**Department of Electrical and Computer Engineering**

**North South University**



**Senior Design Project**

**Amago : A Cloud Based Produce Market for Independent Farmers, Grocers and Retailers**

**Abir Rahman ID # 1530412042**

**Jawad Aziz Khan ID # 1530457042**

**Ishfaq Zaman ID # 1530173642**

**Faculty Advisor:**

**Dr. Mohammad Monirujjaman Khan**

**Associate Professor**

**ECE Department**

**Summer, 2019**

Letter of Transmittal

8th September 2019,

To

Dr. K. M. A. Salam

Professor & Chair,

Department of Electrical and Computer Engineering

North South University, Dhaka

Subject: **Submission of Capstone Project on “Amago: A Cloud Based Produce Market for Independent Farmers, Grocers and Retailers”**

Dear Sir,

With due respect, we would like to submit our **Capstone Project Report** on “**Amago: A Cloud Based Produce Market for Independent Farmers, Grocers and Retailers**” as a part of our BSc program. The report deals with introducing an online digital market for Agricultural trade. This project was very valuable to us as it helped us gain experience from implementing software in the practical field and apply it in real life. We tried to make the most of our competence to meet all the dimensions required for this report.

We will be highly obliged if you kindly receive this report and provide your valuable judgment. It would be our immense pleasure if you find this report useful and informative to have an apparent perspective on the issue.

Sincerely Yours,

.........................................................

Abir Rahman

ECE Department

North South University, Bangladesh

........................................................

Jawad Aziz Khan

ECE Department

North South University, Bangladesh

.........................................................

Ishfaq Zaman

ECE Department

North South University, Bangladesh

APPROVAL

**Abir Rahman** (ID #**1530412042**), **Jawad Aziz Khan** (ID #**1530457042**) and **Ishfaq Zaman** (ID #**1530173642**) from the Electrical and Computer Engineering Department of North South University, have worked together on the Senior Design Project titled “**Amago: A Cloud Based Produce Market for Independent Farmers, Grocers and Retailers**” under the supervision of **Dr. Mohammad Monirujjaman Khan** as partial fulfillment of the requirement for the degree of **Bachelor of Science in Computer Science and Engineering** on **September 2019** and has been accepted as satisfactory.

**Supervisor’s Signature**

…………………………………….

**Dr. Mohammad Monirujjaman Khan**

**Professor**

Department of Electrical Engineering & Computer Science

North South University

Dhaka, Bangladesh.

**Chairman’s Signature**

…………………………………….

**Dr. K. M. A. Salam**

**Professor & Chair**

Department of Electrical Engineering & Computer Science

North South University

Dhaka, Bangladesh.

Declaration

This is to certify that this Project is our original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. Any material reproduced in this project has been properly acknowledged.

Students’ Names & Signatures

1. Abir Rahman

ID 1530412042

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

2. Jawad Aziz Khan

ID 1530457042

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

3. Ishfaq Zaman

ID 1530173642

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

ACKNOWLEDGMENT

First of all, we wish to express our gratitude to the Almighty for giving us the strength to perform our responsibilities and complete the report.

The capstone project program is very helpful to bridge the gap between the theoretical knowledge and real life experience as part of our Bachelor of Science (BSc) program. This report has been designed to have a practical experience through the theoretical understanding.

We also acknowledge our profound sense of gratitude to all the teachers who have been instrumental in providing us the technical knowledge and moral support to complete the project with full understanding.

It is imperative to show our appreciation for our honorable faculty member Dr. Mohammad Monirujjaman Khan for his undivided attention and help to achieve this milestone. Also, our gratefulness is divine to the North South University, ECE department for providing us a course such as CSE 499 in which we could really work on this project and materialize it the way we have dreamt of.

We thank our friends and family for their moral support to carve out this project and always offer their support.

Table of Contents

|  |  |  |
| --- | --- | --- |
| **Index** | **Chapters** | **Page no.** |
|  | Organization of the Report | 9 |
|  | Abstract | 10 |
|  | **Chapter 1: Overview** | 11 |
| 1.1 | Introduction | 12 |
| 1.2 | Project Description | 12 |
| 1.3 | Purpose of Project | 13 |
| 1.4 | Project Goal | 13 |
| 1.5 | Summary | 13 |
|  | **Chapter 2: Related Works** | 14 |
| 2.1 | Introduction | 15 |
| 2.2 | Similar Existing Systems | 15 |
| 2.3 | Problems with the Current System | 16 |
| 2.4 | Proposed Solution | 17 |
| 2.5 | Summary | 17 |
|  | **Chapter 3: System Architecture** | 18 |
| 3.1 | Introduction | 19 |
| 3.2 | System Description | 19 |
| 3.3 | System Design | 20 |
| 3.4 | Summary | 21 |
|  | **Chapter 4: Software Design** | 22 |
| 4.1 | Introduction | 23 |
| 4.2 | Software Module Design | 23 |
| 4.3 | Summary | 26 |
|  | **Chapter 5: System Framework** | 27 |
| 5.1 | Introduction | 28 |
| 5.2 | Step-by-Step Framework Design | 28 |
| 5.3 | Summary | 29 |
|  | **Chapter 6: Skills Obtained** | 30 |
| 6.1 | Introduction | 31 |
| 6.2 | Skills Obtained | 31 |
| 6.3 | Summary | 33 |
|  | **Chapter 7: Working Sheets** | 34 |
| 7.1 | Introduction | 35 |
| 7.2 | Work Breakdown | 35 |
| 7.3 | Financial Plan | 37 |
| 7.4 | Feasibility Study | 38 |
| 7.5 | Summary | 39 |
|  | **Chapter 8: Future Work** | 40 |
| 8.1 | Introduction | 41 |
| 8.2 | Future Scope of Work | 41 |
| 8.3 | Summary | 41 |
|  | **Chapter 9: Design Impact** | 42 |
| 9.1 | Introduction | 43 |
| 9.2 | Environment Impact | 43 |
| 9.3 | Economic Impact | 43 |
| 9.4 | Social Impact | 44 |
| 9.5 | Sustainability | 44 |
| 9.6 | Summary | 45 |
|  | **Chapter 10: Compliance with Standards** | 46 |
| 10.1 | IEEE Standard | 47 |
| 10.2 | US Standard | 47 |
| 10.3 | EU Standard | 47 |
| 10.4 | Summary | 48 |
|  | **Chapter 11: Results** | 49 |
| 11.1 | Introduction | 50 |
| 11.2 | Results Achieved | 50 |
| 11.3 | Summary | 52 |
|  | **Chapter 12: Conclusion** | 53 |
|  | **Bibliography** | 55 |
|  | **Appendix** | 56 |
| 14.1 | Android Code | 57 |
| 14.2 | Server Code | 68 |
| 14.3 | Website Code | 78 |
|  | **List of Figures** | 92 |

# **Organization of the Report**

Chapter 1 : Overview | This chapter provides an overview of the complete project idea.

Chapter 2 : Related Works | This chapter talks about similar and current works done and the proposed solution.

Chapter 3: System Architecture | This chapter highlights the complete technical architecture of the system.

Chapter 4 : Software Design | This chapter goes in depth into the design of the android and web applications.

Chapter 5 : System Framework | This chapter highlights the database and server application design

Chapter 6: Skills Obtained | This chapter talks about the skills that were need to be acquired before and during the duration of the project

Chapter 7: Working Sheets | This chapter talks about the work timeline and procedures, feasibility study and sustainability.

Chapter 8: Future Works | This chapter talks about what future work extensions can be done.

Chapter 9: Design Impact | This chapter highlights the social, economical and environmental impacts of the project.

Chapter 10: Compliance with Standards | This chapter talks about how the project maintains major industry standards.

Chapter 11: Results | This chapter talks about the project results achieved.

Chapter 12: Conclusion | This chapter concludes the whole project.

# 

# **Abstract**

Bangladesh suffers from fluctuating prices of produce due to several factors, such as transport costs, seasonal supply demand changes, forced price inflation by stocking goods when demand is high and many more. Each of these factors is a unique problem but some of the solutions to these problems are data driven. Amago is a digital produce market platform that aims to eliminate inefficient, unfair and unprofitable segments of the country’s existing pipeline of the agricultural trade. The optimization of calculated minimal waste crop yield is a data problem where supply and demand data should be leveraged for prediction that our proposed system could potentially solve. Another point of inflection where profits are unfairly lower are for the farmers when sold to the unregulated middle men. Thus the Amago Platform has been proposed as a better solution for a digital future. The Amago Platform consists of 3 main components that we have implemented successfully - the Server and Application Layer that handles data and queries, the Android app for farmers to post their harvests for sale, and the Web Client for grocers and retailers who would buy from the posted harvests. The platform will serve as a digital open marketplace for both parties and optimizations can be introduced with data analysis and machine learning.

# 

**Chapter 1**

**Overview**

**1.1 Introduction**

Bangladesh suffers from fluctuating prices of produce due to several factors, such as transport costs, seasonal supply demand changes, forced price inflation by stocking goods when demand is high and many more. Each of these factors is a unique problem but some of the solutions to these problems are data driven. Amago is a digital produce market platform that aims to eliminate inefficient, unfair and unprofitable segments of the country’s existing pipeline of the agricultural trade. The optimization of calculated minimal waste crop yield is a data problem where supply and demand data should be leveraged for prediction that our proposed system could potentially solve. Another point of inflection where profits are unfairly lower are for the farmers when sold to the unregulated middle men. Thus the Amago platform has been proposed as a better solution for a digital future.

**1.2 Project Description**

We propose an efficient, semi-automated produce warehousing platform that will quickly accumulate produce directly from the farmers based on market demand and ship them to urban retail grocery stores.

Farmers will post a sell request through an android application. Our system will process these requests and accumulate the produce until reaching maximum storage capacity. Retailers will be able to order through an online web interface, once an order is received it will be directly shipped to their specified store/outlet. The optimisation objective of the platform is to minimise the time the produce spends inside the warehouse.

**1.3 Purpose of Project**

The purpose of the Amago Platform project is to replace or at least augment the manual, inefficient, unregulated, unfair and unprofitable pipeline of the current agricultural trade to better help the economy grow and to provide an attainable future for the producers to scale up their business and move up in the social ladder. As of now, the fresh produce changes hands multiple times on the way to the urbans from the farmlands and each time the price of the produce increases in price while the corrupt middlemen take a part of the cut. But the real problem is the fact that the farmers are nevertheless underpaid at the end of the day.

**1.4 Project Goal**

To create an intuitive and layman friendly platform with Bangla support that can facilitate trade among farmers, producers and grocers, retailers so that both parties can buy and sell at fair government set prices and help the agricultural economy grow.

****

Figure 1.1: Amago Platform Logo

**1.5 Summary**

Amago is a digital cloud based produce market that aims to help the agricultural economy by connecting farmers to grocers such that both can trade at acceptable prices and profit. The platform is meant to be used by laymen and accommodates them thusly by design.

**Chapter 2**

**Related Works**

**2.1 Introduction**

Currently there exists no such solution in the market. Studies show that there are several challenges that have affected the agricultural industry over the years, this includes climate change, inefficient use of water and fertilizers, pests, lack of quality seeds, unfair pricing and insufficient investment in research [1]. There are multiple emerging agro-tech startups that are planning to tackle these challenges.

