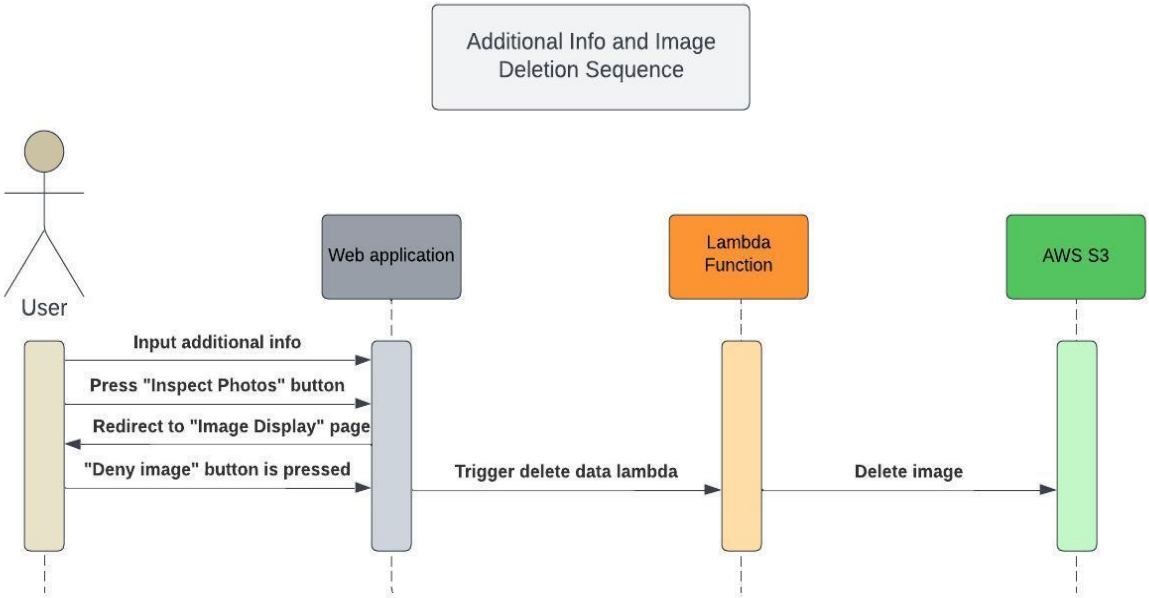

Figure 23: *Additional Info and Image Deletion Sequence Diagram.*

Figure 23 is the continuation of the sequence shown in **Figure 22**. It covers the case in which a user, after entering the additional information required for a report and is redirected to the "Inspect photos" page, decides to deny an image from the report.

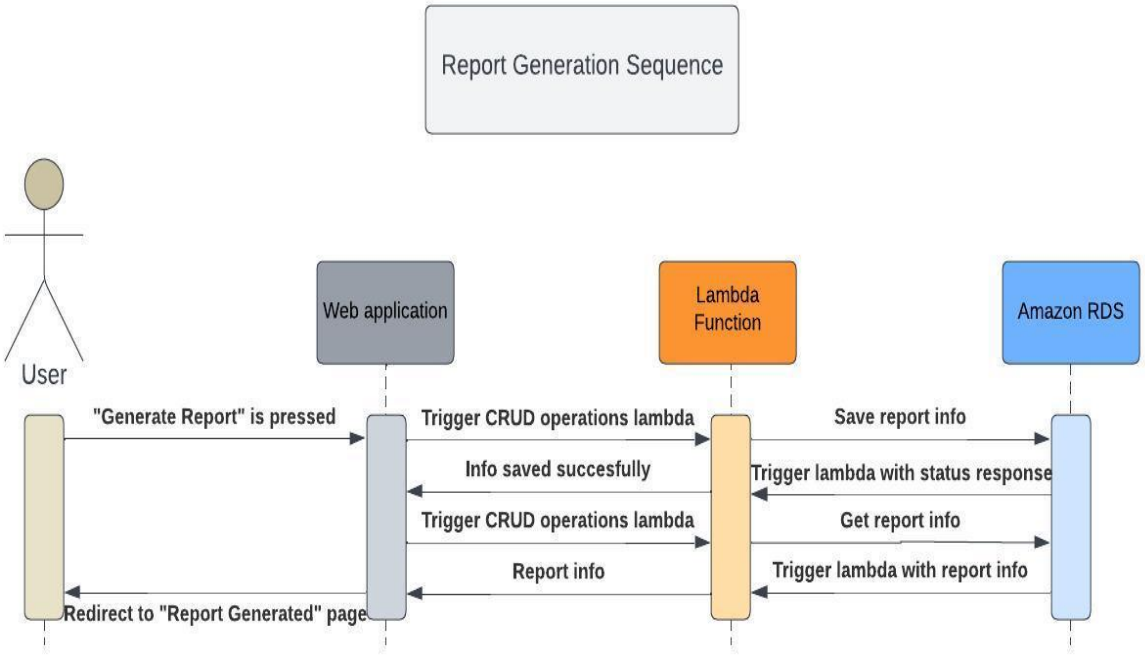

Figure 24: *Report Generation Sequence Diagram.*

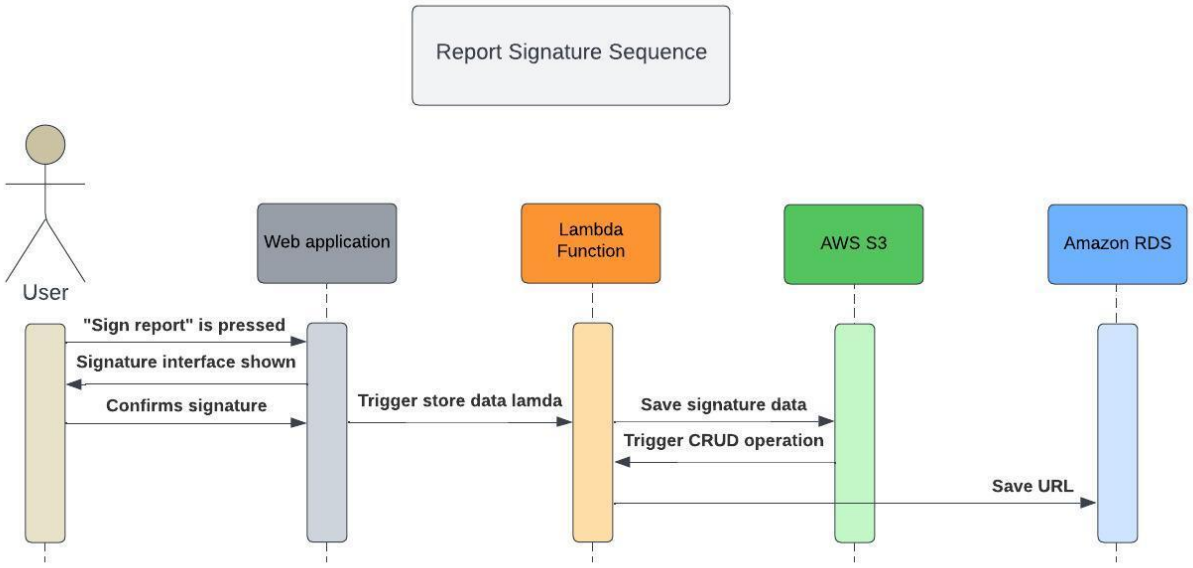

***Figure 25**: Report Signature Sequence Diagram*

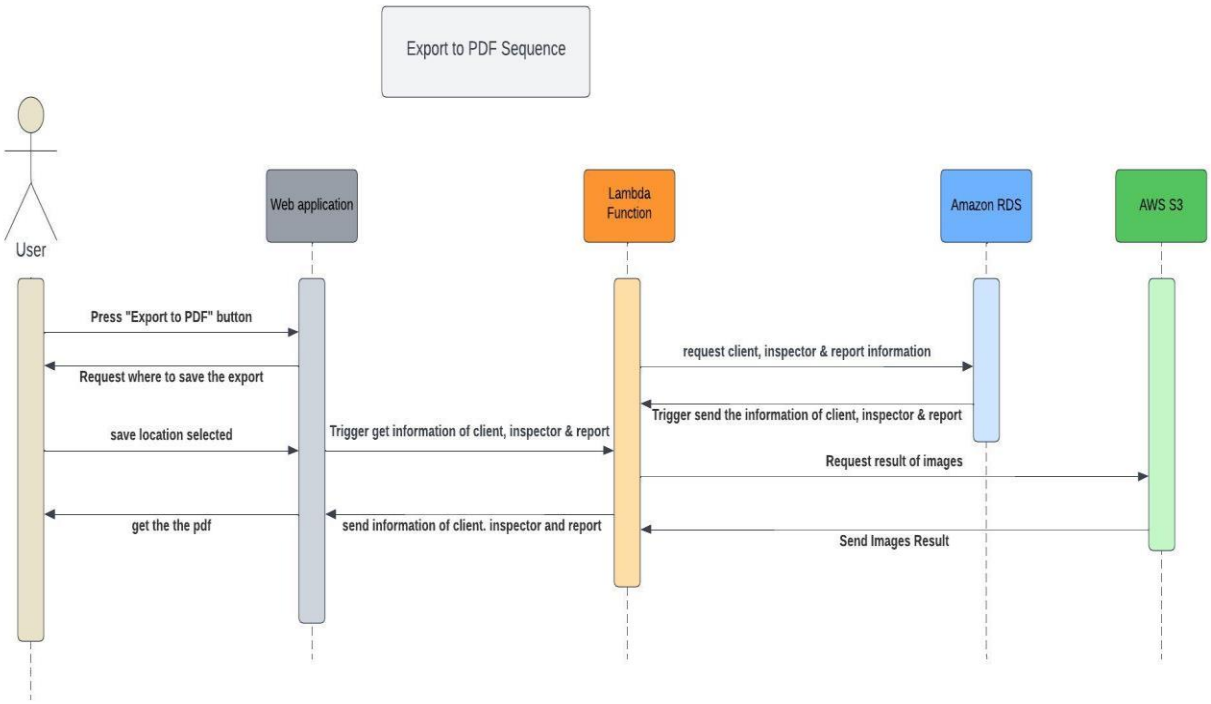

***Figure 26:** Export Report Sequence Diagram*

Figure 25 and **Figure 26** show the sequences for the functions a user would have in the “Report Generated” page. The user has the option to sign the report electronically and export the report to PDF format.

