# 1. INTRODUCTION

B2B Bidding system is a project which provides a friendly format for buying and selling commodities. Users can search and browse for commodity in this application. This application also allows users to sell products through bidding. This application mainly concentrates on maintaining and managing the deals. This application deals with buying and selling the agri products and value added agri products all over the world.

The application has no formal knowledge is needed for the user to use this system. Thus by this all it proves it is user friendly. B2B bidding System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help farmers in better utilization of resources.

# 2. SYSTEM ANALYSIS

System analysis is a step-by-step process used to identify and develop or acquire the software needto control the processing of specific application. System analysis is a continuing activity the stagesof the systems development. System analysis is the process of gathering and interpreting facts, diagnosing problems and using the facts to improve the system. The outputs from the organizationare traced through the various processing that the input phases through in the organization. This involves gathering information and using structured tools for analysis. A detailed study of this process must be made by various techniques like interviews, questionnaires etc .

It is necessary to have such a good system analysis and then by a project development cycle so thatthe project can be completed in a strictly manner and able to finish with the desired time. The analyst must be so careful about his responsibilities.

## 2.1 EXISTING SYSTEM

The existing system is very limited and is fully manual.

Limitation of existing system

- Need computerized equipment
- Lack of communication
- Limited number of suppliers
- It is time consuming.
- Limited customization

## 2.2 PROPOSED SYSTEM

Advantages of proposed system

- Time saving
- Improved communication

- Better data management
- Increased efficiency

## 2.2 SYSTEM REQUIREMENT SPECIFICATION

A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform. An SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-worked situations.

**Customer requirements**

- The system should be fast
- User friendly
- Maintaining security of data
- Efficiency in data retrieval and management

### 2.3.1 Hardware Specifications

| | |
|---|---|
| Processor | : Intel core i5/i7 |
| Speed | : 3.0GHz or higher |
| System bus | : 64bits |
| Memory | : 8GB RAM |
| Hard disk | : 256 GB |
| Monitor | : 15.6" LCD Monitor |
| Keyboard | : 108 keys Enchanced keyboard |

### 2.3.2 Software Specifications

| | |
|---|---|
| Operating System | : Windows 10 |
| Front End | : HTML, Javascript, CSS |
| Back End | : PYTHON |
| Framework | : Django |
| Database | : MySQL |
| IDE | : Visual Studio Code v 1.67 |
| Technology | : PYTHON |
| Web Server | : Django Server |

### 2.3.3 Front End

**HTML**

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically andoriginally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs,lists, links, quotes and other items.

## Javascript

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

## CSS

Cascading Style Sheets (CSs) is a style _sheet language used for describing the presentation of a document written in a mark-up language. Although most often used to set the visual style of web pagesand user interfaces written in HTML and XHTML. the language can be applied to any XML document,including plain XML. SVG and XUL. and is applicable to rendering in speech, or on other media.

Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobileapplications.

CSS is designed primarily to enable the separation of document content from document presentation, including aspects such as the layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification or presentation characteristics, enable multiple HTML pages to share formatting by specifying on all platforms except Windows.

### 2.3.4 Back End

**Python**

**Python** is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0.Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions.

**Framework**

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Django makes it easier to build better web apps quickly and with less code. It is based on MVT (Model View Template) design pattern. The Django is very demanding due to its rapid development feature. It takes less time to build application after collecting client requirement. Django comes with the following design philosophies:

- Loosely Coupled
- Less Coding
- Don't Repeat Yourself (DRY)

Advantages of Django

- **Object-Relational Mapping (ORM) Support:** Django provides a bridge between the data model and the database engine, and supports a large set of database systems including MySQL, Oracle, Postgres, etc. Django also supports NoSQL database through Django-nonrel fork. For now, the only NoSQL databases supported are MongoDB and google app engine.

- **Multilingual Support:** Django supports multilingual websites through its built-in internationalization system. So you can develop your website, which would support multiple languages.
- **Framework Support:** Django has built-in support for Ajax, RSS, Caching and various other frameworks.
- **Administration GUI:** Django provides a nice ready-to-use user interface for administrative activities.

**Database**

**MYSQL**

MySQL server is powerful database and it requires limited programs and used has back end. It supports GUl and more application is developed by help this server. Collection of tables which holds the data is called database. A beginner can create their own database by click home page. ships with no GUI tools to administer MYSQL databases or manage data contained within the databases. Users may use the included command line tools or install MySQL Workbench via a separate download. Many third party GUI tools are also available.

## 2.4 FEASIBILITY ANALYSIS

A feasibility study is an evaluation and analysis of the potential of the proposed project which is basedon extensive investigation and research to give full comfort to the decision makers. Feasibility studies aim to objectively and rationally uncover the strength and weakness of existing business of proposed venture, opportunities and threads as presented by the environment, the resources required to carry through, and ultimately the process for success. In its simplest terms, the two criteria to judge feasibilityare cost required and value to attain. As such, a well-designed feasibility study should provide a historical background of the business or project, description of the product or service, accounting statements, details of the operations and management, marketing research and policies, financial data, legal requirements and tax obligations.

The two aspects in the feasibility study are:

### Technical Feasibility

The technical feasibility centres on the existing system and what extend it can support the proposed addition. The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. The minimum requirements of the system are met by average user. The developer system hasá modest technical requirement as only minimal or null changes are required for implementing system.

Normally associated with the technical feasibility includes:

- Development risk
- Resource availability
- Technology

The proposed system can work without any additional hardware or software support other than the computer system and networks. So, I analysed that the proposed system is much more technically feasible than other systems when comparing with the benefits of the new system.

### Operational Feasibility

Operational feasibility is a measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirementsidentified in the requirements analysis phase of system development.

## 2.5 DATA FLOW DIAGRAM (DFD)

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system. It differs from the flowchart as it shows the data flow instead of the control flow of the program. A data flow diagram can also be used for the visualization of data processing (structured design).

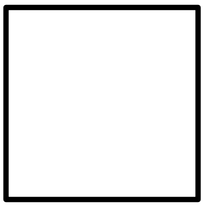
Data Flow Diagrams were invented by Larry Constantine, the original developer of structured design based on Martin and Estrin’s “Data Flow Graph” model of computation.

Data Flow Diagrams (DFD) are one of the three essential perspectives of Structured System Analysis and Design Method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system’s evolution. With a data flow diagram, users are able to visualize how the system will operate, what the system will accomplish and how the system will be implemented. The old system’s data flow diagram can be drawn up and compared with the new system’s data flow diagram can be drawn comparisons to implement a more efficient system. Data flow diagrams can be used to provide the end user with physical idea of where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to report. How many system is developed can be determined through a data flow diagram.

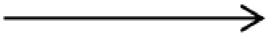
Developing a data flow diagram helps in identifying the transaction data in the data model. There are different notations to draw data flow diagrams, defining different visual representation for process, data stores, data flow and external entities. The first step is to draw a Data Flow Diagram (DFD) also known as “bubble chart” has the purpose of clarifying system requirements and identifying major transformation that will become program in system design. So, it is starting point of the design phase that functionally decompose the requirements specification down to the lowest level of details. DFD consists of series of bubbles joined by lines. The bubbles represent data transformation and the lines represent data flow in the system.

## DFD Symbols

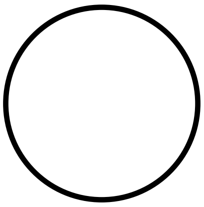
- Square – Defines source or destination of system



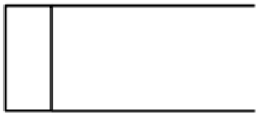
-  Data flow – Identifies data flow circle



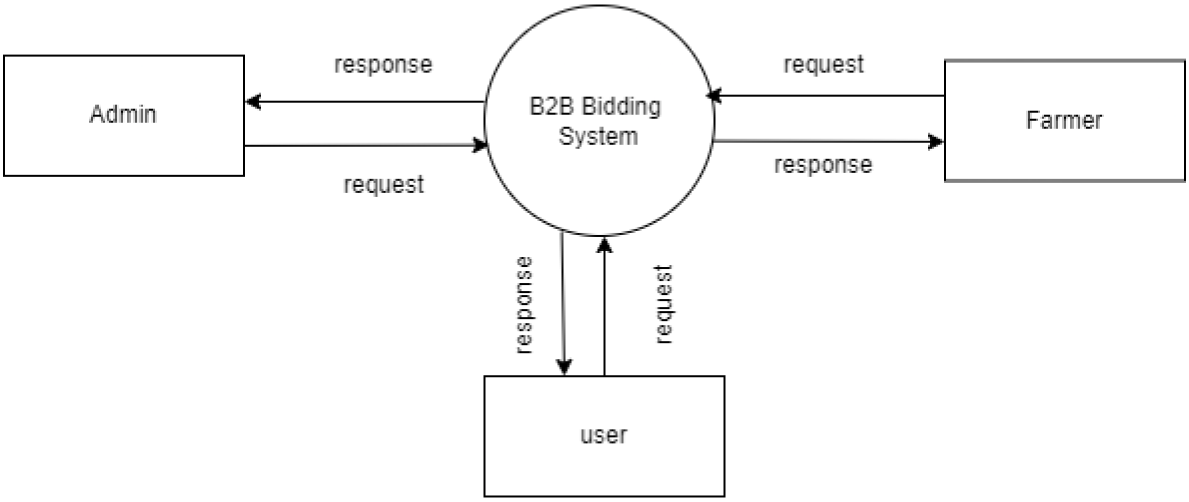
-  Bubble – Represents a process that transforms incoming data to outgoing data



