PROJECT REPORT ON

ONLINE ORGANIC VEG MART

Submitted in partial fulfillment of requirements to

CB354-MINI PROJECT

By
POTTI.HARSHITHA(Y19CB048)
SAMINENI.LAVANYA(Y19CB053)
SHAIK. ABDUL AZEEZ(Y19CB054)

NOVEMBER 2021

COMPUTER SCIENCE AND BUSINESS SYSTEMS
RVR & JC COLLEGE OF ENGINEERING AND TECHNOLOGY
(NAAC A+ GRADE) (Approved by A.I.C.T.E) (Affiliated to Acharya Nagarjuna University) Chowdavaram
GUNTUR – 522 019

R.V.R & J.C.COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE&BUSINESS SYSTEMS

BONAFIDE CERTIFICATE

This is to certify that this project work titled ONLINE ORGANIC VEG MART is bonafide work of S. Lavanya(Y19CB053), P. Harshitha(Y19CB048), Sk. Azeez(Y19CB054) who have carried out the work under my supervision, and submitted in partial fulfillment of the requirements to **CB-354, MINI PROJECT**during the year 2020-2021.

A. Yaswanth Kumar & N. Neelima — Lecturer Incharge

Dr. M.V.P.Chandrashekar — Prof&HOD Of CSBS

ACKNOWLEDGEMENTS

The successful completion of any task would be incomplete without a proper suggestions, guidance and environment. Combination of these three factors acts like backbone to our Project **"ONLINE ORGANIC VEG MART".**

We are very much thankful to **Dr.K.RAVINDRA**, Principal of R.V.R. & J.C. College of Engineering, Guntur for having allowed delivering this Project - I.

We express our sincere thanks to **Dr.M.V.P Chandrashekar**, Professor, Head of the Department computer science and business systems for his encouragement and support to carry out this mini project successfully.

We are very glad to express our special thanks to **A.Yaswanth Kumar & N. Neelima** , Assistent Professor in Department of Information Technology for timely, guidance and providing us with most essential materials required for the completion of this report.

Finally we express our sincere thanks to all the **Teaching** and **Non-Teaching staff** of **CSBS DEPARTMENT** who have contributed for the successful completion of this report.

P.HARSHITHA(Y19CB048)
S.LAVANYA(Y19CB053)
Sk. AZEEZ(Y19CB054)

CONTENTS

Chapter No. & Name

PROBLEM STATEMENT

ONLINE ORGANIC VEG MART

Since the evolution of smart technology, it has made the lives of everyone easier to access data from online through various devices, whether it may be work or entertainment. Advances in technology eases people from doing some mudane tasks. Our project "ONLINE ORGANIC VEG MART" is one which helps people to get organic vegitables and fruits to their door step on one click. It facilitates customers to view products available online, submit online orders for items and services from a store that serves both walk-in customers and online customers.

PROJECT DESCRIPTION:

Here in this project the key roles are played by Admin and User as the admin have to maintain all the data regarding stock available in mart and should place items in website along with that he/she should manage the data of the users as well. Here the user role is to view items, add those items to cart and ordering them through payment. The whole process of this Online Organic Veg Mart is mentioned below.

REGISTRATION:

Users must register to the website before they shop online and they should provide their name, email address, password and mobile number in order to get registered. Once after registration users should login with valid username and password.

ORDERING THROUGH ONLINE:

Users should view items before adding them to cart and place order. The stock availability and replacing new items is take cared by admin.

ONLINE PAYMENT:

After adding the items to cart the user moves to checkout form in order to continue payment. After successful payment the order gets confirmed. Here admin provides multiple mode of payment for safe and secure online transactions and proceeds the confirmed order for delivery.

SOFTWARE REQUIREMENT SPECIFICATIONS

Software Requirements:

- Operating System : windows XP
- Coding language : HTML,CSS,BOOTSTRAP,PHP,JS
- Data Base : MySQL

Hardware Requirements:

- Personal computer with keyboard and mouse maintained with uninterrupted power supply.
- Processor : Intel® core™ i7
- Installed Memory (RAM) : 1.00 GB
- Hard Disc : 40 GB

REQUIREMENTS MODELING

The most important factor for the success of a project is whether the software product satisfies its users' requirements. Models constructed from an analysis perspective focuses determining these requirements. This means Requirement Model includes gathering and documenting facts and requests.

The use case model gives a perspective on many user requirements and models them in terms of what the software system can do for the user. Before the design of software which satisfies user requirements, we must analyze both the logical structure of the problem situation, and also the ways that its logical elements interact with each other. We must also need to verify the way in which different, possibly conflicting, requirements affect each other. Then we must communicate this understanding clearly and unambiguously to those who will design and build the software.

Use-case diagrams graphically represents system behavior (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider's (actor's) perspective. A use-case diagram may contain all or some of the use cases of a system.

A use-case diagram can contain:

- ·actors ("things" outside the system)
- ·use cases (system boundaries identifying what the system should do)
- interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented.

IDENTIFICATION OF ACTORS & USECASES

Identification of ACTORS :

Actors represent system users. They are NOT part of the system. They represent anyone or anything that interacts with the system.