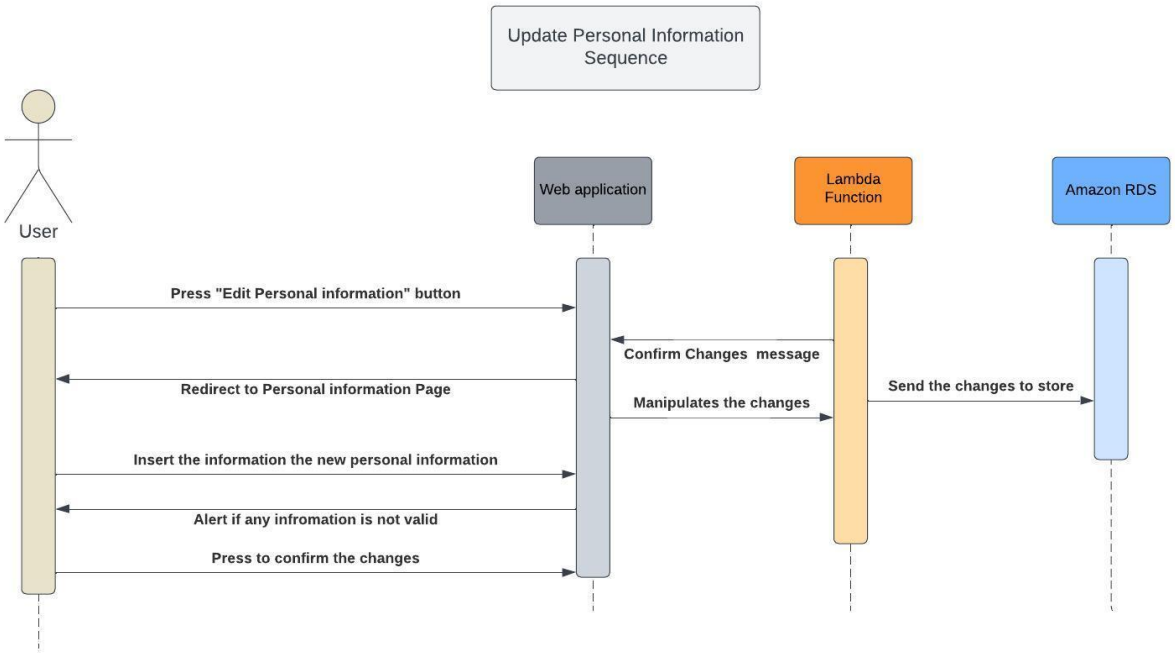

Figure 27: *Update Personal Information Sequence Diagram*

iv. Use Case Diagram (Giomar Santiago)

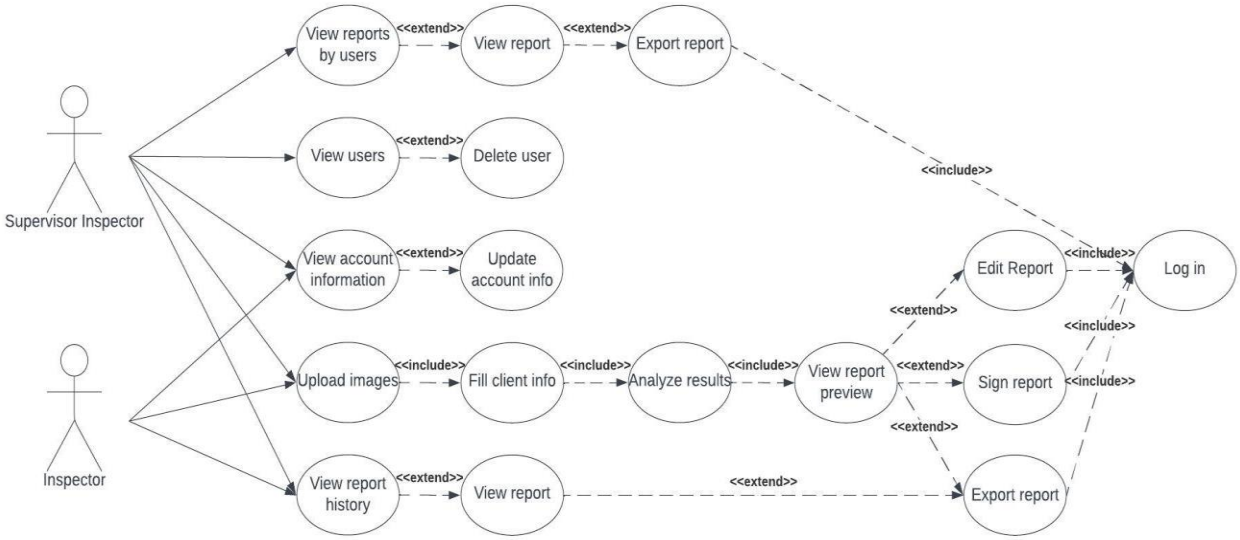

Figure 28: *Use Case Diagram for Supervisor Inspector and Inspector*

v. ER Diagram (Jeremy J. Márquez, Giomar Santiago)

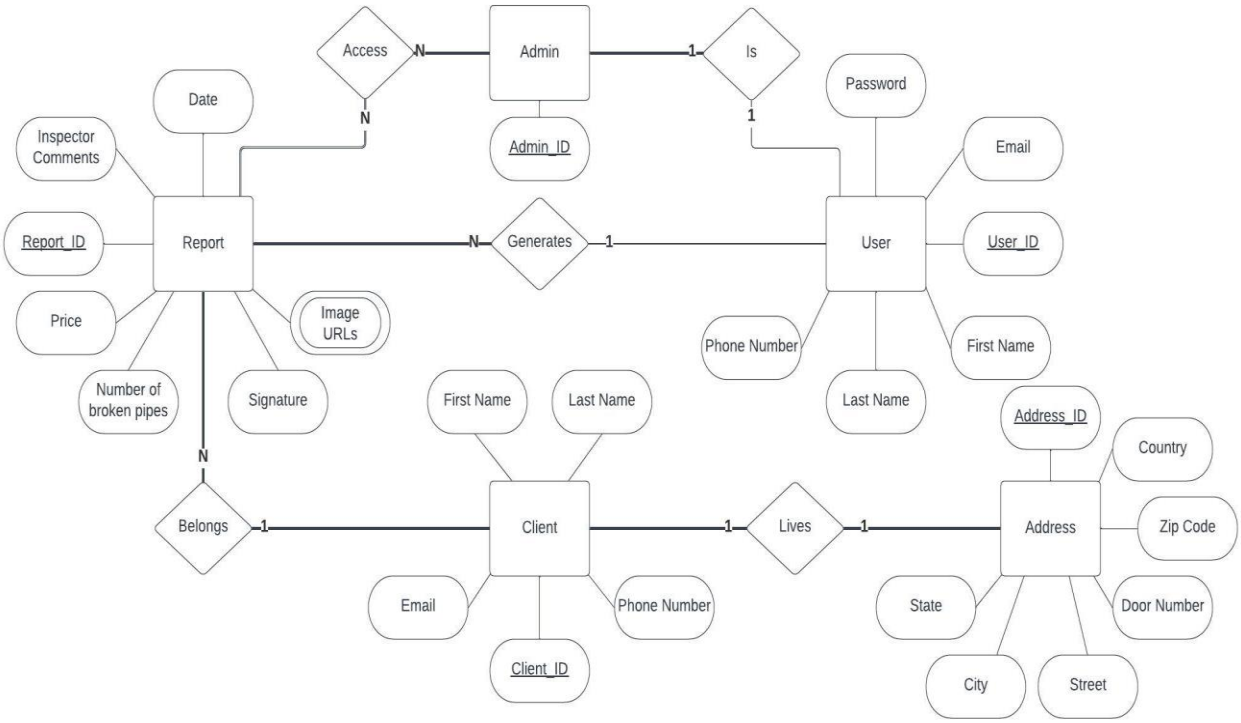

Figure 29: *Entity Relationship Diagram for the database.*

vi. Table Diagram (Giomar Santiago)

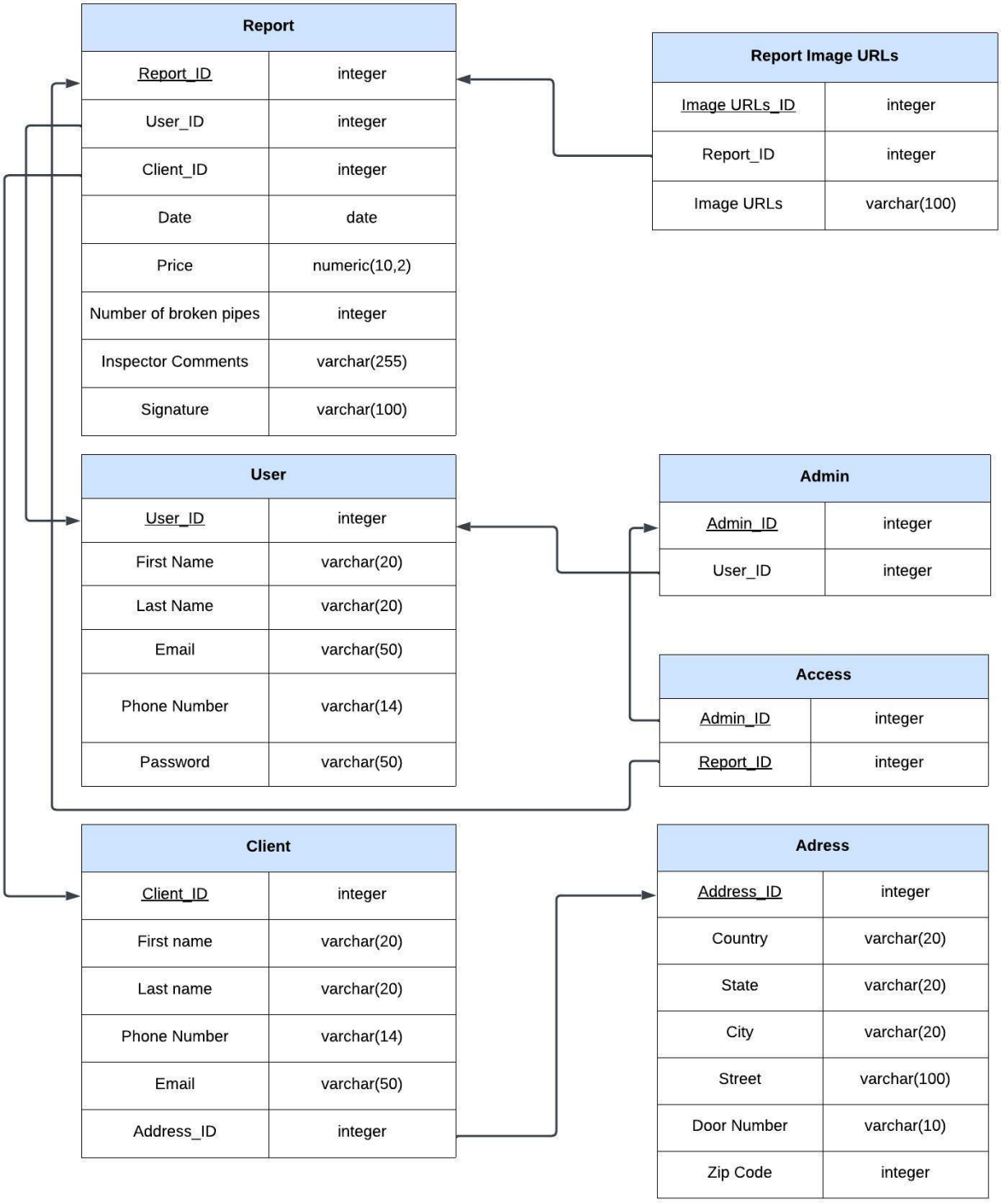

Figure 30: *Table Diagram*

vii. UI Prototype (Kevin O. Valentín)

