* 1. **INTRODUCTION**

SportifyCampus is a web-based system designed to efficiently manage sports events within a college setting. The system aims to simplify event organization, making it easier and less time-consuming. Built upon the Python Django framework, this project offers a robust foundation for seamless event management. Designed with efficiency in mind, SportifyCampus aims to simplify the event management process, minimizing time-consuming tasks and maximizing student engagement. By providing a user-friendly interface, students can effortlessly register and apply for their desired events, eliminating the barriers to participation.

One of the key features of SportifyCampus is its comprehensive event tracking system. Students can easily access event schedules, rules, results, and other pertinent information, ensuring they stay informed every step of the way. This accessibility not only enhances the overall student experience but also encourages greater involvement in sports activities. Moreover, SportifyCampus offers invaluable benefits to event coordinators. By centralizing data management, the platform reduces administrative burdens and mitigates the risk of data loss. This empowers coordinators to allocate their time and resources more effectively, ultimately leading to smoother event execution.

* 1. **PROBLEM STATEMENT**

Inefficient organization and management of college sports events pose significant challenges for both students and event coordinators. Current systems often lack user-friendly interfaces, leading to difficulties in event registration, scheduling, and communication. The cumbersome nature of these processes results in decreased student participation and engagement in sports activities. Additionally, coordinators face considerable administrative burdens, including manual data management and the risk of data loss. The need of the hour is a comprehensive solution that streamlines event management, enhances student participation, and reduces the workload for coordinators. Addressing these challenges requires the development of a web-based system that offers intuitive event registration, accessible event information, and efficient data management capabilities. Such a solution would not only improve the overall experience of college sports events but also foster a vibrant sports culture within the campus community. Here are few issues with the existing system:

* Inefficient Event Organization: Current systems lack user-friendly interfaces and efficient workflows for organizing college sports events, leading to confusion and frustration among students and coordinators.
* Difficulty in Event Registration: Students often encounter challenges when registering for sports events, such as complex forms or unclear procedures, resulting in decreased participation rates.
* Lack of Event Information: Accessing event schedules, rules, and results is often cumbersome and inconvenient, leading to disengagement and missed opportunities for students to participate in sports activities.
* Administrative Burden on Coordinators: Event coordinators face significant administrative tasks, including manual data entry, tracking participant information, and managing event logistics, which can be time-consuming and prone to errors.
* Risk of Data Loss: Traditional methods of data management, such as spreadsheets or paper records, pose a risk of data loss or inconsistency, jeopardizing the integrity of event information and causing logistical challenges.
* Decreased Student Participation: Cumbersome event management processes and lack of accessible information contribute to decreased student interest and participation in sports activities, impacting the overall sports culture within the college community.
  1. **SCOPE AND RELEVANCE OF PROJECT**

Organizing a sports event can be time consuming and tough, so the scope and relevance of the project is very high.

**Scope:**

SportifyCampus encompasses a wide range of functionalities aimed at revolutionizing college sports event management. The system facilitates event registration and application processes for students through a user-friendly interface, streamlining participation. It also provides comprehensive event tracking capabilities, allowing students to access schedules, rules, and results conveniently. Additionally, SportifyCampus offers administrative tools to coordinators, enabling efficient data management and communication. The project scope includes the development of scalable and flexible features built upon the Python Django framework, ensuring adaptability to evolving needs and future enhancements.

**Relevance:**

SportifyCampus addresses critical challenges faced by colleges in organizing sports events efficiently. By simplifying event organization and minimizing time-consuming tasks, the system enhances operational efficiency and reduces administrative burdens for coordinators. Its user-friendly interface promotes greater student engagement and participation in sports activities, fostering a vibrant sports culture within the college community. The comprehensive event tracking system ensures transparency and accessibility to event-related information, empowering students to make informed decisions and stay actively involved. Overall, SportifyCampus is a valuable tool that aligns with the strategic goals of colleges to promote sports and enhance student experiences.

**1.4. OBJECTIVE**

The primary objective of SportifyCampus is to streamline the process of sports event management within the college environment. By offering a user-friendly platform, the project aims to simplify event registration and application procedures for students, thereby increasing participation rates. Additionally, SportifyCampus seeks to enhance accessibility to event information, including schedules, rules, and results, making it easier for students to stay informed and engaged. Ultimately, the project endeavors to cultivate a thriving sports culture within the college community by removing barriers to participation and promoting active involvement in sports activities.

* 1. **INTRODUCTION**

System analysis for SportifyCampus is a pivotal process aimed at thoroughly evaluating the functionality, performance, and architecture of the web-based college sports event management system. Through meticulous examination, this analysis seeks to identify the strengths and weaknesses of SportifyCampus, ensuring it effectively meets the specific requirements of students and event coordinators while enhancing the overall efficiency and effectiveness of managing college sports events. By scrutinizing the various components of SportifyCampus, stakeholders can assess its features, usability, and scalability, enabling informed decisions to optimize its performance. Furthermore, system analysis plays a critical role in pinpointing potential areas for improvement and enhancement, such as event registration processes, event information accessibility, and administrative functionalities. Consequently, the systematic evaluation provided by system analysis forms the groundwork for refining and advancing SportifyCampus, establishing a well-structured and user-friendly platform for managing college sports events.

* 1. **EXISTING SYSTEM**

The current system relies heavily on manual processes for managing college sports events, posing challenges for both students and event coordinators. Students are required to physically visit department offices or approach event coordinators to provide information and register for desired events. This manual registration process is time-consuming and prone to errors, increasing the likelihood of data mix-ups due to paperwork. Moreover, activities such as result announcements and scoreboard updates are also handled manually, adding to administrative burdens and the risk of inaccuracies.

Furthermore, students encounter difficulties in accessing comprehensive information about games conducted across various venues. The lack of a centralized system results in fragmented event information, forcing students to invest significant time and effort in gathering details about games. This inconvenience extends to obtaining essential details such as event timings, scores, and venue locations, which are scattered and challenging to locate within the manual system. Consequently, students may miss important updates or fail to stay informed about event schedules, negatively impacting their participation and overall experience in college sports activities.

* + 1. **LIMITATION OF EXISTING SYSTEM**

**Manual Registration Process**

Having students register manually at the department can be time-consuming for both the students and the staff managing the registrations. It often involves filling out paperwork, waiting in queues, and potentially dealing with human errors during the process.

**Tedious Task of Managing Simultaneous Registrations**

When multiple students try to register simultaneously, it can overwhelm the staff responsible for managing registrations. This can lead to long wait times, frustrated students, and increased chances of errors due to rushed processing.

**Errors in Data Entry**

Manual data entry is inherently prone to errors. Staff may make typos, misinterpret handwriting, or accidentally misplace or lose paperwork. These errors can lead to inaccuracies in student records, which can cause issues down the line, such as incorrect contact information or registration details.

**Difficulty in Tracking Changes**

Keeping track of changes to game schedules, updates to scoreboards, or any other relevant information manually can be challenging. It requires constant vigilance and communication to ensure that all stakeholders are aware of the latest updates, which can be time-consuming and prone to human error.

**Risk of Data Loss or Damage**

Paperwork can be susceptible to damage or loss, whether due to accidents, natural disasters, or simple misplacement. This puts important student registration information at risk and can lead to significant disruptions in the management of student activities and events.

* 1. **PROPOSED SYSTEM**

The proposed College Sports Management System,Sportifycampus offers students a comprehensive platform to access information about various games and their venues. With this system, students can effortlessly obtain details about upcoming events, eliminating the need to navigate through multiple sources. By integrating registration processes into a single application, the system streamlines efficiency, saving students valuable time. Students can register for and participate in events directly from the provided list, simplifying the engagement process. Timely updates on event timings, scores, results, and venues are readily available, enhancing convenience and accessibility. This centralized approach not only benefits students but also facilitates event coordinators in efficiently managing and organizing their respective events.

Moreover, the system ensures the secure storage of essential data, such as student details and participant information, minimizing the risk of errors and data loss. Flexibility is another key feature, as it allows for easy updates to schedules and other event details, ensuring consistency and accuracy.

Furthermore, the inclusion of a dedicated blog section provides students with a platform to share articles and insights about various events, fostering engagement and community participation. This feature enriches the overall user experience and encourages collaborative interaction within the college sports community.

The College Sports Management System offers a comprehensive and user-friendly solution for accessing information, managing events, and fostering community engagement within the college sports domain.

* + 1. **ADVANTAGES OF PROPOSED SYSTEM**

1. Convenience of Registration: Imagine never having to run around campus to register for different sports events. With this application, students can sign up for their favorite games with just a few clicks. Whether it's basketball, soccer, or even chess tournaments, everything is accessible from one central platform, saving students valuable time and effort.

2. Effortless Tracking: No more missing out on game changes or scrambling to find updated schedules. This application keeps students informed in real-time about any modifications to game schedules, venue shifts, or updated scoreboards. Whether it's a last-minute venue change or a delayed kickoff, students stay in the loop effortlessly, ensuring they never miss a moment of the action.

3. Data Security and Accuracy: Gone are the days of worrying about misplaced registration forms or inaccurate data entries. This application ensures the security and integrity of student information, eliminating the risk of data loss or damage. With error-free data management, students can trust that their registration details are handled with precision and confidentiality.

4. Rich Content Experience: Beyond just registration and tracking, this platform offers a dynamic content hub where students can engage with blogs and articles covering the latest sports events and happenings. Whether it's match previews, or post-game analysis, students have access to a wealth of sports-related content, enriching their overall experience and fostering a deeper connection to the sporting community.

5. Increased Student Participation: By streamlining the registration process, providing seamless tracking of game updates, ensuring data security, and offering engaging content, this application serves as a catalyst for boosting student participation in sports events. With the barriers to entry significantly lowered and the overall experience enhanced, more students are encouraged to actively take part in sports activities, fostering a healthier and more vibrant campus community.

* 1. **FEASIBILITY STUDY**

A feasibility study for SportifyCampus serves as a critical analysis to assess the viability and potential success of implementing this web-based system for managing sports events within college campuses. It aims to determine whether the project is practical and worthwhile by evaluating various aspects such as economic, technical, legal, and scheduling considerations. The study seeks to understand the core functionalities of SportifyCampus, identify potential challenges in its implementation, and ascertain its feasibility in terms of available resources, technology requirements, and expected return on investment (ROI). By conducting this feasibility study, the project team can make informed decisions about the viability of SportifyCampus and address any obstacles before committing substantial time and resources.

Furthermore, the study enables SportifyCampus's management to present a compelling case to investors or stakeholders, demonstrating that investing in this platform is a strategic choice with significant potential for success in the competitive landscape of college sports management.Ultimately, the feasibility study plays a crucial role in shaping the strategic direction of SportifyCampus, ensuring that it is well-prepared for successful development and implementation. It provides valuable insights that guide decision-making and helps mitigate risks, ultimately contributing to the platform's effectiveness in simplifying event organization and maximizing student engagement in sports activities on campus.

* + 1. **TECHNICAL FEASIBILITY**

Technical feasibility concerns whether a project can meet its performance objectives. This involves the assessing of whether the system can be developed and implemented using the available technology infrastructure. It examines the factors such as hardware and software requirements, compatibility with existing systems, availability of skilled resources and potential technical challenges. The study helps to determine if the proposed system can be effectively built and integrated within the existing environment. As there is enough hardware and software requirements, the system is technically feasible.

### **OPERATIONAL FEASIBILITY**

Operational feasibility is the 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 requirements identified in the requirements analysis phase of system development.

The operational feasibility assessment focuses on the degree to which the proposed development project fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture and existing business processes.

To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, reducibility, disposability, sustainability, affordability and others. These parameters are required to be considered at the early stages of design if desired operational behaviors are to be realized. A system design and development requires appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phase.

### **2.4.3. ECONOMIC FEASIBILITY**

Economic feasibility is a kind of cost-benefit analysis of the examined project, which assesses whether it is possible to implement it. This term means the assessment and analysis of a project's potential to support the decision-making process by objectively and rationally identifying its strengths, weaknesses, opportunities and risks associated with it, the resources that will be needed to implement the project, and an assessment of its chances of success. It consists of market analysis, economic analysis, technical and strategic analysis.

**2.5 SOFTWARE ENGINEERING PARADIGM APPLIED**

The project follows Agile Process Model. The meaning of Agile is swift or versatile. **“**Agile process model" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

Phases in the Agile model are:

1. **Requirements gathering:** In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.
2. **Design the requirements:** When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.
3. **Construction/ iteration:** When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.
4. **Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug.
5. **Deployment:** In this phase, the team issues a product for the user's work environment.
6. **Feedback:** After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

**3.1 INTRODUCTION**

System design is a process of developing specifications for a candidate system that meet the criteria established in the system analysis. Major steps in design are the preparation of the input forms and output reports in a form applicable to user. The main objective of the system design is to use the package easily by any computer operator. System design is the creative act of invention, developing new inputs, a database, offline files, method, procedure and output for processing business to meet an organization objective. System design builds information gathered during the system analysis.

**3.2 DATABASE DESIGN**

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and effectively. The general objective is to make information easy and quick to access for the users. The general theme behind a database is to integrate all information. Database design is recognized as a standard of management information system and is available virtually for every computer system. In database design several specific objectives are considered.

* Ease of learning and use
* Controlled redundancy
* Data independency
* More information at low cost
* Accuracy and integrity
* Recovery and failure
* Privacy and security

A database is an integrated collection of data and provides centralized access to the data. Usually the centralized data managing the software is called RDBMS. The main significant difference between RDBMS and other DBMS is the separation of data as seen by the program and data has indirect access to store device. This is the difference between logical and physical data.

**3.2.1. ENTITY RELATIONSHIP MODEL**

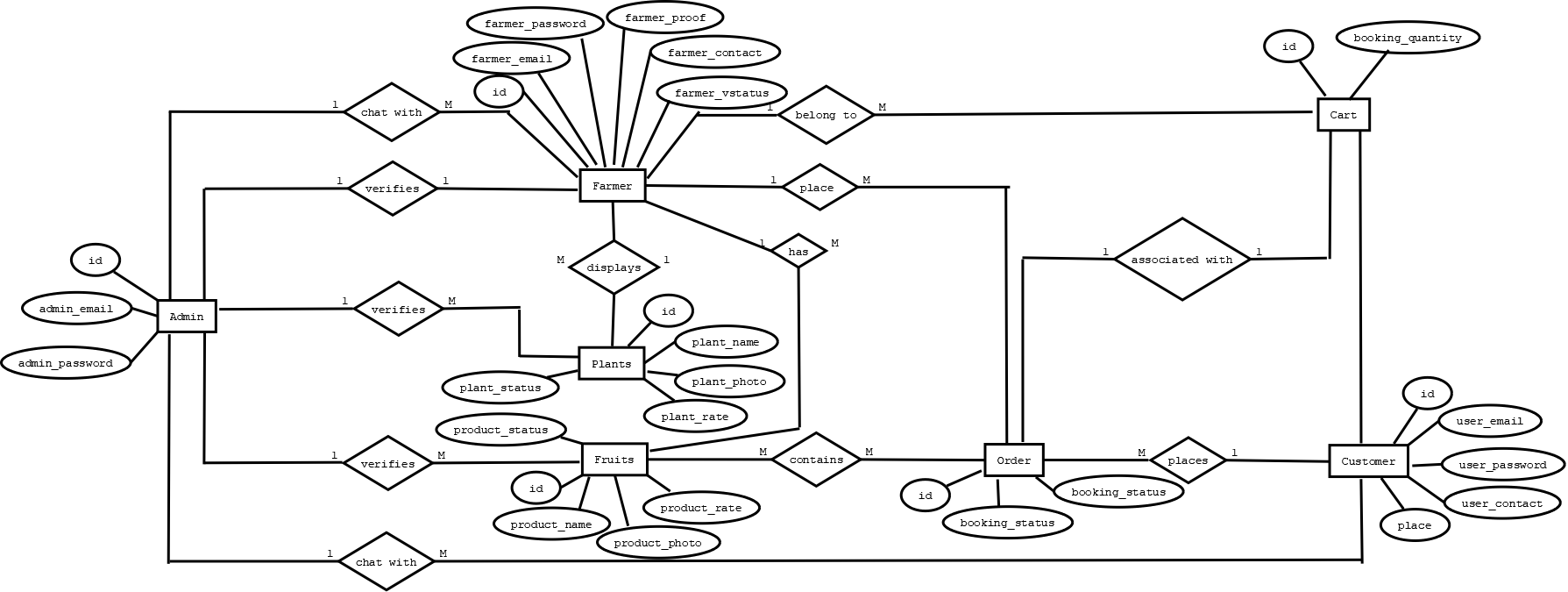
An entity relationship diagram (ERD) is a data modelling technique that graphically illustrates an information system’s entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

The elements of an ERD are:

* Entities:
* Relationships

* Attributes

**E-R DIAGRAM**



**3.2.2 TABLE DESIGN**

Table is a collection of complete details about a particular subject. These data are saved in rows and columns. Hence, rows are called RECORDS and columns of each row are called FIELDS. Data is stored in tables. The items and data, which are entered in the input, form id directly stored in this table using linking of database. I can link more than one table to input forms. I can collect the details from the different tables to display on the output.