**2.2 Similar Existing System**

"Ifarmer.asia" is a Bangladesh based startup that is providing an investment platform for farmers. Farmers can sign up to work for designated farms. These farms are then advertised for sponsorship by the platform. The sponsor and farmer share the profit from the farm giving a commission to the platform.

"Impact Terra", a Myanmar Based startup, is designing an application to help smallholder farmers by providing real-time recommendation of leading agricultural information. Their digital service "Golden Paddy" consist of a web/android application and a Facebook page. Aiming to be an effective tool for efficient farming.

On July 2019, ACI launched ‘Fosholi’ an android app which aims to create an agricultural platform in Bangladesh. It promises to enable 15 million farming families to access high quality agricultural data and services from the app. Fosholi offers intelligent information and advisory services to local crop farmers for free. Fosholi will help farmers by providing information on crop suitability, modern agronomic technology and practice, pest and disease alerts and weather forecasts. Thus, it will improve farmers’ decision making and farming productivity and ensure the overarching national goal of sustainable food and nutrition security.

"Plantix" is an Android based farming assistant tool. It provides crop health checking through the use of computer vision, connects farmers to scientists, experienced farmers and plant experts.

**2.3 Problems with the Current System**

As mentioned before current methods do not address the main logistical and storage related problems found in the agricultural industry.

Ifarmer is an investment platform for farmers and mostly focus on livestock rearing. Currently providing no option for crop farming. Nor do they provide any solution to connecting farmers directly to retail market.

"Impact Terra" and "Plantix" simply is a “Farming Assistant” Tool that provides tips and suggestions from experts to farmers using an app.

ACI Foshli while promising to provide data and logistical solutions to farmers does not have a solid business policy that could incentivise adoption.

**2.4 Proposed Solution**

“Amago" aims to output an online service coupled with a mobile application and a web client that will digitally connect the aforementioned users in a produce marketplace, similar to modern e-commerce services. The platform aims to own multiple warehouses in geographically lucrative location where farmers can opt to store their produce and sell it directly to uban retail markets. Logistical support is also provided through our service. The ability to store produce allow farmers to take their time and sell crops at a higher rate. Therefore the platform provides a higher chance of profitability for farmers automatically incentivising them to use it. The lack of intermediary agents allow for lower prices for both sides in this b2b model.

**2.5 Summary**

While there are many Agriculture related applications most provide a supportive role. No current solution exists in the market that addresses our concern. Amago therefore is filling the void of a much needed service.

**Chapter 3**

**System Architecture**

**3.1 Introduction**

The Amago Platform is an online service and thus would require a backend and frontend system with a suitable communication protocol to successfully function and keep it’s operations active. It would also require a database and accompanying management system to store it’s user data. In this section, we discuss the technology, underlying structure of the system and design principles that have gone into making this project tangible.

**3.2 System Description**

The Amago System is run and managed by 3 components; the PostgreSQL database, the NodeJS Server with the Application Layer, and the Frontend Android App plus the Web Client. The database stores all the user’s necessary information in tuple form. Only necessary information like Name, ID, Phone number, PIN and Location are stored. The Application Layer handles the queries sent to the database dynamically as endpoint URLs are called by the frontend client applications. The Server and Application Layer is written in Javascript and uses NodeJS for high-level control over the server logic.

Accompanying NodeJS are 2 frameworks - Express and Sequelize, which make handling the server logic and communications with the server and clients easier. They are scalable and is proudly touted as being able to take large network traffic load and still maintain atomic transactions with the server.

**3.3 System Design**

The proposed system will operate using a PostgreSQL Relational Database System. The database will log user account details, transaction requests, inventory and geolocation data. The backend will be managed using a Node framework called ExpressJS and queries will be managed by the Sequelize framework. The robust ExpressJS framework is used to create APIs that is used for communicating with the server through an Android app/Web interface. Whereas Sequelize is a database bridging manager that handles queries written in Javascript then dynamically and efficiently translates it for the PostgreSQL hosted database. A diagram of the collective system architecture is given below:



Figure 3.1: Amago System Architecture

Responses to queries are sent back as a JSON entity code. Such responses are very easy to parse and manipulate within the application and is also capable of handling large quantities of data entities, hence the decision to use it as the medium of data being communicated between the client and server. Given below is an example of a JSON response, an element from a JSON Array of the produce from a particular (farmer) user, note the user is denoted by the ‘userID’ and ‘itemType’ denotes what produce it is:

{

"id": 1,

"itemType": 3,

"userID": 15,

"amount": 300,

"price": 500,

"status": 2,

"retailID": null,

"retailName": null,

"createdAt": "2019-08-23",

"updatedAt": "2019-08-23"

}

The integer itemType is parsed and displayed by using a lookup string array that contains a list of all the names of the produce available for the farmers to harvest. This list is also queried from the database if needed or a local copy is maintained on both the Android app and the Website that have matching values.

**3.4 Summary**

The system was designed with stability and scalability in mind with regards to using modern application, server and web technologies. Efficiency and atomicity of the transactions is also to be preserved with the system designed. Overall a holistic component approach was adopted for the design of the Amago System Architecture.

**Chapter 4**

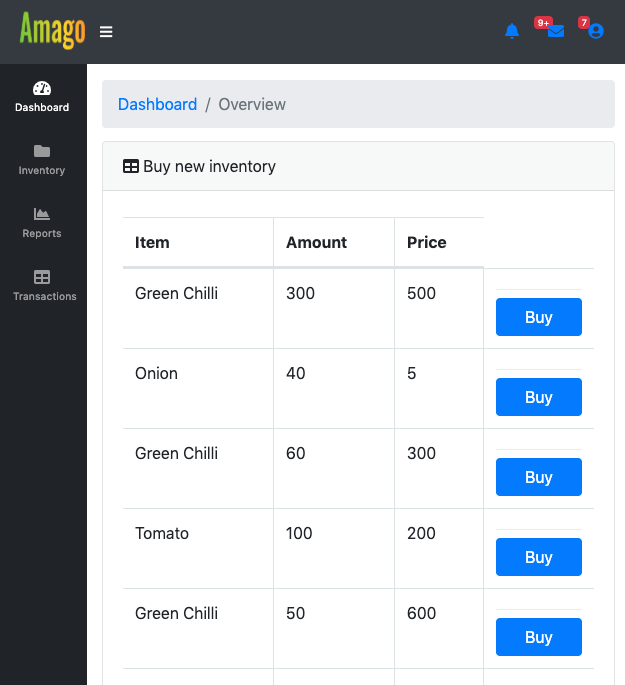
**Software Design**

**4.1 Introduction**

For the completion of the Platform, 2 frontend applications needed to be developed - an Android App for the farmers and a Web Client for Grocers. The two applications were developed separately but while following the Server Application Layer’s endpoints and constraints. The Application Layer was also developed with API endpoints for handling communications and routing.

**4.2 Software Module Design**

Web Client

Our retail clients will be provided with a web interface. The web client is written in Embedded Javascript which is a set of instructions and templates that generate the necessary HTML code based on the user’s query and the state of the database. The server queries and parses the retail user’s account information and dynamically builds the views for the website they will be browsing. The web interface will be a dynamic website designed using HTML, CSS and JavaScript. The clients will be able to place orders for produce based on their needs. Figure 4.1: Screenshot of Retail Website

Android:

The Android app, its API frameworks and the UX will follow Google's "Material” design principles. Standard software patterns and Object Oriented design was considered during the development of the app. It was done in a Linux Android Studio environment and tested on a Virtual Device running Android 5.0. The app was laid out such that non tech savvy farmers will be able to use it based on its easy operations and simple atomic functions.

Another significant way farmers can better use the app with minimum difficulty is by making the entire interface of the app fully in Bangla. This is done with a dynamic locale management system and is maintained by cross translating every possible string used in the interface in a reference resource table as seen below:

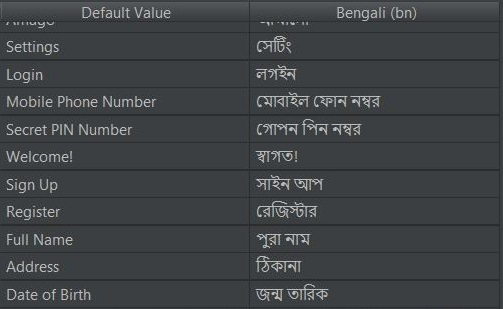


Figure 4.2: Example Screenshot of Translation Table.

The UI surrounding the text is designed with ‘user look flow’ in mind and buttons given large welcoming shapes and colors. The UI design was kept simple minimalistic and large such that even laymen and farmers who are not very tech-savvy would be capable of using the app without difficulty or much training. Text is displayed at a highly readable 24dp and colors are contrastive as they are set up in a triadic pairing.



Figure 4.3: Screenshot of the Login, Registration and Main Harvest pages of the Amago Android app respectively.

Sell requests and Harvest posts through our app are displayed on the main home page in a vertical linear view, with each operation and item segmented as a card. Each operation reloads the home page and calls the server for an updated list of harvests and sell requests. Sell requests are coupled with a status integer denoting the stage of the harvest and the corresponding card is changed to reflect it. For example when a harvest is accepted for delivery, the card can display an icon of a truck, the recipient’s name and location.

Farmers’ login credentials are sent to the server API via the Retrofit framework. Retrofit is a high level third party Android framework dependency that makes sending and processing HTTP Requests vastly convenient, concise and streamlined. This is done by handling all GET and POST operations like regular functions from within Android and passed to the Retrofit Interface class to run. Retrofit functions need to be declared in the Interface class along with it’s required fields. For example, for a ‘login’ POST request the only fields needed would be the phone number and the PIN. The interface function declaration looks something like:

@FormUrlEncoded

@POST("api/auth/login")

Call<ResponseBody> Login(

@Field("phone") String phonenumber,

@Field("password") String pinnumber

);

After declaration of the interface it is only a matter of calling the function, sending the necessary parameters and parsing any responses sent from the server. The return type of the functions are of ‘ResponseBody’ type which can be parsed as a JSON.

**4.3 Summary**

Implementation standards designed by Google were followed for the Android app whilst also accommodating Bangla for farmer usage and accessibility. The Retrofit API was very useful in building the app since it made endpoint calls extremely easy. The Web client smartly uses EmbeddedJS to dynamically generate views.

**Chapter 5**

**System Framework**

**5.1 Introduction**

This chapter provides an overview of the database design and server application design.

**5.2 Framework Design**

One of the major goals of this project was to make sure that the system was designed to be scalable and efficient. Following industry standards, we had started with our database. We choose to go with a relational database, based on the fact that our system would be storing lots of data. Being able to run high level queries were an important aspect. The database relation diagram given in fig 5.1 shows how we planned our design. We tried to make query faster by trying to stick to an integer based system, considering integers take less time to sort. Our design also allows for complex joins that would create some interesting work when working with data. We’ve tried to create as little redundancy as possible.