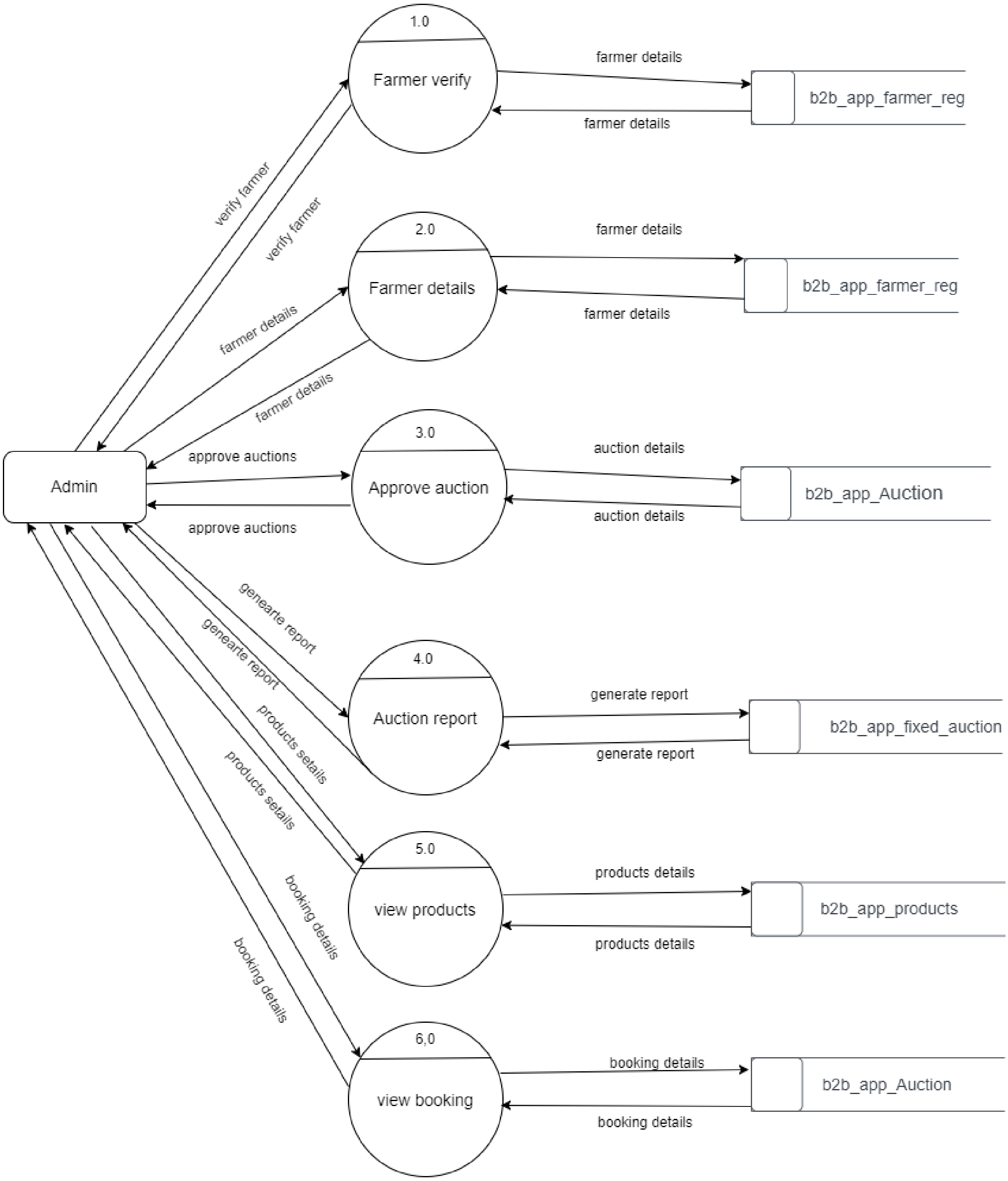
-  Open Rectangle – Data store

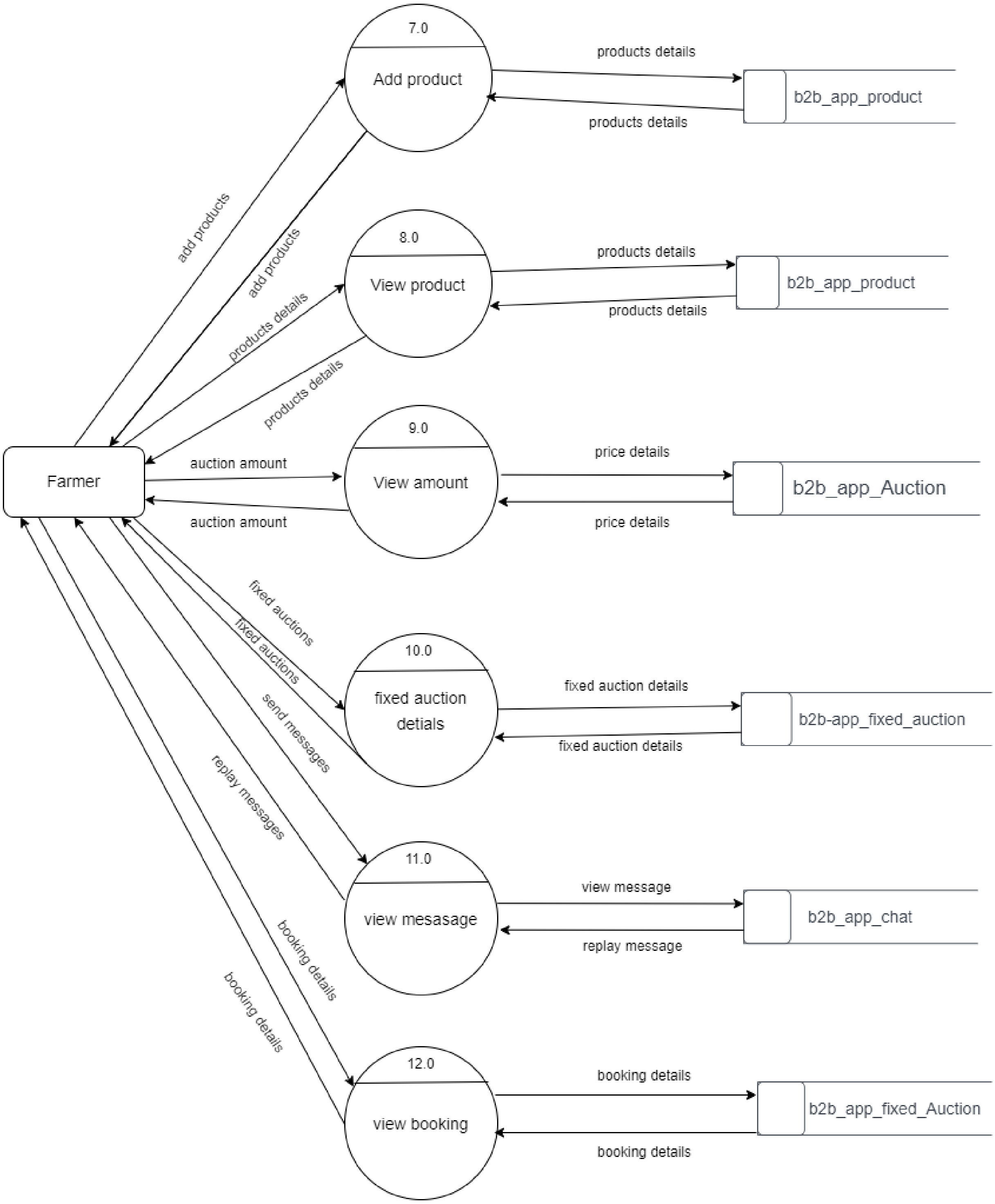


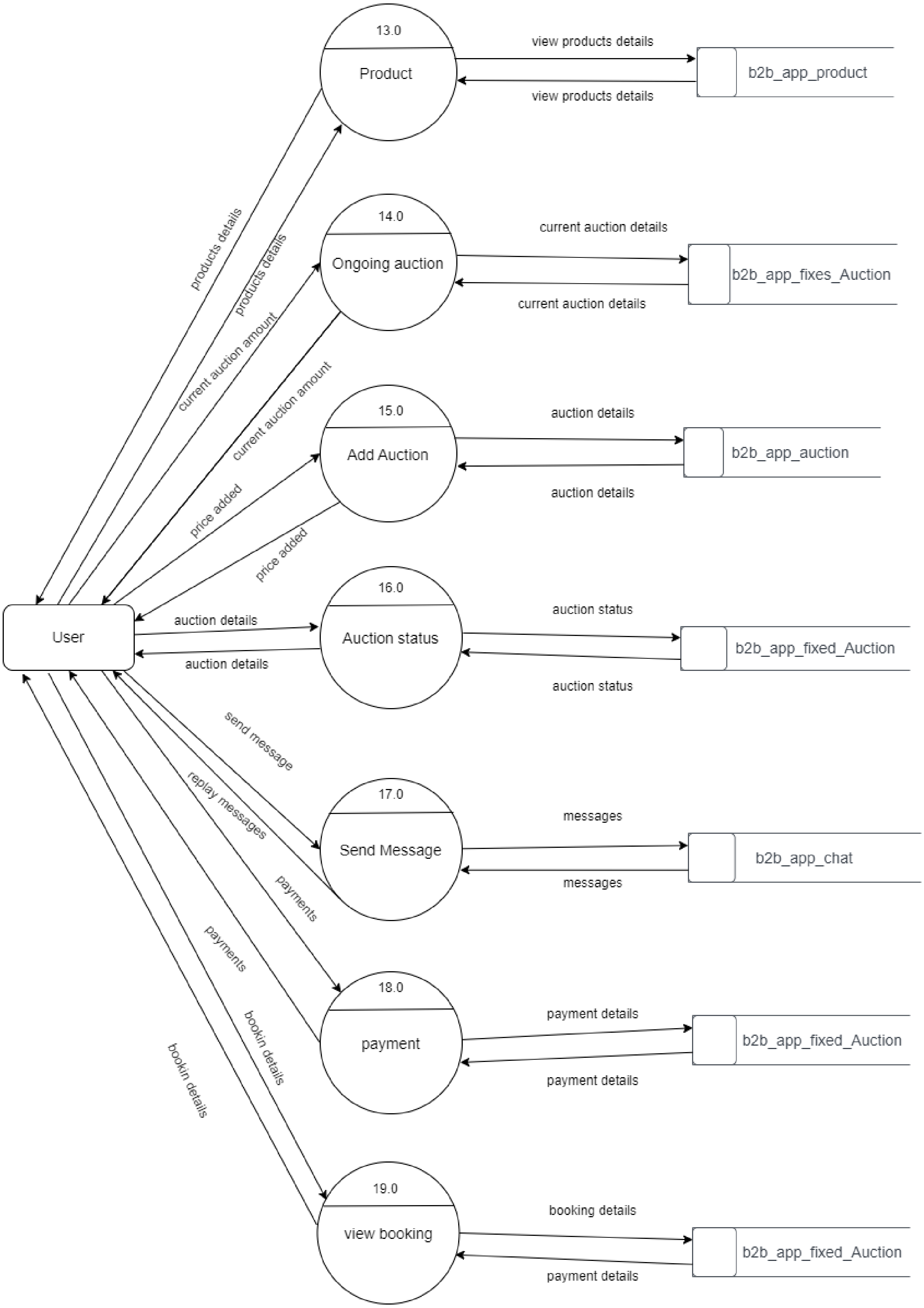
Level **0**


Admin
response
request
B2B Bidding System
request
response
Farmer
response
request
user

Level 1


1.0
Farmer verify
farmer details
farmer details
b2b_app_farmer_reg
verify farmer
verify farmer
2.0
Farmer details
farmer details
farmer details
b2b_app_farmer_reg
farmer details
farmer details
3.0
Approve auction
auction details
auction details
b2b_app_Auction
Admin
approve auctions
approve auctions
genearte report
genearte report
4.0
Auction report
generate report
generate report
b2b_app_fixed_auction
products setails
products setails
5.0
view products
products details
products details
b2b_app_products
booking details
booking details
6,0
view booking
booking details
booking details
b2b_app_Auction


7.0
Add product
products details
b2b_app_product
products details
add products
add products
8.0
View product
products details
b2b_app_product
products details
products details
products details
9.0
View amount
Farmer
auction amount
auction amount
price details
b2b_app_Auction
price details
fixed auctions
fixed auctions
10.0
fixed auction detials
fixed auction details
b2b-app_fixed_auction
fixed auction details
send messages
replay messages
11.0
view mesasage
view message
b2b_app_chat
replay message
booking details
booking details
12.0
view booking
booking details
b2b_app_fixed_Auction
booking details


13.0
Product
view products details
view products details
b2b_app_product
14.0
Ongoing auction
current auction details
current auction details
b2b_app_fixes_Auction
15.0
Add Auction
auction details
auction details
b2b_app_auction
16.0
Auction status
auction status
auction status
b2b_app_fixed_Auction
17.0
Send Message
messages
messages
b2b_app_chat
18.0
payment
payment details
payment details
b2b_app_fixed_Auction
19.0
view booking
booking details
payment details
b2b_app_fixed_Auction
User
products details
products details
current auction amount
current auction amount
price added
price added
auction details
auction details
send message
replay messages
payments
payments
bookin details
bookin details

# 3.SYSTEM DESIGN

## 3.1 INPUT DESIGN

The quality of the system input determines the quality of the system output. Input specification describes the manner in which data enter the system for processing. Input design features can ensure the reliability of the system and produce result from accurate data, or they can result in the production or erroneous information. The input design also determines whether the user can interact efficiently with the system.

In our system almost, all inputs are being taken from the databases. To provide adequate inputs we have to select necessary values from the databases and arrange it to the appropriate controls.

## 3.2 OUTPUT DESIGN

One of the important features of an information system for users is the output produces. Output is the information delivered to users through the information system. Without quality of the output, the entire system appears to be unnecessary that users will avoid using it. Users generally merit the system solely by its output. In order to create the most useful output possible. One works closely with the user though an interactive process. until the result is considered to be satisfactory.

**Admin**

Admin can verify farmer and he can view and delete the details of farmer. He can approve bid amount set by the user. Admin can generate the auction report by date wise and he has control overall the system. He can view the product details and booking.

**Farmer**

Farmer is a part of b2b bidding system. In this module farmer can add products and he can view and delete the details of product. Farmer can see the auction amount set by the user and he can also fix the auctions with message. Farmer can communicate with the user and view booking.

**User**

User can view product details and add bid amount. He can see the status of farmer bid fixed and user can also see the other user added prices. User can chat with farmer and view message replay of farmer. He can see the current auction amount of the product in ongoing auction. User can make payments and view booking.

## 3.3 DATABASE DESIGN

Table name: b2b_app_usertype
Description: User details
Primary key: id
Foreign key: user_id

| Field | Type | Size | Description |
|---|---|---|---|
| id | bigint | | Id of user |
| type | varchar | 50 | type of user |
| user_id | bigint | | Id of user |

Table name: b2b_app_product

Description: products details

Primary key: id

Foreign key: farmer_id

| Field | Type | Size | Description |
|---|---|---|---|
| id | bigint | | Id of product |
| name | varchar | 150 | Name of product |
| image | varchar | 150 | image of product |
| desc | varchar | 150 | Discripion of product |
| quantity | int | | Quantity of product |
| price | int | | Price of product |
| auction_date | varchar | 150 | Auction date |
| delivary_date | varchar | 150 | Delivary date |
| Farmer_id | bigint | | id of farmer |

Table name: b2b_app_fixed_Auction

Description: fixing auction

Primary key: id

Foreign key: customer_id

Foreign key: farmer_id

| Field | Type | Size | Description |
|---|---|---|---|
| id | bigint | | Id of fixing |
| product_name | varchar | 150 | Name of product |
| message | varchar | 150 | message |
| price | int | | Price of product |
| customer_id | bigint | | Id of customer |
| date | varchar | 150 | Auction date |
| farmer_id | bigint | | Id of farmer |
| status | varchar | 100 | Status of auction |
| auction_price | varchar | 100 | Price of auction |
| com_date | varchar | 100 | Date of current day |
| payment | varchar | 100 | payments |

Table name: b2b_app_farmer_reg

Description: farmer details

Primary key: id

Foreign key: user_id

| Field | Type | Size | Description |
|---|---|---|---|
| id | bigint | | Id of farmer |
| address | varchar | 150 | Address of farmer |
| phonenumber | varchar | 150 | Number of farmer |
| user_id | bigint | | Id of user |

Table name: b2b_app_customer_reg

Description: customer details

Primary key: id

Foreign key: user_id

| Field | Type | Size | Description |
|---|---|---|---|
| id | bigint | | Id of custmer |
| address | varchar | 150 | Address of custmer |
| phonenumber | varchar | 150 | Number of custmer |
| user_id | bigint | | Id of user |

Table name: b2b_app_chat

Description: chat details

Primary key: id

Foreign key: farmer_id

Foreign key: product_id

Foreign key: user_id

| Field | Type | Size | Description |
|---|---|---|---|
| id | bigint | | Id of chat |
| message | varchar | 150 | messages |
| status | varchar | 150 | status |
| farmer_id | bigint | | Id of farmer |
| product_id | bigint | | Id of product |
| user_id | bigint | | Id of user |
| replay | varchar | 150 | Replay of messages |

Table name: b2b_app_auction

Description: Auction details

Primary key: id

Foreign key: faremer_id

Foreign key: product_id

Foreign key: customer_id

| Field | Type | Size | Description |
|---|---|---|---|
| id | bigint | | Id of User |
| price | int | | Price of product |
| farmer_id | bigint | | Id of farmer |
| product_id | bigint | | Id of product |
| status | varchar | 150 | status |
| customer_id | bigint | | Id of Customer |
| message | varchar | 150 | message |
| product_name | varchar | 150 | Name of product |
| Admin_status | varchar | 150 | Status of admin |

# 4.SYSTEM TESTING AND IMPLEMENTATION

## 4.1 SYSTEM TESTING

Testing is the process of examining the software to compare the actual behaviour with that of the excepted behavior. The major goal of software testing is to demonstrate that faults are not present. In order to achieve this goal, the tester executes the program with the intent of finding errors. Though testing cannot show absence of errors but by not showing their presence it is considered that these arenot present.

System testing is defined as the process by which one detects the defects in the software. Any software development organization or team has to perform several processes. Software testing is one among them. It is the final opportunity of any programmer to detect and rectify any defects that may have appeared during the software development stage. Testing is a process of testing a program with the explicit intention of finding errors that makes the program fail. In short system testing and quality assurance is a review in software products and related documentation for completion, correctness, reliability and maintainability.

System testing is the first stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct and thegoal will be successfully achieved. A series of testing are performed for the proposed system before the proposed system is ready for user acceptance testing.

The testing steps are:

- Unit testing
- Integration testing
- Validation testing
- Output testing
- Acceptance testing

System Testing provides the file assurance that software once validated mast combined with all other system elements. System testing verifies whether all elements nave been combined properly and that overall system function and performance is achieved. FA the integration of modules, the validation test was carried out over the system. It was that all the modules work well together and meet the overall system function and performance.

**Unit Testing**

Unit testing is carried out screen-wise, each screen being identified as an object. Attention is diverted to individual modules, independently to one another to locate errors. This has enabled the detection oferrors in coding and logic.

Various test cases are prepared. For each module these test cases are implemented and it is checked whether the module is executed as per the requirements and outputs the desired result. In this test eachservice input and output parameters are checked.

In unit testing:

1. Module interface was tested to ensure that information properly flows into and out of theprogram under test.
2. Boundary condition was tested to ensure that module operates properly at boundariesestablished to limit or restrict processing.
3. All independent paths through the control structures were executed to ensure that all statementsin the modules have been executed at least once.
4. Error handling paths were also tested.

## Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing.

Unit tested module were taken and a single program structure was built that has been dictated by the design. Incremental integration has been adopted here.

The modules are tested separately for accuracy and modules are integrated too.th tn. using bottom up integration i.e., by integrating from moving from bottom to the top the system is checked and errors found during integration are rectified.

The entire software was developed and tested in small segments, where errors were easy to locate andrectify. Program builds (group of modules) were constructed corresponding to the successful testing of user interaction, data manipulation analysis, and display processing and database management.

## Validation Testing

Validation testing is done to ensure complete assembly of the error-free software. Validation can be termed successful only if it functions in manner. Reasonably expected by the student under validation is alpha and beta testing. The student-side validation is done in this testing phase. It is checked whetherthe data passed to each student is valid or not. Entering incorrect values does the validation testing and it is checked whether the errors are being considered. Incorrect values are to be discarded. The errors are rectified.

In “University result portal" verifications are done correctly. So, there is no chance for users to enter incorrect values. It will give error messages by using different validations. The validation testing is done very clearly and found it is error free.

### Output Testing

After performing the validation testing the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format.

The output format on the screen was found to be correct as the format was designed in the system design phase according to the user needs. For the hard copy also, the output comes out as specified requirement by the user. Hence output testing does not result in any Correction in the system. output This project is developed based on the user choice. It is user friendly. The output format is very clear to user. Output testing is done on Smart builders correctly.

### Acceptance Testing

Acceptance involves running a suite of tests on the completed system. Each individual test, known as a Case, exercise particular operating condition of the operating condition of users environment or feature of the system, and will result in a pass fail.

## 4.2 SYSTEM IMPLEMENTATION

The implementation is the final state and it is an important phase. It involves the invalid programming system testing, user training and the operational running of developed proposed system that constitutes the application subsystems. A major task of preparing for implementation is education of users, which should really have been taken place much carrier in the project when they were belong involved in the investigation and design work. During the implementation phase system actually take physical shape. In order to develop a system implemented planning is very essential.

The implementation phase of the software development is concerned with translating design specification into source code. The user tests the developed system and changes are made according to their needs. Our system has been successfully implemented.

Before implementation several tests have been conducted to ensure that no errors are encountered during the operation. The implementation phase ends with an evaluation of the system after placing into the operation for a period of time.

The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from old system to new system. The system can be implemented only after testing is done and is found to be working to specifications. The implementation stage is a systems project in its own right.

The implementation stage involves following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of method to achieve change over
- Evaluation of the changeover method

In the case of this project all the screens are designed first. For making it to be executable, codes are written on each screen and performs the implementation by creating the database and connecting to the server. After that the system, is Checked, whether it performs all the transactions Correctly. Then databases are cleared and made it to be usable to the technicians.

# 5.SECURITY TECHNOLOGIES & POLICIES

The protection of computer-based resources that includes hardware, software, data procedures and people against unauthorized use or natural. Disaster is known as System Security. System Security can be divided into four related issues:

- Security
- Integrity
- Privacy
- Confidentiality

**SYSTEM SECURITY** refers to the technical innovations and procedures applied to the hardware and operation systems to protect against deliberate or accidental damage from a defined threat.

**DATA SECURITY** is the protection of data from loss, disclosure, modification and destruction.

**SYSTEM INTEGRITY** refers to the power functioning of hardware and programs, appropriate physical security and safety against external threats such as caves dropping and wiretapping.

**PRIVACY** defines the rights of the user or organizations to determine what information they are willing to share with or accept from others and how the organization can be protected against unwelcome, unfair or excessive dissemination of information about it.

**CONFIDENTIALITY** is a special status given to sensitive information in a database to minimize the possible invasion of privacy. It is an attribute of information that characterizes its need for protection.

**SECURITY IN SOFTWARE** System security refers to various validations on data in form of checks and controls to avoid the system from failing. It is always important to ensure that only valid data is entered and only valid operations are performed on the system.