**TABLES**

1. Table name: tbl\_adminreg

Description: To store admin data

Primary key: id

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| Id | bigint(20) | Id of admin |
| admin\_name | varchar(50) | Name of admin |
| admin\_email | varchar(50) | Email of admin |
| admin\_password | varchar(30) | Password of admin |

1. Table name: tbl\_dept

Description: To store district data

Primary key: id

|  |  |  |
| --- | --- | --- |
| Field name | Datatype | Description |
| Id | CharField | Department id |
| dept\_name | CharField | Department name |

1. Table name: tbl\_course

Description: To store course data

Primary key: id

Foreign key: dept\_id (tbl\_dept)

|  |  |  |
| --- | --- | --- |
| Field Name | Field Type | Description |
| Id | CharField |  |
| course\_name | CharField |  |
| dept\_id | CharField |  |

1. Table name: tbl\_category

Description: To store different categories of event

Primary key: id

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| id | bigint(20) | Id of customer |
| category\_name | varchar(30) | Name of customer |

1. Table name: tbl\_subcategory

Description: To store subcategory data

Primary key: id

Foreign key: category\_id (tbl\_category)

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| id | bigint(20) | Id of subcategory |
| subcategory\_name | varchar(30) | Name of subcategory |
| category\_id | bigint(20) | Category id of subcategory |

6. Table name: tbl\_sem

Description: To store subcategory data

Primary key: id

Foreign key: category\_id (tbl\_category)

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| id | bigint(20) | Id of farmer |
| subcategory\_name | varchar(30) | Name of farmer |
| category\_id | Bigint | Phone number of farmer |

7. Table name: tbl\_academicyear

Description: To store subcategory data

Primary key: id

Foreign key: category\_id (tbl\_category)

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| id | bigint(20) | Id of farmer |
| subcategory\_name | varchar(30) | Name of farmer |
| category\_id | Bigint | Phone number of farmer |

8. Table name: tbl\_coordinator

Description: To store coordinator informatiom

Primary key: id

Foreign key: coordinator\_id (tbl\_farmer)

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| id | bigint(20) | Id of coordinator |
| coordinator\_name | varchar(30) | Store coordinator name |
| coordinator\_gender | varchar(50) | Store coordinator gender |
| coordinator\_contact | varchar(100) | Store coordinator contact |
| coordinator\_email | varchar(10) | Store coordinator email |
| coordinator\_address | bigint(20) | Store coordinator address |
| coordinator\_password | Int | Store coordinator password |
| coordinator\_photo | varchar(100) | Store coordinator photo |
| coordinator\_proof |  | Store coordinator proof |
| coordinator\_status |  | Store status of the coordinator |
| coordinator\_doj |  | Store coordinator date of joining |

9. Table name: tbl\_student

Description: To store student details

Primary key: id

Foreign key: academic\_ic(tbl\_academicyear),sem\_id(tbl\_sem),course\_id(tbl\_course)

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| id | bigint(20) | Store student id |
| student\_name | varchar(30) | Store student name |
| student\_gender | varchar(50) | Store student gender |
| student\_contact | varchar(100) | Store contact of student |
| student\_email | varchar(10) | Store email of the student |
| student\_address | bigint(20) | Store address of the student |
| student\_password | Int | Store password of the student |
| student\_photo | varchar(100) | Store photo of the student |
| student\_proof | varchar(10) | Store proof of the student |
| student\_status |  | Store status of the student |
| student\_doj |  | Store date of joining of the student |
| academic\_id |  | Store academic id |
| course\_id |  | Store course id |
| sem\_id |  | Store sem id |

10. Table name: tbl\_newevent

Description: To store event details

Primary key: id

Foreign key:

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| id | bigint(20) | Id of event |
| event\_gender | varchar(30) | Store event gender |
| event\_date | varchar(50) | Store event date |
| event\_time | varchar(100) | Store event time |
| event\_rules | varchar(10) | Store event rules |
| event\_venue | bigint(20) | Store event venue |
| event\_poster | Int | Store event poster |
| event\_type\_name | Radio | Store event type |
| event\_teammate\_count | varchar(10) | Store event teammate count |
| event\_status |  | Store event status |
| category\_id |  | Store category id |
| subcategory\_id |  | Store subcategory id |
| coordinator\_id |  | Coordinator id |

11. Table name: tbl\_participants

Description: To store participant details

Primary key: id

Foreign key: event\_id (tbl\_newevent),student\_id (tbl\_student)

|  |  |  |
| --- | --- | --- |
| **FIELD NAME** | **FIELD TYPE** | **DESCRIPTION** |
| id | bigint(20) | Store participant id |
| participant\_status | varchar(10) | Store participant status |
| paricipant\_date | Date | Store participant date |
| event\_id | bigint(20) | Store event id |
| student\_id | bigint(20) | Store student id |

12.Table name: tbl\_teammates

Description: To store teammate names of team events

Primary key: id

Foreign key: participant\_id(tbl\_participants)

|  |  |  |
| --- | --- | --- |
| **FIELD NAME** | **FIELD TYPE** | **DESCRIPTION** |
| id | bigint(20) | Store teammate id |
| teammate\_name | bigint(20) | Store teammate name |
| participant\_id | bigint(20) | Store participant id |

13.Table name: tbl\_results

Description: To store results of events

Primary key: id

Foreign key: booking\_id (tbl\_booking), user\_id (tbl\_user), farmer\_id (tbl\_farmer), from\_farmer (tbl\_farmer)

|  |  |  |
| --- | --- | --- |
| **FIELD NAME** | **FIELD TYPE** | **DESCRIPTION** |
| id | bigint(20) | Store result id |
| result\_date | date (10) | Store result date |
| result\_position | bigint(20) | Store result position |
| result\_score | Int | Store result score |
| participant\_id | Int | Store participant id |
| dept\_id | Int | Store department id |

14.Table name: tbl\_notification

Description: To store Notifications to the user

Primary key: id

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| id | bigint(20) | Store notification id |
| notification | Date | Store notification |

15.Table name: tbl\_complaint

Description: To store withdraw details

Primary key: id

Foreign key: farmer\_id (tbl\_farmer)

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
|  | bigint(20) | Store withdraw id |
|  | Int | Store withdraw balance |
|  | Int | Store withdraw amount |
|  | Int | Store withdraw status |
|  | date | Store withdraw date |
|  | time | Store withdraw time |
|  | bigint(20) | Store farmer id |

16.Table name: tbl\_feedback

Description: To store market booking details

Primary key: id

Foreign key: user\_id (tbl\_user),farmer\_id (tbl\_farmer), product\_id (tbl\_product)

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Field Type** | **Description** |
| id | bigint(20) | Store marketbooking id |
| booking\_date | date | Store product id |
| product\_id | bigint(20) | Store comment content |
| booking\_quantity | bigint(20) | Store booking quantity |
| farmer\_id | bigint(20) | Store farmer id |
| user\_id | bigint(20) | Store user id |

**3.3 OBJECT ORIENTED DESIGN**

**UML DIAGRAM**

**3.3.1 Use case Diagram**

The most important aspect is to capture the dynamic behaviour when modelling a system. Dynamic means the behaviour of the system when it is operating. Use case diagram is one of the diagrams that describe the dynamic behaviour of the system. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analysed to gather its functionalities, use cases are prepared and actors are identified. When the initial task is complete, the use case diagrams are modelled to present the outside view.

**3.3.1.1 Symbols in Use-case Diagram:**

An actor represents a coherent set of roles that users of a system play when interacting with the use cases of the system. Actors can be anything-humans, devices or other systems. One physical object can play several roles and so can be modelled by several actors.

Use cases describe what a system does and not how it does. A use case contains multiple scenarios, each of which describes a specific flow of events through the use cases. The behavior of use case is specified by describing the scenarios clearly enough for outsiders to understand.

Connection symbol represent the connection between actor and use cases.

Include symbol include the relationship between the use cases. One use case (base) includes the functionality of another (inclusion case).

**USECASE DIAGRAM**

**3.3.2 SEQUENCE DIAGRAM**

A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a constant of a message sequence chart. A sequence diagram shows the interactions of objects arranged in a time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality. Sequence diagrams are typically associated with use case realizations in the logical view of the system under development

**3.3.2.1 Symbols in Sequence Diagram**

**•Life lines & life line boxes: -** Active objects can be an object which is an instance of a class and can be drawn with rectangular box called the life line box with its name is specified within the box. The long dashed line tailing the object is called life lines.



•**Message:** -It is used to illustrate the communication between different active objects. Each message between object is represented with a message, expressions and filled arrow solid lines from the calling active objects life line to the recipient life lines.

• **Synchronous:** - This type of message is used when it is important that a message is received & completed before execution of control flow begins.

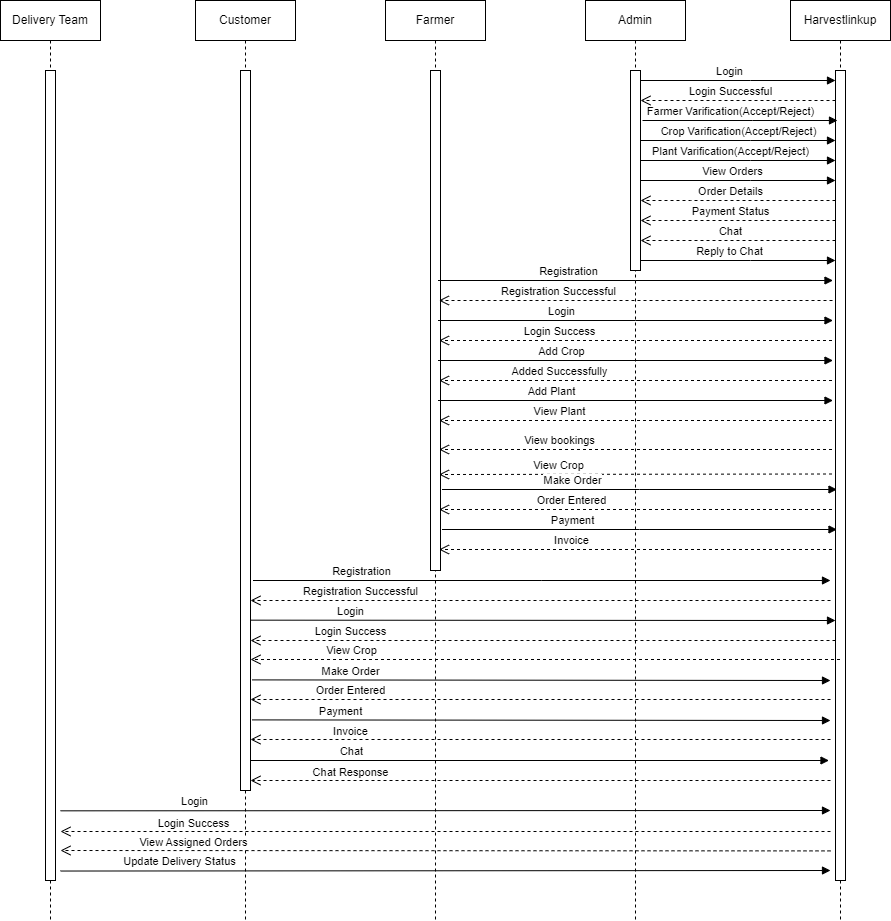
•**Return**: -Return message shows that the control flow has return into the calling active objects and the synchronous messages are completed its operations.

•**Asynchronous**: - This type of message is send from an active object and waits for a responds. This type of message is used when control flow doesn’t lead to interrupted before complete the operation.



• **Flat:** -This type of message is used when there is no distinction between asynchronous and synchronous messages

**SEQUENCE DIAGRAM**



**3.3.3 ACTIVITY DIAGRAM**

Activity diagrams are graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. UML, activity diagrams are intended to model both computational and organizational process. It shows the overall flow of control.

**Symbols in Activity Diagram**

**Action State**: - It is an automatic state once it started execution. It will come to completion without any interruption at the middle of execution.

**Transition**: - When the action or activity state completes the flow of control passes immediately to the next action or activity state. This is represented by using the solid lines.

**Branching:** - It specifies the alternative paths that are taken based on some Boolean expressions.

**Initial node:** - The initial node is a control node from where the activity is invoked. An activity may have more than one initial node.

**Final node**: - The final node shows the end of the activity. An activity can have more than one final node but the first one reached will stop the activity.

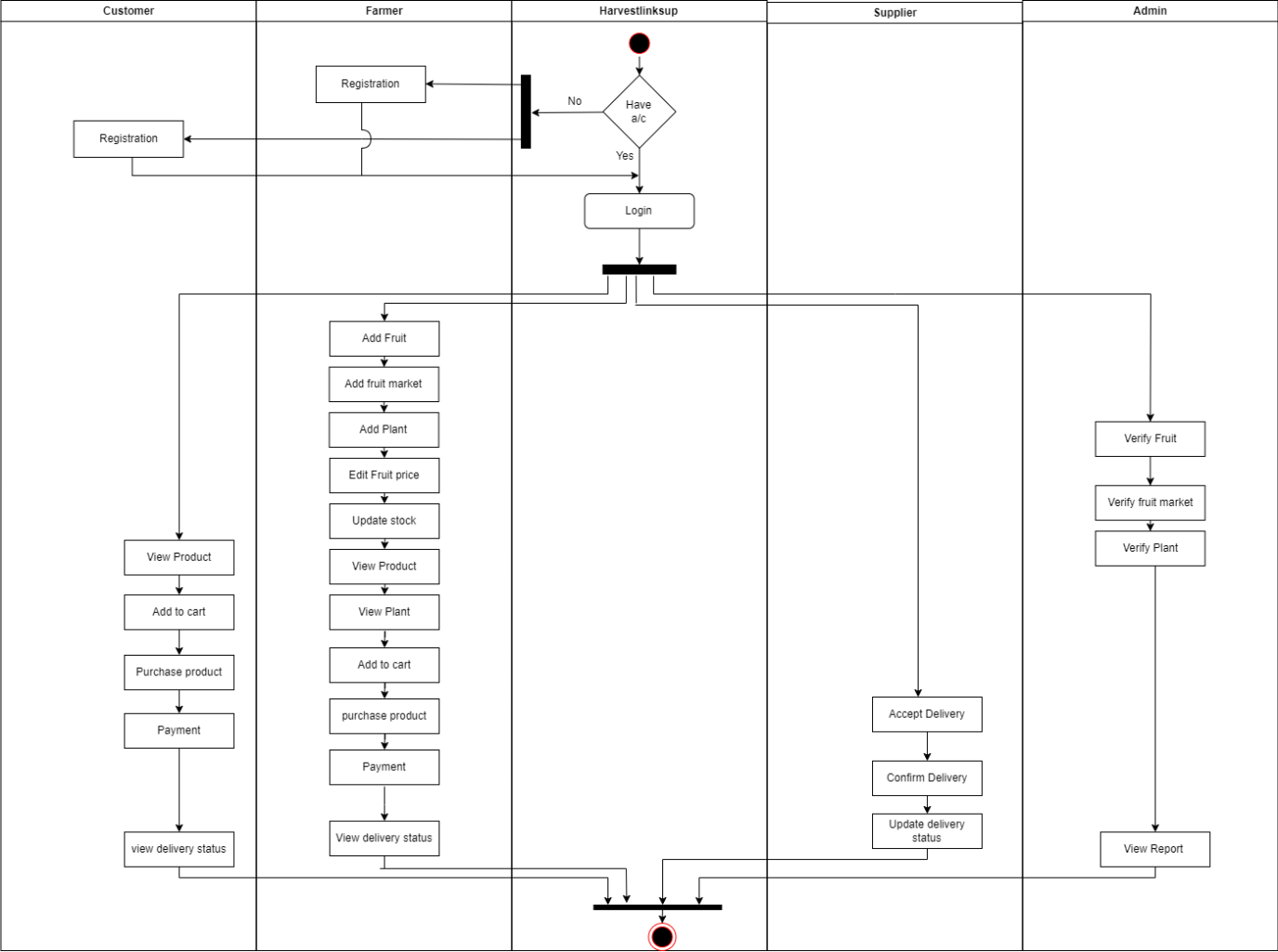
**Join:** - A join represents multiple incoming transitions and provide only one outgoing transitions.

****

**Fork**: - A Fork represents only one incoming transitions and provide multiple outgoing transitions.

****

**ACTIVITY DIAGRAM**



**3.4. MODULAR DESIGN**

Modular design is an important concept in software engineering that focuses on breaking down a system into smaller, manageable and independent modules or components. These modules can be developed and tested independently, making it easier to understand, maintain and extend the system. In the context of an SRS (Software Requirements Specification) document, modular design principles may be referenced or included to describe how the system should be organized and structured.

**3.4.1. MODULES DESCRIPTION**

**Module 1: Registration and Login Management Module**

**• Login**

Registered student, coordinator, and Admin can log into the system using their email and password. The system checks the details entered against the information stored in the database.

**• Session Management**

The system will maintain student, coordinator, Admin, sessions to keep them logged in while they interact with the application. Session expiration will be implemented to ensure security and is logged out when inactive.