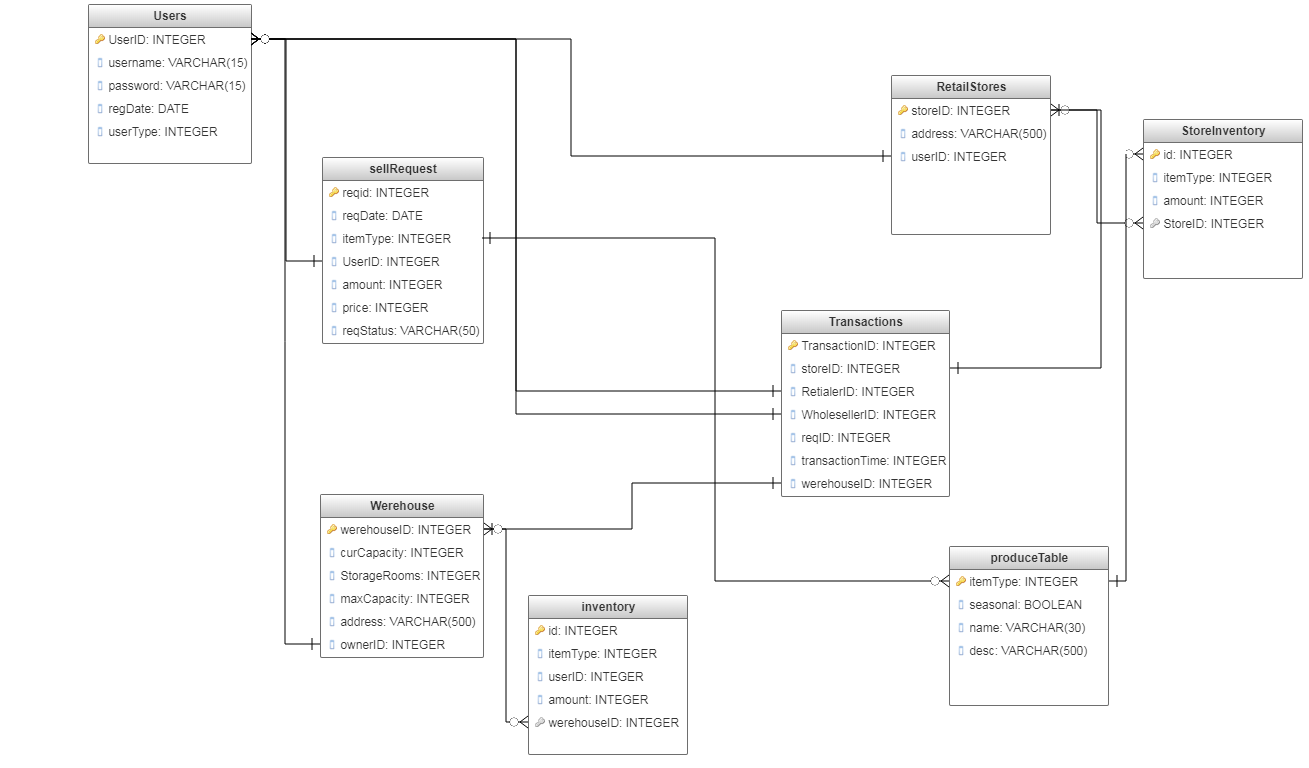


Fig 5.1 Database Relation Diagram

The application server design was based on the MVC or model view controller architecture. We have API routes and web routes all handling the validation and handling of input and output ports. All routing was done using expressJS framework because of its fast build cycle and the fact that it does all the json parsing for us and no extra code needs to be written.

There are specific controllers that contain all the logic and validations required for example say registration. The controllers are designed to be modular so that down the line they can be updated. We have tried to make the system secure and hassle free by using json web tokens for authentication and seamless login. The json web tokens provide a unique hashed code that is only decrypted using a secret key set by us. Hashed information is so secure that, neither can the user nor the developers know how to decrypt it. Additionally there were small handy utility middlewares such as loggers and uuid code generators.

One of the major frameworks we used was Sequelize to help create models of the database for easy query. We used the native sequelize queries for most use cases but also used raw SQL queries for more advanced tasks. The server was hosted on heroku to help our development team collaborate real time and this also made testing remotely easier.

**5.3 Summary**

To conclude, the system was designed to be fast, efficient and scalable. Starting from the database design to application server design, all of it was done after hours of research and learning, trying to maintain industry standard.

**Chapter 6**

**Skills Obtained**

**6.1 Introduction**

This project required us to design a full end to end product solution complete with a RDMS database, server backend, web front end and android application. We therefore received exposure to multiple different technologies and acquired experience in proper software engineering practices.

**6.2 Skills Obtained**

Our project technology stack included java for the android mobile application, node javascript server framework, express web development framework, sequelize a custom ORM for SQL queries, json web tokens, embedded javascript template engine, HTML, CSS and Postgresql. During the start of the project timeline we spent a significant amount of time learning node.js and express.js frameworks. We had chosen these stacks based on their high performance and fast code cycles. Amongst all our members, we were experienced in programming so it was fairly easy to get started with javascript. There was a small learning curve to understanding how the express framework utilized routing protocols for APIs. We had created our database on postgresql and used the sequelize framework as an ORM to help maintain relations more easily. We learned how to create sequelize models and also learned how to make simple queries using the model syntax that sequelize provided. However, later down the line, we discovered that we were unable to do certain complex table queries using sequelize but luckily we were able to query the tables manually with raw SQL queries.

Having prior android application development experience, we were able to jump into production very fast however, our previous experiences were unpleasant due to the way android studio handled API calls. Previously we had to write numerous lines of code to handle API posts and requests inclusive of the manual parsing of data, however this time we spent some time learning about ‘retrofit’. Retrofit is an open source type safe HTTP client for android and java. Time invested in learning retrofit’s implementation saved us a lot of time during the final stages of the production timeline. Lastly, on top of the basics of HTML and CSS we had to learn integrating EJS pr embedded javascript template engine to help us setup the web application. Using ejs was a time constrained decision because due to lack of time, this was the fastest to learn and implement.

|  |  |
| --- | --- |
| Technology | Reasoning |
| Node.js | Node.js is an extremely versatile backend framework that has seen extensive industrial usage. The framework provides excellent tools for creating APIs that will be used by our applications to communicate with our server and database. |
| PostgreSQL | PostgreSQL provides a “complete”, tried and tested implementation of SQL which follows the ACID principles. It natively supports many advanced SQL functionalities and allows for faster deployment with minimal tweaking. |
| Java and Android Studio | Java is a performant and widely known scalable application programming language. It has automated memory management and taking advantage of features is much more streamlined as it is Object Oriented. Android Studio is the platform where the app is to be implemented which allows fast prototyping, API handling and debugging. |
| HTML, CSS, Javascript | HTML is the world standard for building smooth simple but aesthetically pleasing frontend web interfaces. CSS allows for animations and styling while JS lets functions and business logic run in the background of the site and is a highly readable and scalable tool. |
| Python | Python is enriched with many powerful statistical testing and machine learning libraries. These libraries will be used extensively when implementing our statistical testing platform and machine learning model for demand prediction. |

**6.3 Summary**

All in all, over the course of the project time period, we learned and implemented several industry standard technologies. We can now call ourselves ‘experienced’ in these fields and similar tasks would take almost half the time to accomplish.

**Chapter 7**

**Working Sheets**

**7.1 Introduction**

In this chapter, we observe the entire work structure, meaning how the scheduling was maintained throughout the developmental phase. We shall also see the financial foundation of this project and furthermore the feasibility study will also be discussed.

**7.2 Work Breakdown**

The project development was conducted in a number of phases. They were planned as follows:

Development methodology used was Agile. Each cycle was be around 2 weeks long. In each cycle the development team updated the product with major features. A three member team collaborated together to bring the project to life.

*Phase 1: Research*

* This phase was dedicated to researching the technological factors to be used and the market research to collect vital data about the implications of the product.
* Delivered: Research material and datasets to progress to building the AI model for the platform.

*Phase 2: Core Development*

* This phase oversaw the preparation of the minimal viable product (MVP). This includes the UI/UX, frontend and backend development. All the sections was completed to meet the basic requirements of the project.
* Delivered: At the end of this phase, a useable platform was provided which included a mobile application and retail website. Performance of both services were at an acceptable state.

*Phase 3: Performance optimization*

* This phase concerned performance optimization and bug fixes. Performance optimization of the website and backend services were done to a critical extent.
* Deliverable: At the end of this phase, the platform was tuned and optimized for the maximum user experience metrics.

*Phase 4: User testing*

* This phase concerned the end user experience. Testing of the platform was done in this phase, starting from user testing to internal code tests to make sure the system is easily accessible yet secure.
* Deliverable: At the end of this phase a final report was published stating the progress of the project.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Month** | | | | | | | |
| **Major Milestones** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| Requirement Analysis |  |  |  |  |  |  |  |  |
| Market Research and Data Collection |  |  |  |  |  |  |  |  |
| Backend system  development |  |  |  |  |  |  |  |  |
| Android App development |  |  |  |  |  |  |  |  |
| Web Services |  |  |  |  |  |  |  |  |
| Feedback and UI Changes |  |  |  |  |  |  |  |  |
| Final Report |  |  |  |  |  |  |  |  |

Figure 7.1: Gantt Chart

**7.3 Financial Plan**

During the complete duration of the project we utilized our own computers and laptops to minimize costs. Fortunately, one of our team members already owned powerful enough hardware to perform data analysis. We utilized ‘heroku’ to deploy our database, application server and retail website. Using heroku helped us cut costs down and not compromise on performance. Overall, there were almost no costs involved in the production of the platform as we tried to use only open source software and frameworks.

**7.4 Feasibility Study**

During feasibility analysis for this project, the following primary areas of interest are to be considered:

**Technical feasibility**

If we consider the technology stack used to build the ‘Amago’ platform, it can be seen that these are very modern technologies and have large community support. The technology is easy to learn and upgrading should also be easy. The system is built to be modular, so that it can be easily upgraded and maintained. Integration of new technology is also possible with the API routing system that we have programmed in.

Most of the frontend and backend portions of the project can be built with open source tools that are highly well documented. System requirements for our application server and android application are very minimal, hence end users won’t have to bear high computational costs. The technology stack was chosen with thought to make sure that scalability is a major option. Our system can currently handle upto approx. 3000 concurrent users with ease. This number can be extended through investment in better hosting servers.

**Economic feasibility**

The ‘Amago’ platform itself does not require a lot of money to be developed, however putting the system in the hands of users and operational costs are massive. A big chunk of investment is needed to bring the service to the market and initially requires a lot of training sessions to teach our target group of farmers how to use the application and how it may benefit them. All digital aspects of the project are financially feasible but physical aspects of it are not.

**7.5 Summary**

To conclude, we discussed the scheduling processes of the development of ‘Amago’ and additionally talked about the feasibility options of the complete system, starting from production to market.

**Chapter 8**

**Future Work**

**8.1 Introduction**

In this chapter we discuss the future of our project including how we plan to improve the system and what new features we might add down the line.

**8.2 Future Scope of Work**

We plan on leveraging the large data that the system will amass if users join the platform. We plan to apply machine learning to make sensible understanding of all that data. We also plan to introduce new features such as farmer support and automated warehouse management tools. Computer vision models can also be used to help farmers identify crop health and other useful info, similar to "plantix".

**8.3 Summary**

This chapter has described the possible future applications of the design. But there are a lot of possibilities with the designed system. The system may need some research for different applications, though the principle of the designed system will remain as it is, mostly data driven future is expected.

**Chapter 9**

**Design Impact**

**9.1 Introduction**

This section talks about the design impacts of the ‘Amago’ platform. Topics range from environmental impact to social and economical impacts.