The system employees two types check and controls:

**CLIENT-SIDE VALIDATION** Various client-side validations are used to ensure on the client side that only valid data is entered. Client-side validation saves server time and load to handle invalid data. Some checks imposed are:

❖ Forms cannot be submitted without filling up the mandatory data so that manual mistakes of submitting empty fields that are mandatory can be sorted out at the client side to save the server time and load.

❖ Tab-indexes are set according to the need and taking into account the ease of user while working with the system.

**SERVER-SIDE VALIDATION** Some checks cannot be applied at client side. Server-side checks are necessary to save the system from failing and intimating the user that some invalid operation has been performed or the performed operation is restricted. Some of the serverside checks imposed is:

❖ Server-side constraint has been imposed to check for the validity of primary key and foreign key. A primary key value cannot be duplicated. Any attempt to duplicate the primary value results into a message intimating the user about those values through the forms using foreign key can be updated only of the existing foreign key values.

❖ User is intimating through appropriate messages about the successful operations or exceptions occurring at server side.

❖ Various Access Control Mechanisms have been built so that one user may not agitate upon another. Access permissions to various types of users are controlled according to the organizational structure. Only permitted users can log on to the system and can have access according to their category. User name, passwords and permissions are controlled over the server side.

❖ Using server-side validation, constraints on several restricted operations are imposed.

# 6.MAINTENANCE

Software maintenance is the modification of a software product delivery to correct faults, to improve performance or other attributes. Maintenance is the case with which a program can be corrected if any error is encountered, adapted if its environment changes or enhanced if the customer desires a change in requirement. Maintenance follows conversation to extend that changes are necessary to maintain satisfactory operations relative to changes in the user's environment.

Maintenance often includes minor enhancements or corrections to problems that surface in the system's operation. Maintenance is also done based on fixing the problems reported, changing the interface with other software or hardware enhancing the software.

## CATEGORIES OF MAINTENANCE

### Corrective Maintenance

Corrective maintenance is the most commonly used maintenance approach, but it is easy to see its limitations. When equipment fails, it often leads to downtime in production, and sometimes damages other parts. In most cases, this is expensive. Also, if the equipment needs to be replaced, the cost of replacing it alone can be substantial. Reliability of systems maintained by this type of maintenance is unknown and cannot be measured. Corrective maintenance is possible since the consequences of failure or wearing out are not significant and the cost of this maintenance is not great.

### Perfective Maintenance

Modification of a software product alter delivery to improve performance or maintainability. This term is used to describe changes undertaken to expand the existing requirements of the system. A successful piece or software lends to be subjected to a the Succession of changes resulting in an increase in us requirements. Expansion requirements can take the form enhancement of existing system functionality and improvement in computational efficiency**.**

**Adaptive Maintenance**

Modification of a software product performed after delivery to keep a are product usable a changed or changing environment. Adaptive maintenance includes any work initiated as a consequence of moving the software to a different hardware or software platform. It is a change driven by the need to accommodate modifications in the environment of software system. The environment in this context refers to the totality of all conditions and influences which act from outside upon the system. A change to the whole or part of this environment will Warrant a corresponding modification of the software.

**Preventive Maintenance**

Preventive maintenance is a schedule of planned maintenance actions aimed at the prevention of breakdowns and failures. The primary goal of preventive maintenance is to prevent the failure of equipment before it actually occurs. It is designed to preserve and enhance equipment reliability by replacing worn components before they actually fail. Preventive maintenance activities include equipment checks, partial or complete overhauls at specified periods.

Long-term benefits of preventive maintenance include:

- Improved system reliability.
- Decreased cost of replacement.
- Decreased system downtime.

## 7. SCOPE FOR FUTURE ENHANCEMENT

We can consider integration with AI and machine learning Incorporating AI and machine learning technologies can help a B2B bidding system optimize bids and improve targeting. For example, machine learning algorithms can analyze bid data to identify patterns and make recommendations for future bids**.** Providing customization options can help users tailor their bidding strategies to their specific needs. This could include the ability to set custom bidding rules, adjust bid amounts based on time of day or day of the week, or prioritize certain types of bids over others. Providing better reporting and analytics can help users better understand the performance of their bids and make data-driven decisions. This could include visualizations of bid data, custom reports, and integration with third-party analytics tools.

## 8.CONCLUSION

The proposed system B2B bidding is a powerful tool that can help streamline the bidding process, increase efficiency, and drive better results for businesses. By enabling users to bid on contracts ,products, or services in a competitive marketplace, a B2B bidding system helps ensure that the best offer is selected for each transaction. Well- designed and well-implemented B2B bidding system can help business of all sizes time and money, increase competitiveness, and drive growth.

# 9.BIBILOGRAPHY

1. Jalotte, Pankaj. " **SOFTWARE ENGINEERING",** Third Edition, Spring Publishers, Narosa Publishing House.

2. Ramakrishnan, Raghu. and_ Gehrke, Johannes. **DATABASE MANAGEMENT SYSTEM**", Third Edition, McGraw-Hill Professional Publication.

3. Rob, Peter. and Coronel, Carlos. "**DATABASE SYSTEMS DESIGN AND MANAGEMENT",** Fifth Edition, Thomson Learning Publication.

4. Pressman, Roger S. "**SOFTWARE ENGINEERING**", Fifth Edition, McGraw-Hill Publication.

5. Bayross, Ivan. "**SQL/MySQL**" Second Revised Edition, BPB Publications

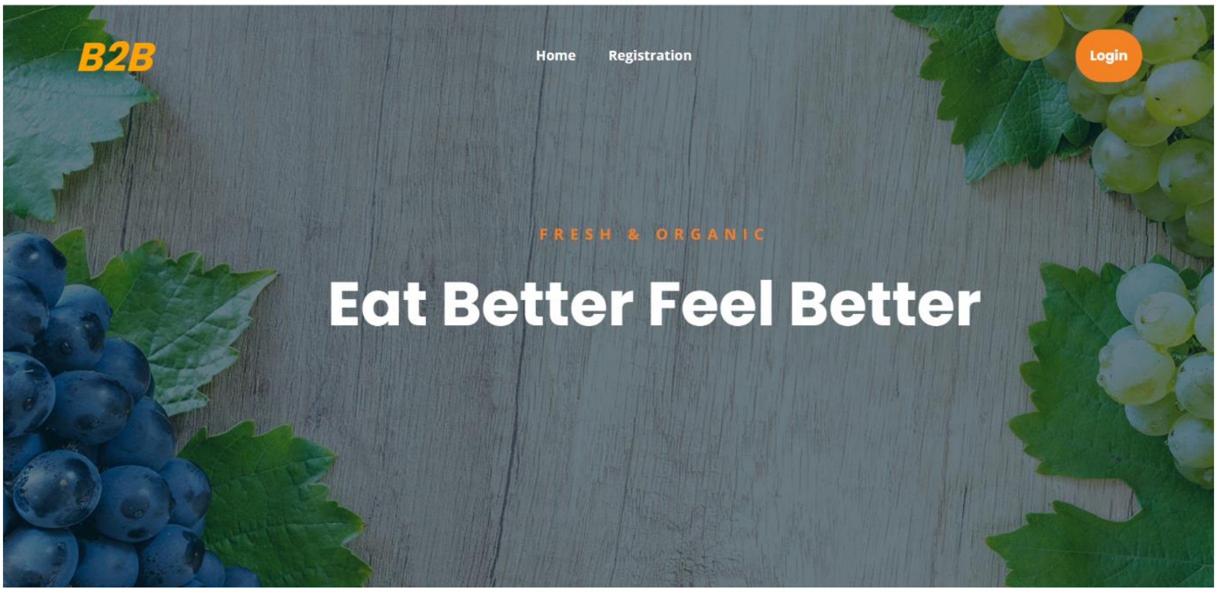
6. https://www.google.com/amp/s/www.geeksforgeeks.org/best-books-to-learn-python-for-beginners-and-experts-in-2019/amp/