**Student**

**• Student Registration**

New students can register by providing necessary information such as name, email address, password and contact number. These details are validated and error cases such as duplicate emails and invalid inputs are handled properly. Upon successful registration, a unique user ID and user profile is created for each customer.

**Coordinator**

**• Coordinator Registration**

Admin is responsible for new coordinator registration. Admin registers them by providing necessary information such as name, email address, password and contact number. These details are validated and error cases such as duplicate emails and invalid inputs are handled properly. Upon successful registration, a unique user ID and user profile is created for each customer.

**Module 2: Event Management Module**

1. **Administrator**

**• View, Verify and Delete product**

Admin can view the product details such as product name, details, category, price and imagebased on product details admin can verify and delete products

1. **Farmer**

**• Add, Edit Products**

This enables the Farmers to add products by giving product name, details, category, price and image and can edit the added products.

**• Update Stock**

Farmers can update the product stock.

**Module 3: Order Management Module**

1. **Administrator**

* View Orders

The system provides a centralized view of all incoming orders. Viewing orders include the customer/ Farmer information, order items and order timestamps.

1. **Customer / Farmer**

* Cart Management

This allows customers to easily add desired products to their shopping carts and displays the cart summary with selected products, quantities and total price. It also enabled the customers to adjust quantities or remove items from the cart. The system also enables real-time availability status to prevent ordering out-of-stock items.

* Checkout Process

This allows to guide the customers through a user-friendly checkout process and required to provide the information needed for further procedures.

* Order History

Each customer has access to their order history which shows their past purchases and details.

**Module 4: participant Management Module**

1. Administrator

* Monitor Payment Transactions

The administrators have access right to view and monitor payment transactions.

* Payment Gateway Integration

This helps the admin to integrate and manage payment gateways for secure and reliable transaction processing.

1. Customer / Farmers

* Payment Status Update

The customer / Farmer can view their status of payment if they have paid.

* Payment Alerts

Proper payment alerts are given for both successful and unsuccessful payment.

**Module 5: result Management Module**

1. Customer / Farmer

* Delivery Notification

Active notifications are received by the customer regarding the delivery.

* Delivery Status Update

The Customer/Farmer can view their status of delivery until the order is received.

1. Delivery Partner

* Update Delivery Status

The courier partner updates the status of the delivery regularly.

**3.5 INPUT DESIGN**

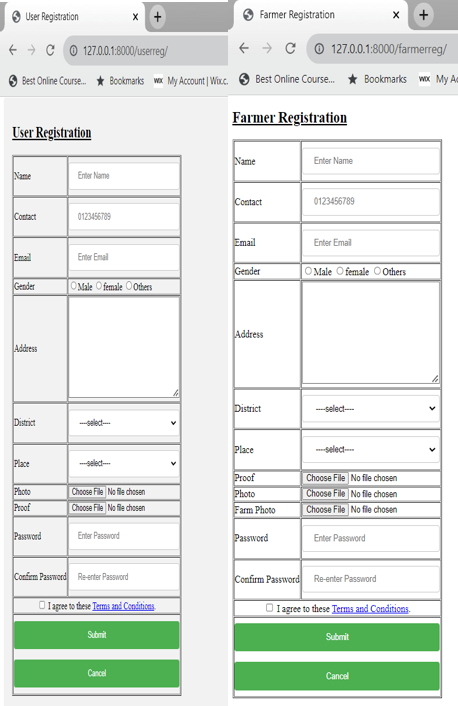
Input design is the process of converting user-oriented input to a based format. Inaccurate input data are the most common cause of errors in data processing. Errors entered by data entry operators can be controlled by input design.

The goal of designing input data is to make data entry as easy, logical and free from errors. When we approach input data design; we design the data source documents that capture the data and then select the media used to enter them into the computer.

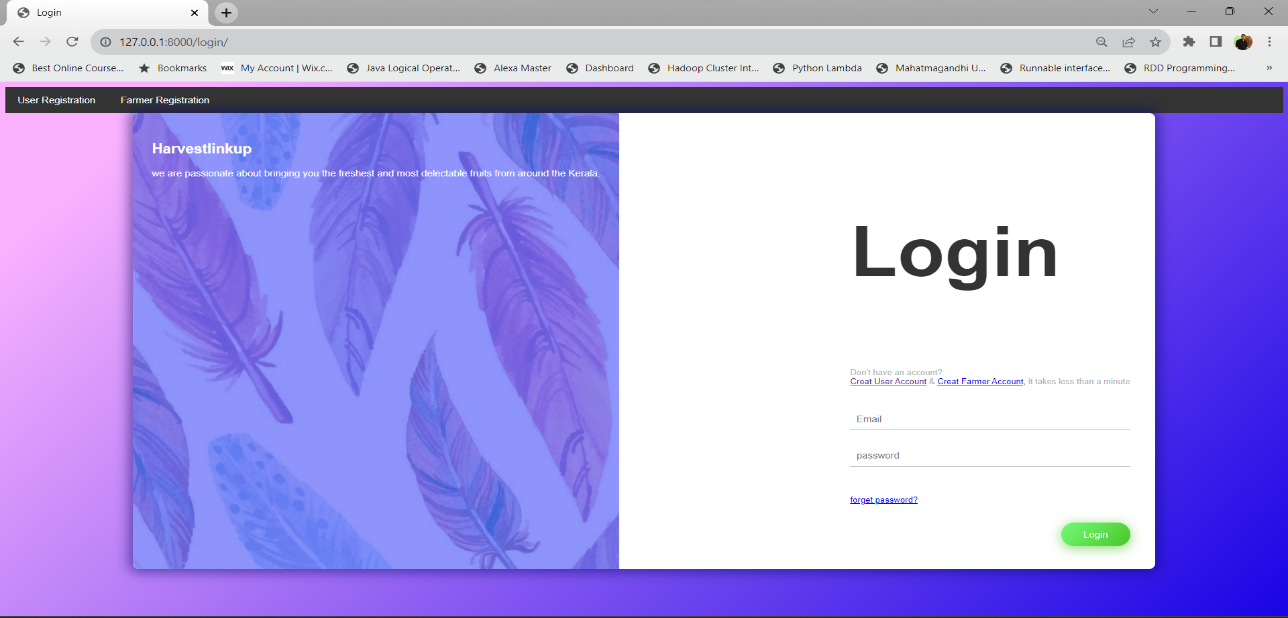
User –friendly screen format can reduce the burden on end users, who are not highly proficient in computers. An important step in input design stage is a design of source document. Source document is the form in which the data are initially captured. The next step is the design of document layout. In the layout organizes the document by placing information, where it will be noticed and establishes the appropriate sequence of items.