**9.2 Environmental Impact**

Environmental impacts can include utilization of arable land and since most of the inventory will be done digitally, there will be significantly less use of physical paper. Overall contributing to a positive environmental impact.

**9.3 Economic Impact**

The expected economic impacts are as follows:

* Farmers are expected to receive fair pricing on their production as ‘Amago’ plans to remove the middle man in the transactions.
* Farmers can stay connected to the market more easily as vital supply demand information will be provided to them
* Help farmers modernize their processes and bring new innovative technologies to them to help them transition into the 21st century.
* Help farmers export their products abroad by trying to create a direct channel for them to operate.
* Create a fair pricing and regulation standard for the drivers of long haul deliveries.

**9.4 Social Impact**

* ‘Amago’ hopes to uplift farming as a profession, by helping farmers earn more and showing that it is a well respected lifestyle. These people feed us and we would go hungry without them.
* Create more jobs for logistic personnel and independent drivers.

**9.5 Sustainability**

Initially, the project requires funding from investors. The initial funding will be used to fund the development phase and a portion of the marketing. Our sustainable business model would be to charge a 2% (subject to change) flat service charge on the overall order. However, we do not plan to solely rely on this but rather utilize the data we collect over time to create some other additional sources of revenue.

Scalability of the project is massive. Initially we would be entering the market as a platform that connects suppliers to retailers in the most efficient way possible but with time as we grow our database with information regarding supply, demand and crop yields, we would be able to diversify into providing market analysis. Not only this, the farmer application can be used to collect vital information about the farmers land and crops, which can be later processed to help farmers obtain higher crop yields based on crop optimization. The farmer application can also work as an information hub for farmers providing advice, a marketplace for farmers to order equipment, fertilizer, other essentials etc.

As mentioned earlier, the project has a high potential for income generation. Initially revenue will be generated on a flat 2%(subject to change) service charge added on top of each transaction. A real life assumption can be made, a seller ‘A’ wants to sell 1 tonne of onions at wholesale price (100 kg = 1562 BDT, set by the ministry of agriculture), therefor total of 15620 BDT. If you add logistics on top of this, then an approximate addition of 5000 BDT(subject to location), and the total becomes 20620 BDT. Amago takes 2% of this, which is about 413 BDT, so the total price retailers pay is 21032 BDT. Assuming Amago initiated 500 of such onion orders, then we make approximately 206500 BDT. This amount is only for one particular product and can be scaled to more products.

After the initial investment, it will take some time to familiarize the product to the market and might not be making any profit. However, in the long run, profit will hopefully be made. To reach a profitable state, more research and development would be needed and this would require further financial support.

The instant benefit of the project will be enjoyed by the farmers and logistic personnel. But down the line it would have a massive impact on the way retailers buy produce. Farm to retail procedure would be structured and tracked leaving no place for bad quality produce to make it to shelves. Research and development will be done based on the data acquired to help farmers make better yields and quality produce.

**9.6 Summary**

This chapter has covered the different types of impacts that our system offers and those has been described and discussed. From the above given impacts we can conclude that our designed system is good enough to use under any circumstances.

**Chapter 10**

**Compliance with Standard**

**10.1 Introduction**

In this section we discuss the consistency of our task with diverse standards. There are a few distinct standards, amongst which the IEEE standards, US standards and European standards are talked about in this part.

**10.2 Compliance with IEEE Standard**

There are a few distinct guidelines put forward by IEEE Standards affiliation. The majority of them however are not material for our framework. We have included idea of operation as for the IEEE standard. A conference paper has additionally been submitted and affirmed by IEEE standards entitled "**Amago: A Cloud Based Produce Market for Independent Farmers, Grocers and Retailers**" that points our work on this task.

**Chapter 10.3 Compliance with US Standard**

ANSI recommends that copyrighted software should only be included for informational purposes, or in forms which do not mandate particular implementations of the standard. Object code should never be included in a standard as a normative requirement. While ANSI opposes use of software standards to mandate particular implementations and believes that the use of software in standards should be avoided to the extent possible, ANSI recognizes that there may be circumstances in which inclusion of some software, provided it is accompanied by adequate legal permissions, may facilitate development of multiple, competing and interoperable implementations of the standard. Examples of such software could include: ·

* Pseudo Code (code that is human readable and similar to programming languages but cannot be directly processed or compiled directly to be processed by hardware that manipulates data according to instructions);
* Schema examples;
* Data structure definitions;
* ASN.1 structure definitions;
* ABNF grammar specifications;
* Example programming instructions that are sufficiently limited in scope that they do not, either singularly or in the aggregate, perform a complete or a substantial part of a function and are illustrative, at most, of limited sections of an independent fully described specification; or
* Sample programming instructions provided solely for conformance testing purposes.

Our project has been established based on the above ANSI principles and it completely relies upon it.

**Chapter 10.4 Summary**

In this section we have examined the different compliant standards and made sure that we are in accordance with. These standards have been put without hesitation so as to control things, guarantee well-being and ensure there are no well-being dangers to the use of distinctive segments. It is imperatively essential to maintain these measures and we have done as such over the span of our task work.

**Chapter 11**

**Results**

**11.1 Introduction**

This chapter of the report contains the results that we achieved throughout the course of using this system.

**11.2 Results achieved**

By investing a significant amount of time on research and discovering technological industry standards, we were able to design a dynamic and flexile backend which made frontend and android development relatively smooth. We found that using Node.js as our primary application server technology would triumph over using raw PHP as it provides much higher scalability due to its asynchronous network command execution. This enables it to handle much higher user traffic compared to PHP. We created an API based server side application that made future expansion and upgrades and development easy. Planning took a major portion of the project timeline because of the extensive scale of the project. The flexible backend allowed us to implement granular APIs to meet the needs of our Android application and web interface. We were able to design an android app which is able to connect wholesalers or independent farmers to urban retailers along with storing accurate logs of transactions and inventory of its users. The UI of the android app is kept simplistic and intuitive with complete bangla localization to make it easy to use by our target demographic.

Additionally to all this, we were able to collect data regarding sales of agricultural products to apply a low level data analysis and prediction models. The idea of this was to create a feedback system to help farmers and retailers both understand the market better and act accordingly. However due to time constraints we were not able to implement the feedback system into our core code. However we played around with the data we collected and found interesting outputs.

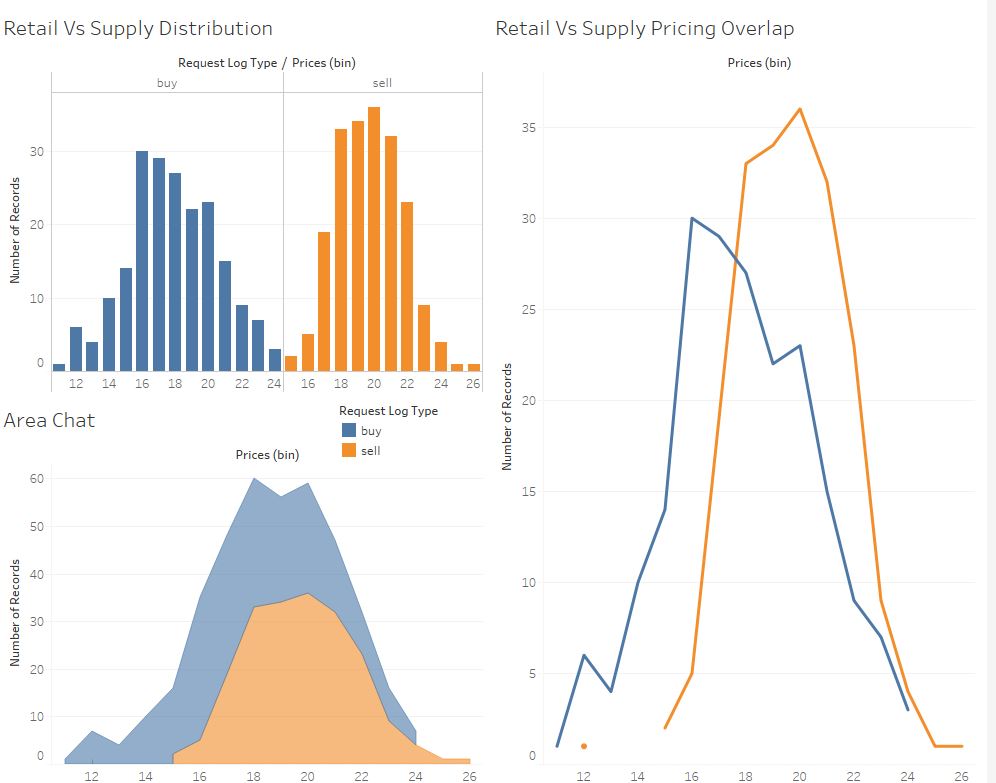


Figure 11.1: Data Analysis Graphs on Pricing

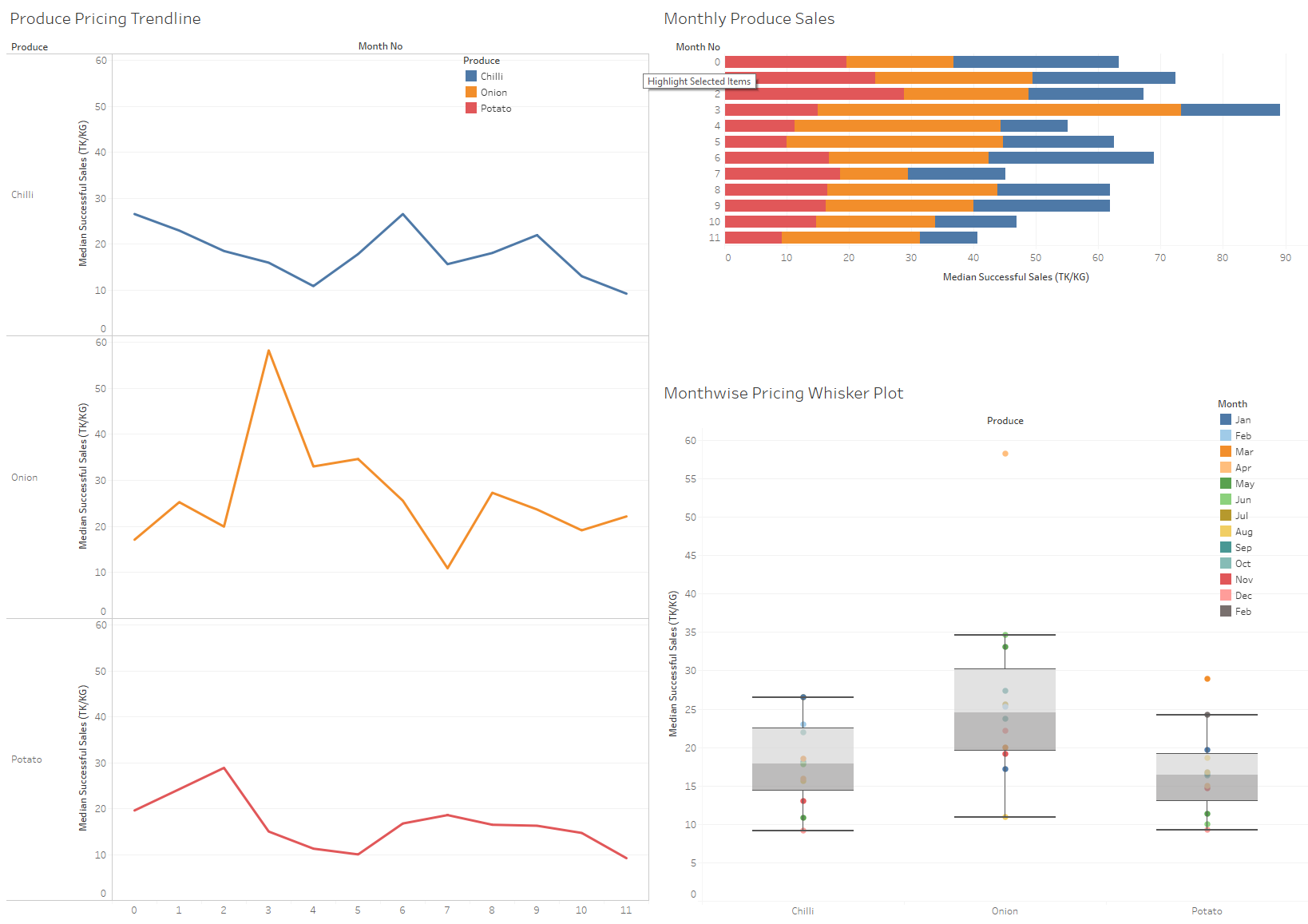


Figure 11.2: Data Analysis Graphs on Harvest

**11.3 Summary**

This chapter has covered the different types of results that we have managed to obtain throughout the course of using this system.