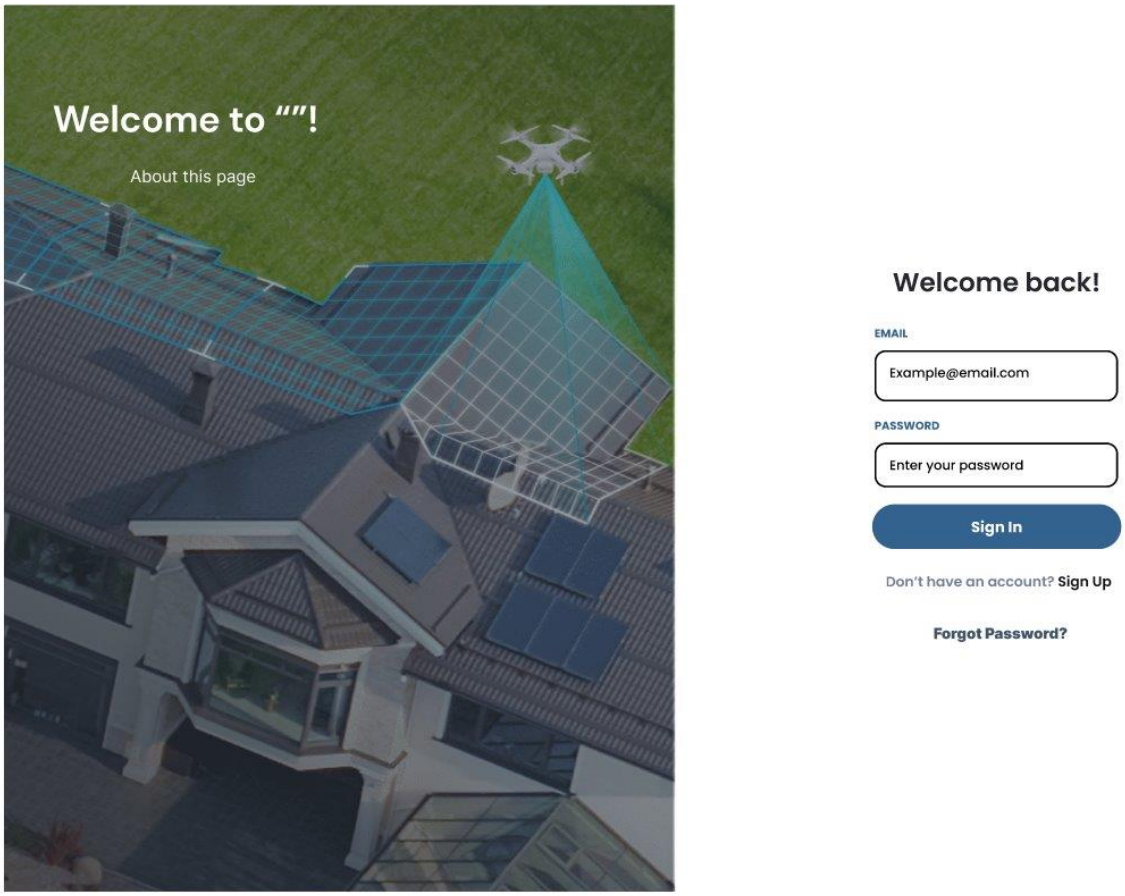

Figure 31: *Login Page*

The login page provides users with a secure gateway to access their accounts. It serves as a crucial entry point for users to gain personalized access to functionalities within the website. requiring them to input their email and password.

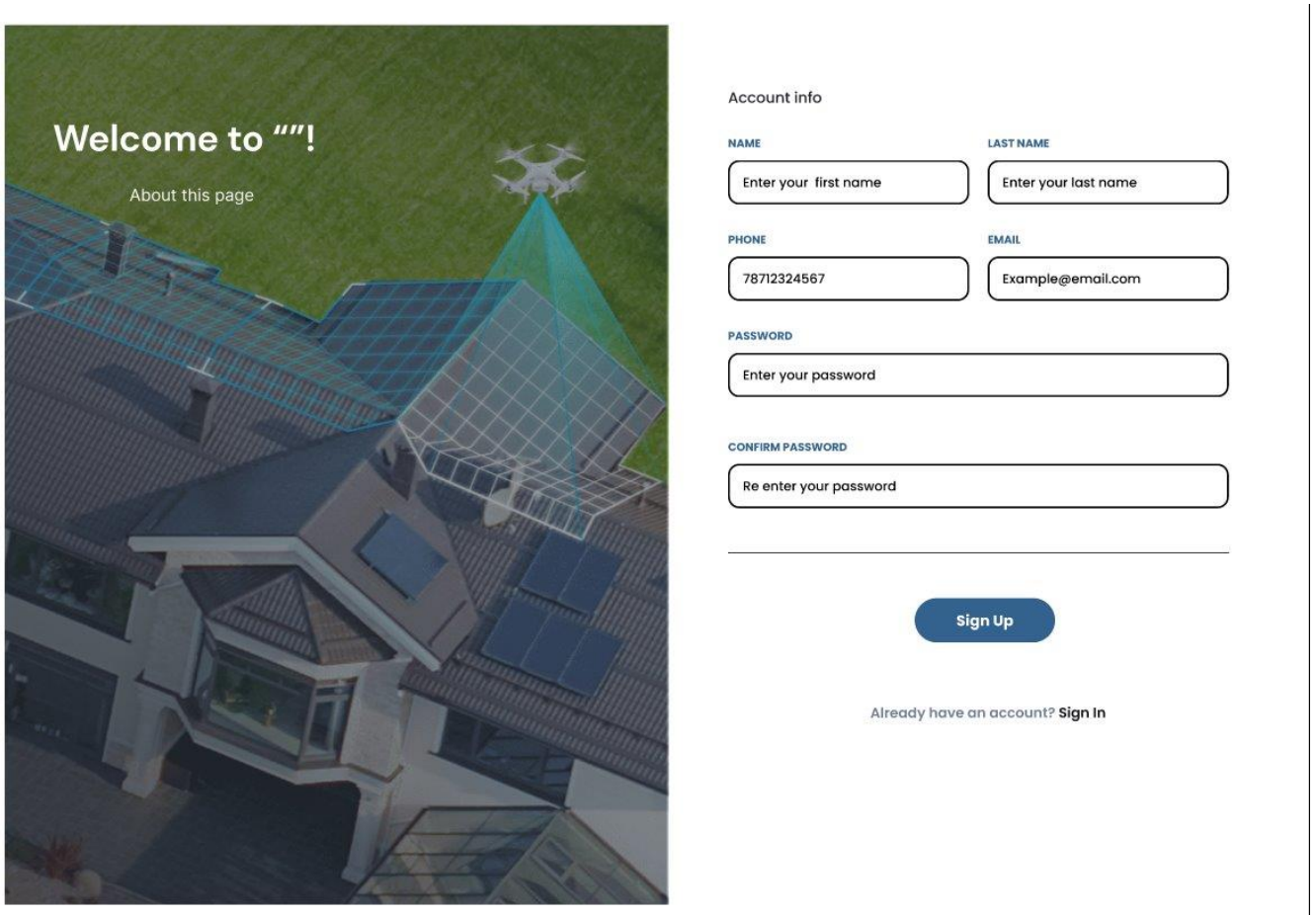

Figure 32: *Create Account Page*

The account creation page offers new users the opportunity to register for an account by providing necessary information such as email address, phone number and password.

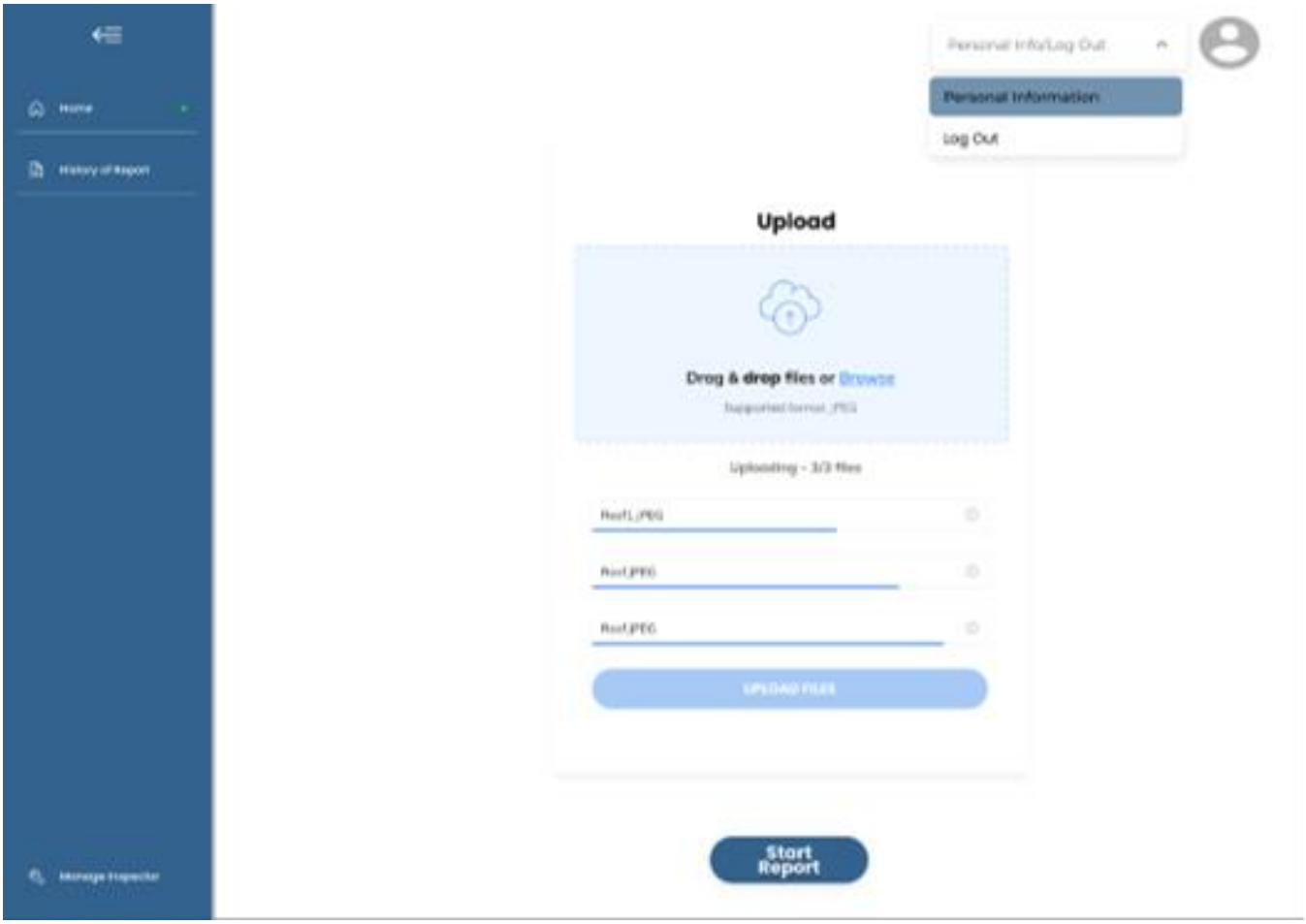

Figure 33: *Home Page*

The sidebar is implemented in all pages serves as a navigational hub with options to access different sections of the webpage. Users can expand it to reveal options such as the Home Page for general navigation, History of Reports for reviewing past activities, and Manage Inspector Page for overseeing inspector-related reports. This streamlined design enhances user experience by providing quick and intuitive access to essential features while maintaining a tidy interface.

The dropdown menu is implemented in all pages features two tabs for user convenience: one tab grants access to personal information, allowing users to view and manage their account details, while the other tab provides a quick option to log out, facilitating secure session closure with just a click. This streamlined design optimizes user interaction by offering essential functionalities within a compact and intuitive dropdown menu.

The home page features an interface with a prominent dropdown box designed for uploading images, simplifying the process of adding visual content to the platform. This intuitive design enhances user experience by providing a straightforward means to upload images.

Figure 34: *Admin Home Page*

The admin home page offers two distinct options: "Manage Inspector" for overseeing and administering inspector-related reports, such as reviewing reports and the option to delete inspector account, and "Start Report" to initiate the process of creating new reports. This interface empowers administrators with efficient access to essential functionalities, streamlining administrative tasks and facilitating seamless workflow management.