7. https://w3schools.invisionzone.com/topic/51932-how-to-do-a-auction-webpage/

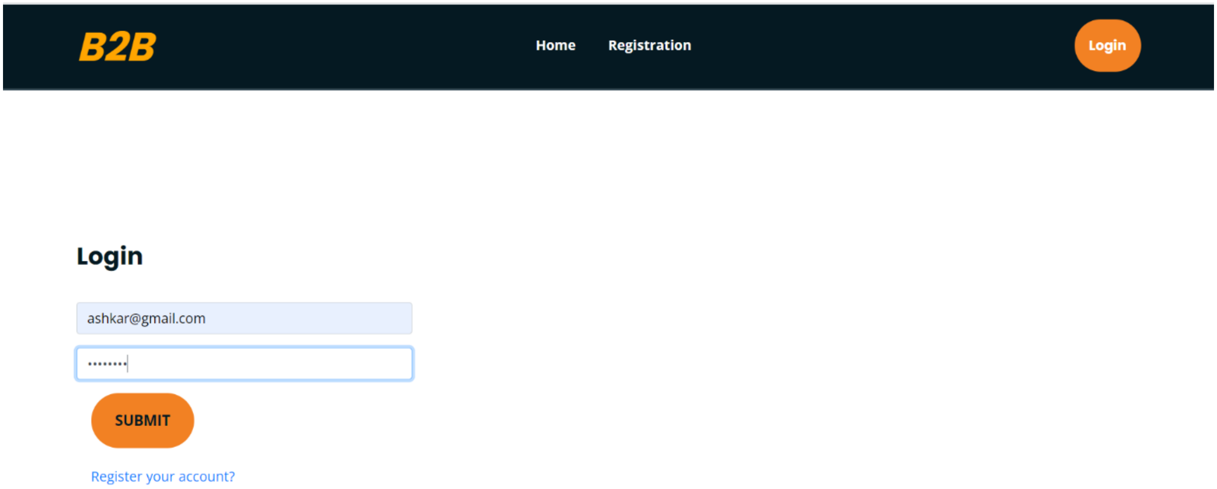
# 10. APPENDIX

## 10.1 Screenshots

### Home page



### Login page

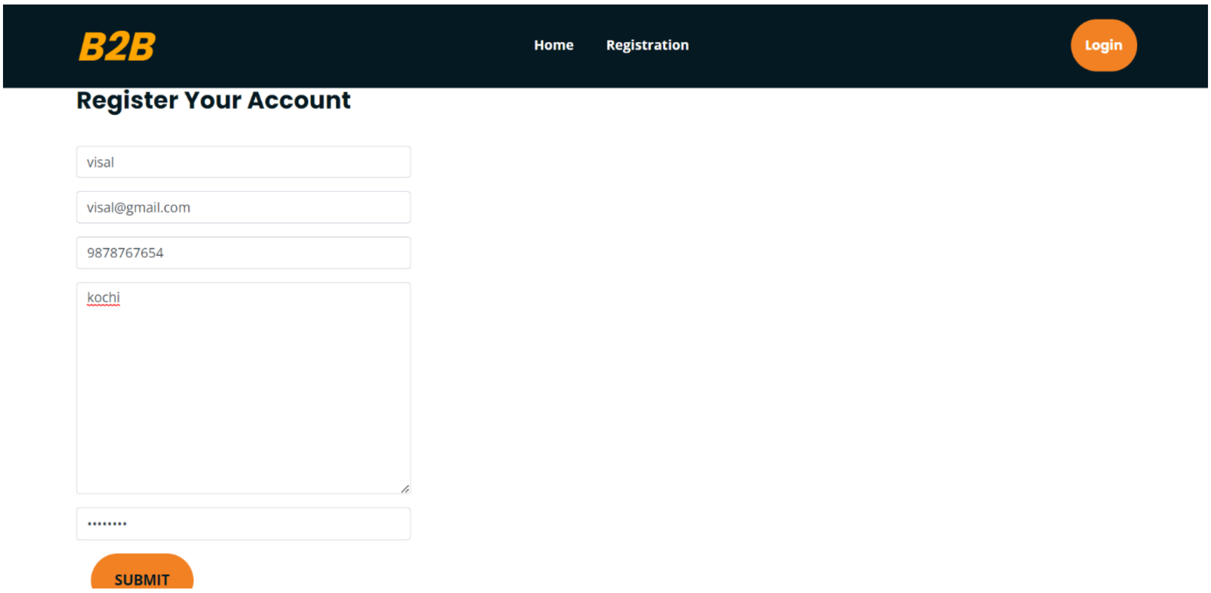


Login

ashkar@gmail.com

••••••••

## Farmer registration



Register Your Account

visal

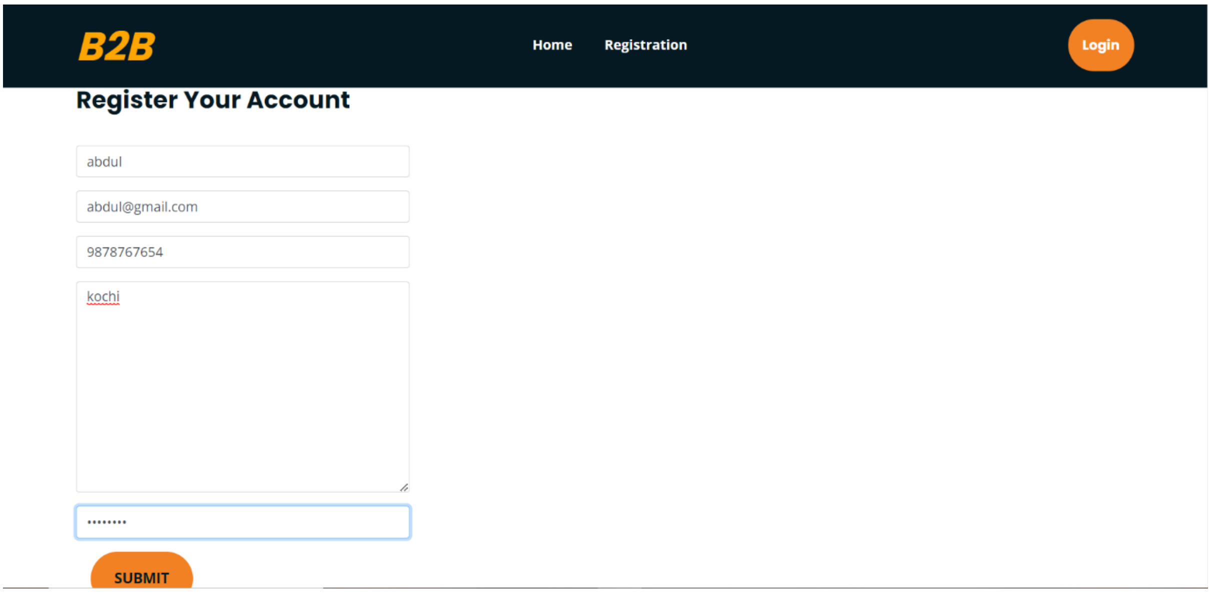
visal@gmail.com

9878767654

kochi

••••••••

## User registration



Register Your Account

abdul

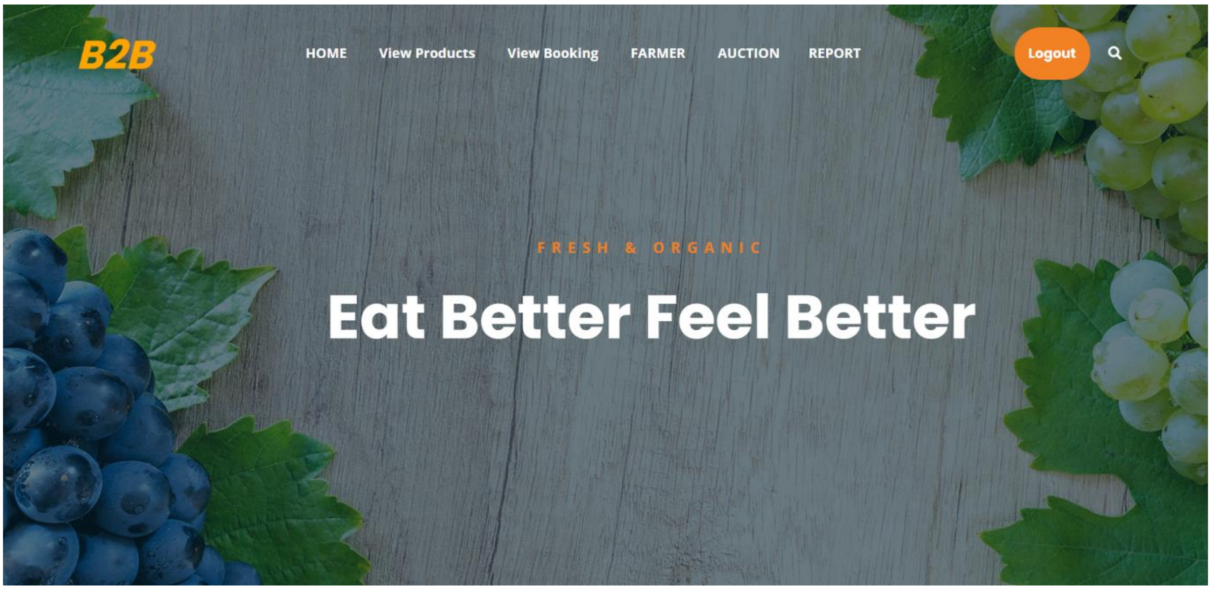
abdul@gmail.com

9878767654

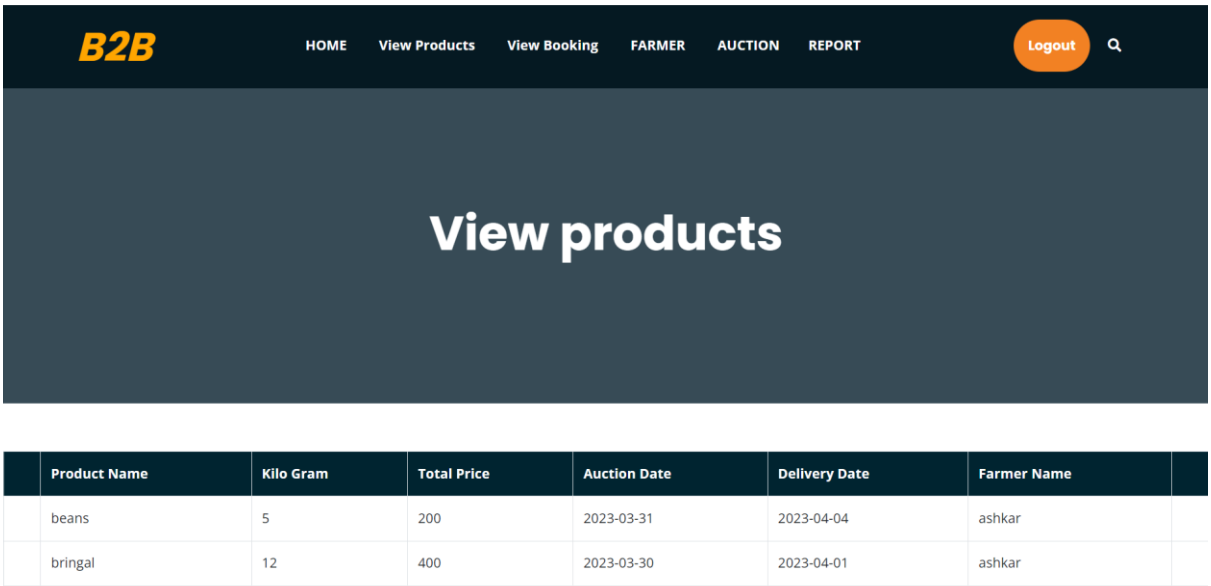
kochi

••••••••

## Admin home page

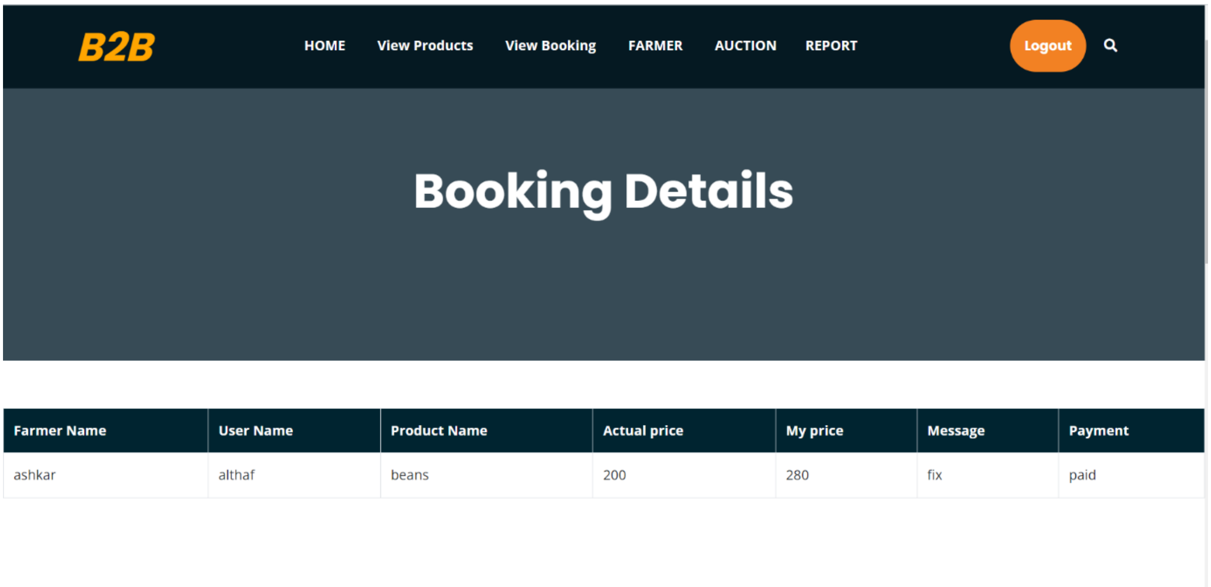


## View Products



| | Product Name | Kilo Gram | Total Price | Auction Date | Delivery Date | Farmer Name | |
|---|---|---|---|---|---|---|---|
| | beans | 5 | 200 | 2023-03-31 | 2023-04-04 | ashkar | |
| | bringal | 12 | 400 | 2023-03-30 | 2023-04-01 | ashkar | |

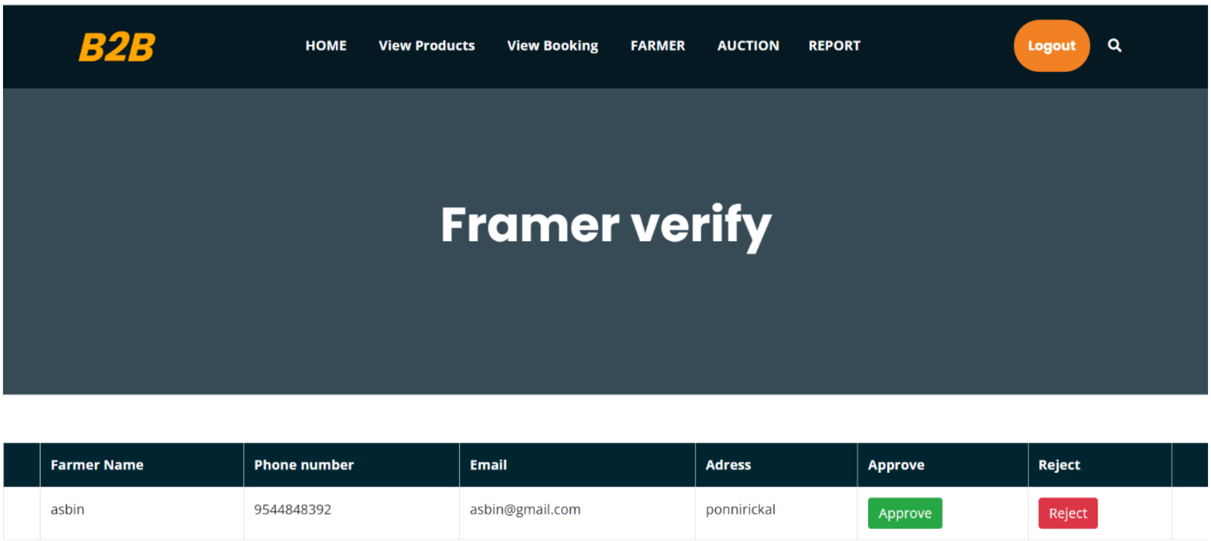
## View Booking



Booking Details

| Farmer Name | User Name | Product Name | Actual price | My price | Message | Payment |
|---|---|---|---|---|---|---|
| ashkar | althaf | beans | 200 | 280 | fix | paid |

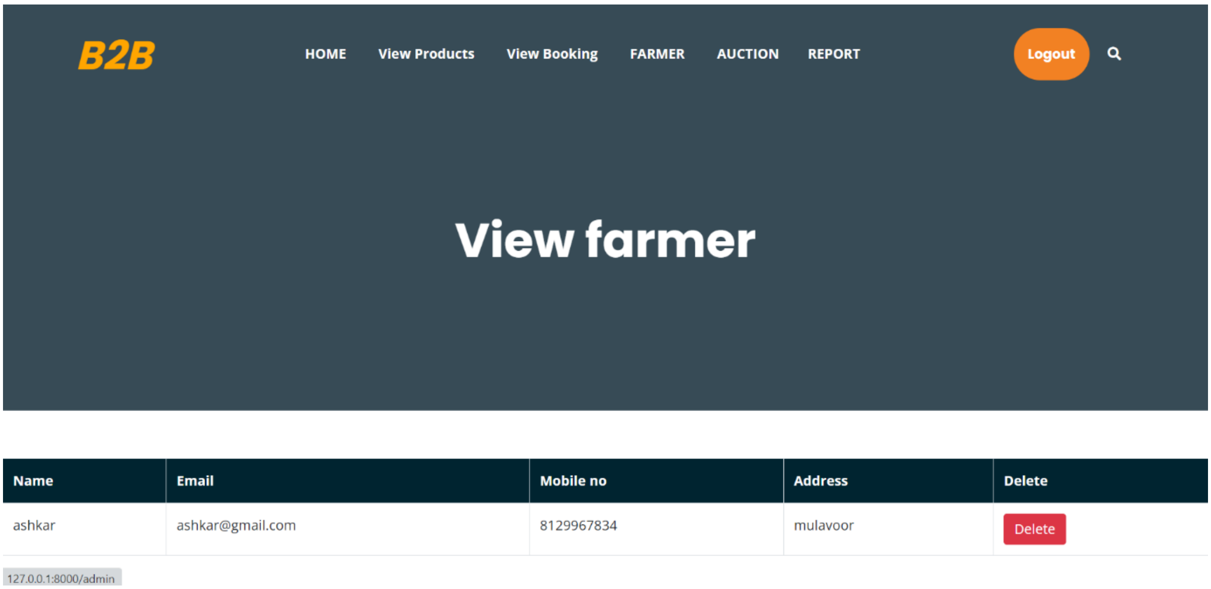
## Farmer Verify



Framer verify

| Farmer Name | Phone number | Email | Adress | Approve | Reject |
|---|---|---|---|---|---|
| asbin | 9544848392 | asbin@gmail.com | ponnirickal | Approve | Reject |

## Farmer Details

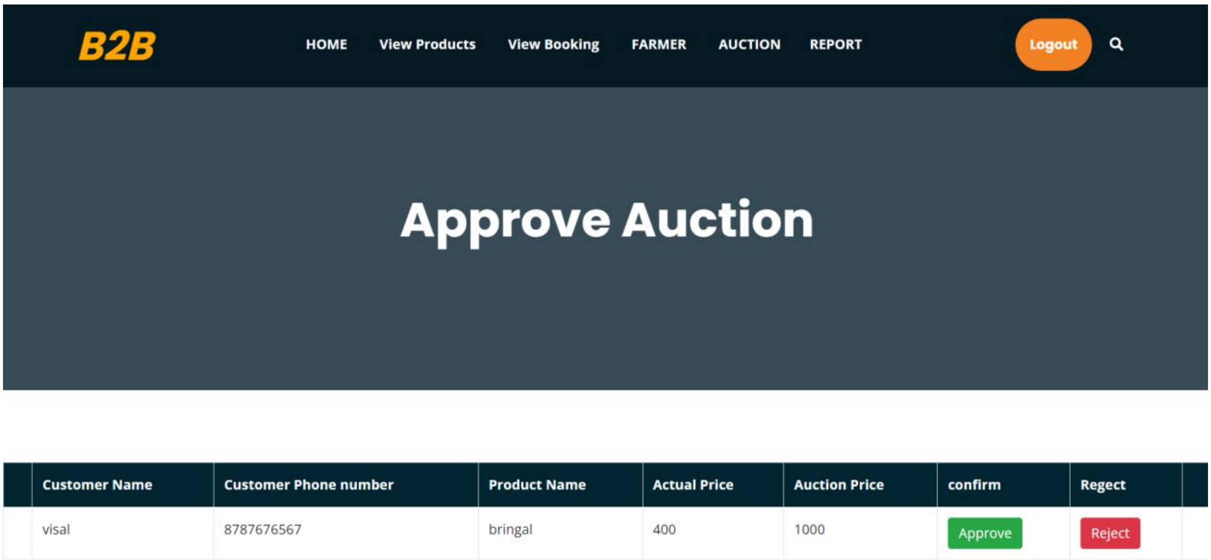


View farmer

| Name | Email | Mobile no | Address | Delete |
|---|---|---|---|---|
| ashkar | ashkar@gmail.com | 8129967834 | mulavoor | Delete |

127.0.0.1:8000/admin

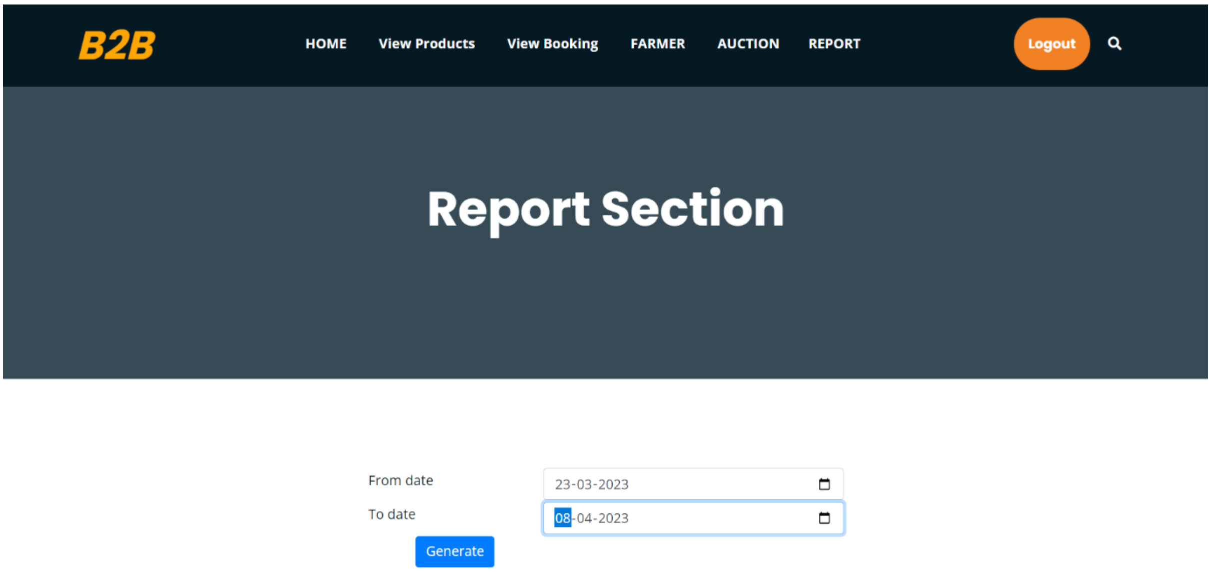
## Approve Auction



Approve Auction

| Customer Name | Customer Phone number | Product Name | Actual Price | Auction Price | confirm | Regect |
|---|---|---|---|---|---|---|
| visal | 8787676567 | bringal | 400 | 1000 | Approve | Reject |