**Chapter 12**

**Conclusion**

We designed a concept application that intends to use technology to streamline the logistical and communication challenges faced in the agricultural industry of our country. It highlights the features farmer and retailer users can take advantage of and emphasizes the optimization of profitability for underpaid agriculture trade.

The proposed solution will truly count as innovation since nothing like this has been implemented before in Bangladesh. Innovation in the agriculture sector has not been scaled using digital technology and hence provides a perfect opportunity for the *‘Amago’* platform to thrive. Production to Retail management in terms of agriculture goods has not been done before. Not only that but management of production yields and logistics has also not been done before at a significant scale. Prediction models for supply and demand will also be a new introduction to the market. These technologies can change the playing field for the good of the consumers and suppliers.

From the beginning of developing this system, our main goal was to develop a fully functional state of the art system which could cater to the masses and it would become a tremendous commercial entity. Our focus was not only to finish the project in due time but to design this system in such a way that it would genuinely be useful in the real world. We wish to continue on our endeavours on this project to actually see it come to life.

**Bibliography**

1. Mohammad H. Mondal, “Challenges And Opportunities Of Sustainable Crop Production In Bangladesh.” Eighth Biennial Agronomy Convention. Bangladesh Society Of Agronomy, 2010.
2. “Bangladesh Agricultural Statistics Yearbook 2012” till “Bangladesh Agricultural Statistics Yearbook 2017”, Bangladesh Bureau of Statistics. DataBD.co
3. iFarmer.Asia, an impact tech startup for innovative online platform which enables anyone to sponsor in farming and livestock in Bangladesh.   
   <https://ifarmer.asia/>
4. Impact Terra. AgroTech Startup with Finance and Farmer Consultancy. <https://www.impactterra.com/>
5. Plantix. Realtime Crop Health, Yield Advice and Feedback.  
   https://plantix.net/en

**Appendix**

**Android Code**

**AmagoItem.java**

package com.sks.amago;

import com.google.gson.annotations.SerializedName;

import java.util.Calendar;

public class AmagoItem {

@SerializedName("uniqueID")

private int uniqueID;

@SerializedName("itemID")

private int itemID;

@SerializedName("dateHarvested")

private String dateHarvested;

@SerializedName("itemPrice")

private int itemPrice;

@SerializedName("itemAmount")

private int itemAmount;

@SerializedName("sellerName")

private String sellerName;

@SerializedName("itemStatus")

private int itemStatus;

public AmagoItem(int id, int n, int w){

uniqueID = id;

itemID = n;

// itemPrice = p;

itemAmount = w;

itemStatus = 1;

dateHarvested = GetDateTaken();

}

public AmagoItem(int id, int n, int w, int p){

uniqueID = id;

itemID = n;

itemPrice = p;

itemAmount = w;

itemStatus = 2;

dateHarvested = GetDateTaken();

}

public AmagoItem(int id, int n, int w, int p, String sn, int is){

uniqueID = id;

itemID = n;

itemPrice = p;

itemAmount = w;

sellerName = sn;

itemStatus = is;

dateHarvested = GetDateTaken();

}

public int getUniqueID() {return uniqueID;}

public int getItemType() {return itemID;}

// public int getItemName() {return getResources().getStringArray(R.array.Produce)[itemID];}

public int getItemPrice() {return itemPrice;}

public int getItemAmount() {return itemAmount;}

public String getSellerName() {return sellerName;}

public int getItemStatus() {return itemStatus;}

public void setUniqueID(int uniqueID) {this.uniqueID = uniqueID;}

public void setItemName(int itemName) {this.itemID = itemName;}

public void setItemPrice(int itemPrice) {this.itemPrice = itemPrice;}

public void setItemAmount(int itemAmount) {this.itemAmount = itemAmount;}

public void setSellerName(String sName) {this.sellerName = sName;}

public void setItemStatus(int status) {this.itemStatus = status;}

public static String GetDateTaken(){

return java.text.DateFormat.getDateTimeInstance().format(Calendar.getInstance().getTime());

}

public String showString(){

return uniqueID+": "+itemID+" "+itemAmount+"kg "+itemPrice+"tk "+itemStatus+"\n";

}

}

**LanguageSelect.java**

package com.sks.amago;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.content.res.Configuration;

import android.content.res.Resources;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.Button;

import android.widget.RadioButton;

import android.widget.RadioGroup;

import android.widget.TextView;

import android.widget.Toast;

import com.sks.amago.Helper.LocaleHelper;

import java.util.Locale;

public class LanguageSelect extends AppCompatActivity {

private RadioGroup radioLangGroup;

private RadioButton radioLangButton;

private Button btnOK;

private TextView textLang;

boolean firsttime;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_language\_select);

radioLangGroup = findViewById(R.id.radiogroupLang);

btnOK = findViewById(R.id.button\_langok);

textLang = findViewById(R.id.textView2);

}

public void LanguagePicked(View view) {

int selectedId = radioLangGroup.getCheckedRadioButtonId();

radioLangButton = (RadioButton) findViewById(selectedId);

String lang = radioLangButton.getText().toString();

SharedPreferences sharedPrefs = getSharedPreferences("com.sks.amago.userprefs", MODE\_PRIVATE);

SharedPreferences.Editor editor = sharedPrefs.edit();

editor.putBoolean("firsttime", false);

editor.apply();

Toast.makeText(this,

lang+" Selected", Toast.LENGTH\_SHORT).show();

if (lang.equalsIgnoreCase(" English")) {

updateView("en");

Locale locale = new Locale("en");

Configuration config = getBaseContext().getResources().getConfiguration();

config.locale = locale;

getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());

}

else {

updateView("bn");

Locale locale = new Locale("bn");

Configuration config = getBaseContext().getResources().getConfiguration();

config.locale = locale;

getBaseContext().getResources().updateConfiguration(config, getBaseContext().getResources().getDisplayMetrics());

}

startActivity(new Intent(LanguageSelect.this, IntroSlider.class));

}

private void updateView(String language) {

Context context = LocaleHelper.setLocale(this, language);

Resources resources = context.getResources();

btnOK.setText(resources.getString(R.string.alright));

textLang.setText(resources.getString(R.string.whatlang));

}

}

**Register.java**

package com.sks.amago;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.Toolbar;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import com.sks.amago.Retrofit.RetrofitClient;

import java.io.IOException;

import okhttp3.ResponseBody;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

import retrofit2.Retrofit;

public class Register extends AppCompatActivity {

EditText editTextFullnamereg;

EditText editTextPhoneNumreg;

EditText editTextPINreg;

Button buttonSignup;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_register);

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

editTextFullnamereg = findViewById(R.id.editText\_fullnamereg);

editTextPhoneNumreg = findViewById(R.id.editText\_phonereg);

editTextPINreg = findViewById(R.id.editText\_PINreg);

buttonSignup = findViewById(R.id.button\_signupreg);

}

public void GotoRegisterDone(View view) {

String fullname = editTextFullnamereg.getText().toString().trim(),

phonenum = editTextPhoneNumreg.getText().toString().trim(),

pinnumber = editTextPINreg.getText().toString().trim();

if (!fullname.isEmpty()) {

if (phonenum.length() == 11) {

if (pinnumber.length() >= 4) {

SharedPreferences sharedPrefs = getSharedPreferences("com.sks.amago.userprefs", MODE\_PRIVATE);

SharedPreferences.Editor editor = sharedPrefs.edit();

editor.putString("amagoPIN", pinnumber);

editor.putString("amagoPhone", phonenum);

editor.putString("amagoFullname", fullname);

editor.apply();

Toast.makeText(this, "Saved!\n" + phonenum + " "

+ pinnumber, Toast.LENGTH\_LONG).show();

Call<ResponseBody> apicall = RetrofitClient.getRetrofitInstance()

.getAPICalls().Register(fullname, pinnumber, phonenum);

apicall.enqueue(new Callback<ResponseBody>() {

@Override

public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {

try {

String s = response.body().string();

Toast.makeText(Register.this, s, Toast.LENGTH\_LONG).show();

} catch (IOException e) {

e.printStackTrace();

}

}

@Override

public void onFailure(Call<ResponseBody> call, Throwable t) {

Toast.makeText(Register.this, t.getMessage(), Toast.LENGTH\_LONG).show();

}

});

this.finish();

} else editTextPINreg.setError("! " + getString(R.string.needpin) + " !");

} else editTextPhoneNumreg.setError("! " + getString(R.string.needphnum) + " !");

} else editTextFullnamereg.setError("! " + getString(R.string.needname) + " !");

}

}

**Login.java**

package com.sks.amago;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.content.res.Resources;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.support.v7.widget.Toolbar;

import android.util.Log;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

import com.sks.amago.Helper.LocaleHelper;

import com.sks.amago.Retrofit.RetrofitClient;

import org.json.JSONException;

import org.json.JSONObject;

import java.io.IOException;

import io.paperdb.Paper;

import okhttp3.ResponseBody;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class Login extends AppCompatActivity {

TextView amagotext;

EditText editTextPhone;

EditText editTextPIN;

Button buttonLogin;

Button buttonSignin;

String userpin, userphone, utoken, userid, username;

Boolean firsttime;

SharedPreferences sharedPrefs;

String tokencheck = " ";

int tokenresponsecode = 0;

String tokencheckresp = " ";

@Override

protected void attachBaseContext(Context newBase) {

super.attachBaseContext(LocaleHelper.onAttach(newBase, "en"));

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

sharedPrefs = getSharedPreferences("com.sks.amago.userprefs", MODE\_PRIVATE);

firsttime = sharedPrefs.getBoolean("firsttime", true);

if(firsttime){

Intent intent = new Intent(Login.this, LanguageSelect.class);

intent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP | Intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(intent);

finish();