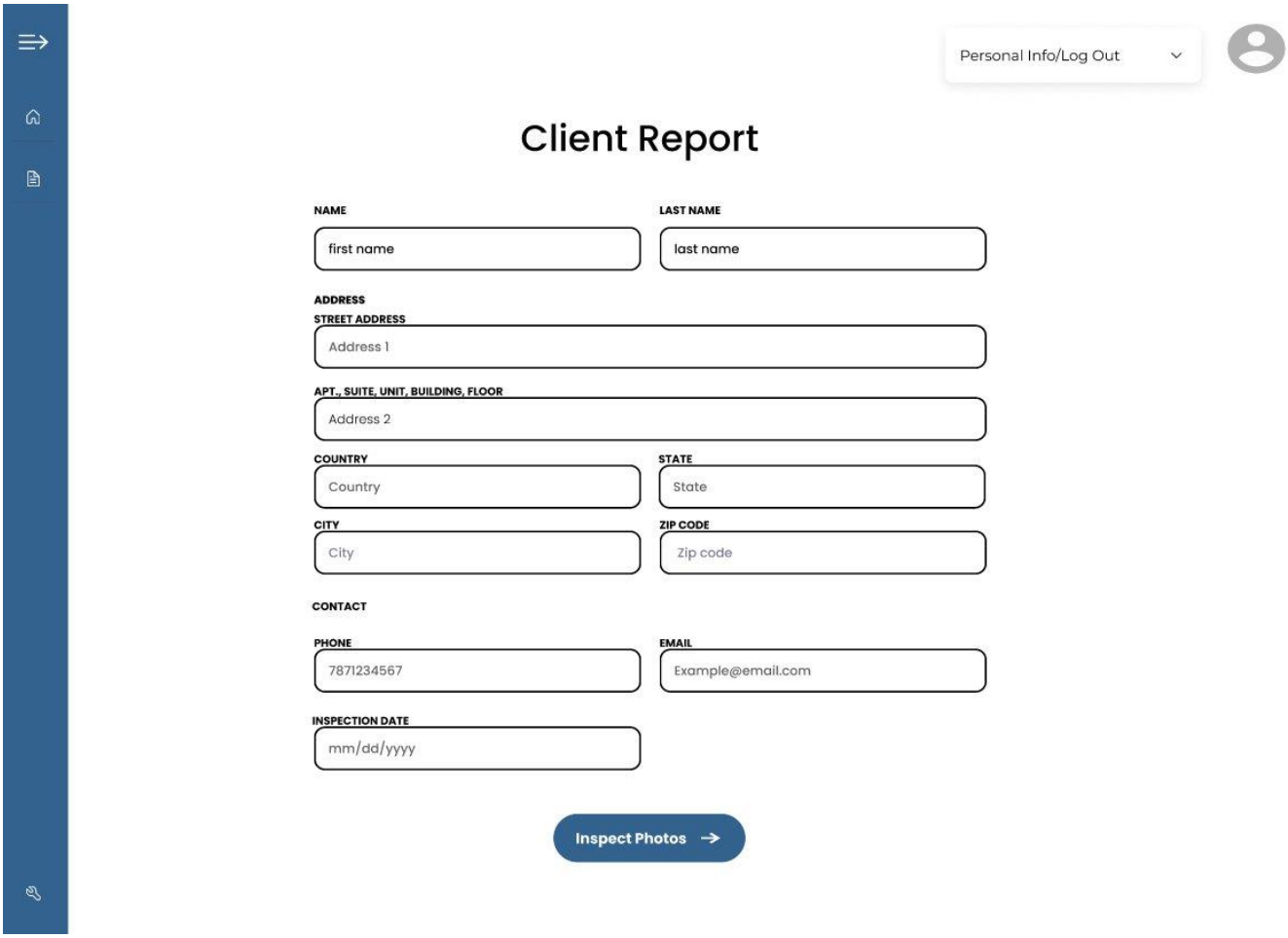

Figure 35: *Report Generation Page*

The report generation page features an intuitive interface with a dedicated section for input of text containing client information such as full name, full address, phone number, inspection date, and email address. Users can conveniently input any missing or additional information directly into corresponding text input fields provided on the page. The design ensures seamless data integration and validation, facilitating efficient generation of comprehensive client reports.

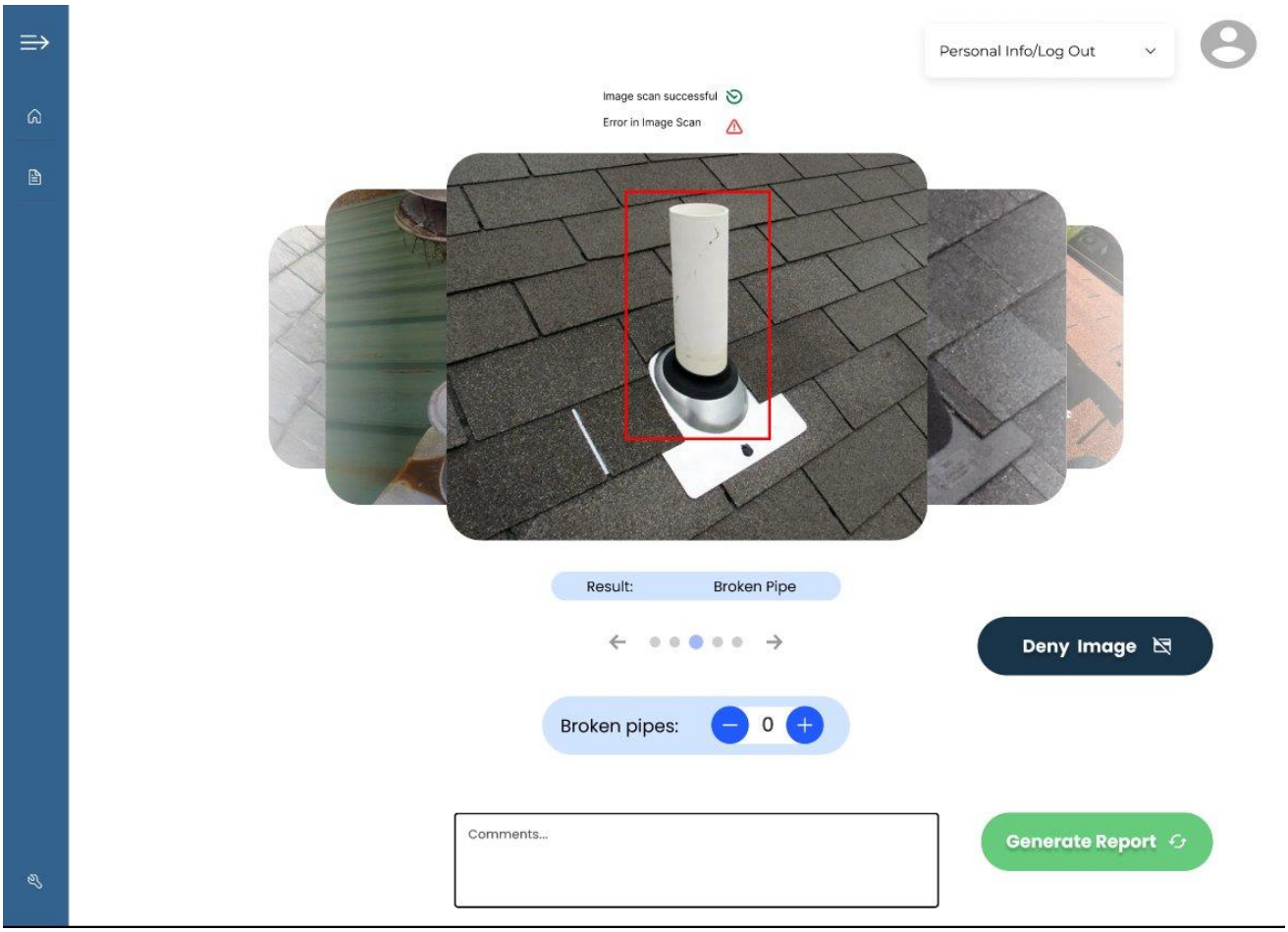

Figure 36: *Image Display Page*

The Display Page for previewing photos analyzed by the ML model provides users with a visually engaging interface. Upon accessing the page, users are presented with a slideshow-style layout showcasing the analyzed photos along with their corresponding results, such as classifications or detected objects. Each photo is accompanied by options to take specific actions: Deny Image, Add Broken Pipe, Retract, and Comment.

The Deny Image option allows users to flag photos for exclusion from further processing or reporting. Add Broken Pipe enables users to mark any identified pipe issues within the image. The Retract feature provides a mechanism to remove erroneous results or actions. Additionally, users can provide contextual feedback and annotations via the Comment option, facilitating collaboration and refining the ML model's accuracy. This interactive and comprehensive interface empowers users to efficiently review and contribute to the analysis process.

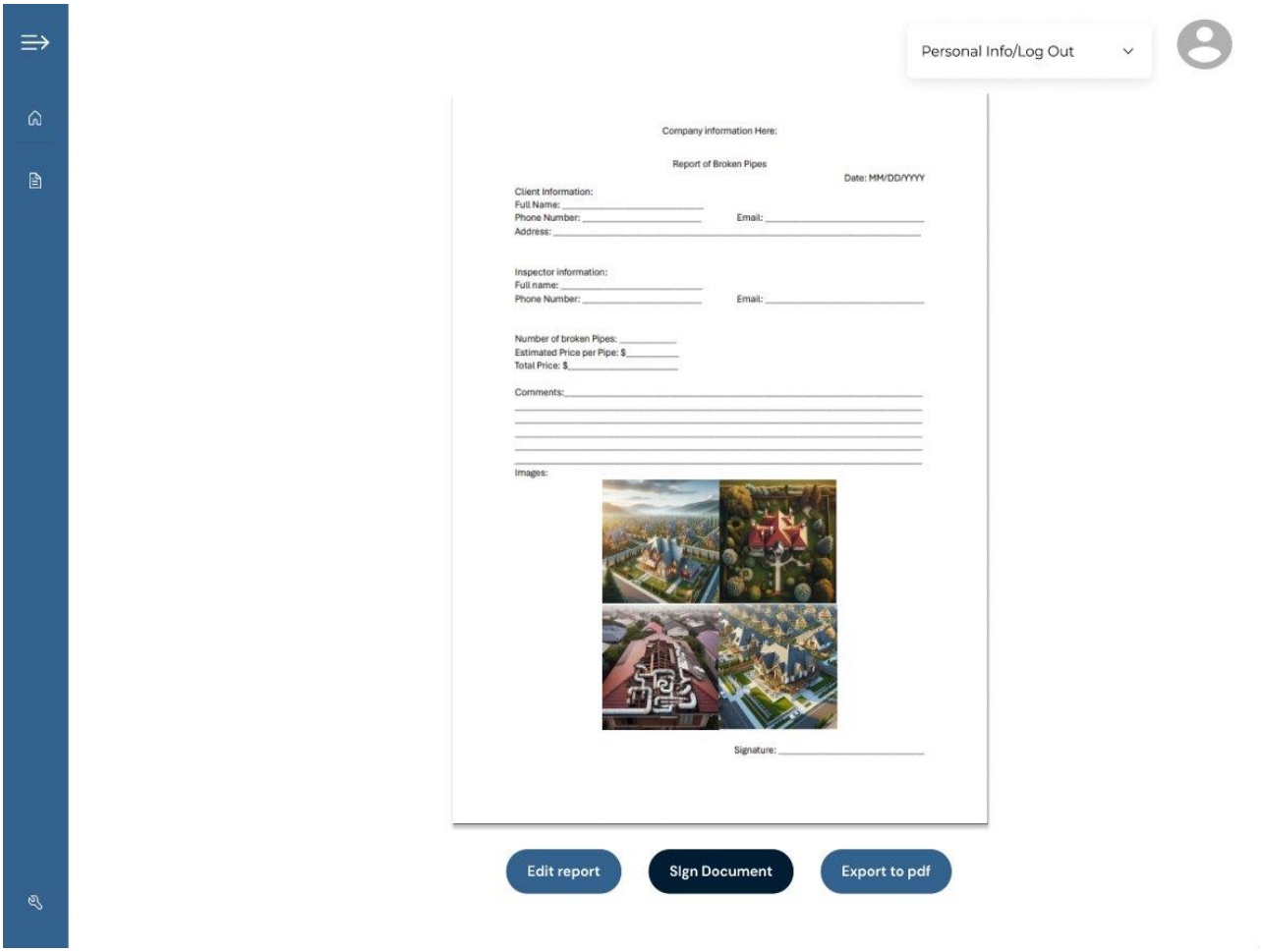

Figure 37: *Report Generated Page*

The Report Generated Page offers users a convenient page to preview and edit reports with ease. Upon accessing the page, users can review the report's content and structure before making any necessary modifications. Once satisfied with the edits, users can electronically sign the document for authentication directly within the webpage. After signing, users can effortlessly export the finalized report to PDF format, ensuring its accessibility and preservation for future reference.