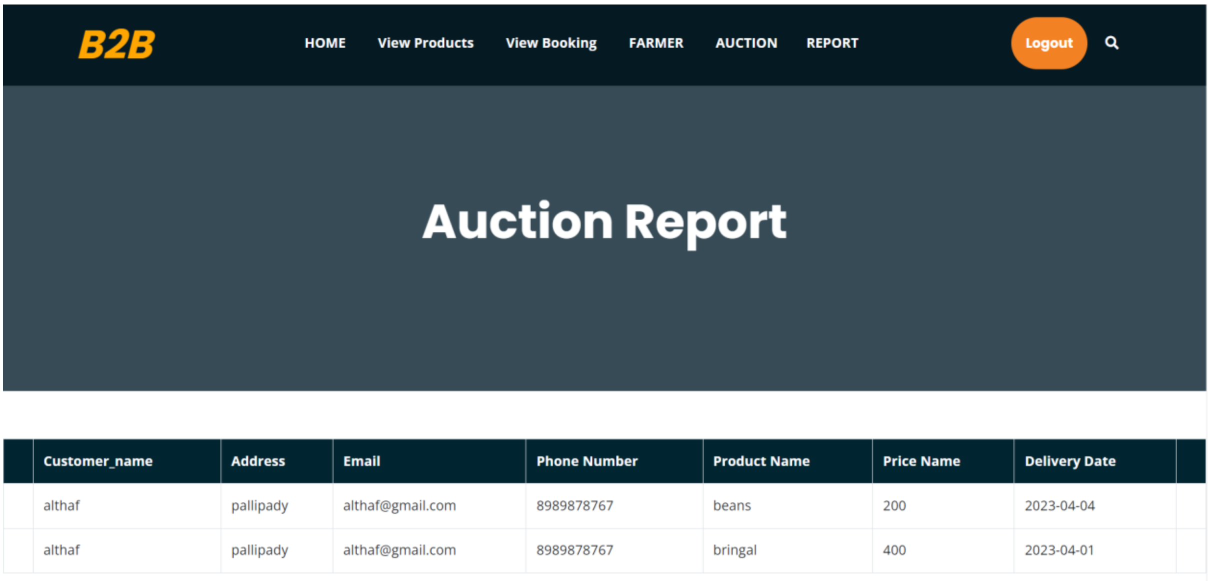
## Auction Report



From date 23-03-2023

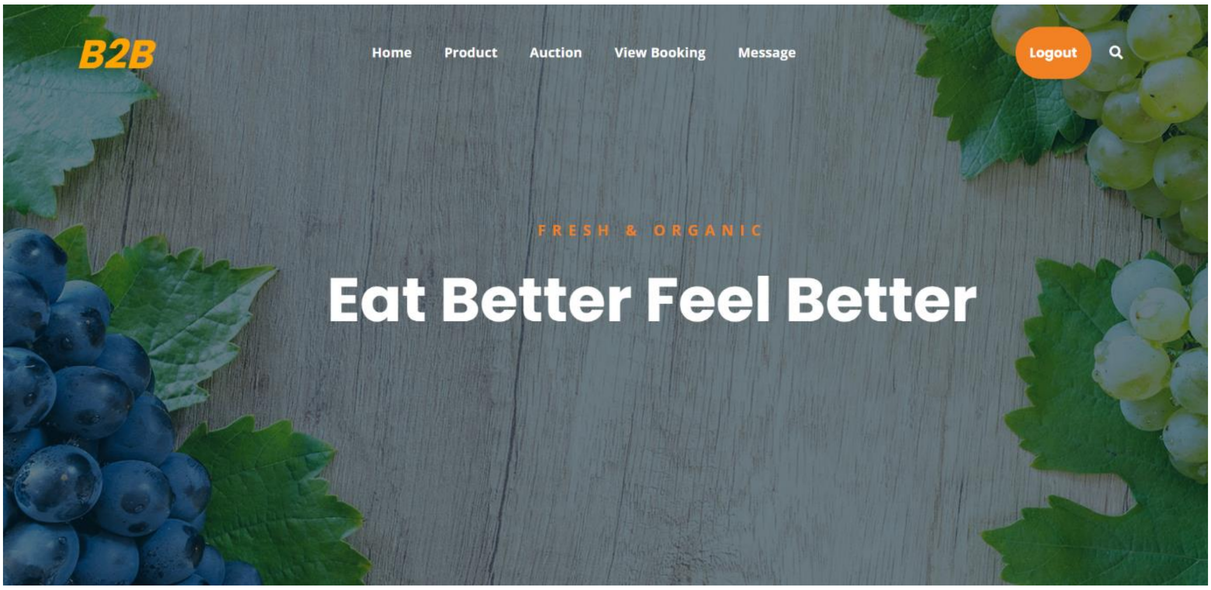
To date 08-04-2023

## Generate Report

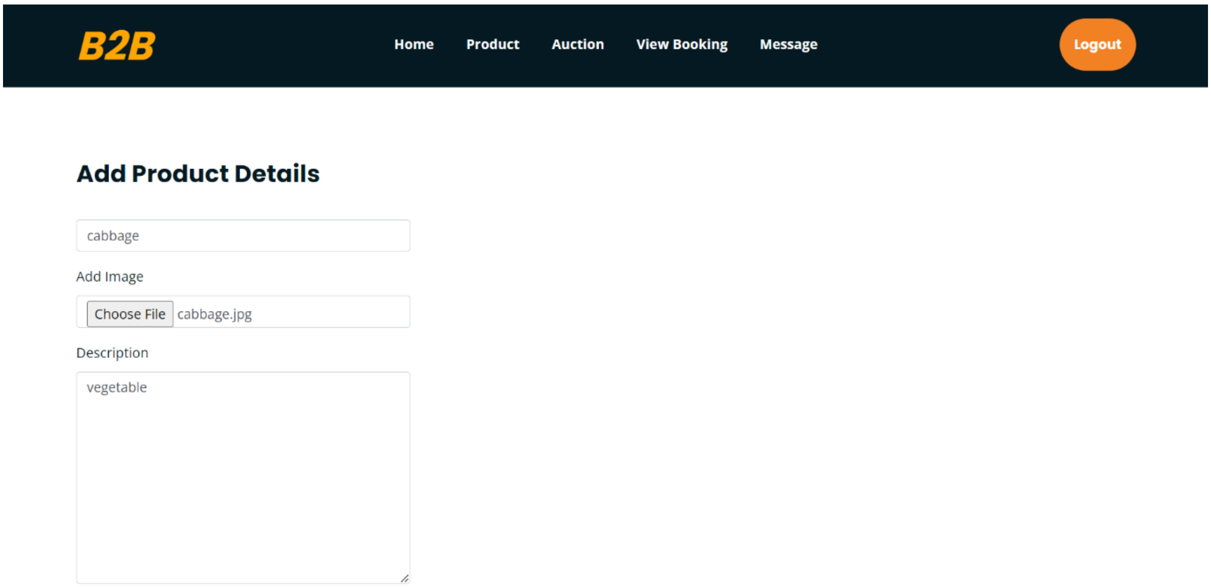


| Customer_name | Address | Email | Phone Number | Product Name | Price Name | Delivery Date |
|---|---|---|---|---|---|---|
| althaf | pallipady | althaf@gmail.com | 8989878767 | beans | 200 | 2023-04-04 |
| althaf | pallipady | althaf@gmail.com | 8989878767 | bringal | 400 | 2023-04-01 |

## Farmer home page



## Add Products



Add Product Details

cabbage

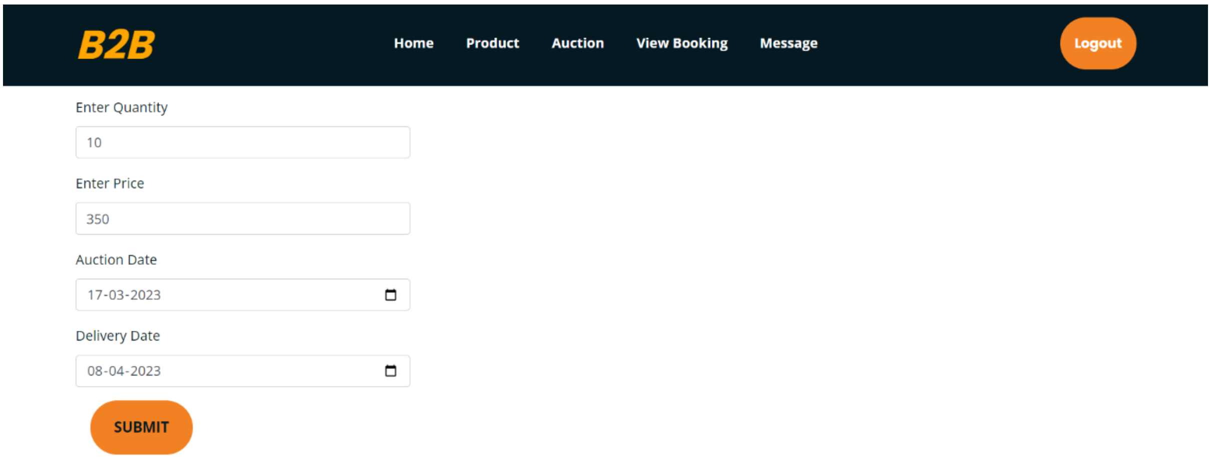
Add Image

Choose File cabbage.jpg

Description

vegetable

## Add product balance



Enter Quantity

10

Enter Price

350

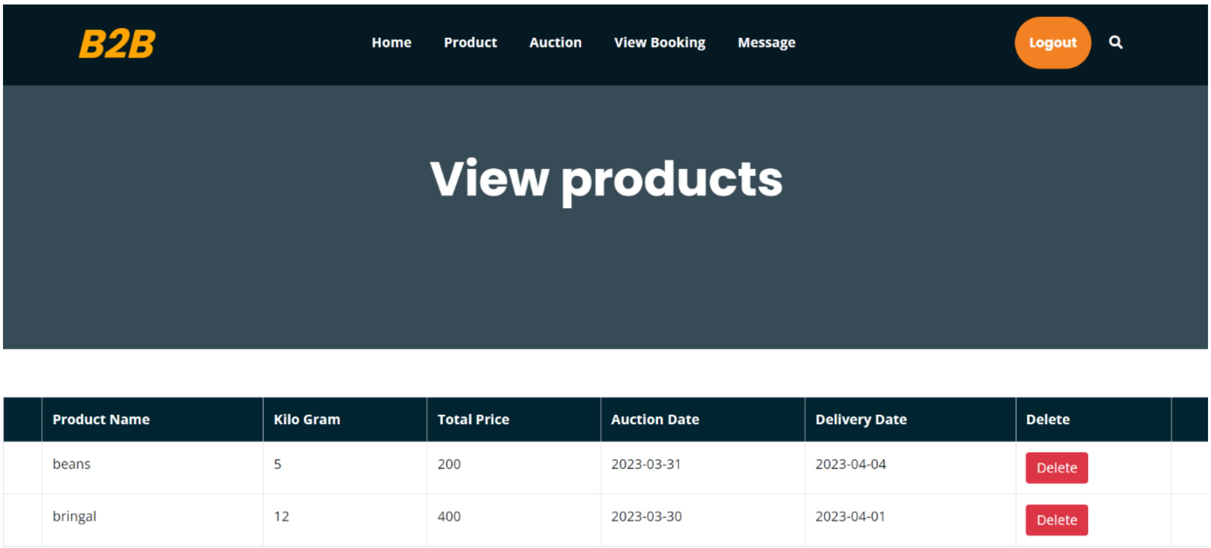
Auction Date

17-03-2023

Delivery Date

08-04-2023

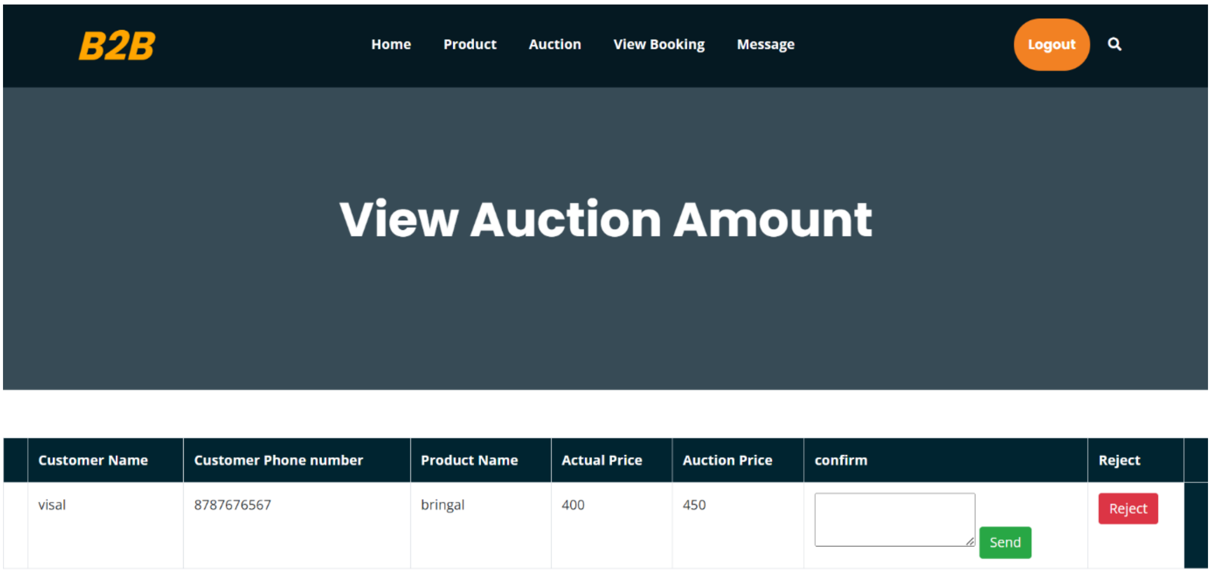
## View products



View products

| Product Name | Kilo Gram | Total Price | Auction Date | Delivery Date | Delete |
|---|---|---|---|---|---|
| beans | 5 | 200 | 2023-03-31 | 2023-04-04 | Delete |
| bringal | 12 | 400 | 2023-03-30 | 2023-04-01 | Delete |

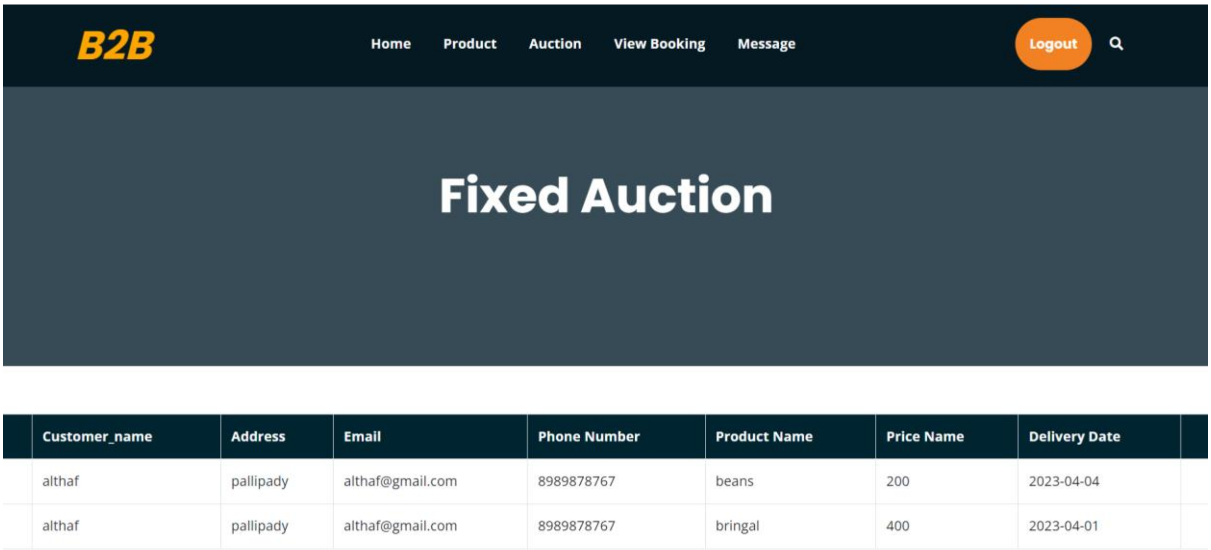
## Auction view amount



View Auction Amount

| Customer Name | Customer Phone number | Product Name | Actual Price | Auction Price | confirm | Reject |
|---|---|---|---|---|---|---|
| visal | 8787676567 | bringal | 400 | 450 | Send | Reject |

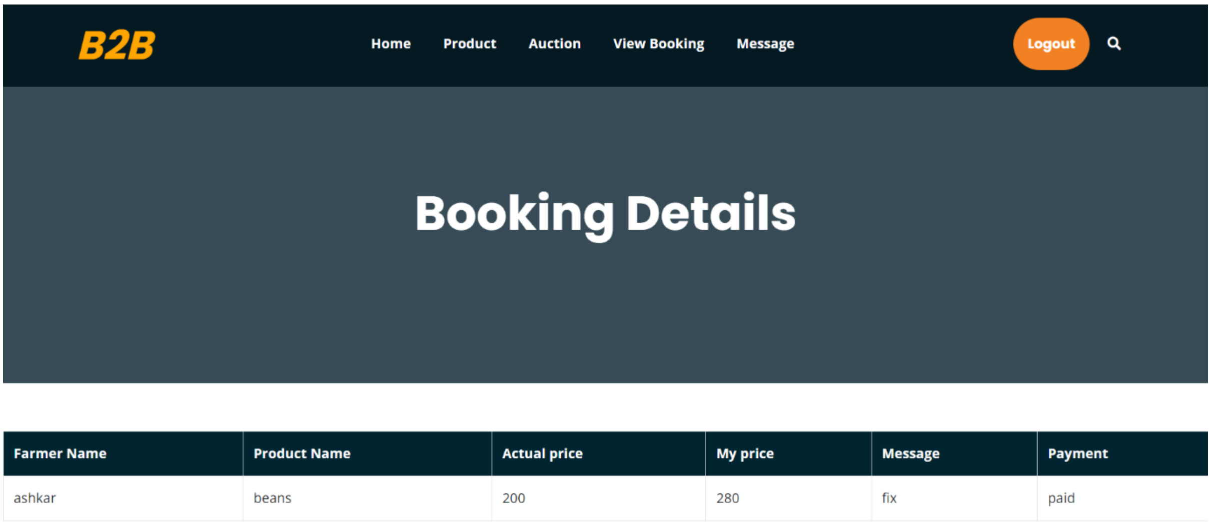
## Fixed Auction Details



Fixed Auction

| Customer_name | Address | Email | Phone Number | Product Name | Price Name | Delivery Date |
|---|---|---|---|---|---|---|
| althaf | pallipady | althaf@gmail.com | 8989878767 | beans | 200 | 2023-04-04 |
| althaf | pallipady | althaf@gmail.com | 8989878767 | bringal | 400 | 2023-04-01 |