}

userphone = sharedPrefs.getString("amagoPhone", "01711499499");

userpin = sharedPrefs.getString("amagoPIN", "0499");

utoken = sharedPrefs.getString("utoken", "tokennotsetpleaselogin");

userid = sharedPrefs.getString("utoken", "tokennotsetpleaselogin");

username = sharedPrefs.getString("utoken", "tokennotsetpleaselogin");

CheckCurrentToken(utoken);

amagotext = findViewById(R.id.textView);

editTextPhone = findViewById(R.id.editText\_phone);

editTextPIN = findViewById(R.id.editText\_PIN);

buttonLogin = findViewById(R.id.button\_login);

buttonSignin = findViewById(R.id.button\_signup);

}

public void GotoMain(View view) {

// userphone = sharedPrefs.getString("amagoPhone", "01711499499");

// userpin = sharedPrefs.getString("amagoPIN", "0499");

Log.i("CheatLoginCreds",userphone + " " + userpin);

Log.i("CheatLoginCreds",editTextPhone.getText().toString() + " " + editTextPIN.getText().toString());

String getphone = editTextPhone.getText().toString();

String getpin = editTextPIN.getText().toString();

Call<ResponseBody> apicall = RetrofitClient.getRetrofitInstance()

.getAPICalls().Login(getphone, getpin);

apicall.enqueue(new Callback<ResponseBody>() {

@Override

public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {

try {

tokencheck = response.body().string();

tokenresponsecode = response.code();

Toast.makeText(Login.this, tokencheck + " - " + tokenresponsecode, Toast.LENGTH\_LONG).show();

Log.i("LOGIN GET TOKEN",tokencheck + " - " + tokenresponsecode);

JSONObject tokenrespjson = new JSONObject(tokencheck);

String tokencheck = tokenrespjson.getString("token");

Log.i("JSONOBJECT TOKEN","" + tokencheck);

SharedPreferences sharedPrefs = getSharedPreferences("com.sks.amago.userprefs", MODE\_PRIVATE);

SharedPreferences.Editor editor = sharedPrefs.edit();

editor.putString("utoken", tokencheck);

editor.apply();

Toast.makeText(Login.this, "Token Saved to Local: "+tokencheck, Toast.LENGTH\_LONG).show();

Log.i("LOGIN SAVE TOKEN","Token Saved to Local: "+tokencheck);

CheckCurrentToken(tokencheck);

}

catch (IOException e) {

e.printStackTrace();

} catch (JSONException e) {

e.printStackTrace();

}

}

@Override

public void onFailure(Call<ResponseBody> call, Throwable t) {

Toast.makeText(Login.this, t.getMessage() +" "+ getString(R.string.crednotmatch), Toast.LENGTH\_LONG).show();

}

});

}

private void CheckCurrentToken(String utoken){

tokencheck = utoken;

Call<ResponseBody> apicall2 = RetrofitClient.getRetrofitInstance()

.getAPICalls().CheckToken(tokencheck);

apicall2.enqueue(new Callback<ResponseBody>() {

@Override

public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {

try {

tokenresponsecode = response.code();

if(tokenresponsecode == 200){

tokencheckresp = response.body().string();

JSONObject tokenlogindeets = new JSONObject(tokencheckresp);

tokenlogindeets = tokenlogindeets.getJSONObject("authData");

userid = tokenlogindeets.getString("id");

username = tokenlogindeets.getString("username");

Toast.makeText(Login.this, "Ok: "+tokencheckresp +" - "+tokenresponsecode

+ "/n"+userid+" - "+username, Toast.LENGTH\_LONG).show();

Log.i("RESPONSE TOKEN OK",""+tokencheckresp+" - "+tokenresponsecode+ "\n"+userid+" - "+username);

SharedPreferences sharedPrefs = getSharedPreferences("com.sks.amago.userprefs", MODE\_PRIVATE);

SharedPreferences.Editor editor = sharedPrefs.edit();

editor.putString("userid", userid);

editor.putString("username", username);

editor.apply();

Intent intent = new Intent(Login.this, MainActivity.class);

intent.addFlags(Intent.FLAG\_ACTIVITY\_CLEAR\_TOP | Intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(intent);

finish();

}

else {

Toast.makeText(Login.this, "Fail: "+tokenresponsecode+" "+getString(R.string.crednotmatch), Toast.LENGTH\_LONG).show();

Log.i("RESPONSE TOKEN FAIL",tokencheck+" "+tokenresponsecode);

}

}

catch (IOException | JSONException e) {

e.printStackTrace();

}

}

@Override

public void onFailure(Call<ResponseBody> call, Throwable t) {

Toast.makeText(Login.this, t.getMessage(), Toast.LENGTH\_LONG).show();

}

});

}

public void GotoRegister(View view) {

startActivity(new Intent(Login.this, Register.class));

}

}

**CallsInterface.java**

package com.sks.amago.Retrofit;

import okhttp3.ResponseBody;

import retrofit2.Call;

import retrofit2.http.Field;

import retrofit2.http.FormUrlEncoded;

import retrofit2.http.GET;

import retrofit2.http.Header;

import retrofit2.http.POST;

import retrofit2.http.Path;

public interface CallsInterface {

@FormUrlEncoded

@POST("api/users/register")

Call<ResponseBody> Register(

@Field("username") String fullname,

@Field("password") String pinnumber,

@Field("phone") String phonenumber

);

@FormUrlEncoded

@POST("api/auth/login")

Call<ResponseBody> Login(

@Field("phone") String phonenumber,

@Field("password") String pinnumber

);

@POST("api/auth/current")

Call<ResponseBody> CheckToken(

@Header("Authorization") String utoken

);

@GET("api/harvest/getHarvest/{id}")

Call<ResponseBody> GetHarvest(@Path("id") String id);

@FormUrlEncoded

@POST("api/harvest/postHarvest")

Call<ResponseBody> PostHarvest(

@Field("userID") String userid,

@Field("itemType") int itemType,

@Field("amount") int amount

);

@FormUrlEncoded

@POST("api/harvest/sellHarvest")

Call<ResponseBody> SellHarvest(

@Field("id") int itemid,

@Field("status") int status,

@Field("price") int price

);

}

**MainActivity.java**

- omitted due to extremely long code ~500 lines 10 pages -

**Server Code**

**server.js**

const express = require('express');

const server = express();

//const cookieParser = require('cookie-parser');

//const session = require('express-session');

//EJS view engine initialization

server.set('view engine', 'ejs');

//Loading static assets for web retail

server.use(express.static(\_\_dirname + '/public'));

// create application/x-www-form-urlencoded parser

const urlencodedParser = express.urlencoded({ extended: false }); // ! Problem has

//Mobile app Routes

const users = require('./routes/api/users');

const auth = require('./routes/api/auth');

const harvest = require('./routes/api/harvest');

const item = require('./routes/api/item');

const sellReq = require('./routes/api/sellReq');

//Web app Routes

const retailUtils = require('./controllers/retailUtils');

const retailUsers = require('./controllers/retailUser');

const retailAuth = require('./controllers/retailAuth');

const retailDashboard = require('./controllers/retailDashboard');

//Logs server changes for debug

const logger = require('./middleware/logger');

server.use(logger);

//Cookie and Sesssion Handling

//server.use(cookieParser());

//server.use(session({ secret: 'this is a secret' }));

//Body Parser Middleware

server.use(express.json());

server.use(express.urlencoded({ extended: false }));

// DB config

const db = require('./config/database');

// DB Connection

/\*

! Connected to amagoProduction DB server

\*/

db.authenticate()

.then(() => console.log('Database connected...'))

.catch(err => console.log('Error: ' + err));

/\*

//Retail webapp routes

server.get('/register', (req, res) => {

res.render('register', { name: 'Register for a retail account' });

});

server.post('/register', urlencodedParser, (req, res) => {

var register = {

first: req.body.firstName,

last: req.body.lastName,

phone: req.body.phone,

password: req.body.inputPassword

};

console.log(register);

res.render('login');

//res.render('home',{

// userValue : student,

// topicHead : 'Student Form'

});

\*/

//use routes

server.use('/api/users', users);

server.use('/api/auth', auth);

server.use('/api/harvest', harvest);

server.use('/api/item', item);

server.use('/api/sellRequest', sellReq);

server.use('/', retailUtils);

server.use('/', retailUsers);

server.use('/', retailAuth);

server.use('/', retailDashboard);

const PORT = process.env.PORT || 5000; // TODO Place this in config file

server.listen(PORT, () => console.log(`Server running on port ${PORT}`));

**api/user.js**

const express = require('express');

const uuid = require('uuid');

const moment = require('moment');

const router = express.Router();

const bcrypt = require('bcryptjs');

//User Model Import

const db = require('../../config/database');

const Users = db.import('../../models/Users\_updated');

//GET all users

router.get(

'/all',

(req, res) => Users.findAll().then(result => res.json(result)) // ! Can't handle large data load, needs to be fixed

);

//GET single user

router.get('/:id', (req, res) =>

Users.findAll({

where: {

userID: req.params.id

}

})

.then(resultID => res.json(resultID))

.catch(err => console.log(err))

);

//Register New Member || Create New User

// ! Need to add Validator

router.post('/register', (req, res) => {

const newMember = {

//userID: uuid.v4(),

//userID: '7',

username: req.body.username,

password: req.body.password,

phone: req.body.phone,

createdAt: moment().format(),

updatedAt: moment().format(),

//email: req.body.email,

userType: '1' // ! userType one set to 1 for farmers

};

// password hashing & saving new user

bcrypt.genSalt(10, (err, salt) => {

bcrypt.hash(newMember.password, salt, (err, hash) => {

if (err) console.log(err);

newMember.password = hash;

console.log(newMember.password);

let {

//userID,

username,

password,

phone,

createdAt,

updatedAt,

userType

} = newMember;

// !Insert into User Table

Users.create({

//userID,

username,

password,

phone,

createdAt,

updatedAt,

userType

})

.then(res.status(200).json({ msg: 'User successfully registered' }))

.catch(err => console.log(err));

});

});

/\*

! Need to check if the user already exists in DB

! and handle accordingly

\*/

});

module.exports = router;

**api/auth.js**

const express = require('express');

const router = express.Router();

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');

//User Model Import

const db = require('../../config/database');

const Users = db.import('../../models/Users\_updated');

//Secret Key access for JWT signature validation

const keys = require('../../config/keys');

//JWT token extractor middleware

const verifyToken = require('../../middleware/jwtMiddleware');

//GET single user

router.post('/loginTest', (req, res) =>

Users.findAll({

where: {

phone: req.body.phone

}

})

.then(result => res.json(result))

.catch(err => console.log(err))

);

// @route POST api/auth/login

// @description Login User | Returns JWT token

// @access Public

router.post('/login', (req, res) =>

Users.findAll({

where: {

phone: req.body.phone

},

raw: true //parsing seqeulize model

})

.then(member => {

//Check for user

if (!member) {

return res.status(404).json({ msg: ' not a registered user' });

}

const password = req.body.password; //password

const memberData = member[0]; // member[0] is an array returned after query

// Check Password

bcrypt.compare(password, memberData.password, (err, isMatch) => {

console.log(err + isMatch + password + memberData.password);

if (isMatch) {

// Passwords match

//Create JWT payload

console.log(memberData.password);

const payload = {

id: memberData.userID,

username: memberData.username

};