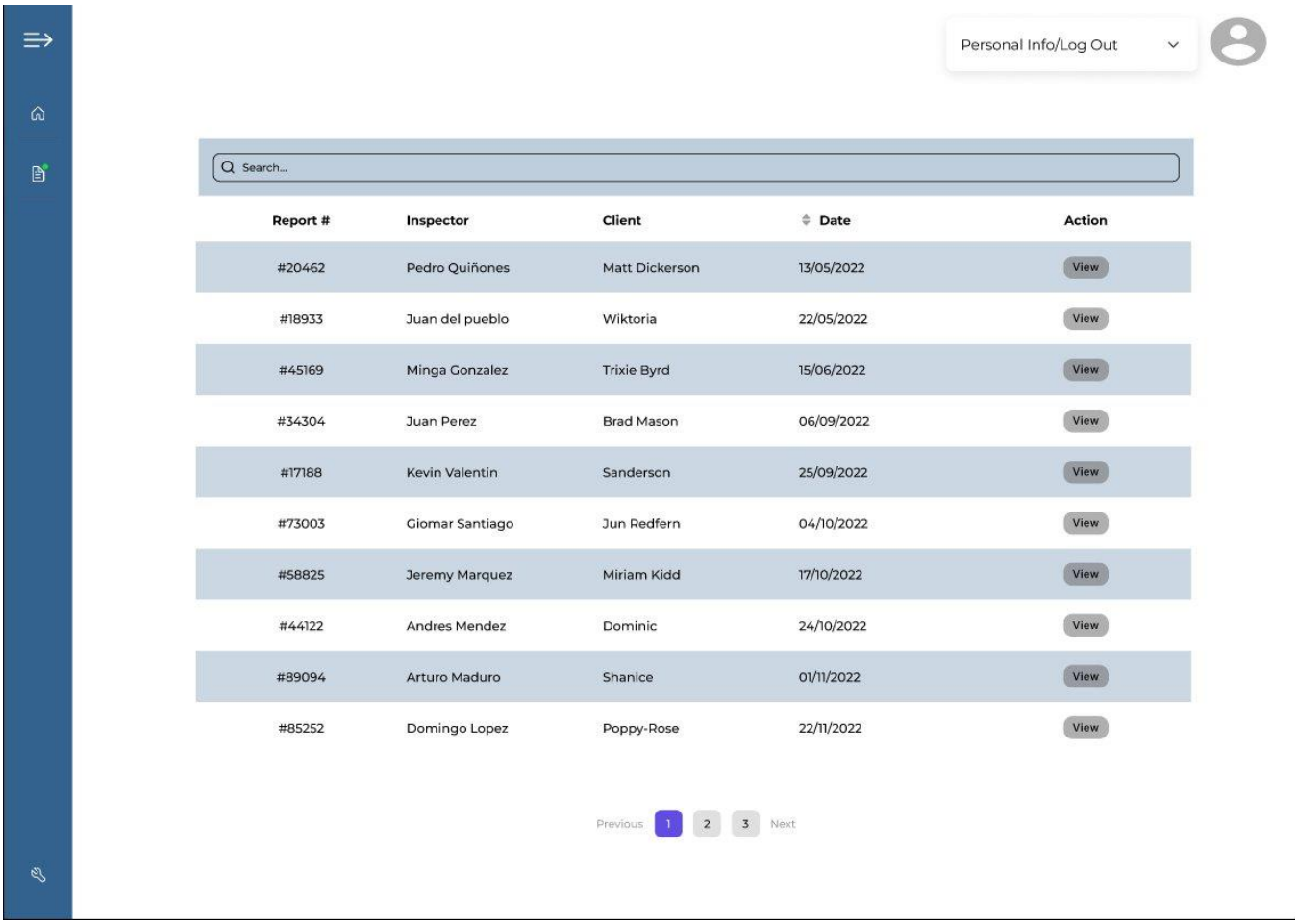

Figure 38: *History of Report Page*

The History of Report Page serves as a centralized repository where users can access past reports generated by that user, organized by date and client. If the user is an admin, the user can see all reports generated by all inspectors. Upon accessing the page, users are presented with a user-friendly interface displaying a chronological list of reports, along with corresponding inspection dates and client names. A search bar feature enables users to efficiently locate specific reports by entering the client's name, streamlining the retrieval process. Also, users can view each report in detail, allowing for comprehensive review and analysis of inspection and view of the past generated report.

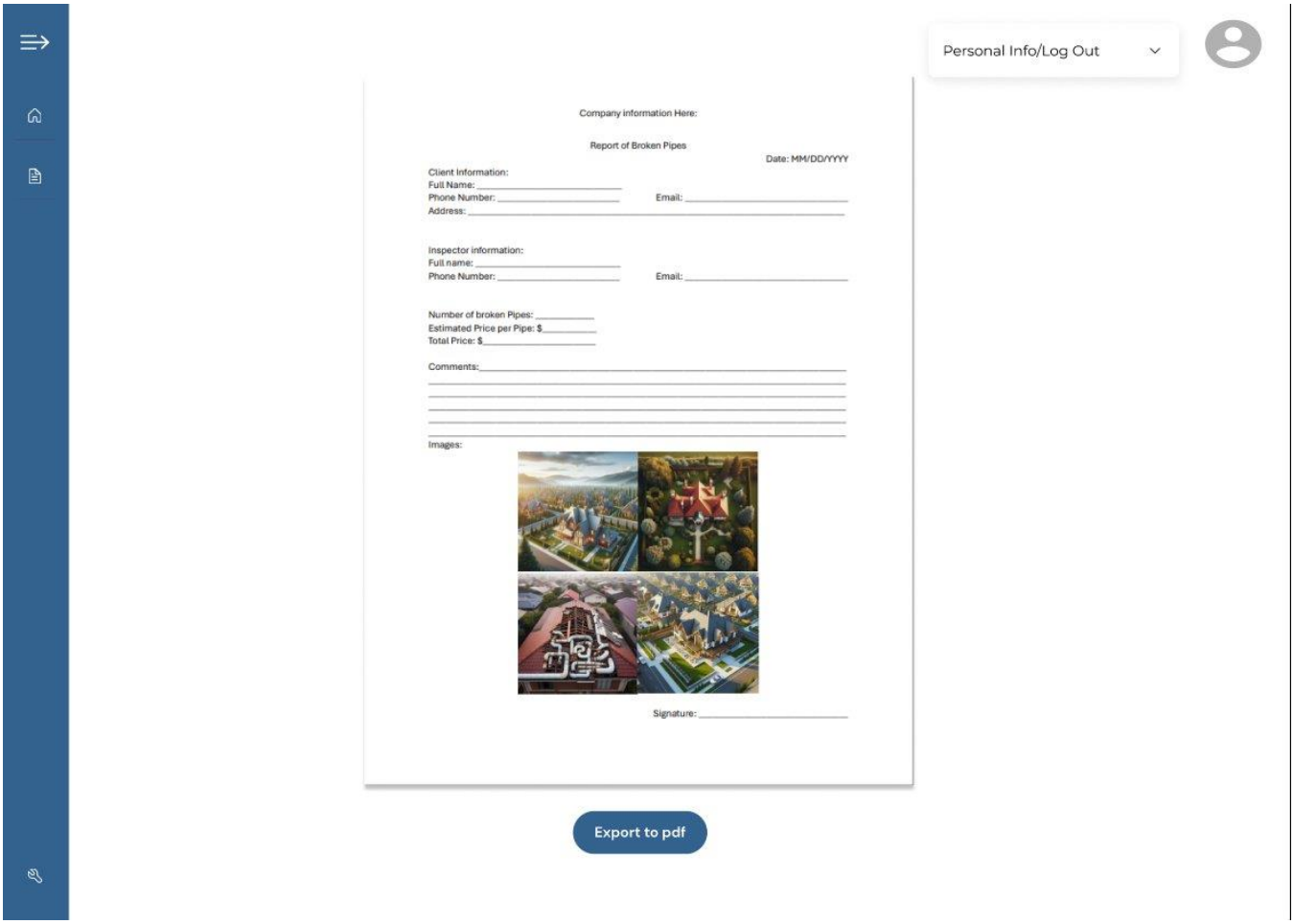

Figure 39. *View Report Page*

The View Report Page is a simple designed page for easy access to inspection reports. Users are provided with a clean interface where they can seamlessly view reports generated by inspectors. The page offers a convenient "Export to PDF" option, allowing users to quickly download the report for offline viewing or sharing. This streamlined functionality ensures that users can effortlessly access and save reports in a universally compatible format.

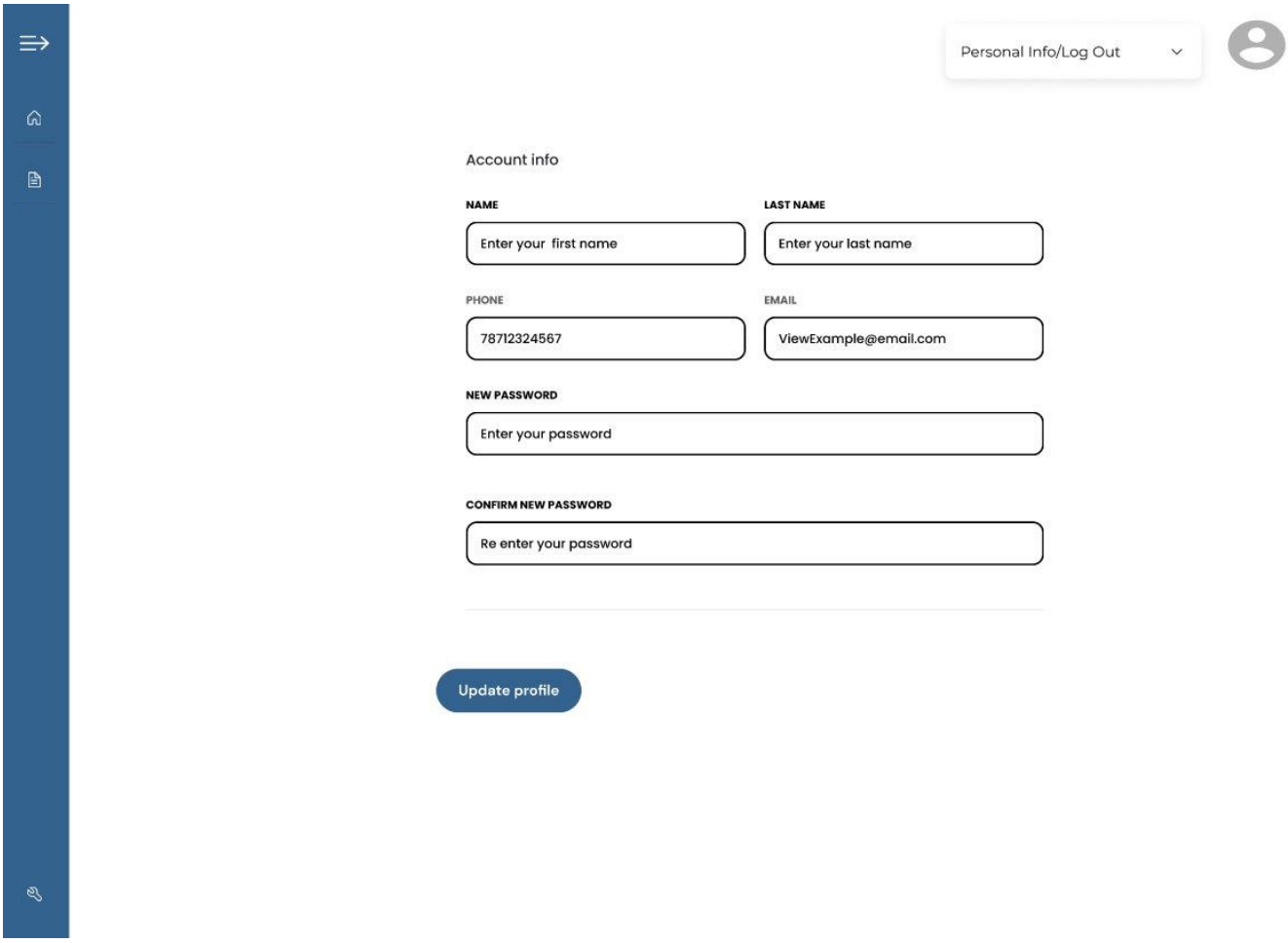

Figure 40. *Personal info Page*

The Personal Info Page serves as a secure portal for users to manage their account details conveniently. Upon accessing the page, users are presented with their current personal information, including their full name, phone number, and email address. Users have the option to update their full name and phone number directly on the page, ensuring their information is always accurate and up to date. Additionally, users can view their email address for reference purposes. For enhanced security, users can also change their password, providing control over their account credentials. This comprehensive functionality empowers users to maintain their personal information and account security effortlessly.