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

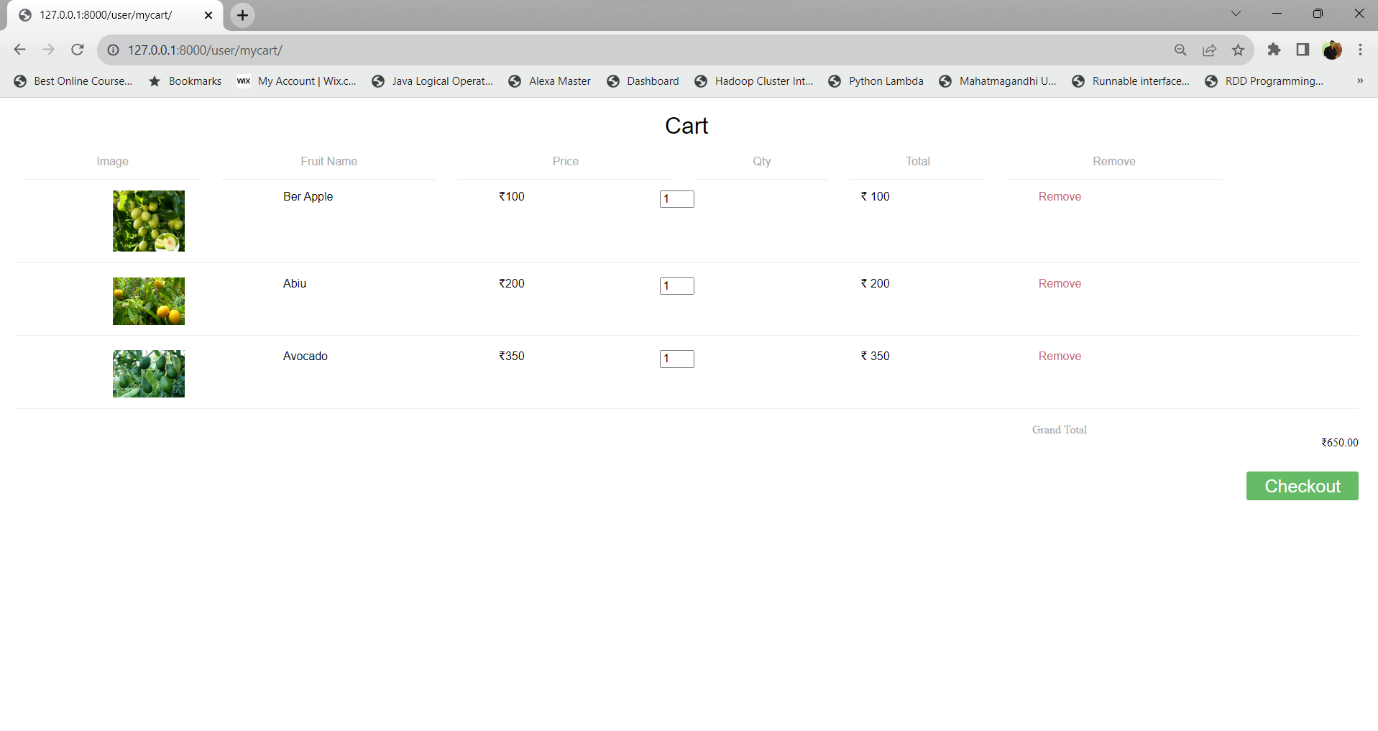
**Customer Registration**



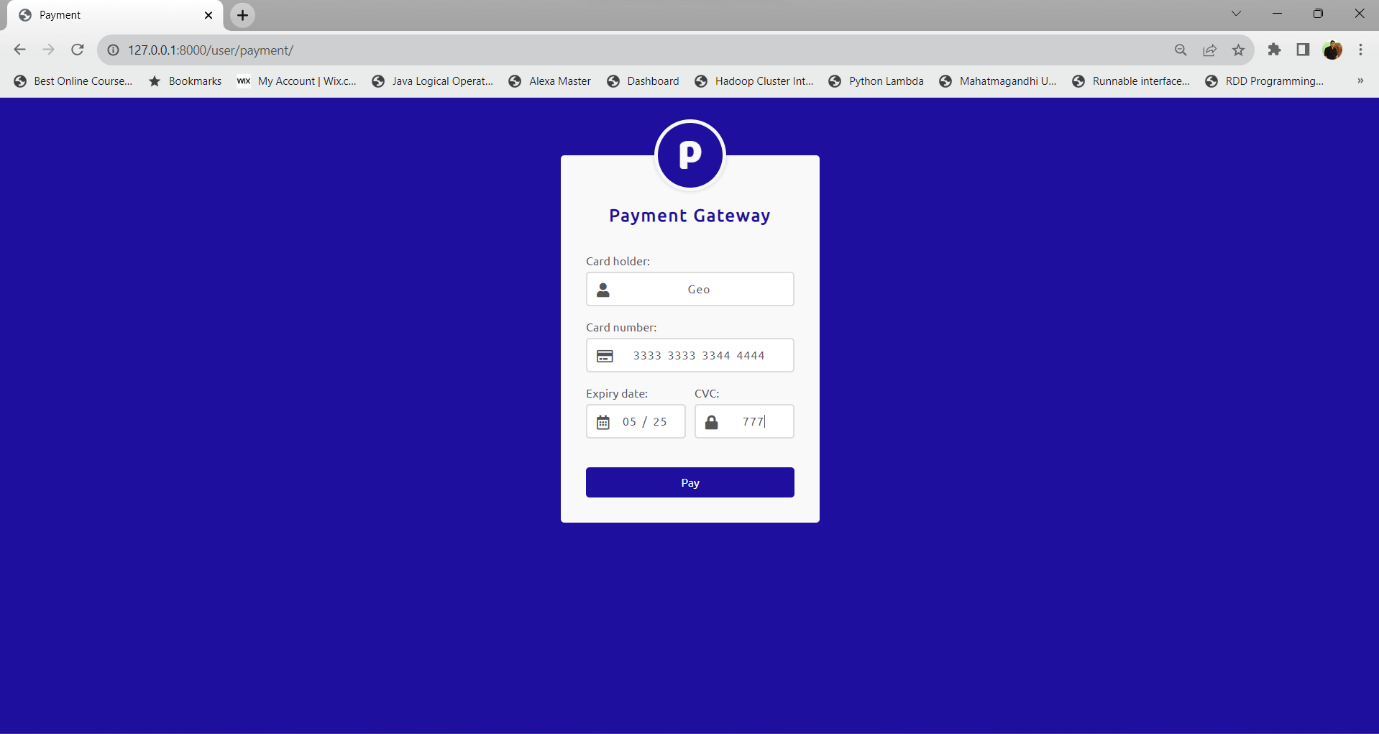
**Login**



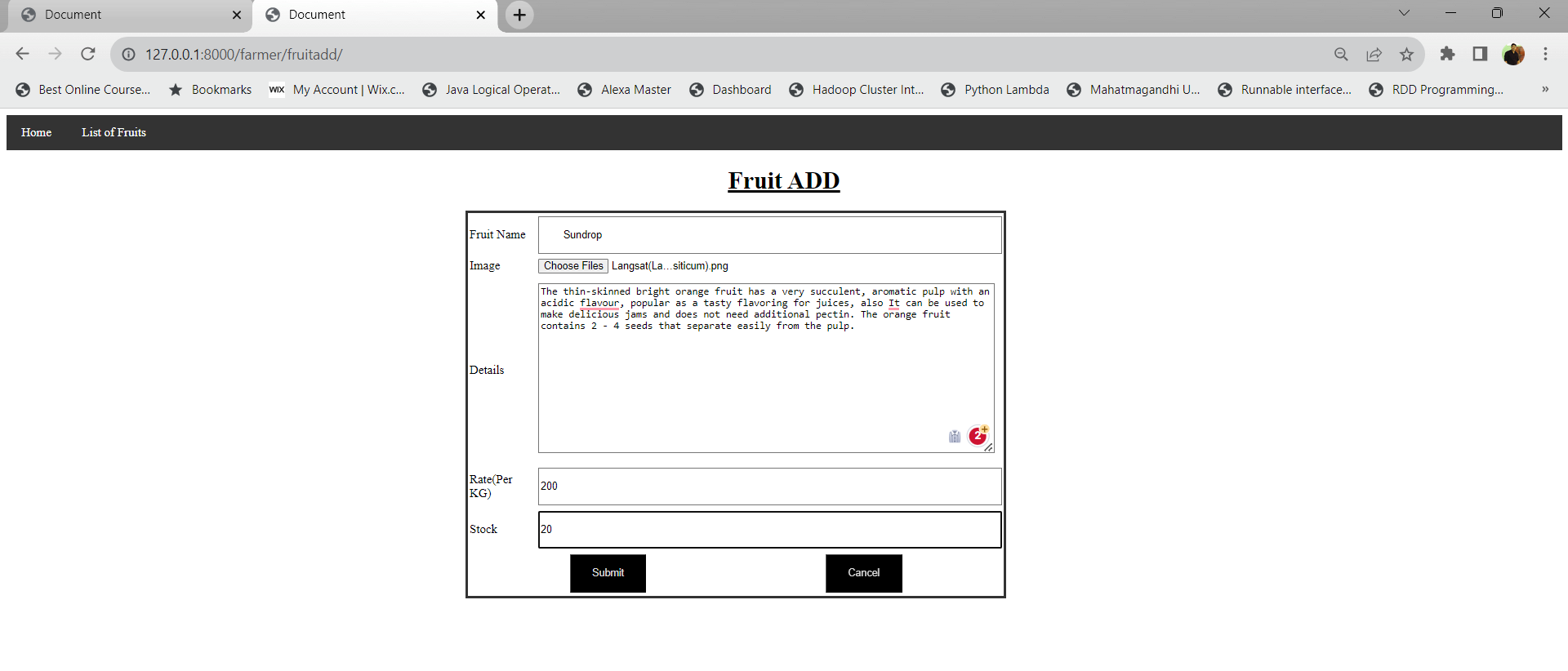
**Cart**



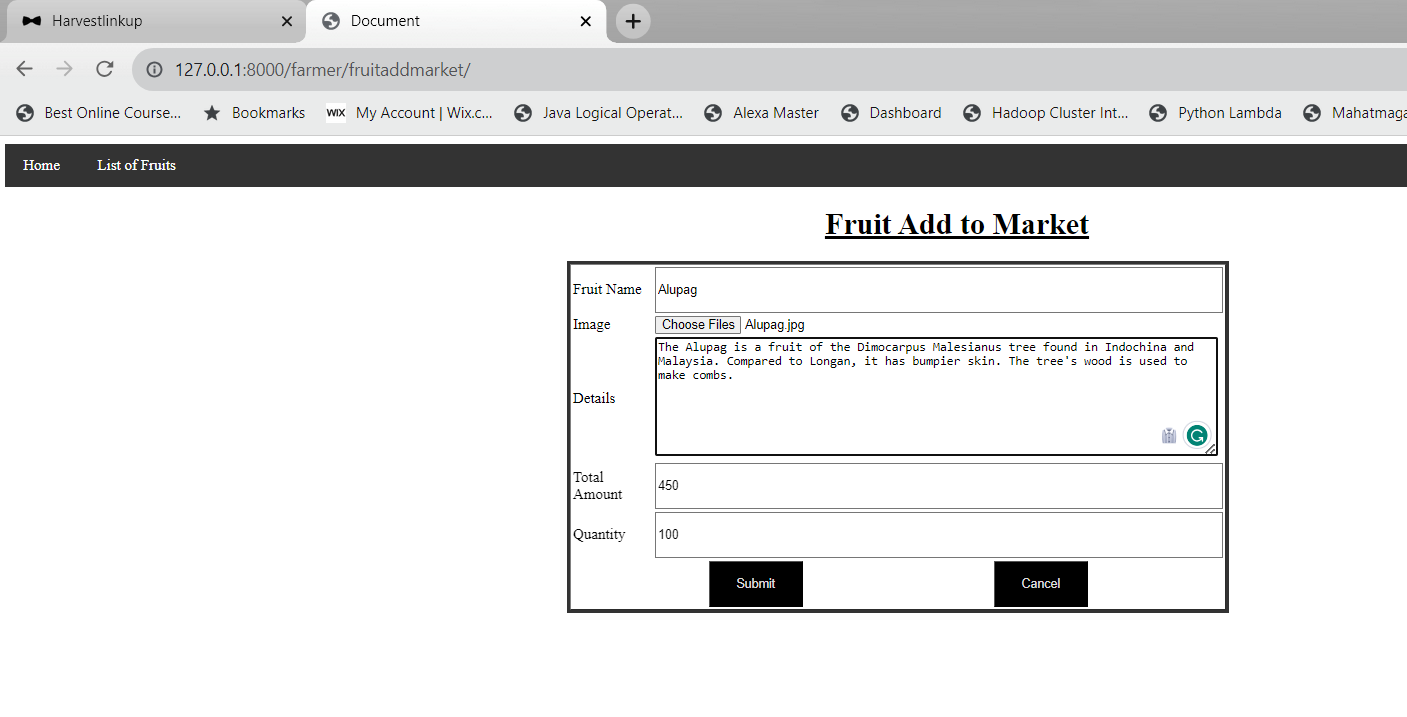
**Payment**

****

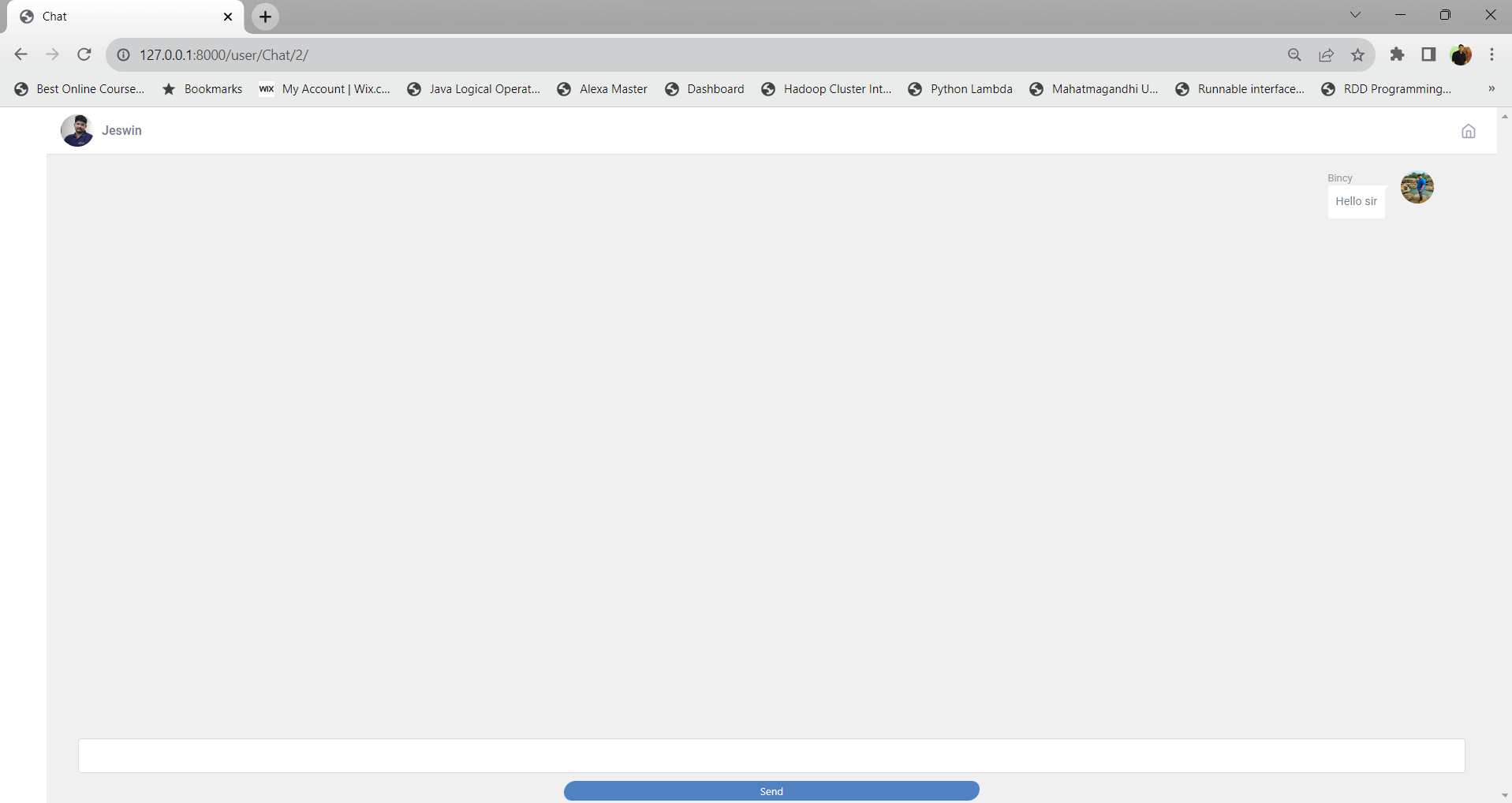
**Farmer Add Fruit**

****

**Farmer Add Fruit to Market**

****

**Customer- Admin Chat**

****

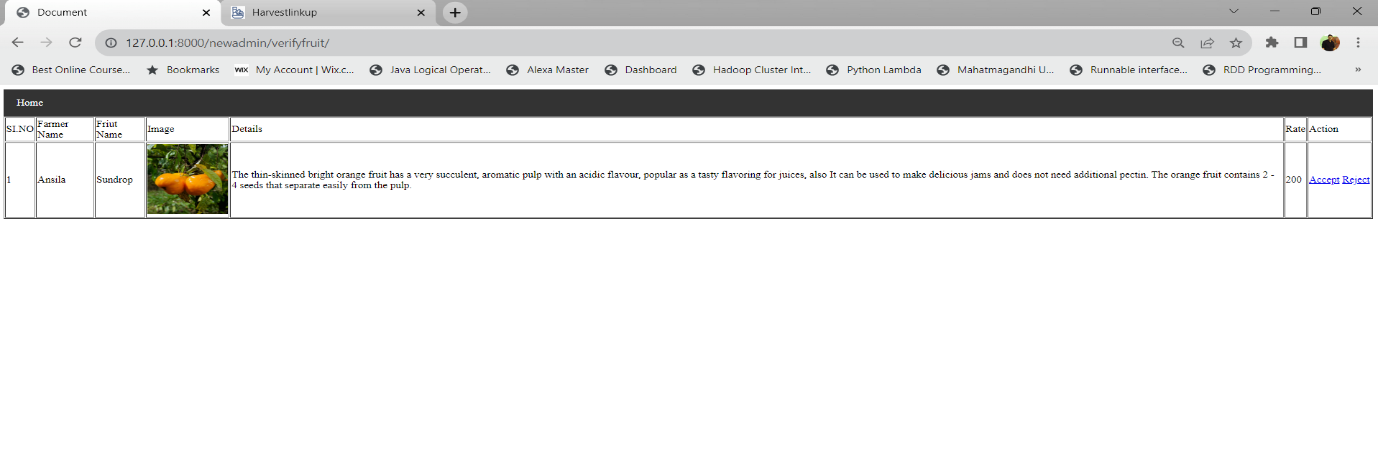
**3.6 OUTPUT DESIGN**

Computer Output is the most important and direct source of information to the user. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship and helps user decision-making.

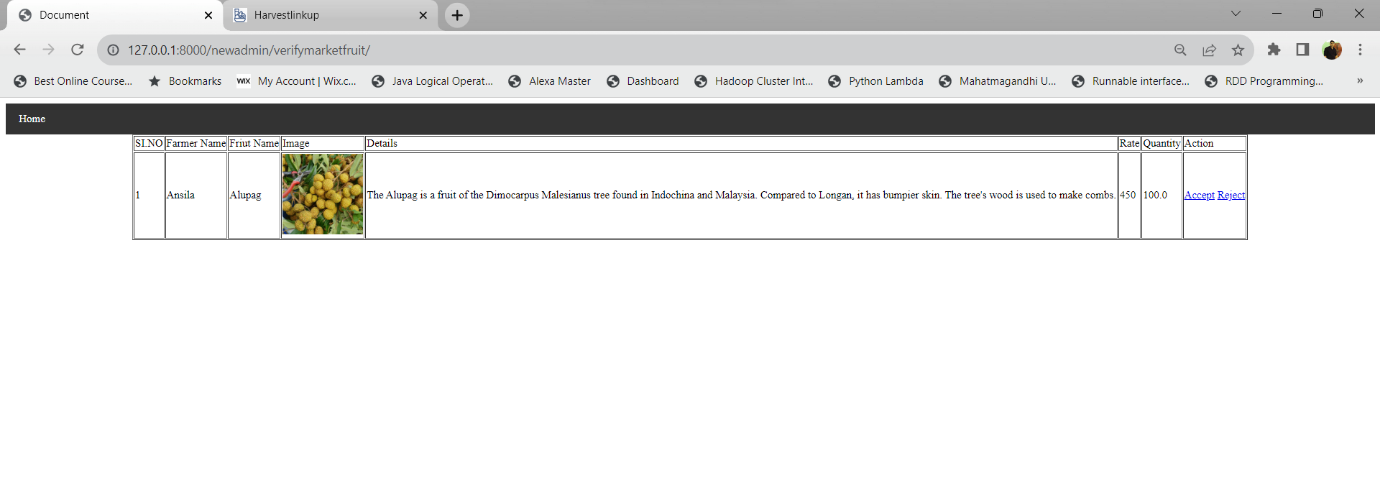
In the output design it is determined how the implementation is to be played for immediate need and also the hard copy output. A major form of input is a hard copy from the printer. Printouts should be designed around the output requirements of the user. Printers, CRT screen display are the examples for providing computer based output. The output design associated with the system includes the various reports of table generations and query executions.

Output design is one of the, most important features of the information system. The logical design of an information system is analogous to an engineering blue print of an auto mobile. It shows the major features and how they are related to one another. The outputs, inputs and databases are designed in this phase.