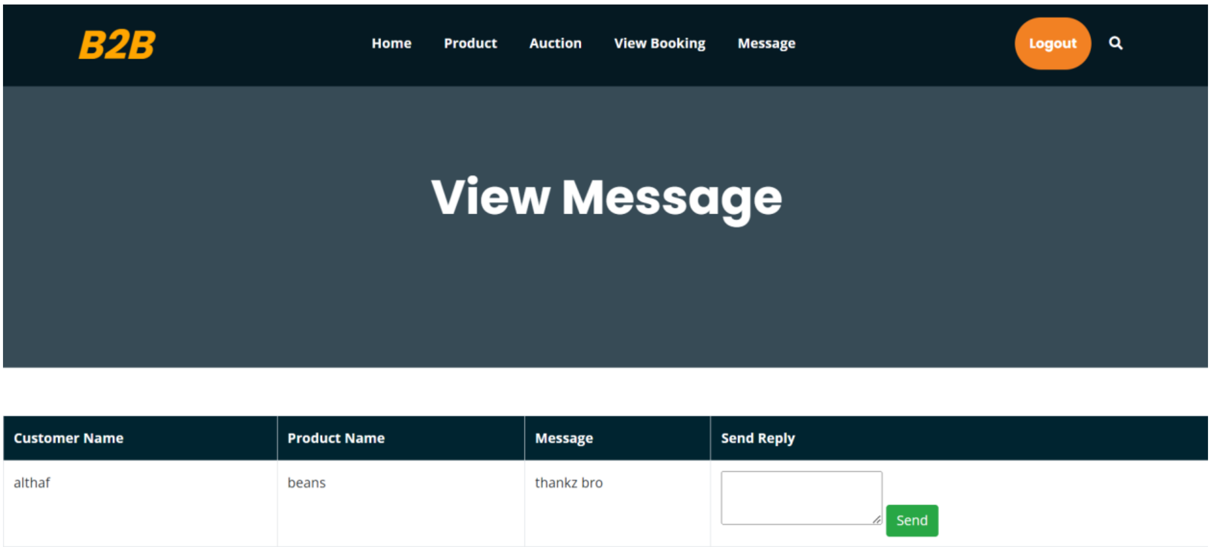
## View Booking



Booking Details

| Farmer Name | Product Name | Actual price | My price | Message | Payment |
|---|---|---|---|---|---|
| ashkar | beans | 200 | 280 | fix | paid |

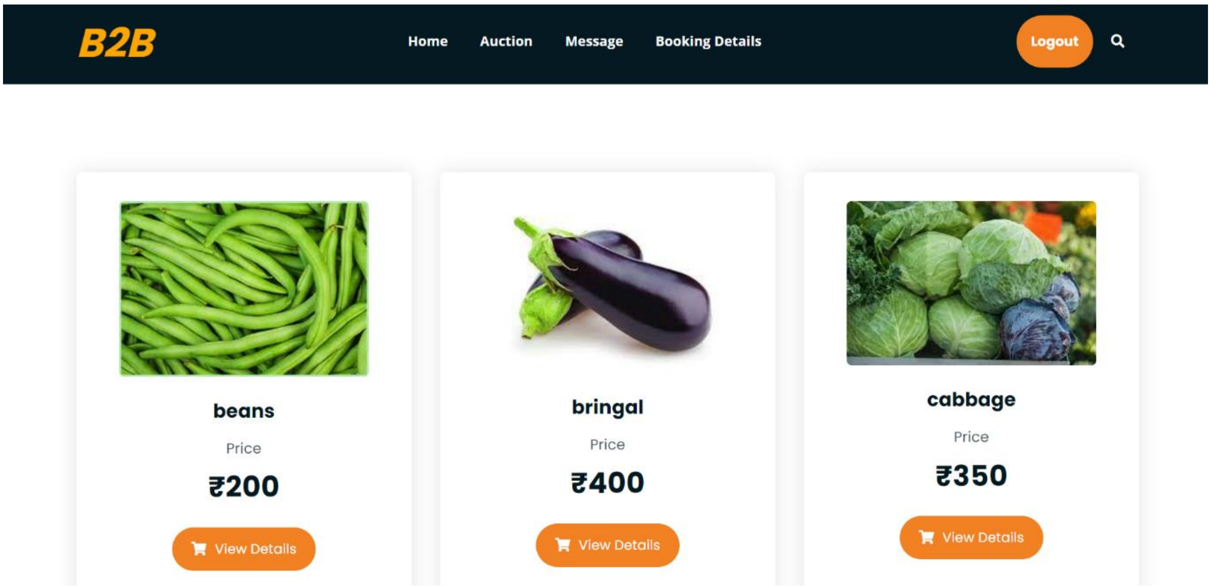
## View Message



View Message

| Customer Name | Product Name | Message | Send Reply |
|---|---|---|---|
| althaf | beans | thankz bro | Send |

## User home page



beans

Price

₹200

bringal

Price

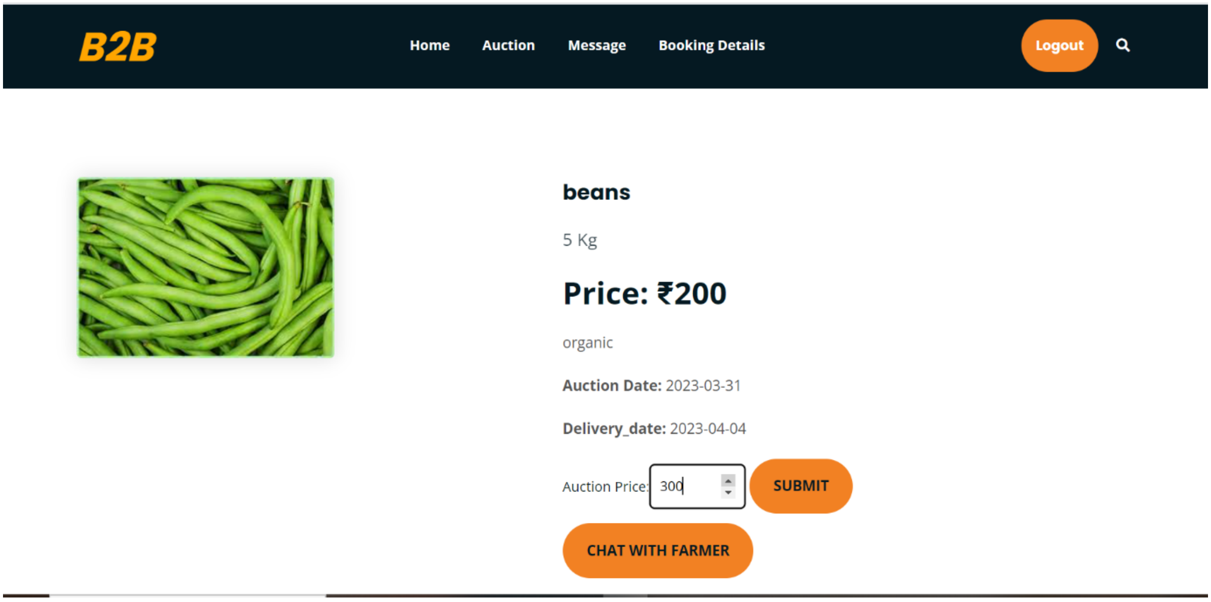
₹400

cabbage

Price

₹350

## View Details



beans

5 Kg

Price: ₹200

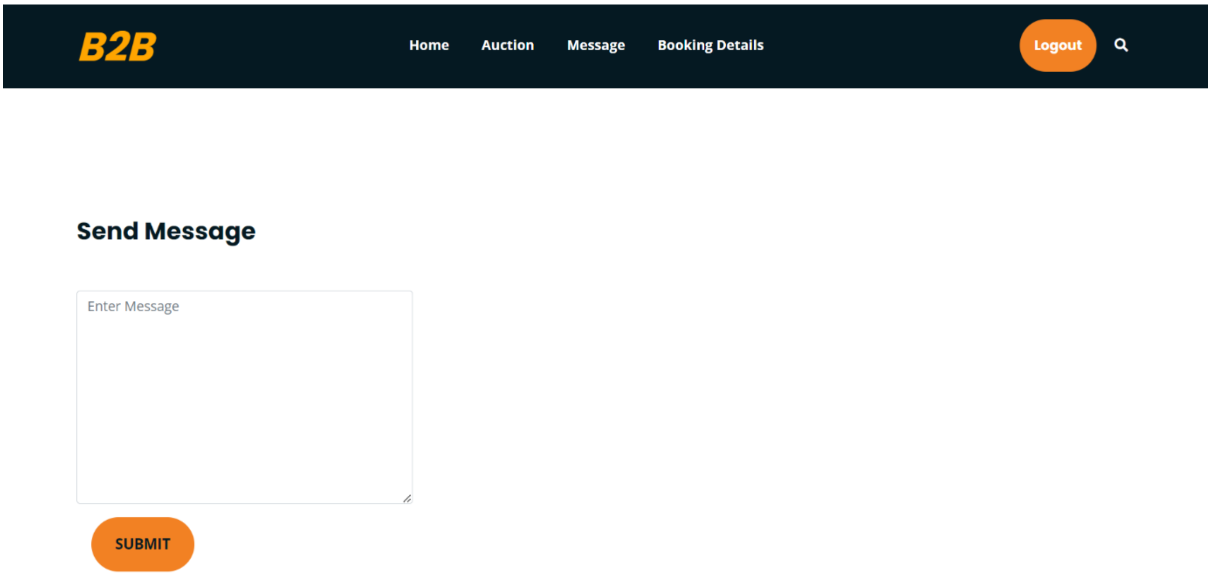
organic

**Auction Date:** 2023-03-31

**Delivery_date:** 2023-04-04

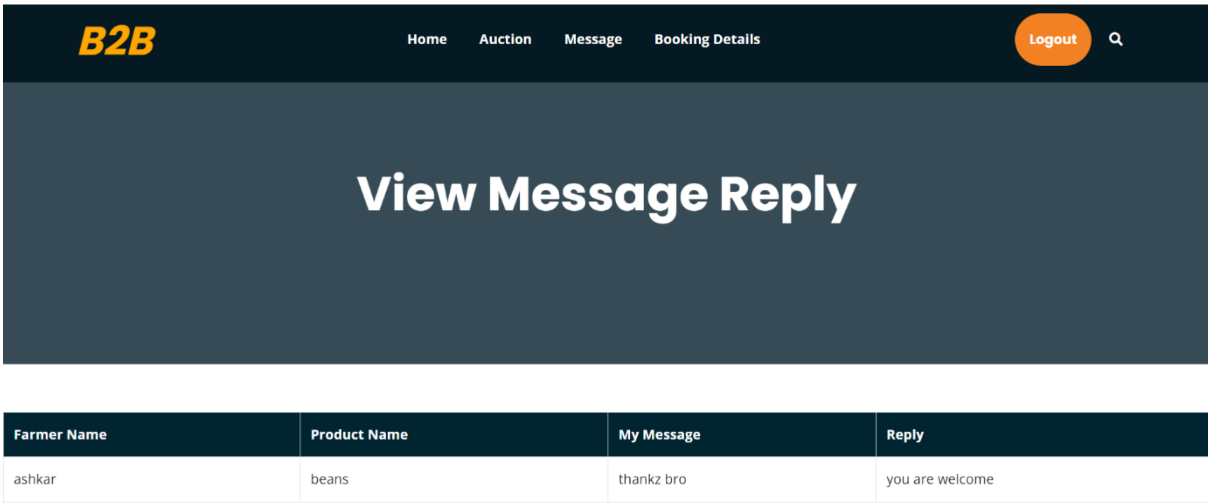
Auction Price: 300

## Chat With Farmer



### Send Message

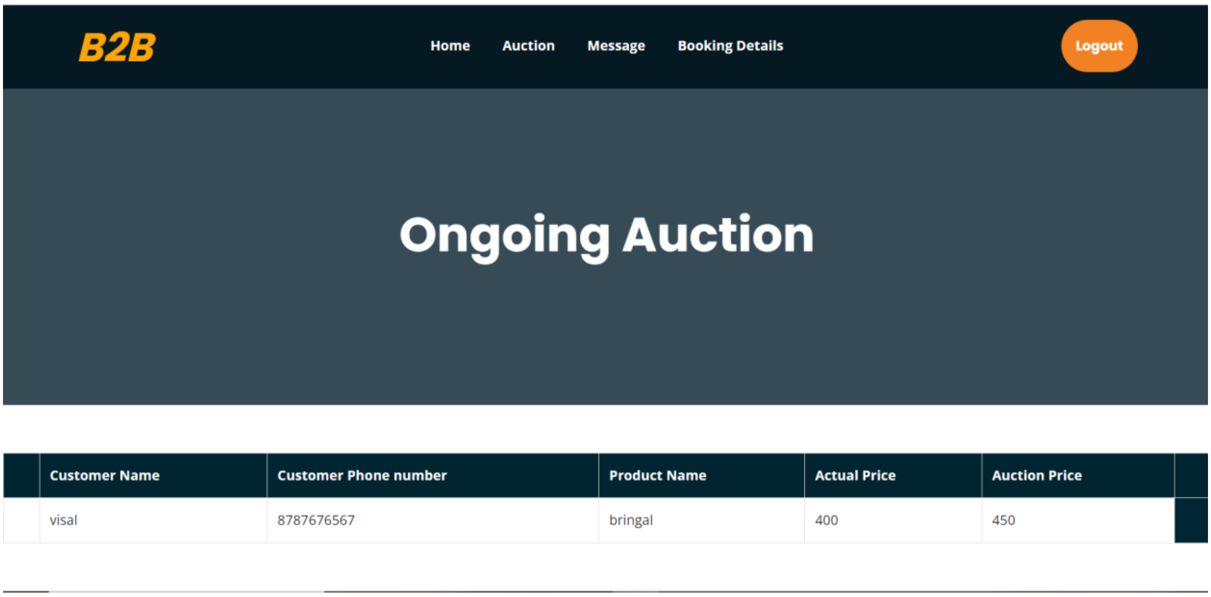
## View Message Replay



### View Message Reply

| Farmer Name | Product Name | My Message | Reply |
|---|---|---|---|
| ashkar | beans | thankz bro | you are welcome |

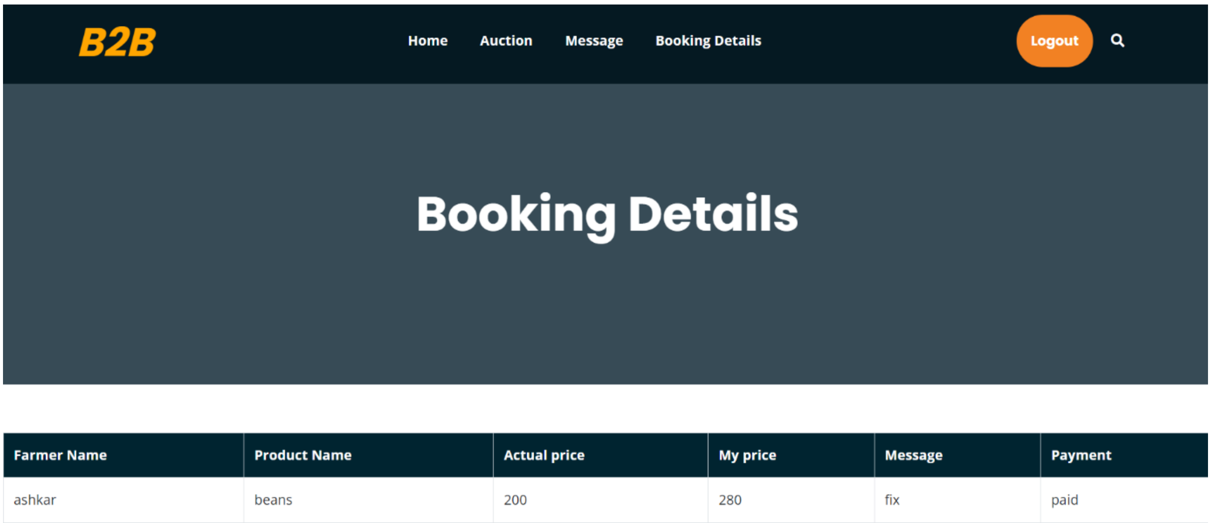
## Ongoing Auction



Ongoing Auction

| | Customer Name | Customer Phone number | Product Name | Actual Price | Auction Price | |
|---|---|---|---|---|---|---|
| | visal | 8787676567 | bringal | 400 | 450 | |