//Sign Token

jwt.sign(

payload,

keys.secretOrKey,

{ expiresIn: 3060 },

(err, token) => {

res.json({

token: 'Bearer ' + token

});

}

);

} else {

// Passwords don't match

return res.status(400).json({ msg: 'Incorrect Password' });

}

});

})

.catch(err => console.log(err))

);

router.post('/current', verifyToken, (req, res) => {

jwt.verify(req.token, keys.secretOrKey, (err, authData) => {

if (err) {

res.sendStatus(403);

} else {

res.json({

authData

});

}

});

});

module.exports = router;

**api/harvest.js**

const express = require('express');

const router = express.Router();

const moment = require('moment');

//User Model Import

const db = require('../../config/database');

const Harvest = db.import('../../models/Inventory');

// @route GET api/harvest/allHarvest

// @description Returns all harvests

// @access Public

router.get(

'/allHarvest',

(req, res) => Harvest.findAll().then(result => res.json(result)) // ! Can't handle large data load, needs to be fixed

);

// @route GET api/harvest/getHarvest/:id

// @description gets users harvest list

// @access Public

router.get('/getHarvest/:id', (req, res) =>

Harvest.findAll({

where: {

userID: req.params.id

}

})

.then(harvest => res.json(harvest))

.catch(err => console.log(err))

);

router.post('/sellHarvest', (req, res) => {

Harvest.update(

{ status: req.body.status, price: req.body.price },

{

where: { id: req.body.id }

}

)

.then(msg => res.json('Sell requested'))

.catch(err => console.log(err));

});

// @route POST api/harvest/postHarvest

// @description posts a harvest to db

// @access Public

router.post('/postHarvest', (req, res) => {

const newHarvset = {

userID: req.body.userID,

itemType: req.body.itemType,

amount: req.body.amount,

status: '1',

createdAt: moment().format(),

updatedAt: moment().format()

};

let { userID, itemType, amount, status, createdAt, updatedAt } = newHarvset;

// !Insert into Inventory Table

Harvest.create({

userID,

itemType,

amount,

status,

createdAt,

updatedAt

})

.then(res.status(200).json({ msg: 'Harvest Added' }))

.catch(err => console.log(err));

});

module.exports = router;

**jwtMiddleware.js**

// FORMAT OF TOKEN

// Authorization: Bearer <access\_token>

// Verify Token

module.exports = function verifyToken(req, res, next) {

// Get auth header value

const bearerHeader = req.headers['authorization'];

//const bearerHeader = req.headers('authorization');

// Check if bearer is undefined

if (typeof bearerHeader !== 'undefined') {

// Split at the space

const bearer = bearerHeader.split(' ');

// Get token from array

const bearerToken = bearer[1];

// Set the token

req.token = bearerToken;

// Next middleware

next();

} else {

// Forbidden

res.sendStatus(403);

}

};

**registrationValidator.js**

const Validator = require("validator");

const isEmpty = require("./is-empty");

module.exports = function validateRegisterInput(data) {

let errors = {};

data.name = !isEmpty(data.name) ? data.name : "";

data.email = !isEmpty(data.email) ? data.email : "";

data.role = !isEmpty(data.role) ? data.role : "";

data.password = !isEmpty(data.password) ? data.password : "";

data.password2 = !isEmpty(data.password2) ? data.password2 : "";

if (!Validator.isLength(data.name, { min: 2, max: 30 })) {

errors.name = "Name must be between 2 and 30 characters";

}

if (Validator.isEmpty(data.name)) {

errors.name = "Name field is required";

}

if (Validator.isEmpty(data.email)) {

errors.email = "Email field is required";

}

if (!Validator.isEmail(data.email)) {

errors.email = "Email is invalid";

}

if (Validator.isEmpty(data.role)) {

errors.role = "Role is required";

}

if (!Validator.isLength(data.password, { min: 6, max: 30 })) {

errors.password = "Password must be at least 6 characters";

}

if (Validator.isEmpty(data.password)) {

errors.password = "Password field is required";

}

if (Validator.isEmpty(data.password2)) {

errors.password2 = "Confirm Password field is required";

}

if (!Validator.equals(data.password, data.password2)) {

errors.password2 = "Passwords must match";

}

return {

errors,

isValid: isEmpty(errors)

};

};

**loginValidator.js**

const Validator = require("validator");

const isEmpty = require("./is-empty");

module.exports = function validateLoginInput(data) {

let errors = {};

data.email = !isEmpty(data.email) ? data.email : "";

// data.role = !isEmpty(data.role) ? data.role : '';

data.password = !isEmpty(data.password) ? data.password : "";

if (!Validator.isEmail(data.email)) {

errors.email = "Email is invalid";

}

if (Validator.isEmpty(data.email)) {

errors.email = "Email field is required";

}

// if (Validator.isEmpty(data.role)) {

// errors.role = 'Role is required';

// }

if (Validator.isEmpty(data.password)) {

errors.password = "Password field is required";

}

return {

errors,

isValid: isEmpty(errors)

};

};

**Website Code**

**Login.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta

name="viewport"

content="width=device-width, initial-scale=1, shrink-to-fit=no"

/>

<meta name="description" content="" />

<meta name="author" content="" />

<title>Amago - Login</title>

<link rel="icon" type="image/ico" href="img/amagoLogo.png" />

<!-- Custom fonts for this template-->

<link

href="vendor/fontawesome-free/css/all.min.css"

rel="stylesheet"

type="text/css"

/>

<!-- Custom styles for this template-->

<link href="css/sb-admin.css" rel="stylesheet" />

</head>

<body class="bg-dark">

<nav class="navbar navbar-expand navbar-dark bg-navlight static-top">

<a class="navbar-brand mr-1" href="home"><img src="img/amago.png" position="center" width="70" height="40"></a>

<!-- Navbar -->

<ul class="navbar-nav ml-auto ml-md-0">

</ul>

</nav>

<div class="container">

<div class="card card-login mx-auto mt-5">

<div class="card-header">Login</div>

<div class="card-body">

<form id="login" method="POST" action="/login">

<div class="form-group">

<div class="form-label-group">

<input

type="phone"

id="phone"

name="phone"

class="form-control"

placeholder="Phone number"

required="required"

autofocus="autofocus"

/>

<label for="phone">Phone number</label>

</div>

</div>

<div class="form-group">

<div class="form-label-group">

<input

type="password"

id="inputPassword"

name="inputPassword"

class="form-control"

placeholder="Password"

required="required"

/>

<label for="inputPassword">Password</label>

</div>

</div>

<div class="form-group">

<div class="checkbox">

<label>

<input type="checkbox" value="remember-me" />

Remember Password

</label>

</div>

</div>

<div>

<button

type="submit"

class="btn btn-primary btn-block"

value="Login"

>

Login

</button>

</div>

</form>

<div class="text-center">

<% if(typeof msg != 'undefined')

{ %> <h3><%= msg %></h3> <%}

%>

<a class="d-block small mt-3" href="register"

>Register an Account</a

>

<a class="d-block small" href="forgot-password"

>Forgot Password?</a

>

</div>

</div>

</div>

<!-- Bootstrap core JavaScript-->

<script src="vendor/jquery/jquery.min.js"></script>

<script src="vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<!-- Core plugin JavaScript-->

<script src="vendor/jquery-easing/jquery.easing.min.js"></script>

</body>

</html>

**Registation.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta

name="viewport"

content="width=device-width, initial-scale=1, shrink-to-fit=no"

/>

<meta name="description" content="" />

<meta name="author" content="" />

<title>Amago- Register</title>

<link rel="icon" type="image/ico" href="img/amagoLogo.png" />

<!-- Custom fonts for this template-->

<link

href="/vendor/fontawesome-free/css/all.min.css"

rel="stylesheet"

type="text/css"

/>

<!-- Custom styles for this template-->

<link href="/css/sb-admin.css" rel="stylesheet" />

</head>

<body class="bg-dark">

<nav class="navbar navbar-expand navbar-dark bg-navlight static-top">

<a class="navbar-brand mr-1" href="home"

><img src="img/amago.png" position="center" width="70" height="40"

/></a>

<!-- Navbar -->

<ul class="navbar-nav ml-auto ml-md-0"></ul>

</nav>

<div class="container">

<div class="card card-register mx-auto mt-5">

<div class="card-header">Register for a Retail Account</div>

<div class="card-body">

<form id="register" method="POST" action="/register">

<div class="form-group">

<div class="form-row">

<div class="col-md-6">

<div class="form-label-group">

<input

type="text"

id="firstName"

name="firstName"

class="form-control"

placeholder="First name"

required="required"

autofocus="autofocus"

/>

<label for="firstName">First name</label>

</div>

</div>

<div class="col-md-6">

<div class="form-label-group">

<input

type="text"

id="lastName"

name="lastName"

class="form-control"

placeholder="Last name"

required="required"

/>

<label for="lastName">Last name</label>

</div>

</div>

</div>

</div>

<div class="form-group">

<div class="form-label-group">

<input

type="phone"

id="phone"

name="phone"

class="form-control"

placeholder="Phone Number"

required="required"

/>

<label for="phone">Phone Number</label>

</div>

</div>

<div class="form-group">

<div class="form-row">

<div class="col-md-6">

<div class="form-label-group">

<input

type="password"

id="inputPassword"

name="inputPassword"

class="form-control"

placeholder="Password"

required="required"

/>

<label for="inputPassword">Password</label>

</div>

</div>

<div class="col-md-6">

<div class="form-label-group">

<input

type="password"

id="confirmPassword"

name="confirmPassword"

class="form-control"

placeholder="Confirm password"

required="required"

/>

<label for="confirmPassword">Confirm password</label>

</div>

</div>

</div>

</div>

<div>

<button

type="submit"

class="btn btn-primary btn-block"

value="Register"

>

Register

</button>

</div>

</form>

<div class="text-center">

<a class="d-block small mt-3" href="login">Login Page</a>

<a class="d-block small" href="forgot-password.html"

>Forgot Password?</a

>

</div>

</div>

</div>

</div>

<!-- Bootstrap core JavaScript-->

<script src="/vendor/jquery/jquery.min.js"></script>

<script src="/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<!-- Core plugin JavaScript-->

<script src="/vendor/jquery-easing/jquery.easing.min.js"></script>

</body>

</html>

**Index.ejs**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<meta

name="viewport"

content="width=device-width, initial-scale=1, shrink-to-fit=no"

/>

<meta name="description" content="" />

<meta name="author" content="" />

<title>Amago - Dashboard</title>

<link rel="icon" type="image/ico" href="img/amagoLogo.png" />

<!-- Custom fonts for this template-->

<link

href="/vendor/fontawesome-free/css/all.min.css"

rel="stylesheet"

type="text/css"

/>

<!-- Page level plugin CSS-->

<link

href="/vendor/datatables/dataTables.bootstrap4.css"

rel="stylesheet"

/>

<!-- Custom styles for this template-->

<link href="/css/sb-admin.css" rel="stylesheet" />

</head>

<body id="page-top"

<h1>Logged in as : <%= data.id %></h1>

<nav class="navbar navbar-expand bg-navlight bg-dark static-top">

<a class="navbar-brand mr-1" href="index"

><a class="navbar-brand mr-1" href="home"