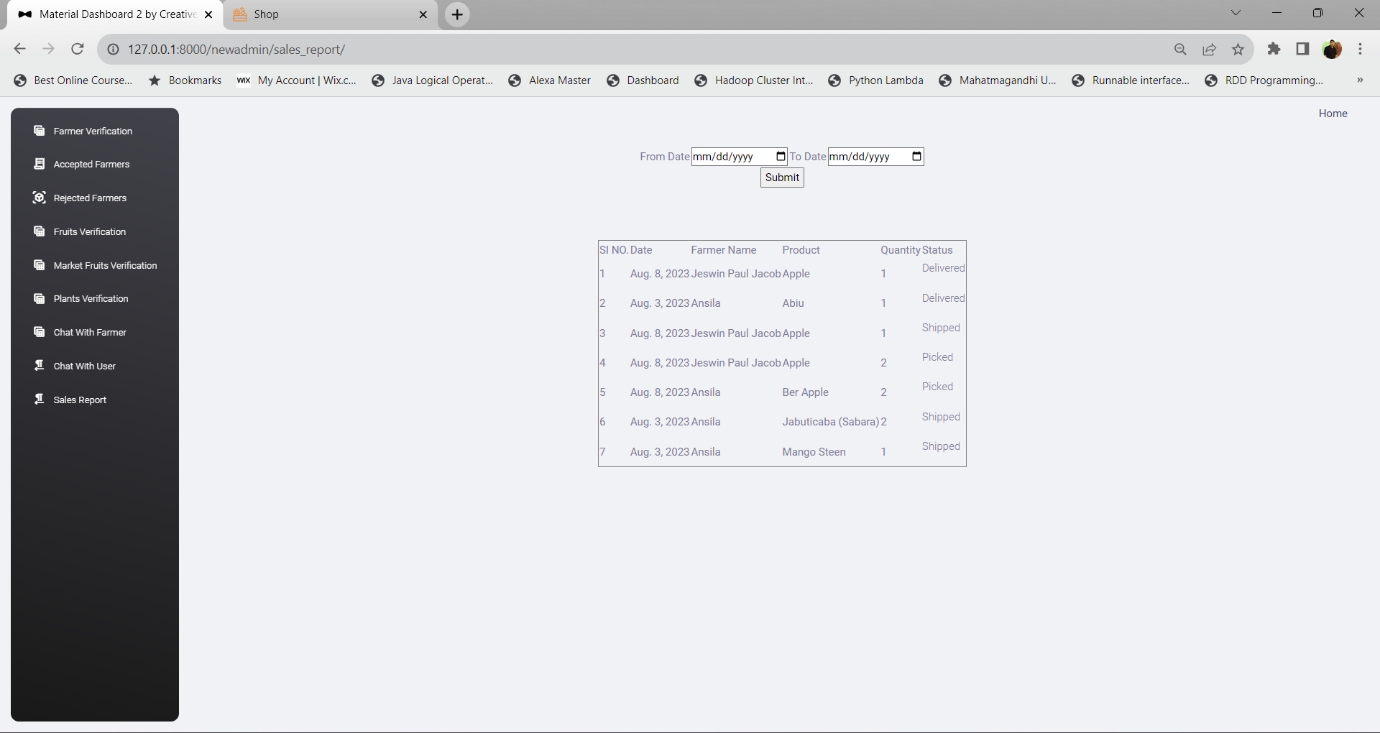
**Admin View Fruit**



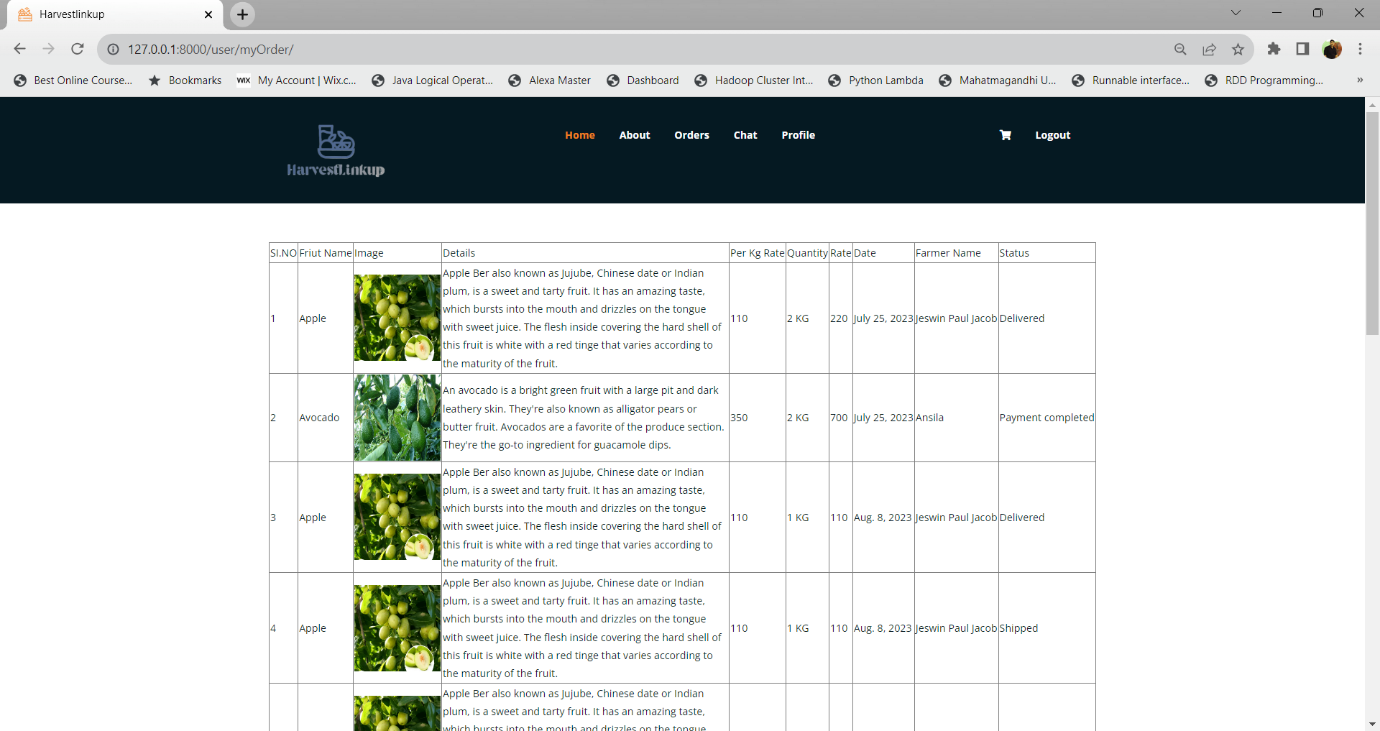
**Admin View Fruit in Market**

****

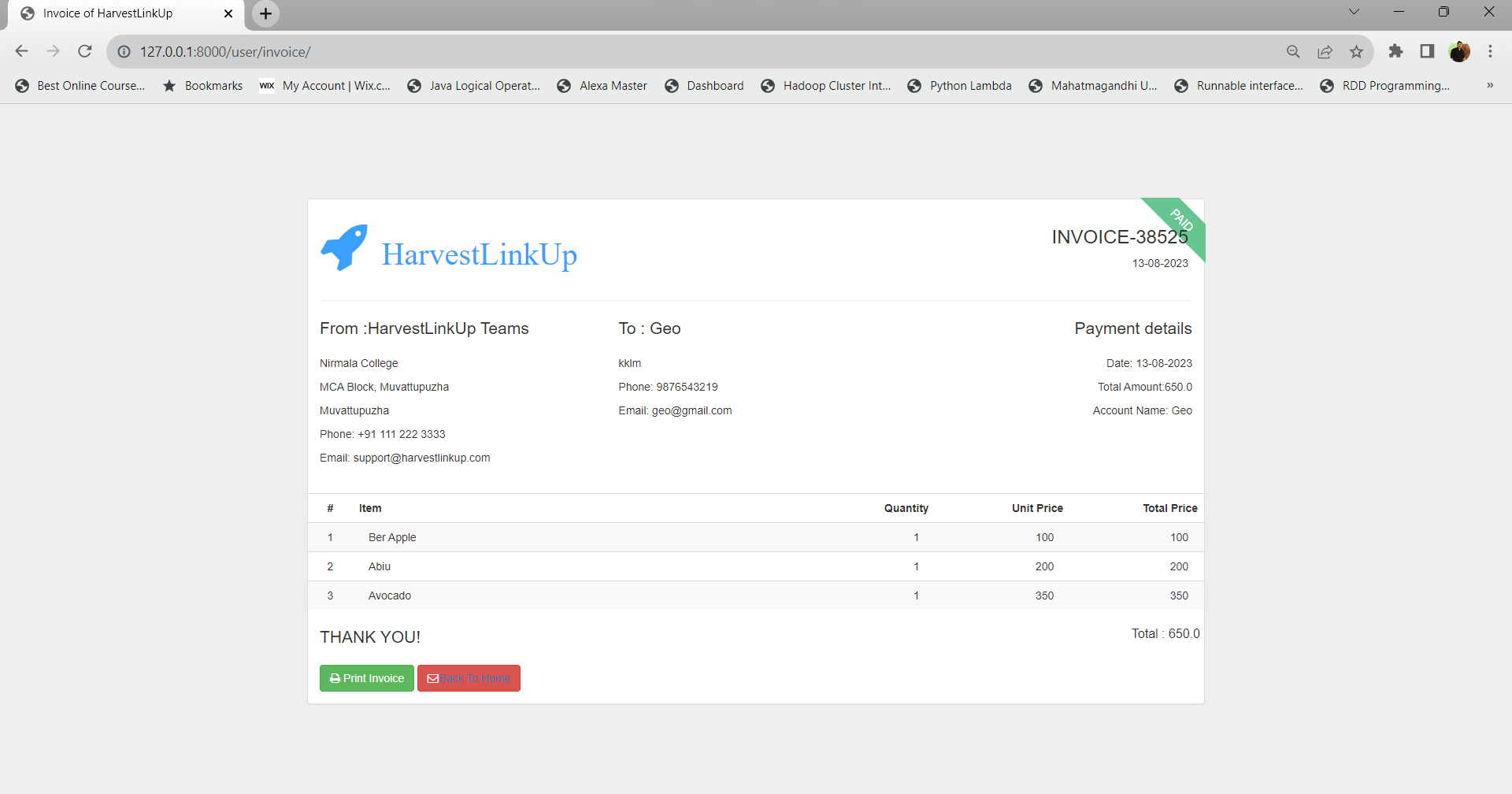
**Admin View Report**

****

**Customer My Orders**



**Invoice**

****

**4.1 INTRODUCTION**

The unique technical and operating characteristics of an IT system and its associated environment include the hardware, software, firmware, communications capability, organization and physical location. The system environment is primarily the set of variables that define or control certain aspects of process execution. From the system-management point of view, it is important to ensure the user is set up with the correct values at log in. Most of these variables are set during system initialization.

**4.2 SOFTWARE REQUIREMENT SPECIFICATION**

After the analyst has collected all the required information regarding the software to be developed and has removed all completeness, inconsistences and anomalies from the specification, he starts to systematically organize the requirements in the form of an SRS document. The software developer refer to the SRS document to make sure that they develop exactly what the customer requires. The SRS document helps the maintenance engineers to understand the functionality of the system. An SRS document should clearly specify;

• Functional requirements

• Non -functional requirements

• Goals of implementation

The software for the development of the proposed system is as follows.

**Software Interface**

Operating system : Windows

Web Browsers : Google Chrome/Firefox/Internet Explorer

Web Design : HTML, CSS, Java Script

Front End : Python 3.10 Django

Back End : SQLite

**4.3 HARDWARE REQUIREMENT SPECIFICATION**

The most common set of requirements defined by any operating system or software application is the physical computer resources also known as hardware. The hardware requirements list is often accompanied by a hardware compatibility list.

The hardware compatibility lists are tested, it may be compatible and sometimes incompatible for some hardware devices for a particular operating system or application. Selection of hardware configuration is very important task related to the software development. The processor should be powerful to handle all the operations. The hard disk should have the sufficient capacity to solve the database and the application. The minimum requirement for the implementation of the system is one machine and internet connection.

Microprocessor : AMD Ryzen 3

Processor Speed : 500MHz

Cache Memory : 512 KB

System bus : 32bits

RAM : 4 GB

Hard Disk : 1TB

Key Board : Standard keyboard

Mouse : MS Serial Mouse

**4.4 TOOLS, PLATFORMS**

**4.4.1** **FRONT END TOOL**

1. **PYTHON 3.10**

Python 3.10 is a major release of the Python programming language, offering various new features, optimizations, and improvements over its predecessor, Python 3.9. Released on October 4, 2021, Python 3.10 brought several enhancements that aimed to make the language more powerful, expressive, and developer-friendly.

Some key features of Python 3.10 include:

1. **Pattern Matching (PEP 634):** One of the most significant additions to Python 3.10 is the introduction of pattern matching with the match statement. Pattern matching allows Developers to write more concise and readable code when dealing with complex Conditional expressions.
2. **Parenthesized Context Managers (PEP 634):** This feature allows combining multiple Context managers into a single parenthesized expression, improving code readability and organization.
3. **Parenthesized Comprehensions (PEP 612):** Python 3.10 introduced support for using Parentheses in list comprehensions and generator expressions, making them more Consistent with other language constructs.
4. **Precise Types (PEP 563 and PEP 586):** Python 3.10 provided more precise types for Variables and function annotations, enabling better static analysis and improved type Checking.
5. **New Syntax Features**: Python 3.10 introduced several syntax enhancements, including Better error messages, the use of the case keyword in match statements, and improved Error handling for unpacking.
6. **Performance Improvements**: As with most Python releases, Python 3.10 included

Performance optimizations that resulted in faster execution for certain code patterns.

Python 3.10 continued to build upon the language's strengths, emphasizing readability, Simplicity, and support for various programming paradigms. Developers welcomed these new features, as they provided more powerful tools to express ideas and solve complex problems in a cleaner and more concise manner.

1. **DJANGO**

Django is a high level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid – for support.

Django was initially developed between 2003 and 2005 by a web team who were responsible for creating and maintaining newspaper websites. Django is now a thriving, collaborative open source project, with many thousands of users and contributors. While it does still have some features that reflect its origin, Django has evolved into a versatile framework that is capable of developing any type of website.

High – profile sites that use Django include: Disqus, Instagram, Knight Foundation, MacArthur Foundation, Mozilla, National Geographic, Open Knowledge Foundation, Pinterest, and Open Stack etc.

**FEATURES OF DJANGO**

* Complete
* Versatile
* Secure
* Scalable
* Maintainable
* Portable

1. **HTML**

HTML (Hypertext Mark-up Language) is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation (CSS) or functionality/behavior (JavaScript).

"Hypertext" refers to links that connect web pages to one another, either within a single website or between websites. Links are a fundamental aspect of the Web. By uploading content to the Internet and linking it to pages created by other people, you become an active participant in the World Wide Web. HTML uses "mark-up" to annotate text, images, and other content for display in a Web browser.

### **CASCADING STYLE SHEETS (CSS)**

CSS, short for Cascading Style Sheets, is a fundamental technology used in web development to control the presentation and layout of HTML documents. It provides a way to separate the content of a web page from its appearance, allowing developers to define styles once and apply them consistently to multiple elements throughout a website.

Key points about CSS:

1. Styling HTML Elements: CSS is used to define how HTML elements should be displayed on a web page. It allows developers to control aspects like font size, color, margins, padding, positioning, and more.
2. Separation of Concerns: One of the key principles of web development is the separation of concerns. CSS enables this by keeping the design and layout separate from the HTML structure. This separation makes it easier to update styles without affecting the underlying content.
3. Selectors and Declarations: CSS uses selectors to target HTML elements and declarations to specify the styles for those elements. Selectors can target elements by tag name, class, ID, or other attributes.
4. Cascading and Specificity: The term "cascading" in CSS refers to the ability of styles to flow from parent elements to their child elements. Additionally, CSS uses a specificity hierarchy to determine which styles should apply when multiple rules target the same element.
5. External Style Sheets: CSS can be written inline within HTML elements, embedded in the head section of an HTML document, or placed in external .css files. External style sheets promote code reusability and ease of maintenance across multiple pages.
6. Media Queries: CSS includes media queries that allow developers to apply different styles based on the characteristics of the user's device, such as screen size, resolution, or orientation. This enables responsive web design, optimizing the layout for different devices like desktops, tablets, and smartphones.
7. CSS Preprocessors: To enhance the capabilities of CSS, developers often use CSS preprocessors like Sass and Less. These preprocessors introduce variables, nesting, and other programming-like constructs to make CSS code more maintainable and organized.
8. Browser Compatibility: While modern web browsers generally support CSS standards, developers often need to consider cross-browser compatibility to ensure consistent rendering across different platforms.

CSS plays a crucial role in shaping the visual appearance of websites and web applications. Together with HTML and JavaScript, CSS forms the backbone of modern web development, enabling developers to create appealing, user-friendly, and responsive web experiences. By mastering CSS, web developers can design visually stunning and well-structured websites that cater to diverse user preferences and device types

**5. JAVA SCRIPT**

JavaScript is a programming language that allows scripting of events, objects, and actions to create Internet applications. It is a website development environment that will allow the creation of Interactive Web Pages. The coding techniques are capable of accepting a client's requests and processing these requests. The web site development environment should faster than existing solutions and has been successfully used in highly demanding production environments for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet. also provide the facility for 'validating' user input. With JavaScript, forms are a consideration in nearly every pages you design. Capturing user requests is traditionally done via a ‘form’. So the website need to have facilities to create forms. Text fields and text areas can dynamically change in response to user responses.

**4.4.2 BACKEND TOOL**

**1. SQLite**

SQLite is a lightweight, server less, and self-contained relational database management system that is commonly used in small to medium-scale applications. Django, on the other hand, is a high-level Python web framework that provides a convenient way to build web applications following the Model-View-Controller (MVC) architectural pattern.

In Django, SQLite is one of the built-in database backends, and it is the default database engine used when you start a new Django project. SQLite is an excellent choice for development and testing purposes, as well as for small to medium-sized applications that don't have high concurrent user loads...

**4.4.3. OPERATING SYSTEM**

**WINDOWS 10**

Windows 10 is a series of personal computer operating system produced by Microsoft as part of its Windows NT family of operating systems. It is the successor to Windows 8.1, and was released to s on July 15, 2015. Windows 10 receives new builds on an ongoing basis, which are available at no additional cost to users, in addition to additional test builds of windows 10 which are available to windows insiders. The latest stable build of Windows 10 is

version 1903(may 2019 update).Device in enterprise environments can receive these updates at a slower pace ,or use long term support milestones that only receive critical updates ,such as security patches, over their ten-year Lifespan of extended support One of windows 10’s

Most notable features is its support of universal apps, an expansion of the Metro-style apps first introduced in windows 8. Universal apps can be designed to run across multiple Microsoft product families with nearly identical code including PCs, tablets, smartphones and Mixed Reality. The windows user interface was revised to handle transitions between a mouse-oriented interface and a touchscreen optimized interface based on available input devices-particularly on 2-in-1 PCs, both interfaces include an updated start menu which incorporates elements of windows 7’s traditional start menu with tiles of windows 8.

**5.1 INTRODUCTION**

Systems implementation is the process of defining how the information system should be built (i.e., physical system design), ensuring that the information system is operational and ensuring that it meets the quality standards. Implementation is the process of converting a new or a revised system design into an operational one. Conversion means changing from one system to another. The objective is to put the system into operation while holding costs, risks and personnel irritation to minimum. It involves:

• Creating computer compatible files.

• Training the operating staff.

• Installing terminals and hardware.

Implementation includes all those activities that take place to convert from the old system to new system. The new system is totally new which replace the existing manual system with a major modification. Proper implementation is essential to provide a reliable system to meet the organizations requirements. The method of implementation and the time scale to be adopted is found initially. Then the system is tested properly and the users are trained with the new procedures.

**5.2 CODING**

Coding is done with the help of Integrated Development Environment (IDE) tools. An integrated development environment (IDE) is a software application that helps programmers develop software code efficiently. It increases developer productivity by combining capabilities such as software editing, building, testing and packaging in an easy-to-use application. For this project the IDEs used are Visual Studio IDE which is free, open source and has a worldwide community of users and developers

**5.2.1 CODING STANDARDS**

Different modules specified in the design document are coded in the coding phase according to the module specifications. The main goal of the coding phase is to code from the design document prepared after the design phase through a high-level language and then to unit test the code.

Good software development organizations want their programmers to maintain a well-defined and standard style of coding called coding standards. They usually make their own coding standards and guidelines depending on what suits their organization the best and based on the types of software they develop. It is very important for the programmers to maintain coding standards, otherwise the code will be rejected during code review.

**Purpose of Having Coding Standards:**

* Coding standards give a uniform appearance to the codes written by different engineers.
* It improves readability and maintainability of the code and it reduces complexity also.
* It helps in code reuse and helps to detect error easily.
* It promotes sound programming practices and increases efficiency of the programmers.

**Some of the coding standards are given below:**

1. Limited use of Globals.
2. Standard headers for different modules.
3. Naming conventions for local variables, global variables, constants and functions. 4. Indentation.
4. Error return values and exception handling conventions.
5. Avoid using a coding style that is too difficult to understand.
6. Avoid using an identifier for multiple purposes.
7. Code should be well documented.
8. Length of functions should not be very large.
9. Try not to use GOTO statement.