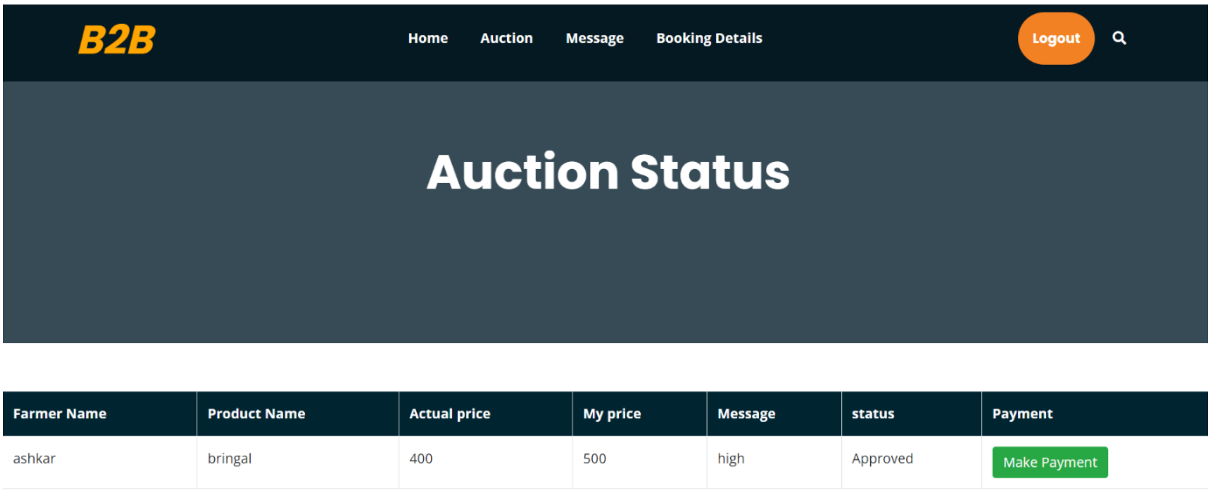
## Booking Details



Booking Details

| Farmer Name | Product Name | Actual price | My price | Message | Payment |
|---|---|---|---|---|---|
| ashkar | beans | 200 | 280 | fix | paid |

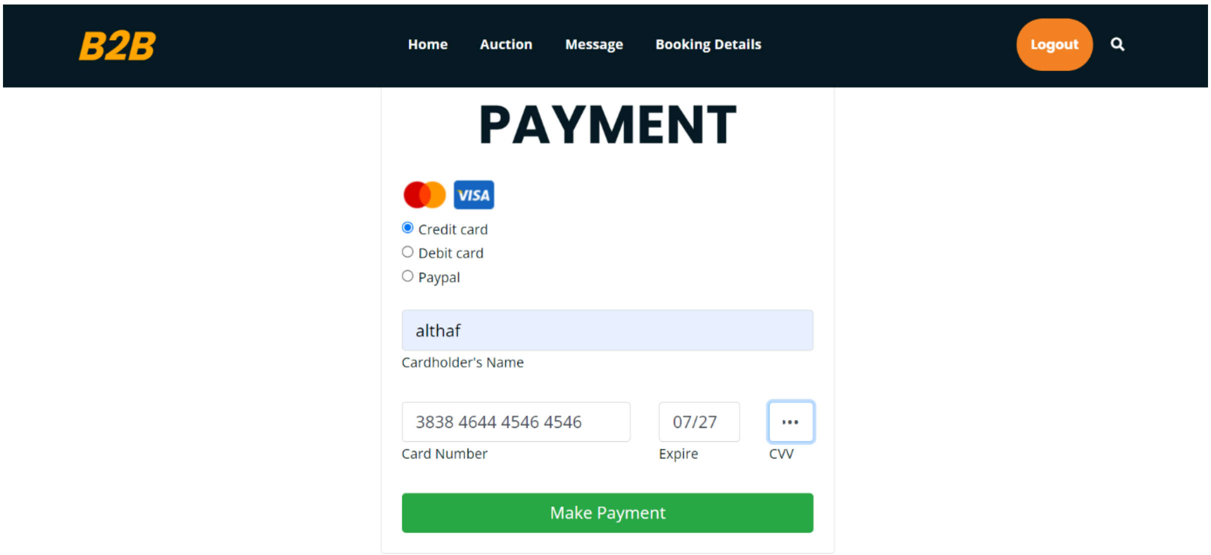
### Your Auction Status



Auction Status

| Farmer Name | Product Name | Actual price | My price | Message | status | Payment |
|---|---|---|---|---|---|---|
| ashkar | bringal | 400 | 500 | high | Approved | Make Payment |

### Make Payment



PAYMENT

● Credit card
○ Debit card
○ Paypal

Cardholder's Name: althaf

Card Number: 3838 4644 4546 4546

Expire: 07/27

CVV: •••

Make Payment

**10.2 Codes**

**Login.html**

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
    {% if message %}
<script>
alert("{{ message }}")
</script>
  {% endif %}
      <div class="breadcrumb-section breadcrumb-bg">
            <div class="container">
                  <div class="row">
                        <div class="col-lg-8 offset-lg-2 text-center">
                              <div class="breadcrumb-text">
                                    <h1>Login </h1>
                              </div>
                        </div>
                  </div>
            </div>
      </div>
      <div class="contact-from-section mt-150 mb-150">
            <div class="container">
                  <div class="row">
                        <div class="col-lg-8 mb-5 mb-lg-0">
```

```
                              <div class="form-title">

                                    <h2>Login </h2>

                              </div>

                              <div id="form_status"></div>

                              <div class="contact-form">

                                    <form action="" method="post">

                  {% csrf_token %}

            <div class="row">

                <div class="col-6 ">

                  {% csrf_token %}

                  <div class="form-group">

                    <input type="text" class="form-control" id="contact-name"
name="email" placeholder="Enter Your Username" >

                  </div>

                </div>

            </div>

            <div class="row">

                <div class="col-6 ">

                  <div class="form-group">

                    <input type="password" class="form-control" name="password"
id="contact-email" placeholder="Enter Your Password">

                  </div>

                </div>

            </div>

            <div class="col-12">

<p><input type="submit" value="Submit"></p> <a href="User_Registration">Register your
account?</a>
```

```
                                </div>

                            </div>

                        </form>

            </div>

            </div>

            </div>

            </div>

            </div>

{% endblock %}
```

**Farmer_reg.html**

```
{% extends 'base.html' %}

{% load static %}

{% block content %}

        {% if message %}

<script>

alert("{{ message }}")

</script>

    {% endif %}

            <div class="breadcrumb-section breadcrumb-bg">

                        <div class="container">

                                    <div class="row">

                                                <div class="col-lg-8 offset-lg-2 text-center">

                                                            <div class="breadcrumb-text">

                                                                        <h1>Farmer Registration </h1>

                                                            </div>
```

```
                                        </div>

                              </div>

                    </div>

          </div>

          <div class="contact-from-section mt-150 mb-150">

                    <div class="container">

                              <div class="row">

                                        <div class="col-lg-8 mb-5 mb-lg-0">

                                                  <div class="form-title">

                                                            <h2>Register Your Account </h2>

                                                  </div>

<div id="form_status">

</div>

                    <div class="contact-form">

                        <form action="" method="post">

                    {% csrf_token %}

                    <div class="row">

                           <div class="col-6 ">

                              {% csrf_token %}

                              <div class="form-group">

 <input type="text" class="form-control" id="contact-name" name="name"
placeholder="Enter Name" autocomplete="off" onkeypress="return (event.charCode > 64
&& event.charCode < 91) || (event.charCode > 96 && event.charCode < 123) ||
(event.charCode==32)" required>

                              </div>

                           </div>

          </div>
```

```
<div class="row">

                    <div class="col-6 ">

                        <div class="form-group">

                           <input type="text" class="form-control" name="email" id="contact-email" placeholder="Enter Your Email" autocomplete="off" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,63}$" required>

                        </div>

                     </div>

                </div>

        <div class="row">

                      <div class="col-6">

                        <div class="form-group">

                           <input type="text" class="form-control" id="" name="phone" placeholder="Enter Mob Number">

                        </div>

                     </div>

                     </div>

                   <div class="row">

                    <div class="col-6">

                        <div class="form-group">

                           <textarea class="form-control" name="address" id="message" cols="30" rows="10"  placeholder="Address" autocomplete="off"></textarea>

                        </div>

                     </div>

                   </div>

                   <div class="row">

                   </div>

                     <div class="row">
```

```
                        <div class="col-6">
                          <div class="form-group">
                             <input type="password" class="form-control" id="contact-username" name="password" placeholder="Enter Password" pattern=".{8,}" title="Eight or more characters" required>
                          </div>
                       </div>
                       </div>
                          <p><input type="submit" value="Submit"></p>
                       </div>
                    </div>
                                                 </form>
                                        </div>
                               </div>
                          </div>
                    </div>
          </div>
{% endblock %}
```

**User_reg.html**

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
      {% if message %}
<script>
alert("{{ message }}")
</script>
```

```
 {% endif %}

        <div class="breadcrumb-section breadcrumb-bg">

                <div class="container">

                        <div class="row">

                                <div class="col-lg-8 offset-lg-2 text-center">

                                        <div class="breadcrumb-text">

                                                <h1>User Registration </h1>

                                        </div>

                                </div>

                        </div>

                </div>

        </div>

        <div class="contact-from-section mt-150 mb-150">

                <div class="container">

                        <div class="row">

                                <div class="col-lg-8 mb-5 mb-lg-0">

                                        <div class="form-title">

                                                <h2>Register Your Account </h2>

                                        </div>

                                        <div id="form_status"></div>

                                        <div class="contact-form">

        <form action="" method="post">

{% csrf_token %}

        <div class="row">

                      <div class="col-6 ">

                        {% csrf_token %}
```

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

                                        <input type="text" class="form-control" id="contact-name"
name="name" placeholder="Enter Your Name" autocomplete="off" onkeypress="return
(event.charCode > 64 && event.charCode < 91) || (event.charCode > 96 && event.charCode
< 123) || (event.charCode==32)" required>

                                    </div>

                                </div>

                        </div>

            <div class="row">

                                <div class="col-6 ">

                                    <div class="form-group">

                                        <input type="text" class="form-control" name="email" id="contact-
email" placeholder="Enter Your Email" autocomplete="off" pattern="[a-z0-9._%+-]+@[a-z0-
9.-]+\.[a-z]{2,63}$" required>

                                    </div>

                                </div>

                        </div>

            <div class="row">

                                 <div class="col-6">

                                    <div class="form-group">

                                        <input type="text" class="form-control" id="" name="phone"
placeholder="Enter Mob Number">

                                    </div>

                                </div>

                                </div>

                            <div class="row">

                             <div class="col-6">

                                    <div class="form-group">
                                        <textarea class="form-control" name="address" id="message"
cols="30" rows="10"  placeholder="Address" autocomplete="off"></textarea>

                                    </div>

                                </div>

                            </div>

                            <div class="row">

                            </div>

                                <div class="row">

                                 <div class="col-6">

                                    <div class="form-group">

                                        <input type="password" class="form-control" id="contact-
username" name="password" placeholder="Enter Password" pattern=".{8,}" title="Eight or
more characters" required>

                                    </div>

                                </div>

                                </div>

                  <div class="col-12">

                                    <p><input type="submit" value="Submit"></p>

                                </div>

                            </div>

                                                                        </form>

                                                            </div>

                                                </div>

                                    </div>

                        </div>

            </div>

{% endblock %}
```

**View.py**

```
from django.contrib.auth import authenticate, login

from django.contrib.auth.models import User

from django.shortcuts import render, redirect

from django.views.generic import TemplateView


from B2B_App.models import UserType, Customer_Reg, Farmer_Reg


class IndexView(TemplateView):

    template_name = 'index.html'


class Registration(TemplateView):

    template_name = 'user_reg.html'


    def post(self, request, *args, **kwargs):

        fullname = request.POST['name']

        address = request.POST['address']

        email = request.POST['email']

        phone = request.POST['phone']

        password = request.POST['password']


        try:

            user = User.objects.create_user(username=email, password=password,
first_name=fullname, email=email,last_name=1)

            user.save()

            reg = Customer_Reg()
            reg.user = user

            reg.address = address

            reg.phonenumber = phone

            reg.save()

            usertype = UserType()

            usertype.user = user

            usertype.type = 'user'

            usertype.save()

            messages = "Register Successfully."


            return render(request, 'index.html', {'message': messages})

        except:

            messages = "Username already used!.."

            return render(request, 'index.html', {'message': messages})


class farmer_reg(TemplateView):

    template_name = 'farmer_registration.html'


    def post(self, request, *args, **kwargs):

        fullname = request.POST['name']

        address = request.POST['address']

        email = request.POST['email']

        phone = request.POST['phone']

        password = request.POST['password']


        try:
            user = User.objects.create_user(username=email, password=password,
first_name=fullname, email=email,last_name=0)
            user.save()
            reg = Farmer_Reg()
            reg.user = user
            reg.address = address
            reg.phonenumber = phone
            reg.save()
            usertype = UserType()
            usertype.user = user
            usertype.type = 'farmer'
            usertype.save()
            messages = "waiting for approval."

            return render(request, 'index.html', {'message': messages})
        except:
            messages = "Username already used!.."
            return render(request, 'index.html', {'message': messages})


class Login(TemplateView):
    template_name = 'login.html'


    def post(self, request, *args, **kwargs):
        email = request.POST['email']
        password = request.POST['password']
        user = authenticate(username=email, password=password)
        if user is not None:
            login(request, user)

            if user.last_name == '1':

                if user.is_superuser:

                    return redirect('/admin')

                elif UserType.objects.get(user_id=user.id).type == "user":

                    return redirect('/user')

                elif UserType.objects.get(user_id=user.id).type == "farmer":

                    return redirect('/farmer')

            else:

                return render(request, 'login.html', {'message': " User Account Not Authenticated"})

        else:

            return render(request, 'login.html', {'message': "Invalid Username or Password"})
```

**model.py**

```
from django.contrib.auth.models import User

from django.db import models


# Create your models here.

class UserType(models.Model):

    user = models.ForeignKey(User,on_delete=models.CASCADE)

    type = models.CharField(max_length=50)


class Customer_Reg(models.Model):

    user = models.ForeignKey(User, on_delete=models.CASCADE)

    address = models.CharField(max_length=100,null=True)

    phonenumber = models.CharField(max_length=100,null=True)
```

```
class Farmer_Reg(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    address = models.CharField(max_length=100,null=True)
    phonenumber = models.CharField(max_length=100,null=True)

class Product(models.Model):
    name=models.CharField(max_length=200,null=True)
    image = models.ImageField(upload_to='images/', null=True)
    desc=models.TextField(null=True)
    quantity=models.IntegerField(null=True)
    price=models.IntegerField(null=True)
    auction_date=models.CharField(max_length=100,null=True)
    delivery_date=models.CharField(max_length=100,null=True)
    farmer = models.ForeignKey(Farmer_Reg, on_delete=models.CASCADE,null=True)

class chat(models.Model):
    message = models.CharField(max_length=250, null=True)
    product=models.ForeignKey(Product, on_delete=models.CASCADE, null=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE, null=True)
    farmer = models.ForeignKey(Farmer_Reg, on_delete=models.CASCADE,null=True)
    status = models.CharField(max_length=250, null=True)
    reply = models.CharField(max_length=250, null=True)

class Auction(models.Model):
    price = models.IntegerField(null=True)
    product = models.ForeignKey(Product, on_delete=models.CASCADE, null=True)

    customer = models.ForeignKey(Customer_Reg, on_delete=models.CASCADE, null=True)

    farmer = models.ForeignKey(Farmer_Reg, on_delete=models.CASCADE, null=True)

    status=models.CharField(max_length=100,null=True)

    product_name=models.CharField(max_length=100,null=True)

    message=models.CharField(max_length=100,null=True)

    admin_status=models.CharField(max_length=100,null=True)


class fixed_auction(models.Model):

    product_name=models.CharField(max_length=100,null=True)

    message=models.CharField(max_length=100,null=True)

    price=models.CharField(max_length=100,null=True)