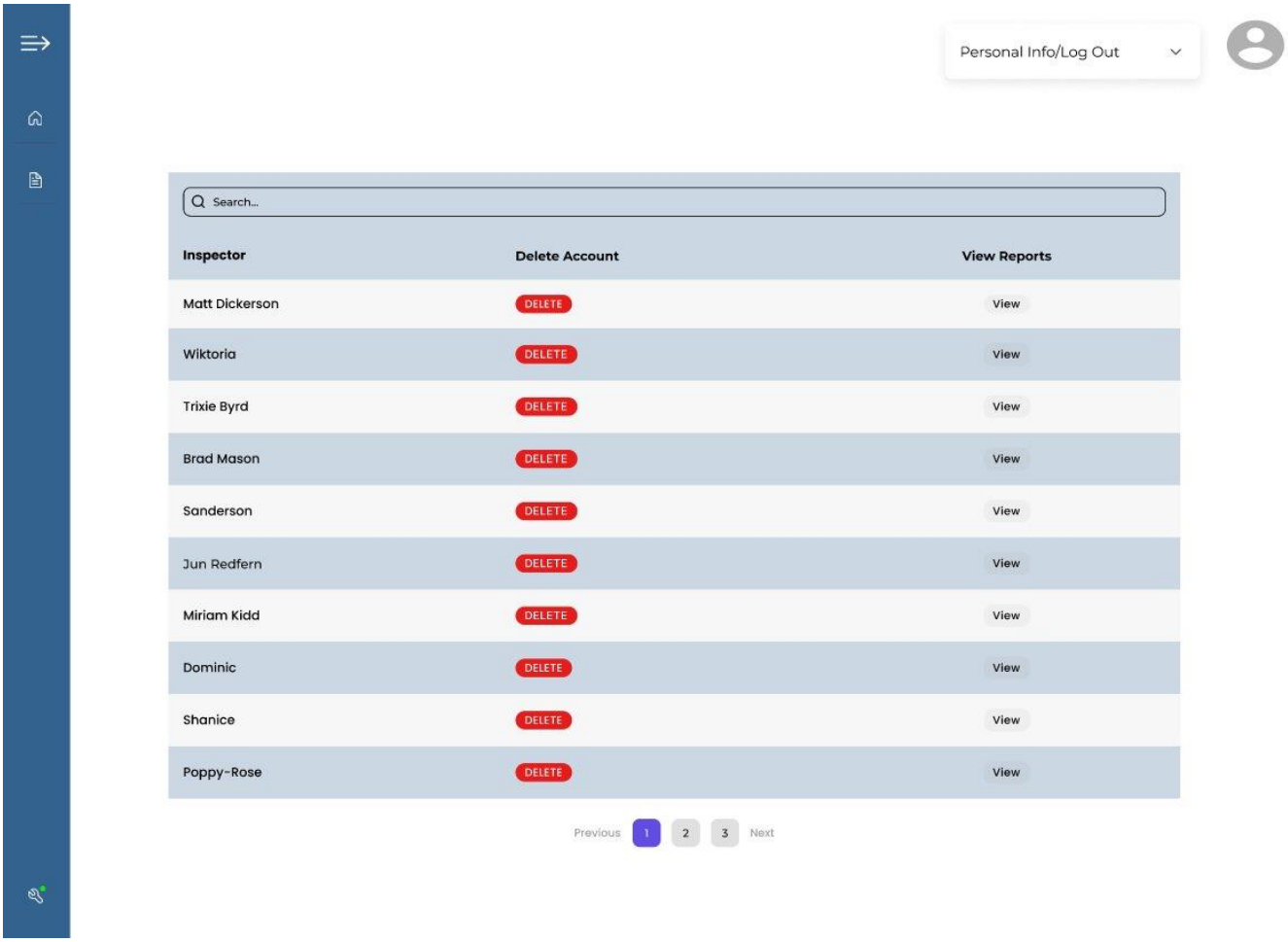

Figure 41: *Manage Inspector Page*

The Manage Inspector Page is a centralized hub for overseeing inspector accounts and their associated activities. It features a comprehensive table listing all inspectors. Within this table, administrators can easily identify each inspector's profile and their respective generated reports. Admins are empowered with the ability to delete inspector accounts if necessary, ensuring efficient management of personnel. Additionally, the page provides seamless access to view reports specifically generated by each inspector, facilitating oversight and quality assurance. This streamlined interface enhances administrative control and oversight over inspector-related operations.

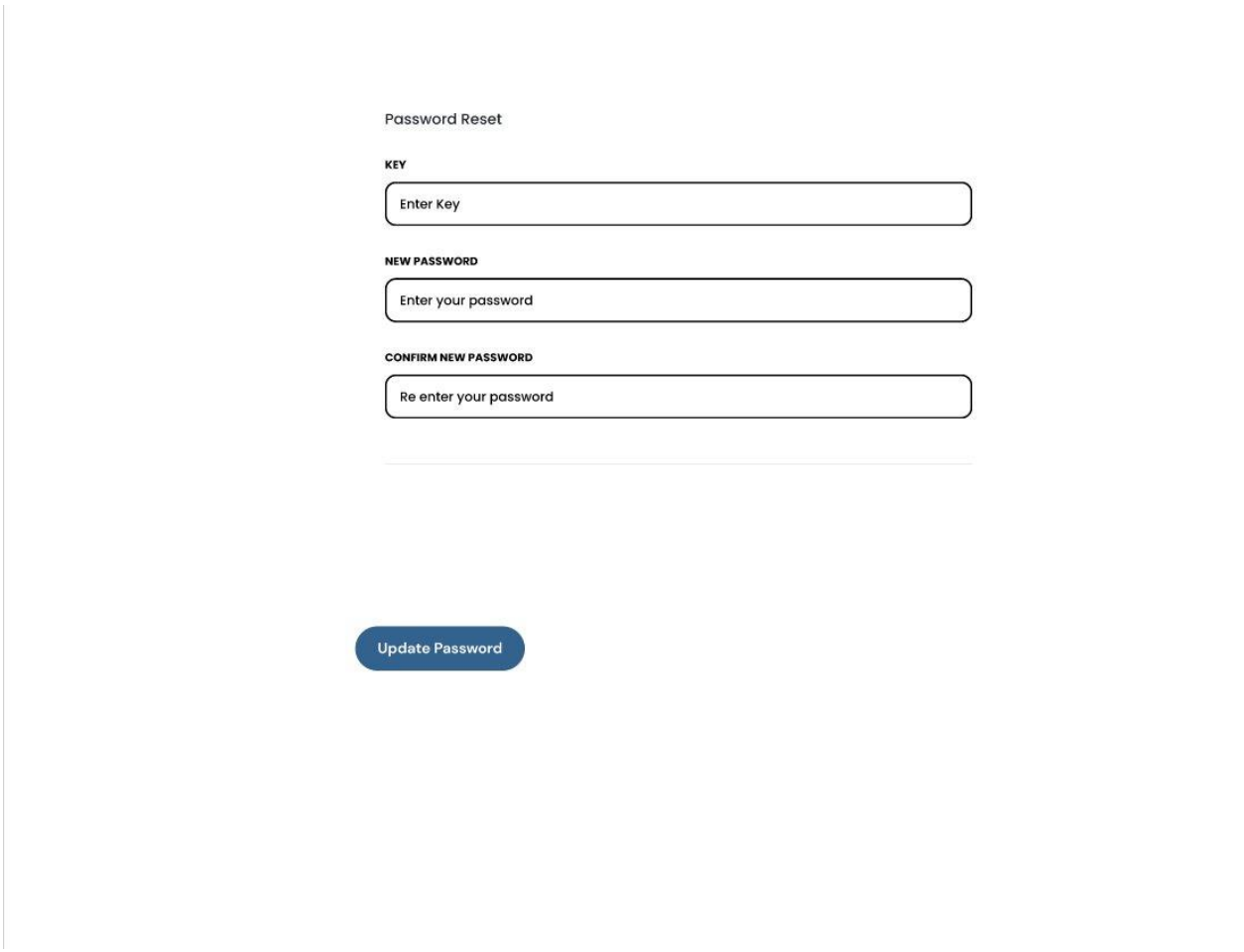

Figure 42. *Password Reset Page*

The Password Reset Page offers users a secure and straightforward process for resetting their account passwords. Users first input a unique key provided to them, typically sent via email. Once the key is verified, users are prompted to enter a new password and confirm it to ensure accuracy. This two-step authentication process safeguards against unauthorized access and ensures the security of the account. By providing a streamlined interface for password reset, users can quickly regain access to their accounts while maintaining the integrity of their personal information.

G. Testing Plan (Kevin O. Valentín)

Test Component	Test Type	Test Procedure	Test Passing Result	Test Fail Result	Contingency Strategy
Upload Functionality	Functional Test	Attempt to upload various photos.	Photos are uploaded without errors.	Upload fails, or errors occur during the process.	Debug and fix any issues with the upload functionality. Retest until successful.
Machine Learning Model Precision	Precision Test	Provide the ML model with a set of test images containing both broken and intact pipes.	ML model achieves at least 80% precision in identifying broken pipes.	Precision of the model is below 80%.	Review and optimize the ML model, adjust parameters, or provide additional training data. Retest until required precision is achieved.
Report Generation	Functional Test	Upload a set of photos containing broken pipes and verify that the system generates a report with client information and identified broken pipes.	A report is generated accurately, containing all elements necessary for the report.	Report is missing information or does not show the correct information.	Debug and fix any issues with report generation. Ensure all necessary information is included accurately. Retest until successful.
User Interface	User Interface Test	Interact with the web application's interface to upload photos and view reports.	The UI is fully functional, and the user can interact	The UI is not fully functional, and the user cannot interact with	Improve the UI's functionality, debug any

			with the whole system.	every part of the system.	issues and retest.
Database Integrity	Database Test	Verify that the database stores uploaded information.	Uploaded information is stored correctly in the database.	Data integrity issues are detected, such as missing or corrupted records.	Investigate and resolve database integrity issues. Ensure proper data validation and error handling. Retest to confirm data integrity.
User Authenticati on	Security Test	Test the user authentication mechanism to ensure secure login and access control.	Users can successfully log in with valid credentials and access authorized features.	Authentication fails, unauthorized access is granted, or security vulnerabilities are detected.	Review and strengthen authentication mechanisms, fix vulnerabilities, and ensure secure access control. Retest to verify improvements.
ML Model Response	Functional	Test the machine learning model response to the web application.	Predictions and image URLs are obtained in the standard JSON format on the front end	JSON does not contain all expect information (Image URLs and predictions)	Verify response generation and packing in the ML model code
Email Notification	Functional Test	Test the functionality of email notifications,	Email notifications are sent correctly for	Email notifications are not sent, or errors occur	Debug and fix issues with email notification

		including sending notifications and handling errors.	various events, and errors are handled gracefully.	during the notification process.	functionality, ensure proper email configuration, and handle errors effectively. Retest to verify successful email notification functionality.
Browser Compatibility	Compatibility Test	Test the web application's compatibility with different web browsers.	Web application functions correctly and displays consistently across various browsers.	Compatibility issues are encountered, causing rendering errors or functionality issues on certain browsers.	Identify and address browser-specific bugs or inconsistencies, optimize CSS and JavaScript code, and conduct thorough cross-browser testing. Retest to ensure consistent browser compatibility.