### **5.2.2. SAMPLE CODES**

**Login.html**

<!DOCTYPE html>

<html lang="en">

<head>

<style>

.navbar

{

overflow: hidden;

background-color: #333;

}

/\* Style the navigation bar links \*/

.navbar a {

float: left;

display: block;

color: white;

text-align: center;

padding: 14px 20px;

text-decoration: none;

}

/\* Right-aligned link \*/

.navbar a.right {

float: right;

}

/\* Change color on hover \*/

.navbar a:hover {

background-color: #ddd;

color: black;

}

body {

background-image: linear-gradient(135deg, #FAB2FF 10%, #1904E5 100%);

background-size: cover;

background-repeat: no-repeat;

background-attachment: fixed;

font-family: "Open Sans", sans-serif;

color: #333333;

}

.box-form {

margin: 0 auto;

width: 80%;

background: #FFFFFF;

border-radius: 10px;

overflow: hidden;

display: flex;

flex: 1 1 100%;

align-items: stretch;

justify-content: space-between;

box-shadow: 0 0 20px 6px #090b6f85;

}

@media (max-width: 980px) {

.box-form {

flex-flow: wrap;

text-align: center;

align-content: center;

align-items: center;

}

}

.box-form div {

height: auto;

}

.box-form .left {

color: #FFFFFF;

background-size: cover;

background-repeat: no-repeat;

background-image: url("https://i.pinimg.com/736x/5d/73/ea/5d73eaabb25e3805de1f8cdea7df4a42--tumblr-backgrounds-iphone-phone-wallpapers-iphone-wallaper-tumblr.jpg");

overflow: hidden;

}

.box-form .left .overlay {

padding: 30px;

width: 100%;

height: 100%;

background: #5961f9ad;

overflow: hidden;

box-sizing: border-box;

}

.box-form .left .overlay h1 {

font-size: 10vmax;

line-height: 1;

font-weight: 900;

margin-top: 40px;

margin-bottom: 20px;

}

.box-form .left .overlay span p {

margin-top: 30px;

font-weight: 900;

}

.box-form .left .overlay span a {

background: #3b5998;

color: #FFFFFF;

margin-top: 10px;

padding: 14px 50px;

border-radius: 100px;

display: inline-block;

box-shadow: 0 3px 6px 1px #042d4657;

}

.box-form .left .overlay span a:last-child {

background: #1dcaff;

margin-left: 30px;

}

.box-form .right {

padding: 40px;

overflow: hidden;

}

@media (max-width: 980px) {

.box-form .right {

width: 100%;

}

}

.box-form .right h5 {

font-size: 6vmax;

line-height: 0;

}

.box-form .right p {

font-size: 14px;

color: #B0B3B9;

}

.box-form .right .inputs {

overflow: hidden;

}

.box-form .right input {

width: 100%;

padding: 10px;

margin-top: 25px;

font-size: 16px;

border: none;

outline: none;

border-bottom: 2px solid #B0B3B9;

}

.box-form .right .remember-me--forget-password {

display: flex;

justify-content: space-between;

align-items: center;

}

.box-form .right .remember-me--forget-password input {

margin: 0;

margin-right: 7px;

width: auto;

}

.box-form .right button {

float: right;

color: #fff;

font-size: 16px;

padding: 12px 35px;

border-radius: 50px;

display: inline-block;

border: 0;

outline: 0;

box-shadow: 0px 4px 20px 0px #49c628a6;

background-image: linear-gradient(135deg, #70F570 10%, #49C628 100%);

}

label {

display: block;

position: relative;

margin-left: 30px;

}

label::before {

content: ' \f00c';

position: absolute;

font-family: FontAwesome;

background: transparent;

border: 3px solid #70F570;

border-radius: 4px;

color: transparent;

left: -30px;

transition: all 0.2s linear;

}

label:hover::before {

font-family: FontAwesome;

content: ' \f00c';

color: #fff;

cursor: pointer;

background: #70F570;

}

label:hover::before .text-checkbox {

background: #70F570;

}

label span.text-checkbox {

display: inline-block;

height: auto;

position: relative;

cursor: pointer;

transition: all 0.2s linear;

}

label input[type="checkbox"] {

display: none;

}

</style>

<meta charset="UTF-8">

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

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login</title>

</head>

<body>

<nav class="navbar">

<a href="{% url 'newguest:userreg' %}" class="w3-button w3-bar-item">User Registration</a>

<a href="{% url 'newguest:farmerreg' %}" class="w3-button w3-bar-item">Farmer Registration</a>

</nav>

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

{% csrf\_token %}

<!-- <table align="center" border="1">

<br><br><br>

<tr>

<td>Email</td>

<td><input type="email" name="txt\_email"></td>

</tr>

<tr>

<td>Password</td>

<td><input type="password" name="txt\_password"></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="submit" value="Login" id=""></td>

</tr>

</table> -->

<div class="box-form">

<div class="left">

<div class="overlay">

<h2>Harvestlinkup</h2>

<p>we are passionate about bringing you the freshest and most delectable fruits from around the Kerala.</p>

</div>

</div>

<div class="right">

<h5>Login</h5>

<p>Don't have an account? <br><a href="{% url 'newguest:userreg' %}">Creat User Account</a> &

<a href="{% url 'newguest:farmerreg' %}">Creat Farmer Account,</a> it takes less than a minute</p>

<div class="inputs">

<input type="email" name="txt\_email" placeholder="Email">

<br>

<input type="password" name="txt\_password" placeholder="password" >

</div>

<br><br>

<div class="remember-me--forget-password">

<!-- Angular -->

<!-- <label>

<input type="checkbox" name="item" checked/>

<span class="text-checkbox">Remember me</span>

</label> -->

<p><a href="{% url 'newguest:forgotpass' %}">forget password?</a></p>

</div>

<br>

<button>Login</button>

</div>

</div>

</form>

</body>

</html>

<script>

{% if msg %}

alert("{{msg}}");

{% endif %}

</script>

**customerregistration.html**

<!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=1.0">

<title>User Registration</title>

<style>

input[type=text], select {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

display: inline-block;

border: 1px solid #ccc;

border-radius: 4px;

box-sizing: border-box;

}

input[type=email], select {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

display: inline-block;

border: 1px solid #ccc;

border-radius: 4px;

box-sizing: border-box;

}

input[type=password], select {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

display: inline-block;

border: 1px solid #ccc;

border-radius: 4px;

box-sizing: border-box;

}

input[type=submit] {

width: 100%;

background-color: #4CAF50;

color: white;

padding: 14px 20px;

margin: 8px 0;

border: none;

border-radius: 4px;

cursor: pointer;

}

input[type=submit]:hover {

background-color: #45a049;

}

input[type=reset] {

width: 100%;

background-color: #4CAF50;

color: white;

padding: 14px 20px;

margin: 8px 0;

border: none;

border-radius: 4px;

cursor: pointer;

}

input[type=reset]:hover {

background-color: #45a049;

}

div {

border-radius: 5px;

background-color: #f2f2f2;

padding: 20px;

}

</style>

</head>

<body>

<div>

<form action="" method="POST" enctype="multipart/form-data" id="UserForm" data-places-url="{% url 'newguest:Ajax\_place' %}" >

<h2 ><u>User Registration</u></h2>

{% csrf\_token %}

<table border="2">

<tr>

<td>Name</td>

<td><input required type="text" name="txt\_name" title="Name Allows Only Alphabets,Spaces and First Letter Must Be Capital Letter" placeholder="Enter Name" pattern="^[A-Z]+[a-zA-Z ]\*$"/></td>

</tr>

<tr>

<td>Contact</td>

<td><input type="text" name="txt\_contact" required="" autocomplete="off" placeholder="0123456789" pattern="[0-9]{10}" required="" maxlength="10"></td>

</tr>

<tr>

<td>Email</td>

<td><input type="email" name="txt\_email" autocomplete="off" pattern="[^ @]\*@[^ @]\*" placeholder="Enter Email" id="email" required=""></td>

</tr>

<tr>

<td>Gender</td>

<td><input type="radio" name="gender" value="M">Male

<input type="radio" name="gender" value="F">female

<input type="radio" name="gender" value="O">Others

</td>

</tr>

<tr>

<td>Address</td>

<td><textarea name="txt\_address" id="" cols="30" rows="10" required=""></textarea></td>

</tr>

<tr>

<td>District</td>

<td>

<Select name="sel\_district" id="sel\_dist" required="">

<option value="">----select----</option>

{% for i in DIS%}

<option value="{{i.id}}">{{i.district\_name}}</option>

{% endfor %}

</Select>

</td>

</tr>

<tr>

<td>Place</td>

<td><Select name="sel\_place" id="sel\_plc" required="">

<option value="">----select----</option>

</Select></td>

</tr>

<tr>

<td>Photo</td>

<td><input type="file" name="txt\_photo" required=""></td>

</tr>

<tr>

<td>Proof</td>

<td><input type="file" name="txt\_proof" required="">

</td>

</tr>

<tr>

<td>Password</td>

<td><input type="password" name="txt\_password" placeholder="Enter Password" pattern="(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{8,}" title="Must contain at least one number and one uppercase and lowercase letter, and at least 8 or more characters" required></td>

</tr>

<tr>

<td>Confirm Password</td>

<td><input type="password" name="txt\_password1" placeholder="Re-enter Password" pattern="(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{8,}" title="Must contain at least one number and one uppercase and lowercase letter, and at least 8 or more characters" required></td>

</tr>

<tr>

<td colspan="2" align="center"><input id="checkbox" type="checkbox" required="" />

<label for="checkbox"> I agree to these <a href="{% url 'newguest:termsuser' %}">Terms and Conditions</a>.</label></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="submit" value="Submit"><input type="reset" value="Cancel"></td>

</tr>

</table>

</form>

</div>

</body>

</html>

<script>

{% if msg %}

alert("{{msg}}");

{% endif %}

</script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$("#sel\_dist").change(function () {

var did = $(this).val();

var ur = $("#UserForm").attr("data-places-url");

$.ajax({

url: ur,

data: { DIST: did, },

success: function (data) {

$("#sel\_plc").html(data);

},

});

   });

</script>

**farmerregistration.html**

<!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=1.0">

<title>Farmer Registration</title>

</head>

<style>

input[type=text], select {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

display: inline-block;

border: 1px solid #ccc;

border-radius: 4px;

box-sizing: border-box;

}

input[type=email], select {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

display: inline-block;

border: 1px solid #ccc;

border-radius: 4px;

box-sizing: border-box;

}

input[type=password], select {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

display: inline-block;

border: 1px solid #ccc;

border-radius: 4px;

box-sizing: border-box;

}

input[type=submit] {

width: 100%;

background-color: #4CAF50;

color: white;

padding: 14px 20px;

margin: 8px 0;

border: none;

border-radius: 4px;

cursor: pointer;

}

input[type=submit]:hover {

background-color: #45a049;

}

input[type=reset] {

width: 100%;

background-color: #4CAF50;

color: white;

padding: 14px 20px;

margin: 8px 0;

border: none;

border-radius: 4px;

cursor: pointer;

}

input[type=reset]:hover {

background-color: #45a049;

}

div {

border-radius: 5px;

background-color: #f2f2f2;

padding: 20px;

}

</style>

<body>

<form action="" method="POST" enctype="multipart/form-data" id="FarmerForm" data-places-url="{% url 'newguest:Ajax\_place' %}" >

{% csrf\_token %}

<h2 ><u>Farmer Registration</u></h2>

<table border="1">

<tr>

<td>Name</td>

<td><input type="text" name="txt\_name" title="Name Allows Only Alphabets,Spaces and First Letter Must Be Capital Letter" placeholder="Enter Name" pattern="^[A-Z]+[a-zA-Z ]\*$"/></td>

</tr>

<tr>

<td>Contact</td>

<td><input type="text" name="txt\_contact" required="" autocomplete="off" placeholder="0123456789" pattern="[0-9]{10}" required="" maxlength="10"></td>

</tr>

<tr>

<td>Email</td>

<td><input type="email" name="txt\_email" autocomplete="off" pattern="[^ @]\*@[^ @]\*" placeholder="Enter Email" id="email" required=""></td>

</tr>

<tr>

<td>Gender</td>

<td><input type="radio" name="gender" value="M">Male

<input type="radio" name="gender" value="F">female

<input type="radio" name="gender" value="O">Others

</td>

</tr>

<tr>

<td>Address</td>

<td><textarea name="txt\_address" id="" cols="30" rows="10" required=""></textarea></td>

</tr>

<tr>

<td>District</td>

<td>

<Select name="sel\_district" id="sel\_dist" required="">

<option value="">----select----</option>

{% for i in DIS%}

<option value="{{i.id}}">{{i.district\_name}}</option>

{% endfor %}

</Select>

</td>

</tr>

<tr>

<td>Place</td>

<td><Select name="sel\_place" id="sel\_plc" required="">

<option value="">----select----</option>

</Select></td>

</tr>

<tr>

<td>Proof</td>

<td><input type="file" name="txt\_proof" required="">

</tr>

<tr>

<td>Photo</td>

<td><input type="file" name="txt\_photo" required=""></td>

</tr>

<tr>

<td>Farm Photo</td>

<td><input type="file" name="txt\_farmphoto" required=""></td>

</tr>

<!-- <tr>

<td>License Number</td>

<td><input type="text" name="txt\_licenseno" ></td>

</tr>

<tr>

<td>License Proof</td>

<td><input type="file" name="txt\_licensephoto"></td>

</tr> -->

<tr>

<td>Password</td>

<td><input type="password" name="txt\_password" placeholder="Enter Password" pattern="(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{8,}" title="Must contain at least one number and one uppercase and lowercase letter, and at least 8 or more characters" required></td>

</tr>

<tr>

<td>Confirm Password</td>

<td><input type="password" name="txt\_password1" placeholder="Re-enter Password" pattern="(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{8,}" title="Must contain at least one number and one uppercase and lowercase letter, and at least 8 or more characters" required></td>

</tr>

<tr>

<td colspan="2" align="center"><input id="checkbox" type="checkbox" required="" />

<label for="checkbox"> I agree to these <a href="{% url 'newguest:termsfarmer' %}">Terms and Conditions</a>.</label></td>

</tr>

<tr>

<td colspan="2" align="center"><input type="submit" value="Submit"><input type="reset" value="Cancel"></td>

</tr>

</table>

</form>

</body>

</html>

<script>

{% if msg %}

alert("{{msg}}");

{% endif %}

</script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<script>

$("#sel\_dist").change(function () {

var did = $(this).val();

var ur = $("#FarmerForm").attr("data-places-url");

$.ajax({

url: ur,

data: { DIST: did, },

success: function (data) {

$("#sel\_plc").html(data);

},

});

   });

</script>

**Customer functions**

from django.shortcuts import render,redirect

from Guest.models import \*

from User.models import \*

from Farmer.models import \*

from django.db.models import Q

from django.core.paginator import Paginator

import random

from datetime import datetime

from django.conf import settings

from django.core.mail import send\_mail

from django.contrib import messages

# Create your views here.

def loginpage(request):

userreg=tbl\_user.objects.get(id=request.session['uid'])

return render(request,"User/Loginpage.html",{'USER':userreg})

def logout(request):

del request.session['uid']

return redirect('newguest:login')

def myprofile(request):

userreg=tbl\_user.objects.get(id=request.session['uid'])

return render(request,"User/Myprofile.html",{'USER':userreg})

def edit(request):

edit=tbl\_user.objects.get(id=request.session['uid'])

if request.method=="POST" and request.FILES :

edit.user\_photo=request.FILES.get('img\_file')

edit.user\_name=request.POST.get('txt\_name')

edit.user\_contact=request.POST.get('txt\_contact')

edit.user\_email=request.POST.get('txt\_email')

edit.user\_address=request.POST.get('txt\_address')

edit.save()

return redirect('user:myprofile')

elif request.method=="POST":

edit.user\_name=request.POST.get('txt\_name')

edit.user\_contact=request.POST.get('txt\_contact')

edit.user\_email=request.POST.get('txt\_email')

edit.user\_address=request.POST.get('txt\_address')

edit.save()

return redirect('user:myprofile')

else:

return render(request,"USER/Editprofile.html",{'USER':edit})

def changepassword(request):

editpass=tbl\_user.objects.get(id=request.session['uid'])

if request.method=="POST":

psw=editpass.user\_password

curpass=request.POST.get('txt\_password')

if psw==curpass:

new=request.POST.get('txt\_new\_password')

cnew=request.POST.get('txt\_new\_password1')

if new==cnew:

editpass.user\_password=new

editpass.save()

return redirect('user:myprofile')

else:

return render(request,"User/Changepassword.html",{'msg':"Password Mismatch!"})

else:

return render(request,"User/Changepassword.html",{'msg':"Password Incorrect!"})

else:

return render(request,"User/Changepassword.html")

def productview(request,pid):

product=tbl\_product.objects.get(id=pid)

request.session["pdid"]=pid

usrid=tbl\_user.objects.get(id=request.session["uid"])

comments=tbl\_comments.objects.filter(product\_id=product)

book=tbl\_booking.objects.filter(user\_id=usrid)

for i in book:

bkid=i.id

crt=tbl\_cart.objects.filter(booking\_id\_\_id=bkid,product\_id=pid,booking\_id\_\_booking\_status\_\_gte=1)

if request.method=="POST":

tbl\_comments.objects.create(comment\_content=request.POST.get('comment'),user\_id=usrid,product\_id=product)

return render(request,"User/Productviewmore.html",{"PRO":product,'CMNT':comments,'user':usrid,'BOOK':crt})

else:

return render(request,"User/Productviewmore.html",{"PRO":product,'CMNT':comments,'user':usrid})

def comment\_delete(request,cmid):

product=tbl\_product.objects.get(id=request.session["pdid"])

comments=tbl\_comments.objects.filter(product\_id=product)

comments.delete()

return redirect("user:searchproduct")

def Addtocart(request,pid):

product=tbl\_product.objects.all()

if 'uid' in request.session:

message=""

prod=tbl\_product.objects.get(id=pid)

userid=tbl\_user.objects.get(id=request.session["uid"])

bcount=tbl\_booking.objects.filter(user\_id=userid,booking\_status=0).count()

if bcount>0:

book=tbl\_booking.objects.get(user\_id=userid,booking\_status=0)

bpcount=tbl\_cart.objects.filter(booking\_id=book,product\_id=prod).count()