><img

src="img/amago.png"

position="center"

width="70"

height="40"/></a

></a>

<button

class="btn btn-link btn-sm text-white order-1 order-sm-0"

id="sidebarToggle"

href="#"

>

<i class="fas fa-bars"></i>

</button>

<!-- Navbar Search -->

<form

class="d-none d-md-inline-block form-inline ml-auto mr-0 mr-md-3 my-2 my-md-0"

>

<div class="input-group">

<input

type="text"

class="form-control"

placeholder="Search for..."

aria-label="Search"

aria-describedby="basic-addon2"

/>

<div class="input-group-append">

<button class="btn btn-primary" type="button">

<i class="fas fa-search"></i>

</button>

</div>

</div>

</form>

<!-- Navbar -->

<ul class="navbar-nav ml-auto ml-md-0">

<li class="nav-item dropdown no-arrow mx-1">

<a

class="nav-link dropdown-toggle"

href="#"

id="alertsDropdown"

role="button"

data-toggle="dropdown"

aria-haspopup="true"

aria-expanded="false"

>

<i class="fas fa-bell fa-fw"></i>

<span class="badge badge-danger">9+</span>

</a>

<div

class="dropdown-menu dropdown-menu-right"

aria-labelledby="alertsDropdown"

>

<a class="dropdown-item" href="#">Action</a>

<a class="dropdown-item" href="#">Another action</a>

<div class="dropdown-divider"></div>

<a class="dropdown-item" href="#">Something else here</a>

</div>

</li>

<li class="nav-item dropdown no-arrow mx-1">

<a

class="nav-link dropdown-toggle"

href="#"

id="messagesDropdown"

role="button"

data-toggle="dropdown"

aria-haspopup="true"

aria-expanded="false"

>

<i class="fas fa-envelope fa-fw"></i>

<span class="badge badge-danger">7</span>

</a>

<div

class="dropdown-menu dropdown-menu-right"

aria-labelledby="messagesDropdown"

>

<a class="dropdown-item" href="#">Action</a>

<a class="dropdown-item" href="#">Another action</a>

<div class="dropdown-divider"></div>

<a class="dropdown-item" href="#">Something else here</a>

</div>

</li>

<li class="nav-item dropdown no-arrow">

<a

class="nav-link dropdown-toggle"

href="#"

id="userDropdown"

role="button"

data-toggle="dropdown"

aria-haspopup="true"

aria-expanded="false"

>

<i class="fas fa-user-circle fa-fw"></i>

</a>

<div

class="dropdown-menu dropdown-menu-right"

aria-labelledby="userDropdown"

>

<a class="dropdown-item" href="#">Settings</a>

<a class="dropdown-item" href="#">Activity Log</a>

<div class="dropdown-divider"></div>

<a

class="dropdown-item"

href="login"

data-toggle="modal"

data-target="#logoutModal"

>Logout</a

>

</div>

</li>

</ul>

</nav>

<div id="wrapper">

<!-- Sidebar -->

<ul class="sidebar navbar-nav">

<li class="nav-item active">

<a class="nav-link" href="index.html">

<i class="fas fa-fw fa-tachometer-alt"></i>

<span>Dashboard</span>

</a>

</li>

<li class="nav-item dropdown">

<a

class="nav-link dropdown-toggle"

href="#"

id="pagesDropdown"

role="button"

data-toggle="dropdown"

aria-haspopup="true"

aria-expanded="false"

>

<i class="fas fa-fw fa-folder"></i>

<span>Inventory</span>

</a>

<div class="dropdown-menu" aria-labelledby="pagesDropdown">

<h6 class="dropdown-header">Login Screens:</h6>

<a class="dropdown-item" href="login">Login</a>

<a class="dropdown-item" href="register">Register</a>

<a class="dropdown-item" href="forgot-password">Forgot Password</a>

<div class="dropdown-divider"></div>

<h6 class="dropdown-header">Other Pages:</h6>

<a class="dropdown-item" href="404">404 Page</a>

<a class="dropdown-item" href="blank">Blank Page</a>

</div>

</li>

<li class="nav-item">

<a class="nav-link" href="charts">

<i class="fas fa-fw fa-chart-area"></i>

<span>Reports</span></a

>

</li>

<li class="nav-item">

<a class="nav-link" href="tables">

<i class="fas fa-fw fa-table"></i>

<span>Transactions</span></a

>

</li>

</ul>

<div id="content-wrapper">

<div class="container-fluid">

<!-- Breadcrumbs-->

<ol class="breadcrumb">

<li class="breadcrumb-item">

<a href="#">Dashboard</a>

</li>

<li class="breadcrumb-item active">Overview</li>

</ol>

<!-- Icon Cards-->

<div class="row">

<div class="col-xl-3 col-sm-6 mb-3">

<div class="card text-white bg-primary o-hidden h-100">

<div class="card-body">

<div class="card-body-icon">

<i class="fas fa-fw fa-comments"></i>

</div>

<div class="mr-5">26 New Messages!</div>

</div>

<a class="card-footer text-white clearfix small z-1" href="#">

<span class="float-left">View Details</span>

<span class="float-right">

<i class="fas fa-angle-right"></i>

</span>

</a>

</div>

</div>

<div class="col-xl-3 col-sm-6 mb-3">

<div class="card text-white bg-warning o-hidden h-100">

<div class="card-body">

<div class="card-body-icon">

<i class="fas fa-fw fa-list"></i>

</div>

<div class="mr-5">11 New Tasks!</div>

</div>

<a class="card-footer text-white clearfix small z-1" href="#">

<span class="float-left">View Details</span>

<span class="float-right">

<i class="fas fa-angle-right"></i>

</span>

</a>

</div>

</div>

<div class="col-xl-3 col-sm-6 mb-3">

<div class="card text-white bg-success o-hidden h-100">

<div class="card-body">

<div class="card-body-icon">

<i class="fas fa-fw fa-shopping-cart"></i>

</div>

<div class="mr-5">123 New Orders!</div>

</div>

<a class="card-footer text-white clearfix small z-1" href="#">

<span class="float-left">View Details</span>

<span class="float-right">

<i class="fas fa-angle-right"></i>

</span>

</a>

</div>

</div>

<div class="col-xl-3 col-sm-6 mb-3">

<div class="card text-white bg-danger o-hidden h-100">

<div class="card-body">

<div class="card-body-icon">

<i class="fas fa-fw fa-life-ring"></i>

</div>

<div class="mr-5">13 New Tickets!</div>

</div>

<a class="card-footer text-white clearfix small z-1" href="#">

<span class="float-left">View Details</span>

<span class="float-right">

<i class="fas fa-angle-right"></i>

</span>

</a>

</div>

</div>

</div>

<!--Purchase New Harvest-->

<div class="card mb-3">

<div class="card-header">

<i class="fas fa-table"></i>

Buy new inventory

</div>

<div class="card-body">

<div class="table-responsive">

<table

class="table table-bordered"

id="dataTable"

width="100%"

cellspacing="0"

>

<thead>

<tr>

<th>Item</th>

<th>Amount (in kg)</th>

<th>Price (BDT)</th>

</tr>

</thead>

<tbody>

<% data.harv.forEach(result => {%>

<tr>

<td><%= result.produce\_name %></td>

<td><%= result.amount %></td>

<td><%= result.price%> BDT</td>

<th> <div class="dropdown-divider"></div>

<a

class="btn btn-primary btn-block"

href="blank"

data-toggle="modal"

data-target="#purchaseModal"

>Buy</a

>

</div></th>

</tr>

<%} ); %>

</tbody>

</table>

</div>

</div>

<div class="card-footer small text-muted">

Updated yesterday at 11:59 PM

</div>

</div>

<!-- Area Chart Example-->

<div class="card mb-3">

<div class="card-header">

<i class="fas fa-chart-area"></i>

Purchase History

</div>

<div class="card-body">

<canvas id="myAreaChart" width="100%" height="30"></canvas>

</div>

<div class="card-footer small text-muted">

Updated yesterday at 11:59 PM

</div>

</div>

</div>

<!-- /.container-fluid -->

<!-- Sticky Footer -->

<footer class="sticky-footer">

<div class="container my-auto">

<div class="copyright text-center my-auto">

<span>Copyright © Amago 2019</span>

</div>

</div>

</footer>

</div>

<!-- /.content-wrapper -->

</div>

<!-- /#wrapper -->

<!-- Scroll to Top Button-->

<a class="scroll-to-top rounded" href="#page-top">

<i class="fas fa-angle-up"></i>

</a>

<!-- Logout Modal-->

<div

class="modal fade"

id="logoutModal"

tabindex="-1"

role="dialog"

aria-labelledby="exampleModalLabel"

aria-hidden="true"

>

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Ready to Leave?</h5>

<button

class="close"

type="button"

data-dismiss="modal"

aria-label="Close"

>

<span aria-hidden="true">×</span>

</button>

</div>

<div class="modal-body">

Select "Logout" below if you are ready to end your current session.

</div>

<div class="modal-footer">

<button

class="btn btn-secondary"

type="button"

data-dismiss="modal"

>

Cancel

</button>

<a class="btn btn-primary" href="login">Logout</a>

</div>

</div>

</div>

</div>

<!-- Product Purchase Modal-->

<div

class="modal fade"

id="purchaseModal"

tabindex="-1"

role="dialog"

aria-labelledby="exampleModalLabel"

aria-hidden="true"

>

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Checkout</h5>

<button

class="close"

type="button"

data-dismiss="modal"

aria-label="Close"

>

<span aria-hidden="true">×</span>

</button>

</div>

<div class="modal-body">

Are you sure you want to buy this item?

</div>

<div class="modal-footer">

<button

class="btn btn-secondary"

type="button"

data-dismiss="modal"

>

Cancel

</button>

<a class="btn btn-primary" type="submit" href="blank">Purchase</a>

</div>

</div>

</div>

</div>

<!-- Bootstrap core JavaScript-->

<script src="vendor/jquery/jquery.min.js"></script>

<script src="vendor/bootstrap/js/bootstrap.bundle.min.js"></script>

<!-- Core plugin JavaScript-->

<script src="vendor/jquery-easing/jquery.easing.min.js"></script>

<!-- Page level plugin JavaScript-->

<script src="vendor/chart.js/Chart.min.js"></script>

<script src="vendor/datatables/jquery.dataTables.js"></script>

<script src="vendor/datatables/dataTables.bootstrap4.js"></script>

<!-- Custom scripts for all pages-->

<script src="js/sb-admin.min.js"></script>

<!-- Demo scripts for this page-->

<script src="js/demo/datatables-demo.js"></script>

<script src="js/demo/chart-area-demo.js"></script>

</body>

</html>

List of Figures

|  |  |  |
| --- | --- | --- |
| **Figure no.** | **Figure Caption** | **Page no.** |
|  |  |  |
| 1.1 | Amago Platform logo | 12 |
| 3.1 | Amago System Architecture | 19 |
| 4.1 | Screenshot of Retail Website | 22 |
| 4.2 | Example of Screenshot of Translation Table | 23 |
| 4.3 | Screenshot of Amago App Pages | 24 |
| 5.1 | Data Relation Diagram | 27 |
| 7.1 | Gantt Chart | 36 |
| 11.1 | Data Analysis Graphs on Pricing | 50 |
| 11.2 | Data Analysis Graphs on Harvest | 51 |
|  |  |  |