    Auction_price=models.CharField(max_length=100,null=True)

    payment=models.CharField(max_length=100,null=True)

    date=models.CharField(max_length=100,null=True)

    status=models.CharField(max_length=100,null=True)

    customer = models.ForeignKey(Customer_Reg, on_delete=models.CASCADE, null=True)

    farmer = models.ForeignKey(Farmer_Reg, on_delete=models.CASCADE, null=True)

    com_date=models.DateField(null=False, blank=False, auto_now=True)
```

**admin_urls.py**

```
from django.urls import path

from B2B_App.admin_views import Indexview, Farmer_varify, ApproveView,
RejectView,Generate_complaints ,view_farmer ,Delete_farmer, auction_view,
auction_approve, view_product, auction_reject,\

    booking
```

```
urlpatterns = [


    path('',Indexview.as_view()),

    path('Farmer_varify',Farmer_varify.as_view()),

    path('approve',ApproveView.as_view()),

    path('reject', RejectView.as_view()),

    path('Generate_complaints',Generate_complaints.as_view()),

    path('view_farmer',view_farmer.as_view()),

    path('Delete_farmer',Delete_farmer.as_view()),

    path('view_product',view_product.as_view()),

    path('auction_approve',auction_view.as_view()),

    path('auc_approve',auction_approve.as_view()),

    path('auction_reject',auction_reject.as_view()),

    path('booking',booking.as_view()),

]

def urls():

    return urlpatterns, 'admin','admin'
```

**admin_view.py**

```
from django.contrib.auth.models import User

from django.shortcuts import render

from django.views import View

from django.views.generic import TemplateView

from B2B_App import farmer_urls, farmer_views


from B2B_App.models import Auction, Farmer_Reg,fixed_auction, Product
```

```
class Indexview(TemplateView):
    template_name = 'admin/admin_index.html'


class Farmer_varify(TemplateView):
    template_name = 'admin/farmer_varify.html'


    def get_context_data(self, **kwargs):
        context = super(Farmer_varify, self).get_context_data(**kwargs)


        shop = Farmer_Reg.objects.filter(user__last_name='0', user__is_staff='0',
user__is_active='1')


        context['shop'] = shop
        return context


class ApproveView(View):
    def dispatch(self, request, *args, **kwargs):
        id = request.GET['id']
        user = User.objects.get(pk=id)
        user.last_name = '1'
        user.save()
        return render(request, 'admin/admin_index.html', {'message': " Account Approved"})


class RejectView(View):
    def dispatch(self, request, *args, **kwargs):
        id = request.GET['id']
        user = User.objects.get(pk=id)

        user.last_name = '1'

        user.is_active = '0'

        user.save()

        return render(request, 'admin/admin_index.html', {'message': "Account Removed"})


class Generate_complaints(TemplateView):

    template_name = 'admin/report.html'


    def post(self, request, *args, **kwargs):




        f = request.POST['f']

        t = request.POST['t']

        print(f)

        print(t)

        host = (fixed_auction.objects.filter(date__lte=f,status='Approved') |
fixed_auction.objects.filter(

            date__lte=t,status='Approved'))


        return render(request, 'admin/report1.html', {'g':host})


class view_farmer(TemplateView):

    template_name = 'admin/view_farmer.html'


    def get_context_data(self, **kwargs):

        context = super(view_farmer, self).get_context_data(**kwargs)
        f = Farmer_Reg.objects.all()


        context['f'] = f
        return context


class Delete_farmer(TemplateView):
    def dispatch(self, request, *args, **kwargs):
        try:
            id = request.GET['id']
            farmer = Farmer_Reg.objects.get(id=id)
            farmer.delete()
            message = "Farmer Removed"
        except Farmer_Reg.DoesNotExist:
            message = "Farmer Not Found"


        return render(request, 'admin/view_farmer.html', {'message': message})


class auction_view(TemplateView):
    template_name = 'admin/approve_auction.html'


    def get_context_data(self, **kwargs):


        context = super(auction_view,self).get_context_data(**kwargs)
        feed=Auction.objects.filter(admin_status="added")
        context['feed'] = feed
        return context


class auction_approve(View):

    def dispatch(self, request, *args, **kwargs):

        id = request.GET['id']

        act = Auction.objects.get(id=id)

        act.admin_status = 'approved'

        act.save()

        return render(request, 'admin/admin_index.html', {'message': " Auction Approved"})


class auction_reject(View):

    def dispatch(self, request, *args, **kwargs):

        id = request.GET['id']

        b = Auction.objects.get(pk=id)

        b.status='Rejected'

        b.save()

        return render(request,'admin/admin_index.html',{'message':" Removed"})


class booking(TemplateView):

    template_name ='admin/booking.html'

    def get_context_data(self, **kwargs):


        context = super(booking,self).get_context_data(**kwargs)

        feed=fixed_auction.objects.filter(payment="paid")

        context['feed'] = feed

        return context
```

```
class view_product(TemplateView):
    template_name = 'admin/view_product.html'
    def get_context_data(self, **kwargs):
        context = super(view_product,self).get_context_data(**kwargs)
        view_pr = Product.objects.all()
        context['view_pr'] = view_pr
        return context
```

**farmer_urls.py**

```
from django.urls import path
from B2B_App.farmer_views import Indexview, Add_Product, Auction_Amount,
Auction_submit, view_product, Delete_product, \
    Fixed_Action, view_message, auction_reject, booking


urlpatterns = [
    path('',Indexview.as_view()),
    path('Add_Product',Add_Product.as_view()),
    path('Auction_Amount',Auction_Amount.as_view()),
    path('Auction_submit',Auction_submit.as_view()),
    path('view_product',view_product.as_view()),
    path('delete',Delete_product.as_view()),
    path('Fixed_Action',Fixed_Action.as_view()),
    path('view_message',view_message.as_view()),
    path('auction_reject',auction_reject.as_view()),
    path('booking',booking.as_view()),
]
```

```
def urls():

    return urlpatterns, 'farmer','farmer'
```

**farmer_view.py**

```
from django.contrib.auth.models import User

from django.core.files.storage import FileSystemStorage

from django.shortcuts import redirect, render

from django.views.generic import TemplateView,View


from B2B_App.models import Product, Farmer_Reg, Auction, fixed_auction, chat

class Indexview(TemplateView):

    template_name = 'farmer/farmer_index.html'




class Add_Product(TemplateView):

    template_name = 'farmer/add_product.html'


    def post(self, request,*args,**kwargs):


        farmer = Farmer_Reg.objects.get(user_id=self.request.user.id)

        name = request.POST['name']

        price = request.POST['price']

        quantity=request.POST['qty']

        desc = request.POST['desc']

        a_date = request.POST['a_date']

        d_date = request.POST['d_date']
        image = request.FILES['image']

        fii = FileSystemStorage()

        filesss = fii.save(image.name, image)

        se = Product()

        se.farmer = farmer

        se.name = name

        se.auction_date=a_date

        se.delivery_date=d_date

        se.quantity=quantity

        se.image=filesss

        se.price = price

        se.desc = desc

        se.save()


        return render(request, 'farmer/farmer_index.html', {'message': "Product Added"})


class Auction_Amount(TemplateView):

    template_name = 'farmer/view_auction_amount.html'


    def get_context_data(self, **kwargs):


        context = super(Auction_Amount,self).get_context_data(**kwargs)

        view_fe = Farmer_Reg.objects.get(user_id=self.request.user.id)

        feed=Auction.objects.filter(farmer_id=view_fe.id, admin_status="approved",
status="added")

        context['feed'] = feed

        return context
    def post(self, request, *args, **kwargs):
        search =request.POST['search']
        feed =
Auction.objects.filter(product_name__icontains=search,status='added').order_by('price')
        return render(request, 'farmer/search.html', {'feed': feed})


class Auction_submit(TemplateView):
    template_name = 'farmer/view_auction_amount.html'
    def post(self, request, *args, **kwargs):


        id = request.POST['id']
        name = request.POST['name']
        cus_price = request.POST['cus_price']
        price = request.POST['price']
        d_date = request.POST['d_date']
        cu_name = request.POST['cu_name']
        f_name = request.POST['f_name']
        id2 = request.POST['id2']
        action = request.POST['action']
        if Auction.objects.filter(product_id=id2,status='confirm'):
            print("fvvfsff")
            return render(request, 'farmer/farmer_index.html', {'message': "Already Added"})
        else:
            act = Auction.objects.get(id=id)
            act.message = action
            act.status = 'confirm'
            act.save()
            a = fixed_auction()
            a.price = price
            a.product_name = name
            a.date = d_date
            a.message=action
            a.customer_id = cu_name
            a.farmer_id=f_name
            a.Auction_price=cus_price
            a.status='Approved'
            a.payment='added'
            a.save()

            return render(request, 'farmer/farmer_index.html', {'message': "Price Confirmed"})


class auction_reject(View):
    def dispatch(self, request, *args, **kwargs):
        id = request.GET['id']
        b = Auction.objects.get(pk=id)
        b.status='Rejected'
        b.save()
        return render(request,'farmer/farmer_index.html',{'message':" Removed"})


class view_product(TemplateView):
    template_name = 'farmer/view_product.html'
    def get_context_data(self, **kwargs):
        context = super(view_product,self).get_context_data(**kwargs)

        f = Farmer_Reg.objects.get(user_id=self.request.user.id)

        view_pr = Product.objects.filter(farmer_id=f.id)

        context['view_pr'] = view_pr

        return context


class Delete_product(TemplateView):

    def dispatch(self,request,*args,**kwargs):

        id = request.GET['id']

        Product.objects.get(id=id).delete()

        return render(request,'farmer/View_product.html',{'message':"Product Removed"})




class Fixed_Action(TemplateView):

    template_name = 'farmer/fixed_action.html'

    def get_context_data(self, **kwargs):


        context = super(Fixed_Action,self).get_context_data(**kwargs)

        f = Farmer_Reg.objects.get(user_id=self.request.user.id)

        view_pr = fixed_auction.objects.filter(farmer_id=f.id)

        context['view_pr'] = view_pr

        return context


class view_message(TemplateView):

    template_name = 'farmer/view_message.html'
    def get_context_data(self, **kwargs):


        context = super(view_message,self).get_context_data(**kwargs)

        f = Farmer_Reg.objects.get(user_id=self.request.user.id)

        view_pr = chat.objects.filter(farmer_id=f.id)

        context['view_pr'] = view_pr

        return context


    def post(self, request, *args, **kwargs):

        # complaint = actions.objects.get(user_id=self.request.id)

        id = request.POST['id']

        action = request.POST['reply']

        act = chat.objects.get(id=id)

        # act.complaint=complaint

        act.reply = action

        act.status = 'replied'

        act.save()

        return render(request, 'farmer/farmer_index.html', {'message': "Replied"})


class booking(TemplateView):

    template_name ='farmer/booking.html'


    def get_context_data(self, **kwargs):

        context = super(booking,self).get_context_data(**kwargs)

        cus = Farmer_Reg.objects.get(user_id=self.request.user.id)

        feed=fixed_auction.objects.filter(farmer_id=cus.id, payment="paid")
```

```
      context['feed'] = feed

      return context
```

**user_urls.py**

```
from django.urls import path

from B2B_App.user_views import Indexview, Product_details, Add_Action, Status_Auc,
Send_message, message_reply, Auction_view, Payment

   Make_Payment,booking

urlpatterns = [


   path('',Indexview.as_view()),

   path('Product_details',Product_details.as_view()),

   path('Add_Action',Add_Action.as_view()),

   path('Auction_Status',Status_Auc.as_view()),

   path('Send_message',Send_message.as_view()),

   path('message_reply',message_reply.as_view()),

   path('Auction_view',Auction_view.as_view()),

   path('Payment',Payment.as_view()),

   path('Make_Payment',Make_Payment.as_view()),

   path('booking',booking.as_view()),

]

def urls():

   return urlpatterns, 'user','user'
```

**user_view.py**

```
from django.contrib.auth.models import User

from django.shortcuts import render

from django.views.generic import TemplateView
from B2B_App.models import Product, Farmer_Reg, Auction, Customer_Reg, fixed_auction,
chat


class Indexview(TemplateView):
    template_name = 'user/product.html'


    def get_context_data(self, **kwargs):
        context = super(Indexview,self).get_context_data(**kwargs)
        view_pp = Product.objects.all()
        context['view_pp'] = view_pp
        return context


class Product_details(TemplateView):
    template_name = 'user/product_details.html'
    def get_context_data(self, **kwargs):
        id =self.request.GET['id']


        context = super(Product_details, self).get_context_data(**kwargs)


        single_view = Product.objects.get(id=id)
        shop = Product.objects.get(id=id)


        context['single_view'] = single_view
        return context


class Add_Action(TemplateView):
    template_name ='user/product_details.html'


    def dispatch(self, request, *args, **kwargs):

        pid = request.POST['id']

        id2 = request.POST['id2']

        qunty = request.POST['price']


        cust = Customer_Reg.objects.get(user_id=self.request.user.id)

        product=Product.objects.get(pk=pid)

        shop = Product.objects.get(pk=pid)


        shopp = Farmer_Reg.objects.get(id=shop.farmer_id)

        ca = Auction()

        ca.customer = cust

        ca.product_name=id2

        ca.farmer_id = shopp.id

        ca.product = product

        ca.price = qunty

        ca.status = 'added'

        ca.admin_status = 'added'

        ca.save()

        return render(request, 'user/product.html', {'message': " price added"})


class Status_Auc(TemplateView):

    template_name ='user/My_Auction_Status.html'

    def get_context_data(self, **kwargs):
        context = super(Status_Auc,self).get_context_data(**kwargs)

        cus = Customer_Reg.objects.get(user_id=self.request.user.id)

        feed=fixed_auction.objects.filter(customer_id=cus.id, payment="Added")

        context['feed'] = feed

        return context


class Send_message(TemplateView):

    template_name = 'user/chat.html'

    def get_context_data(self, **kwargs):

        context = super(Send_message, self).get_context_data(**kwargs)

        id = self.request.GET['id']

        replay = Product.objects.get(id=id)

        context['repl'] = replay

        return context


    def post(self, request, *args, **kwargs):

        user = User.objects.get(id=self.request.user.id)

        product = request.POST['product']

        pro = Product.objects.get(id=product)

        message = request.POST['message']

        fe = chat()

        fe.user = user

        fe.farmer_id=pro.farmer_id

        fe.product_id=product

        fe.message = message

        fe.status='added'
        fe.save()

        return render(request, 'user/product.html', {'message': "Message Send "})


class message_reply(TemplateView):

    template_name = 'user/view_message_reply.html'

    def get_context_data(self, **kwargs):

        context = super(message_reply,self).get_context_data(**kwargs)

        usid = self.request.user.id

        replay = chat.objects.filter(user_id=usid,status='replied')

        context['replay'] = replay

        return context


class Auction_view(TemplateView):

    template_name = 'user/view_auction_amount.html'

    def get_context_data(self, **kwargs):


        context = super(Auction_view,self).get_context_data(**kwargs)

        feed=Auction.objects.filter(admin_status="approved", status="added")

        context['feed'] = feed

        return context


class Payment(TemplateView):

    template_name= 'user/payment.html'

    def get_context_data(self, **kwargs):


        context = super(Payment,self).get_context_data(**kwargs)
        feed=Auction.objects.filter(admin_status="approved", status="added")

        context['feed'] = feed

        return context


class Make_Payment(TemplateView):

    template_name= 'user/payment.html'

    def dispatch(self,request,*args,**kwargs):

        pid = self.request.user.id

        cus = Customer_Reg.objects.get(user_id=self.request.user.id)

        ch=fixed_auction.objects.filter(customer_id=cus.id)

        print(ch)

        for i in ch:

            i.payment='paid'

            i.save()

        return render(request,'user/user_index.html',{'message':" payment Successfull, Check
Booking Details"})


class booking(TemplateView):

    template_name ='user/booking.html'

    def get_context_data(self, **kwargs):



        context = super(booking,self).get_context_data(**kwargs)

        cus = Customer_Reg.objects.get(user_id=self.request.user.id)

        feed=fixed_auction.objects.filter(customer_id=cus.id, payment="paid")

        context['feed'] = feed

        return context
```