if bpcount>0:

message="AlreadyAddedtoCart"

return render(request,"User/Searchproduct.html",{"mess":message,'page\_obj':product})

else:

tbl\_cart.objects.create(product\_id=prod,booking\_id=book)

message="AddedtoCart"

return render(request,"User/Searchproduct.html",{"mess":message,'page\_obj':product})

else:

tbl\_booking.objects.create(user\_id=userid)

bid=tbl\_booking.objects.filter(user\_id=userid,booking\_status=0).count()

if bid>0:

b=tbl\_booking.objects.get(user\_id=userid,booking\_status=0)

ids=b.id

bc=tbl\_booking.objects.get(id=ids)

tbl\_cart.objects.create(booking\_id=bc,product\_id=prod)

message="AddedtoCart"

return render(request,"User/Searchproduct.html",{"mess":message,'page\_obj':product})

else:

return render(request,"User/Searchproduct.html",{"mess":message,'page\_obj':product})

else:

return redirect("newguest:login")

def mycart(request):

if 'uid' in request.session:

userid=tbl\_user.objects.get(id=request.session["uid"])

if request.method=="POST":

return redirect("user:payment")

else:

bcount=tbl\_booking.objects.filter(user\_id=userid,booking\_status=0).count()

if bcount>0:

bid=tbl\_booking.objects.get(user\_id=userid,booking\_status=0)

ids=bid.id

request.session["bookingsid"]=ids

bc=tbl\_booking.objects.get(id=ids)

cartob=tbl\_cart.objects.filter(booking\_id=bc)

return render(request,"User/Mycart.html",{'data':cartob})

else:

return render(request,"User/Mycart.html")

else:

return redirect("newguest:login")

def MyOrder(request):

if 'uid' in request.session:

usrid=tbl\_user.objects.get(id=request.session["uid"])

prdt=tbl\_cart.objects.filter(booking\_id\_\_user\_id=usrid,booking\_id\_\_booking\_status\_\_gte=1)

return render(request,"User/Myorders.html",{'PRO':prdt})

else:

return redirect("Guest:login")

def get\_qnty(request):

if 'uid' in request.session:

qty=request.GET.get('QTY')

alt=request.GET.get('ALT')

cart=tbl\_cart.objects.get(id=alt)

cart.booking\_quantity=qty

cart.save()

return redirect('user:mycart')

else:

return redirect("newguest:login")

def Cancelorder(request,boid):

if 'uid' in request.session:

bid=tbl\_booking.objects.get(id=boid)

bid.booking\_status=4

bid.save()

return redirect("user:Myorder")

else:

return redirect("newguest:login")

def payment(request):

if 'uid' in request.session:

usrid=tbl\_user.objects.get(id=request.session["uid"])

if request.method=="POST":

ids=tbl\_booking.objects.get(id=request.session["bookingsid"])

ccount=tbl\_cart.objects.filter(booking\_id=ids).count()

if ccount>0:

cdata=tbl\_cart.objects.filter(booking\_id=ids)

for i in cdata:

pdata=tbl\_product.objects.get(id=i.product\_id.id)

fid=pdata.farmer

price1=float(pdata.product\_rate)

fullrate=float(price1\*i.booking\_quantity)

fprofit=float(fullrate\*0.8)

aprofit=float(fullrate\*0.18)

sprofit=float(fullrate\*0.02)

ids.booking\_status=1

ids.save()

tbl\_payment.objects.create(payment\_amount=fullrate,booking\_id=ids,farmer\_profit=fprofit,

admin\_profit=aprofit,supplier\_profit=sprofit,user\_id=usrid,farmer\_id=fid)

stock=float(pdata.stock)

bal=stock-float(i.booking\_quantity)

pdata.stock=bal

name=pdata.farmer.farmer\_name

email1=pdata.farmer.farmer\_email

qty=i.booking\_quantity

pname=pdata.product\_name

send\_mail(

'Respected sir/madam '+name, #subject

"\rYour Order of " + str(qty) + " for " + pname + " is Successfully Confirmed By HarvestLinkUp and Proceed To welcome you to HarvestLinkUp.your Username is " + email1 + ".\n This is from HarvestLinkUp team thank you for signing up to our service. \r\n If you have any questions and if there is anything we can improve please reply to this email. \r\n we were always happy to help!. \r\n \r\n Team HarvestLinkUp.\n Thank you.",#body

settings.EMAIL\_HOST\_USER,

[email1],

)

pdata.save()

return redirect("user:processingpayment")

else:

return render(request,"User/Payment.html",{'Usr':usrid})

else:

return redirect("newguest:login")

def processingpayment(request):

if 'uid' in request.session:

return render(request,"User/runpayment.html")

else:

return redirect("newguest:login")

def paysucess(request):

if 'uid' in request.session:

return render(request,"User/paysucessful.html")

else:

return redirect("newguest:login")

def Invoice(request):

if 'uid' in request.session:

Total="0.0"

billno=random.randint(10000,99999)

today = datetime.now()

today=today.strftime("%d-%m-%Y")

userid=tbl\_user.objects.get(id=request.session["uid"])

# uobj=tbl\_cart.objects.filter(booking\_\_user=userid).last()

bkid=tbl\_booking.objects.get(id=request.session["bookingsid"])

# bcount=tbl\_booking.objects.filter(user=userid,booking\_status=1).count()

# if bcount>0:

# bid=tbl\_booking.objects.get(user=userid,booking\_status=1)

# ids=bid.id

# request.session["bookingsid"]=ids

# bc=tbl\_booking.objects.get(id=ids)

uobj=tbl\_cart.objects.filter(booking\_id=bkid)

for i in uobj:

bqty=float(i.booking\_quantity)

prate=float(i.product\_id.product\_rate)

total=float(bqty\*prate)

Total = float(Total) + total

return render(request,"User/Invoice.html",{'billno':billno,'today':today,'userdata':userid,'data':uobj,'T':Total})

else:

return redirect("newguest:login")

def adminchat(request):

admin=tbl\_admin.objects.all()

return render(request,"User/AdminChat.html",{'ADM':admin})

#--------Chat--------#

def chatuser(request,cid):

chatobj = tbl\_admin.objects.get(id=cid)

if request.method == "POST":

cied = request.POST.get("cid")

# print(cied)

ciedobj = tbl\_admin.objects.get(id=cied)

sobj = tbl\_user.objects.get(id=request.session["uid"])

content = request.POST.get("msg")

# print(cied)

# print(content)

Chat.objects.create(

from\_user=sobj, to\_admin=ciedobj, content=content, from\_admin=None, to\_user=None, from\_farmer=None, to\_farmer=None)

return render(request, 'User/Chat.html', {"chatobj": chatobj})

else:

return render(request, 'User/Chat.html', {"chatobj": chatobj})

def loadchatuser(request):

cid = request.GET.get("cid")

request.session["cid"] = cid

+ cid1 = request.session["cid"]

# print(cid1)

# print(cid)

shopobj = tbl\_admin.objects.get(id=cid)

# print(userobj)

sid = request.session["uid"]

# print(sid)

suserobj = tbl\_user.objects.get(id=request.session["uid"])

# chatobj1 = Chat.objects.filter(Q(to\_user=suserobj) | Q(

# from\_user=suserobj), Q(to\_shop=shopobj) | Q(from\_shop=shopobj))

# print(chatobj1) # send message

# print(chatobj2) # recived msg

chatobj = Chat.objects.raw(

"select \* from User\_chat c inner join Guest\_tbl\_user u on (u.id=c.from\_user\_id) or (u.id=c.to\_user\_id) WHERE c.from\_admin\_id=%s or c.to\_admin\_id=%s order by c.date", params=[(cid1), (cid1)])

print(chatobj.query)

return render(request, 'User/Load.html', {"obj": chatobj, "sid": sid, "shop": shopobj, "userobj": suserobj})

def cremove(request,cid):

farme=tbl\_cart.objects.get(id=cid)

farme.delete()

return redirect('user:mycart')

def command(request):

usrid=tbl\_user.objects.get(id=request.session["uid"])

command=tbl\_comments.get(id=request.session['uid'])

if request.method=="POST":

command.comment\_content=request.POST.get('comment')

command.save()

return redirect('user:productview')

else:

return redirect('user:productview')

def about(request):

return render(request,"User/about.html")

# def searchproduct(request):

# product=tbl\_product.objects.filter(product\_status=1)

# page = request.GET.get('page', 1)

# paginator = Paginator(product, 6)

# page\_obj = paginator.get\_page(page)

# if request.method=="POST":

# text=request.POST.get('txt\_search')

# page\_obj=tbl\_product.objects.filter(Q(product\_name\_\_icontains=text) | Q(farmer\_\_farmer\_name\_\_icontains=text),product\_status=1)

# return render(request,"User/Searchproduct.html",{'page\_obj':page\_obj})

# else:

# return render(request,"User/Searchproduct.html",{'page\_obj':page\_obj})

def searchproduct(request):

user=tbl\_user.objects.get(id=request.session['uid'])

product=tbl\_product.objects.filter(product\_status=1)

page\_ob=tbl\_productmarket.objects.filter(product\_status=1)

page = request.GET.get('page', 1)

paginator = Paginator(product, 9)

page\_obj = paginator.get\_page(page)

if request.method=="POST":

text=request.POST.get('txt\_search')

page\_obj=tbl\_product.objects.filter(Q(product\_name\_\_icontains=text) | Q(farmer\_\_farmer\_name\_\_icontains=text),product\_status=1)

page\_ob=tbl\_productmarket.objects.filter(Q(product\_name\_\_icontains=text) | Q(farmer\_\_farmer\_name\_\_icontains=text),product\_status=1)

return render(request,"User/Searchproduct.html",{'page\_obj':page\_obj,'page\_ob':page\_ob})

else:

return render(request,"User/Searchproduct.html",{'page\_obj':page\_obj,'page\_ob':page\_ob})

def ajax\_Fruits(request):

if request.GET.get("mid")=="0":

fdata=tbl\_product.objects.filter(product\_status=1)

return render(request,"User/AjaxFruits.html",{'page\_obj':fdata})

elif request.GET.get("mid")=="1":

fdata=tbl\_productmarket.objects.filter(product\_status=1)

return render(request,"User/AjaxFruits.html",{'page\_ob':fdata})

def marketproducts(request,pid):

if 'uid' in request.session:

product=tbl\_productmarket.objects.get(id=pid)

usrid=tbl\_user.objects.get(id=request.session["uid"])

# comments=tbl\_comments.objects.filter(product\_id=product)

if request.method=="POST":

tbl\_comments.objects.create(comment\_content=request.POST.get('comment'),user\_id=usrid,product\_id=product)

return render(request,"User/MarketPdtView.html",{"PRO":product,'user':usrid})

else:

return render(request,"User/MarketPdtView.html",{"PRO":product,'user':usrid})

else:

return redirect('newguest:login')

def mboooking(request,pid):

pdata=tbl\_productmarket.objects.get(id=pid)

request.session['pdid']=pdata.id

usid=tbl\_user.objects.get(id=request.session["uid"])

if request.method=="POST":

qty=float(request.POST.get('quantity'))

tbl\_marketbooking.objects.create(product=pdata,booking\_quantity=qty,user=usid)

mdata=tbl\_marketbooking.objects.latest('id')

request.session['mtid']=mdata.id

return redirect('user:mpayment')

else:

return render(request,"User/MarketBooking.html",{'PDT':pdata})

def mpayment(request):

if 'uid' in request.session:

usid=tbl\_user.objects.get(id=request.session["uid"])

pdata=tbl\_productmarket.objects.get(id=request.session['pdid'])

mdata=tbl\_marketbooking.objects.get(id=request.session['mtid'])

mid=mdata.id

if request.method=="POST":

fid=pdata.farmer

rate=float(pdata.product\_rate)

fprofit=float(rate\*0.8)

aprofit=float(rate\*0.18)

sprofit=float(rate\*0.02)

tbl\_payment.objects.create(payment\_amount=rate,farmer\_profit=fprofit,admin\_profit=aprofit,

supplier\_profit=sprofit,user\_id=usid,farmer\_id=fid,market\_booking=mdata)

stock=float(pdata.quantity)

bal=stock-float(mdata.booking\_quantity)

name=pdata.farmer.farmer\_name

email1=pdata.farmer.farmer\_email

qty=mdata.booking\_quantity

pname=pdata.product\_name

send\_mail(

'Respected sir/madam '+name, #subject

"\rYour Order of " + str(qty) + " for " + pname + " is Successfully Confirmed By HarvestLinkUp and Proceed To welcome you to HarvestLinkUp.your Username is " + email1 + ".\n This is from HarvestLinkUp team thank you for signing up to our service. \r\n If you have any questions and if there is anything we can improve please reply to this email. \r\n we were always happy to help!. \r\n \r\n Team HarvestLinkUp.\n Thank you.",#body

settings.EMAIL\_HOST\_USER,

[email1],

)

pdata.quantity=bal

pdata.save()

return redirect("user:mprocessingpayment")

else:

return render(request,"User/MarketPayment.html",{'USER':usid,'PD':pdata})

else:

return redirect("newguest:login")

def mprocessingpayment(request):

if 'uid' in request.session:

return render(request,"User/Mrunpayment.html")

else:

return redirect("newguest:login")

def mpaysucess(request):

if 'uid' in request.session:

return render(request,"User/Mpaysucessful.html")

else:

return redirect("newguest:login")

def MInvoice(request):

if 'uid' in request.session:

billno=random.randint(10000,99999)

today = datetime.now()

today=today.strftime("%d-%m-%Y")

usid=tbl\_user.objects.get(id=request.session["uid"])

uobj=tbl\_marketbooking.objects.filter(user=usid).last()

return render(request,"User/MarketInvoice.html",{'billno':billno,'today':today,'userdata':usid,'data':uobj})

else:

return redirect("newguest:login")

def market\_bookings(request):

if 'uid' in request.session:

usid=tbl\_user.objects.get(id=request.session["uid"])

prdt=tbl\_marketbooking.objects.filter(user=usid)

return render(request,"User/MarketBookings.html",{'PRO':prdt})

else:

return redirect("Guest:login")

## 5.3 UNIT TESTING

Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance. A unit is a single testable part of a software system and tested during the development phase of the application software. The purpose of unit testing is to test the correctness of isolated code, to fix bugs early in the development cycle and to save costs, to help the developers to understand the code base and enable them to make changes quickly.

Unit is the smallest testable part of software. Unit testing is used to validate that individual units of source code are working properly. In object-oriented programming, the smallest unit is a method and it contain a base/super class, abstract class or derived/child class but in procedural programming language a unit may be an individual program, function, procedure, etc., while The main advantage of the unit testing is used to improve the quality of code and save the tester's time and effort.

A test plan outlines the approach and strategy for conducting unit testing on the Harvestlinkup. It defines the scope of testing, the functionalities to be tested, and the tools or frameworks used for testing. Additionally, it details the resources, test environment, and schedule for executing the tests.

Test cases are specific scenarios and input combinations that evaluate the behaviour and correctness of individual components or functionalities in the system. Each test case has a set of preconditions, test data, and expected outcomes.

**5.3.1 Example Test Plan & Test Cases:**

**Test Case Id:** TC1

**Tests Used:** Black Box Testing/White Box Testing

**Correct Data**  name: Aju

Email: aju@gmail.com

Password: 3333

Phone number: 9447997304

**Function:** New User Registration

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SI NO** | **STEP** | **TEST DATA** | **EXPECTED RESULT** | **ACTUAL RESULT** | **STATUS** |
| 1. | Enter correct  registration details  and submit | Registration information | Registered  successfully | Registered  successfully | Success |
| 2. | Enter email  without @ symbol | Email:jeringmail.com | Invalid email | Invalid email | Success |
| 4. | Enter the contact number less than or greater than 10 characters | 94479973 | Invalid phone number | Invalid phone number | Success |

**Test Case:** TC2

**Test Used:** Black Box Testing/White Box Testing

Correct Data: Username: geo@gmail.com

Password: geo123

**Function:** Registered User Login

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SI NO** | **STEP** | **TEST DATA** | **EXPECTED RESULT** | **ACTUAL RESULT** | **STATUS** |
| 1. | Enter username only click login button | Username: aju@gmail.com | Please enter password | Please enter password | Success |
| 2. | Enter incorrect password or username | Username: aju@gmail.com Password: 0000 | Wrong username or password | Wrong username or password | Success |
| 3. | Enter password and username correctly | Username: geo@gmail.com Password: 3333 | Login successful | Login successful | Success |

**Test Case:** TC3

**Test Used:** Black Box Testing/White Box Testing

Correct Data: Username: aparna@gmail.com

Password: 5678

**Function:** Registered coordinator Login

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SI NO** | **STEP** | **TEST DATA** | **EXPECTED RESULT** | **ACTUAL RESULT** | **STATUS** |
| 1. | Enter username only click login button | Username:  aparna@gmail.com | Please enter password | Please enter password | Success |
| 2. | Enter incorrect password or username | Username: aparna@gmail.com  Password: 56789 | Wrong username or password | Wrong username or password | Success |
| 3. | Enter password and username correctly | Username: aparna@gmail.com Password: 5678 | Login successful | Login successful | Success |

**Test Case :** TC4

**Test Used:** Black Box Testing/White Box Testing

Correct Data: Product name: Ber Apple

Price: 110

Quantity: 100

**Function:** Registered Admin Event Adding

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SI NO** | **STEP** | **TEST DATA** | **EXPECTED RESULT** | **ACTUAL RESULT** | **STATUS** |
| 1. | Enter correct Event data |  | Event Added  successfully | Event Added  successfully | Success |
| 2. | Product adding with  empty data | No data | Alert to enter necessary data | Alerts tto enter required fields | Success |
| 3. | Product Adding with Invalid data | Product name: Ber Apple  Price: 110  Quantity: -10 | Alert user to enter valid data | Alerts user to Enter valid data | success |

**6.1 INTRODUCTION**

The purpose of the system testing is to identify and correct errors in the candidate system. Testing is an important element of the software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of the software as a system element and the costs associated with a software failure are motivated forces for well-planned through testing. Software testing is a critical element of software quality assurance and represents the ultimate quality review of specifications, design and code generation. Once the source code has been generated, the program should be executed before the customer gets it with the specific intend of finding and removing all errors, test must be designed using disciplined techniques. Testing technique provides the systematic guidance for designing tests. Testing presents an interesting challenge for the software engineer to hold software from an abstract concept to an acceptable implementation. Testing is the process of executing a program for finding errors. A good test is one that has high probability of finding an uncovered error.