File Upload Size Limit	Functional Test	Test the maximum file size limit for uploads and ensure proper handling of large files.	Files up to the specified size limit are successfully uploaded, and the application handles large files efficiently.	Upload fails for files exceeding the size limit, or the application becomes unresponsive when processing large files.	Implement file size validation mechanisms, optimize file processing algorithms, and provide clear error messages for oversized uploads. Retest to verify proper handling of file size limits.

User Permissions	Security Test	Test the enforcement of user permissions and access controls to prevent unauthorized actions.	Users are restricted to performing only authorized actions based on their assigned permissions and roles.	Users can access restricted functionality or perform unauthorized actions due to permission-related vulnerabilities.	Review and strengthen access control mechanisms, ensure proper enforcement of user permissions, and conduct thorough security testing to identify and mitigate permission-related risks. Retest to confirm secure user permissions and access controls.
Error Handling	Functional	Test how the web application handles errors and unexpected situations.	Users receive informative error messages when encountering issues, guiding them on how to resolve or retry actions.	Users encounter cryptic error messages or experience crashes without clear instructions for resolution.	Implement comprehensive error logging, user-friendly error messages, and graceful degradation.

Table 16: *Testing Plan*

H. Economic Analysis (Juan D. Pérez)

The following tables include the amount of money the team already spent from February 13, 2024, to March 8, 2024, by the column of Budget Spent. The team needed to update the supplies budget because the team noticed they need more Amazon Web Services, and they need to reduce costs as well. The supplies the team were going to use are shown on Table 21 and the supplies the team are going to use now are shown on Table 22. Because the team updated the supplies, the estimated budget was reduced by 3%, with the current budget being $87,105.19 as shown on Table 23. This project will have a recurring cost of supplies because Amazon Web Services are being used with an annual cost of $1,842.00. This price will remain at this value if the application does not need more resources to improve performance, user capacity, or add another feature that another needs Amazon Web Services. Supplies, miscellaneous, additional benefits, and indirect costs have not yet been used. The supplies will begin to be used for the implementation phase. In Figures 3 and 42 are shown a summary of the expenses the team incurred between the dates mentioned in which one represents the costs incurred with the estimated budget and the other represents the current costs.

Human Resources						
Name	Position	Salary Hourly	Hours Worked	Total Work Period (Weekly)	Total Salary	Budget Spent
Jeremy J. Márquez González	Project Manager Computer Engineer	$25.00	15	10	$3,750.00	$1,500.00
Giomar Santiago Galloza	Computer Engineer	$22.50	15	10	$3,375.00	$1,350.00
Kevin O. Valentin Avilés	Computer Engineer	$22.50	15	10	$3,375.00	$1,350.00
Juan D. Pérez Sepúlveda	Computer Engineer	$22.50	15	10	$3,375.00	$1,350.00
Nayda Santiago	Senior Consultant	$100.00	5	10	$5,000.00	$2,000.00
Isidoro Couvertier	Senior Consultant	$100.00	5	10	$5,000.00	$2,000.00
Alberto A. Canela Cubero	Consultant	$50.00	5	10	$2,500.00	$1,000.00
Total					$26,375.00	$10,550.00

***Table 17**: Human Resources expenses*

Fringe Benefits								
Position	Benefits			Weeks	Quantity of Employees	Total per Position	Total	Budget Spent
	Social Security	Health Insurance	401k					
PM - Computer Engineer	$23.25	$200.00	$24.38	10	1	$976.25	$976.25	$390.50
Computer Engineer	$20.93	$200.00	$21.94	10	3	$928.63	$2,785.88	$1,114.35
Senior Consultant	$31.00	$200.00	$32.50	10	2	$1,135.00	$2,270.00	$908.00
Consultant	$15.50	$200.00	$16.25	10	1	$817.50	$817.50	$327.00
Total						$3,857.38	$6,849.63	$2,739.85

Table 18: *Fringe Benefits for Project manager, Computer Engineers, Senior Consultants and Consultants.*

Additional Benefits						
Position	Benefits		Position Quantity	Total per Position	Total	Budget Spent
	Overtime	Vacations				
PM/Computer Engineer	$750.00	$500.00	1	$1,250.00	$1,250.00	$0.00
Computer Engineer	$675.00	$450.00	3	$1,125.00	$3,375.00	$0.00
Senior Consultants	$3,000.00	$2,000.00	2	$5,000.00	$10,000.00	$0.00
Consultants	$160.00	$1,000.00	1	$1,160.00	$1,160.00	$0.00
Total				$8,535.00	$15,785.00	$0.00

Table 19: *Additional Benefits for Project managers, Computer Engineers, Senior Consultants and Consultants.*

Equipment		
Equipment	Price	Budget Spent
MSI Laptop	$1,000.00	$1,000.00
1900 MSI GS65 stealth 8SE	$1,900.00	$1,900.00
Lenovo Legion 7	$1,700.00	$1,700.00
XPS 15 9500	$2,000.00	$2,000.00
Total	$6,600.00	$6,600.00

Table 20: *Equipment for the realization of the project*

Supplies			
Supply	Monthly Price	Total	Budget Spent
Amazon Aurora DB	$211.70	$846.80	$0.00
Amazon RDS	$386.50	$1,546.00	$0.00
Total	$598.20	$2,392.80	$0.00

Table 21: *Estimated supplies for the realization of the project*

Supplies			
Supply	Monthly Price	Total	Budget Spent
AWS Amplify	$9.14	$27.42	$0.00
Amazon Cognito	$5.00	15	$0.00
Amazon API Gateway	$4.67	$14.01	$0.00
Lambda Functions	$15.01	45.03	$0.00
Amazon S3	$35.69	$107.07	$0.00
Amazon RDS (MySQL)	$45.59	136.77	$0.00
Amazon SageMaker	$38.40	$115.20	$0.00
Total	$153.50	$460.50	$0.00

Table 22: *Actual supplies for the realization of the project*

Budget			
Category	Budget Spent	Actual Cost	Estimated Cost
Human Resources	$10,550.00	$26,375.00	$26,375.00
Fringe Benefits	$2,739.85	$6,849.63	$6,849.63
Additional Benefits	$0.00	$15,785.00	$15,785.00
Equipment	$6,600.00	$6,600.00	$6,600.00
Supplies	$0.00	$460.50	$2,392.80
Miscellaneous	$0.00	$2,000.00	$2,000.00
Indirect Costs	$0.00	$29,035.06	$30,001.21
Total	$19,889.85	$87,105.19	$90,003.64

Table 23: *Budget Summary with budget spent actual and estimated cost.*

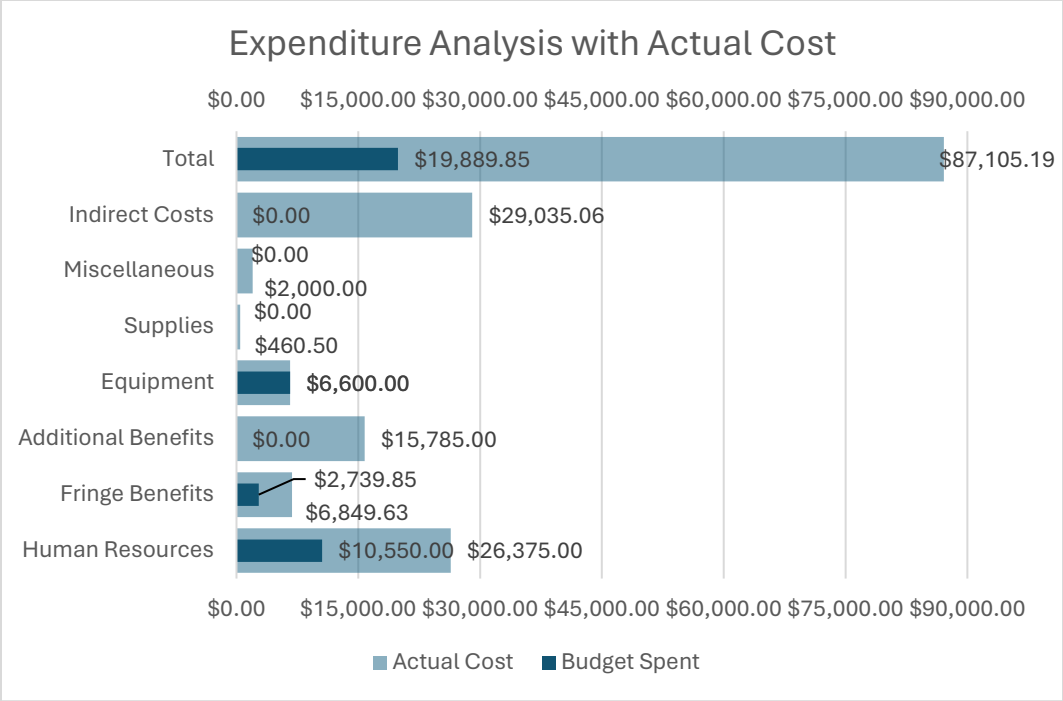

Figure 42: *Actual Cost Expenditure Summary*

I. **Task Progress and Gantt Chart** (Jeremy J. Márquez)

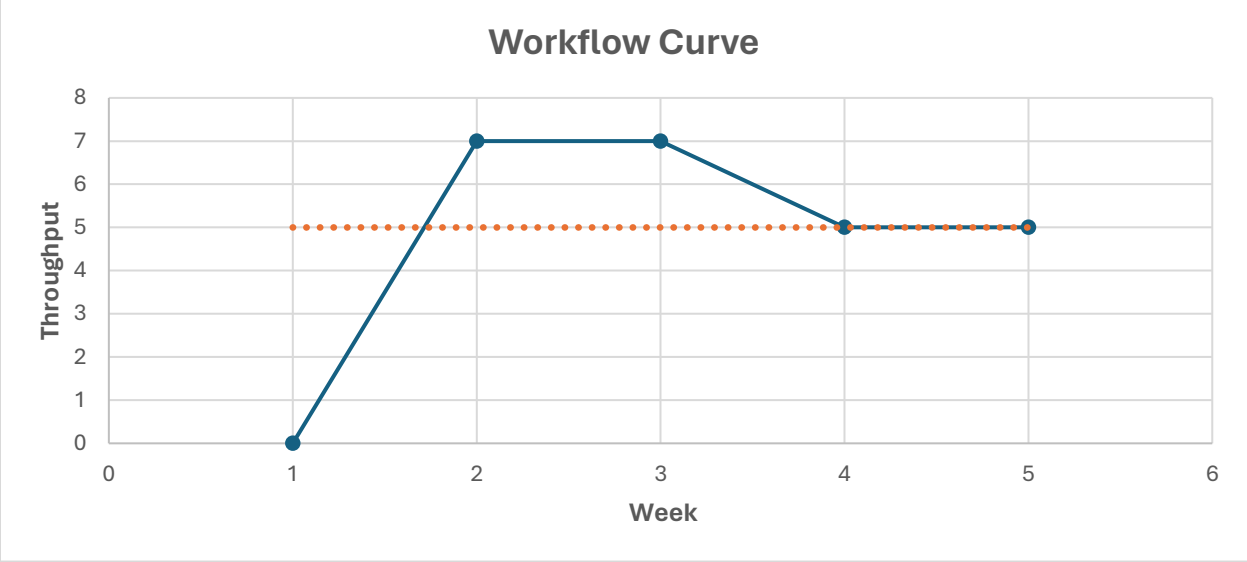

Figure 43: *Workflow Curve of the Team Throughput Tasks for Each Week*

Bit Busters experienced early delays in the research phase due to failure in calculating starting times and the required time to complete research. The team worked with an expected throughput of five tasks per week and in four of the five weeks that have passed since starting the

project, it has been achieved as shown in **Figure 43**. It is important to consider that new tasks were added in response to requirements and demands for the project. Due to this the total number of tasks went from 34 to 47. This included the SRS Document and all diagrams which represented a 17-day delay in the design phase. In response to all these delays and unexpected demands, the expected throughput was changed from five tasks per week to 6. This was done to finish the 23 remaining tasks in four weeks and have one week remaining to handle unforeseen challenges. The Kanban Board completed tasks and Gantt Chart were updated with all additional tasks and completion dates (**Figure 44, Figure 45 and Figure 46**).