System testing was conducted in order to detect errors and for comparing then the final system with the requirement specification report. That is, whether the system meets requirements. During testing the software was executed with a set of test cases and the output of the program for the test cases was evaluated to determine if the program is performing as it was expected to.

System testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartially. It has both functional and non-functional testing.

**6.2 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. In this testing, unit tested modules are combined into subsystems, which are then tested. Integration testing aims at whether the modules can be integrated properly and then combined to produce the output. The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items.

In sportifycmpus unit tested forms are integrated together to perform integration testing. This testing verifies that the updating in one form will make proper changes in the next form linked to it. It also checks the proper flow of system according to different inputs.

**6.3 SYSTEM TESTING**

The important and essential part of the system development phase, after designing and developing the software is system testing. In system testing, the functionalities of the system are tested from end-to-end perspective. It includes both functional and non-functional testing.

The behaviour of whole system is tested as defined by the scope of project. It includes test based on risk, requirement specification, use cases and other high-level description of the system. It is the final test that is used to verify the system to be delivered the specifications. All modules are completely tested when there is no error. Whole system is tested after combined together. Test all the requirements that satisfies or not. All modules are completely tested to confirm there is no error.

**6.3.1TEST PLAN & TEST CASES**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl no** | **STEP** | **TEST DATA** | **EXPECTED RESULT** | **ACTUAL RESULT** | **STATUS** |
| 1 | Enter Correct Registration details and submit | Registration Information | Registered Successfully | As Expected | Success |
| 2 | Enter correct Login Details and submit | Email and password | Login Successful | As Expected | Success |
| 3 | View Events | Solo, team events | Displays the product based on 2 categories | As expected | Success |
| 4 | Apply Event | Event ID | Alert the application is successful | As Expected | Success |
| 8 | Report | Booking details | Based on booking details invoice is generated | As Expected | success |
| 9 | Manufacturer adds product | Product Details | Product Added. | As Expected | Success |
| 10 | Product Based Comments | Product ID, Comment Details | Comment Added | As Expected | Success |

**7.1. INTRODUCTION**

System maintenance is an umbrella term that encompasses various forms of computer maintenance needed to keep a system running. The two main components of system maintenance are preventive and corrective maintenance. Preventive maintenance involves taking measures to help keep the system functioning, whereas corrective maintenance involves the replacement or repair of a system or its components after they have already failed.

**7.2. MAINTENANCE**

Maintenance projects can be used for enhancement or defect fixing. Enhancement projects are related to implementing new developments. It is basically modifications of software after it has been delivered in order to correct errors, to improve performance or other properties of IT systems .The key problems of software maintenance in IT systems include both management and technical problems. Important management issues are: adjusting to customer priorities, appropriate system maintenance personnel, cost estimation. The key technical issues are limited understanding, impact analysis, testing, measuring system maintainability, and infrastructure maintenance .It is well known that the guarantee of business survival is its continuous development and improvement of solutions, and thus also the continuous and adequate development of IT systems, combined with maintenance work carried out regularly on all components of the production environment.

**Types of system maintenance**

1. Corrective Maintenance:

Corrective maintenance is focused on identifying, diagnosing, and fixing defects or issues in the software after it has been deployed or released. The goal is to restore the software to its intended functionality and address problems that arise during its operational phase. This type of maintenance is reactive and is performed in response to incidents, bug reports, or user feedback indicating unexpected behavior or errors in the system.

The SRS should outline how the development team will handle corrective maintenance, including the process for reporting and tracking issues, the severity levels for different types of problems, and the timeline for resolving them. It should also include details about testing and validation procedures to ensure that the corrections are effective and do not introduce new issues.

1. Preventive Maintenance:

Preventive maintenance involves taking proactive measures to prevent potential issues and problems in the software before they occur. The primary aim is to enhance the system's stability and reliability, reducing the likelihood of corrective maintenance needs. Preventive maintenance can include activities such as code reviews, performance monitoring, and updating software components to the latest versions to address known vulnerabilities.

In the SRS, the preventive maintenance section should outline the strategies and practices that the development team will follow to minimize the risk of defects and system failures. This may involve regular system audits, security assessments, and adherence to coding standards and best practices.

1. Perfective Maintenance:

Perfective maintenance focuses on enhancing the software's functionality based on user feedback or evolving business requirements. Unlike corrective maintenance, which addresses defects, perfective maintenance involves making improvements to existing features to meet users' needs better, enhance performance, or introduce new capabilities.

In the SRS, the perfective maintenance section should detail how the development team will handle requests for enhancements and changes to the software's functionalities. It should define the process for gathering and prioritizing user feedback, conducting impact assessments, and implementing approved enhancements while ensuring that the existing functionalities remain unaffected.

1. Adaptive Maintenance:

Adaptive maintenance involves modifying the software to accommodate changes in its operating environment, such as hardware updates, software dependencies, or compliance with new regulations. The goal of adaptive maintenance is to ensure the software remains compatible and functional within the evolving ecosystem.

In the SRS, the adaptive maintenance section should describe how the software will be adapted to changes in the environment or technology landscape. It should include procedures for identifying environmental changes, assessing their impact on the system, and planning and executing necessary updates or modifications.

**8.1 INTRODUCTION**

The security of a computer system is a crucial task. It is a process of ensuring the confidentiality and integrity. Security is one of most important as well as the major task in order to keep all the threats or other malicious tasks or attacks or program away from the computer’s software system. A system is said to be secure if its resources are used and accessed as intended under all the circumstances, but no system can guarantee absolute security from several of various malicious threats and unauthorized access.

The security of a system can be threatened via two violations:

Threat: A program that has the potential to cause serious damage to the system.

Attack: An attempt to break security and make unauthorized use of the assets.

Security violations affecting the system can be categorized as malicious and accidental threats. Malicious threats as the name suggests are a kind of harmful computer code or web script designed to create system vulnerabilities leading to back doors and security breaches. Accidental threats on the other hand, are comparatively easier to be protected.

**8.2 OPERATING SYSTEM-LEVEL SECURITY**

The process of ensuring OS availability, confidentiality and integrity is known as operating system security. OS security refers to the processes or measures taken to protect the operating system from dangers, including viruses, worms, malware and remote hacker intrusions. Operating system security comprises all preventive-control procedures that protect any system assets that could be stolen, modified or deleted if OS security is breached. Anti-virus software is usually installed to avoid malware.

**8.3 DATABASE LEVEL SECURITY**

Database security refers to the range of tools, controls and measures designed to establish and preserve database confidentiality, integrity and availability. This system has only a single admin. The chances of insider threat are too low. Validation is done in each registration so that the human error is handled well.

**8.4 SYSTEM-LEVEL SECURITY**

System-level security refers to the architecture, policy and processes that ensure data and system security on individual computer systems. It facilitates the security of stand-alone and/or network computer systems/servers from events and processes that can exploit or violate its security or stature. It ensures that system access is granted only to legitimate and trusted individuals and applications. The key objective behind system-level security is to keep system secure regardless of security policies and processes at other levels. If other layers or levels are breached, the system must have the ability to protect itself.

**9.1 INTRODUCTION**

Planning and scheduling are distinct but inseparable aspects of managing the successful project. The process of planning primarily deals with selecting the appropriate policies and procedures in order to achieve the objectives of the project. Scheduling converts the project action plans for scope, time cost and quality into an operating timetable. The translating of the project criteria for scope, time, cost, and quality and the requirements for human resources, communications, risk and procurement into workable “machinery” for the project team a critical interface juncture for the project team. Taken together with the project plan and budget, the schedule becomes the major tool for the management of projects. In addition, the integrated cost-time schedule serves as the fundamental basis for monitoring and controlling project activity throughout its life cycle.

This basic level paper addresses the integrated processes of planning and scheduling of multifacet/multidisciplinary programs. The paper presents a working level summary of the major Project Management topics involved in the planning process. The paper also details a systematic process for transforming the Project Plan into the Schedule and the use of the Project Schedule as a model for project control. Intended for the project management novice, the paper concludes with a suggested professional development scheme.

**9.2 PLANNING A SOFTWARE PROJECT**

Project planning is at the heart of the project life cycle, and tells everyone involved where you’re going and how you’re going to get there. The planning phase is when the project plans are documented, the project deliverables and requirements are defined, and the project schedule is created. It involves creating a set of plans to help guide your team through the implementation and closure phases of the project. The plans created during this phase will help you manage time, cost, quality, changes, risk, and related issues. They will also help you control staff and external suppliers to ensure that you deliver the project on time, within budget, and within schedule.

The purpose of the project planning phase is to:

• Establish business requirements.

• Establish cost, schedule, list of deliverables, and delivery dates.

• Establish resources plans.

• Obtain management approval and proceed to the next phase.

The basic processes of project planning are:

• Scope planning– specifying the in-scope requirements for the project to facilitate creating the work breakdown structure.

• Preparation of the work breakdown structure – spelling out the breakdown of the project into tasks and sub-tasks.

• Project schedule development – listing the entire schedule of the activities and detailing their sequence of implementation.

• Resource planning – indicating who will do what work, at which time, and if any special skills are needed to accomplish the project tasks.

• Budget planning – specifying the budgeted cost to be incurred at the completion of the project.

• Procurement planning– focusing on vendors outside your company and subcontracting.

• Risk management– planning for possible risks and considering optional contingency plans and mitigation strategies.

• Quality planning – assessing quality criteria to be used for the project

**9.2.1 STEPS INVOLVED IN PLANNING A SYSTEM**

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process. The following figure is a graphical representation of the various stages of a typical SDLC.

A typical Software Development Life Cycle consists of the following stages:

1. Planning and Requirement Analysis
2. Defining Requirements
3. Designing the Product Architecture
4. Building or Developing the Product
5. Testing the Product
6. Deployment in the Market and Maintenance

**PROJECT SCHEDULING**

|  |  |
| --- | --- |
| **TASK** | **DATE** |
| Study The Existing System | 15-04-2023 |
| Identify The Problem | 26-04-2023 |
| Defining The System | 05-05-2023 |
| Database design | 30-05-2023 |
| Input Design | 10-06-2023 |
| Coding | 15-06-2023 |
| Testing | 15-07-2023 |
| Building a Prototype | 31-07-2023 |

**10.1 INTRODUCTION**

An enhancement is any product change or upgrade that increases software or hardware capabilities beyond original client specifications. Enhancements allow software and hardware product performance scalability.

This project is developed to increase the selling of products of small and medium industry farmers. And management of different reports and various customers can buy products through online platform.

**10.2 MERITS OF THE SYSTEM**

The system is newly automated and there are a lot of merits to this system. They are:

* User-friendly system: The system is designed in a manner that anyone can use this without providing any training. They can easily understand the way the system works.
* Faster performance of the system is a great advantage.
* Simple input: There are simple input screens which are easy to understand.
* Anyone from anywhere can access the system.

**10.3 LIMITATIONS OF THE SYSTEM**

Not every system is perfect and complete. Every system has its own limitations.

* High speed internet connectivity is needed. A mobile phone or a computer with a high-speed internet connection is highly recommended.
* Security**:** The security risks can be
  + client / server risk
  + data transfer and transaction risk
  + virus risk
* Legal issues**:** These issues arises when the customer data is fall in the hands of strangers.
* If the scope of the project is more widen then it may affect the budget of the software.
* Wrong decision making will result in the malfunction of the system.

**10.4 FUTURE ENHANCEMENT OF THE SYSTEM**

The system has been designed in such a way that it can be modified with very little

Effort when such needs arise in the future. New features can be added with slight modifications of software which make it easy to expand the scope of this project. Though the system is working on various assumptions, it can be modified easily to any kind of requirements.

This project has a very vast scope in future. The following are the future scope for this project.

* Real-time tracking using GPS cam be implemented in the system for efficient order tracking for the customers who choose home delivery.
* We can develop a messaging system that can alert the customers effectively when the order is out for delivery.
* Timely deliveries make customers satisfied. We can specify the time limit for delivery by the customers in future.
* We can develop a messaging system that can alert efficiently when an order is out for delivery.
* Offers and coupons can be provided to the customers.
* Changes in price based on market trend can be predicted by including the machine learning concepts.

**11.1 ORGANIZATION PROFILE**

**Name:** Progressive Software Solutions & Training

Progressive Software Solutions & Training is a fast developing, well established Information Technology enterprise with highly talented and efficient staff meeting the needs of clients throughout the world. The company is so popular that the clients feed the company with repeat orders, having faith in the capabilities. The staff includes management and engineering personnel, programmers, web developers, graphic designers, business analysts, technical writers etc., apart from the regular commercial and administrative hands. The motto of this reliable software provider is “DO BEST; GET BEST”. Progressive Software Solutions & Training committed to provide what they have promised. Progressive Software Solutions & Training strives for the satisfaction of their clients and dedicates themselves for the better services to their customers. Progressive Software Solutions & Training are famous in providing high quality, cost cutting, requirement oriented and value-added web and software solutions with remarkable cyber services. Progressive Software Solutions & Training are specialist in developing ever rewarding internet existence for ambitious companies which are in the hangover of swift moving internet facilities. Their aim is to grow with such organizations in a very fruitful manner. Their main concentration is to collect business oriented technological know-how and utilize them for the wellbeing of business concerns. Progressive Software Solutions & Training experts in various technologies are committed to deliver effective and creative service to customers.

Address: Progressive Software Solutions & Training

Tech Floor, Koyas Tower

P.O. Junction, Muvattupuzha

Ernakulam, Kerala – 686661

**11.2 DOCUMENT GLOSSARY, FIGURES, TABLES**

**Abbreviations**

**DBMS** -- DATABASE MANAGEMENT SYSTEM

**IDE** -- INTEGRATED DEVELOPMENT ENVIRONMENT

**HTML** -- HYPERTEXT NARKUP LANGUAGE

**CSS --** CASCADING STYLE SHEETS

## LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Fig.No** | **Figure Name** | **Page Number** |
| 1 | ER DIAGRAM | 11 |
| 2 | SEQUENCE DIAGRAM | 23 |
| 3 | ACTIVITY DIAGRAM | 26 |
| 4 | USE CASE DIAGRAM | 21 |

## LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **Table No** | **Table Name** | **Page Number** |
| 1 | ADMIN | 12 |
| 2 | DISTRICT | 12 |
| 3 | PLACE | 12 |
| 4 | USER | 13 |
| 5 | FARMER | 13 |
| 6 | PRODUCT | 14 |
| 7 | PRODUCTMARKET | 14 |
| 8 | PLANT | 15 |
| 9 | BOOKING | 16 |
| 10 | CART | 16 |
| 11 | PAYMENT | 16 |
| 12 | COMMENTS | 17 |
| 13 | WITHDRAW | 17 |
| 14 | MARKETBOOKING | 18 |
| 15 | CHAT | 18 |
| 16 | DELIVERY | 19 |

**11.3 REFERENCES**

**BOOKS**

1. Software Engineering A Practitioner’s Approach, Roger S Pressman, McGrawhill International, Edition, Sixth Edition.

2. Software Engineering, K K Agarwal and Yogesh Singh, New age international, Third Edition

3. Database Management Systems,Raghu Ramakrishnan and Johannes Gehrke, Third Edition, McGraw Hill, 2003.

4. An Introduction to Python by Guido Van Rossum, Fred L.Drake, Network Theory Limited. 5. Programming and Problem Solving with Python, Ashok Namdev Kamthane & Amit Ashok Kamthane, McGrawHill Education (India) Private Limited

6. Django for Beginners: Build websites with Python and Django Paperback – March 7, 2018 by William S. Vincent

**WEBSITE**

* <https://www.w3schools.com/django/django_views.php>
* <https://www.tutorialspoint.com/jquery-selectors.html>
* <https://www.javapoint.com/javascript-arrays.html>