The team has completed 24 of 47 tasks (51%) with the research and design phase and is starting the implementation phase by March 19, 2024.

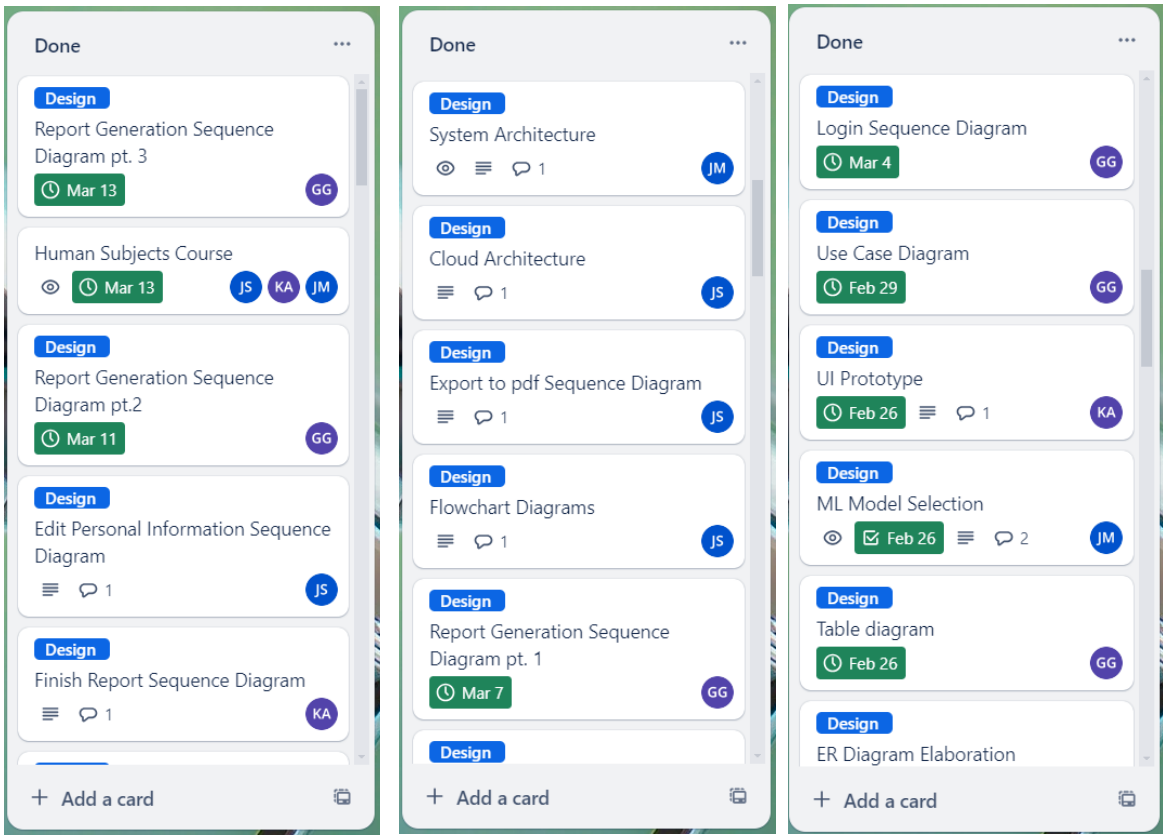

Figure 44: *Completed Tasks*

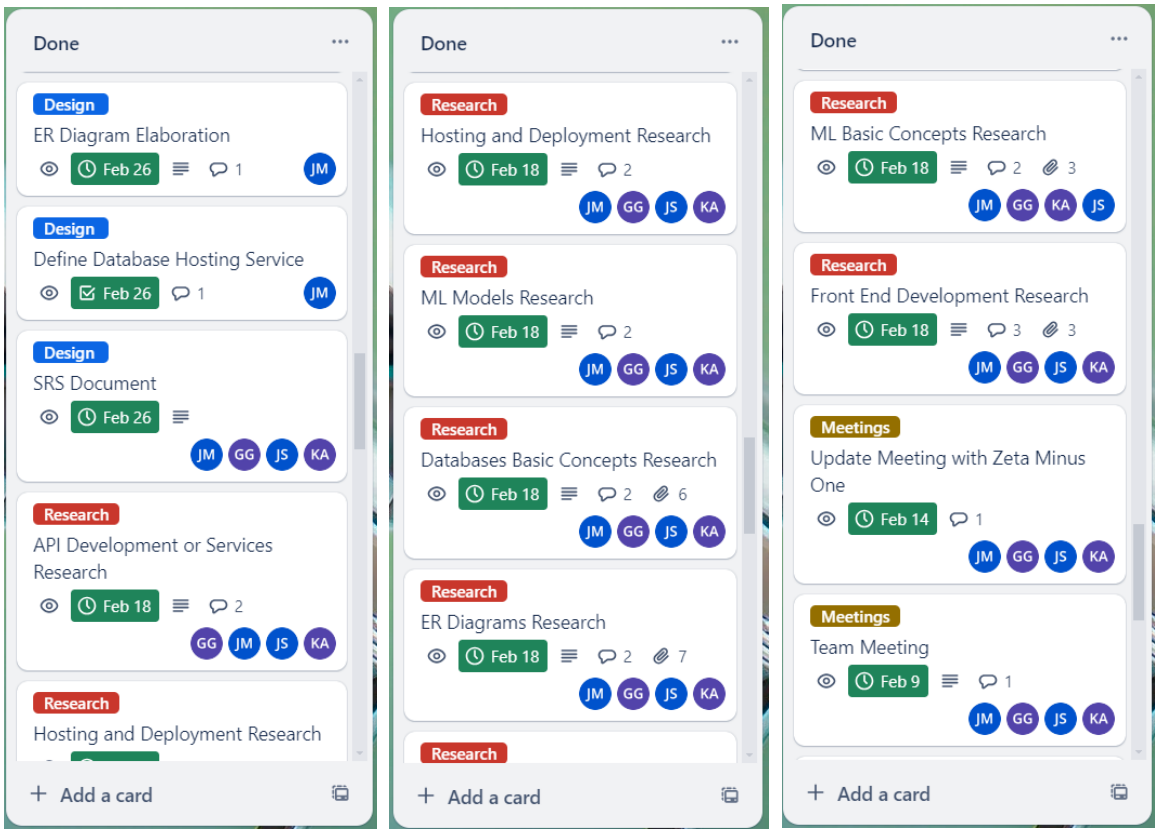

Figure 45: *Completed Tasks Continuation*

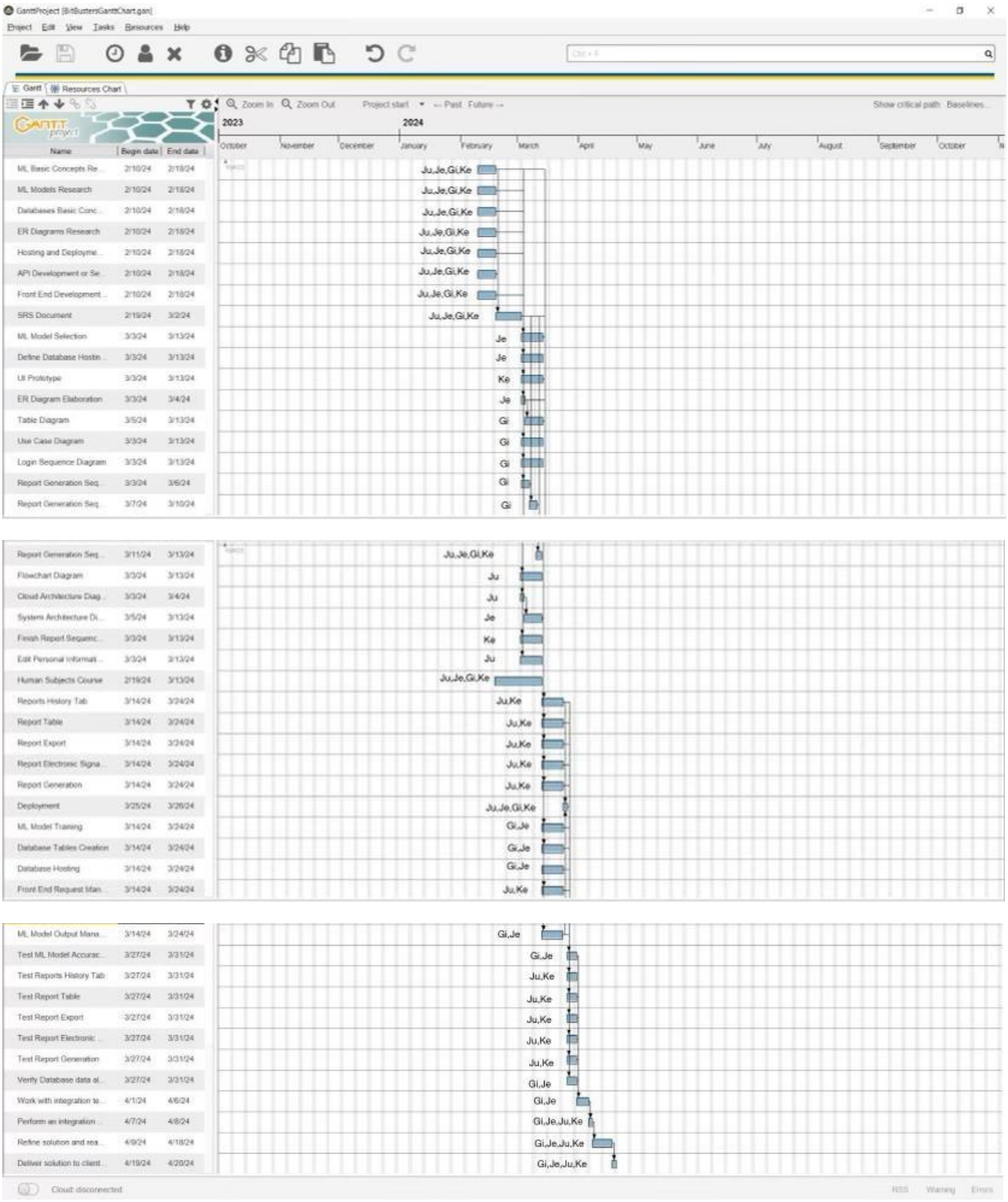

Figure 46: *Gantt Chart*

J. Requirements Agreement

Client Approval

Name	Date	Signature
Eric John Barbosa Lopez	Mar 2, 2024	Eric Barbosa

Figure 47: *Client Requirements Approval*