An actor is someone or something that:

- Interacts with or uses the system
- Provides input to and receives information from the system
- Is external to the system and has no control over the use cases

Actors are discovered by examining:

- Who directly uses the system
- Who is responsible for maintaining the system
- External hardware used by the system
- Other systems that need to interact with the system

The needs of the actor are used to develop use cases. This insures that the system will be what the user expected.

Graphical depiction:

An actor is a stereotype of a class and is depicted as a "stickman" on a use-case diagram. For example,

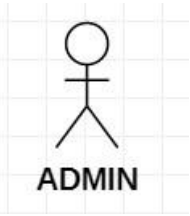
Student

Actors identified in the information system are:

1) Admin
2) User

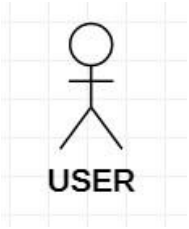
1) Admin: Admin has to login into his account
    - to view his/her profile,
    - to view user's login report,
    - to update his/her profile,
    - to change his/her password request

   and he has to logout the account after the desired actions complete.



2) User: User has to login into his/her account
   - to view his/her profile,
   - to view his/her categories,
   - to update his/her cart,
   - users finally makes order

and he/her has to logout the account after the desired actions complete.

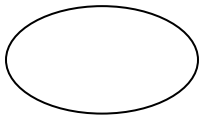


Identification of Use-Cases Or Sub Use-Cases

Use case can be described as a specific way of using the system from a user's perspective. A more detailed description might characterize a use case as:

- A pattern of behavior the system exhibits
- A sequence of related transactions performed by an actor and the system

The UML notation for use case is:

Login 

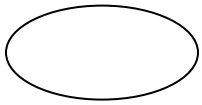
Purpose of use case's:

- Well structured use cases denote essential system or subsystem behaviours only, and are neither overly general nor too specific.
- A use case represents a functional requirement of the system as a whole
- Use cases represent an external view of the system
- A use case describes a set of sequences, in which each sequence represents the interaction of the things outside the system with the system itself.

Use-cases identified for face recognition based attendance management system are:

1.**Use-case name:** SIGNUP

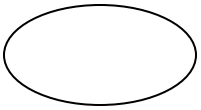
This is a use case which is used by actor to log on to the system and view the available set of operations that he can perform



Signup

2.**Use-case name:** LOGIN

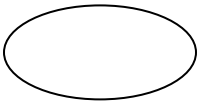
System allows user or admin to view his/her profile and can be able to select available functionalities respectively.



Login

3.**Use-case name:**

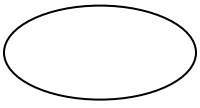
This use case allows the user or admin to change the password and secure delivery contact information.



Categories

6. **Use-case name:** ORDER

This use case allows admin to start marking attendance for a specified amount of time in two sessions i.e, morning session and afternoon session.

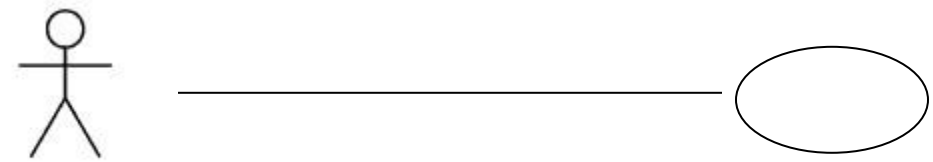


Order

Identification of RELATIONS

Association Relationship:

An association provides a pathway for communication. The communication can be between use cases, actors, classes or interfaces. If two objects are usually considered independently, the relationship is an association.



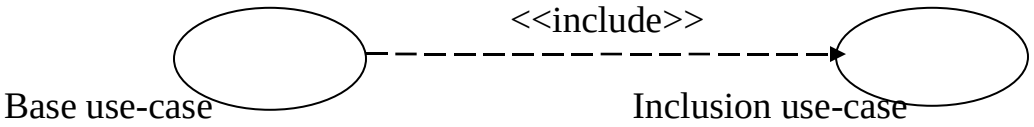
Dependency Relationship:

A dependency is a relationship between two model elements in which a change to one model element will affect the other model element. Use a dependency relationship to connect model elements with the same level of meaning.

We can provide here

1.Include relationship:

It is a stereotyped relationship that connects a base use case to an inclusion use case .An include relationship specifies how the behavior in the inclusion use case is used by the base use case.

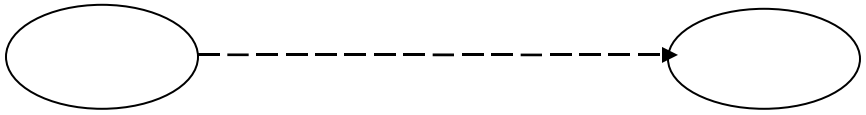


2.Extend relationship:

It is a stereotyped relationship that specifies how the functionality of one use case can be inserted into the functionality of another use case.

<<extend>> is used when you wish to show that a use case provides additional functionality that may be required in another use case.

<<extend>>



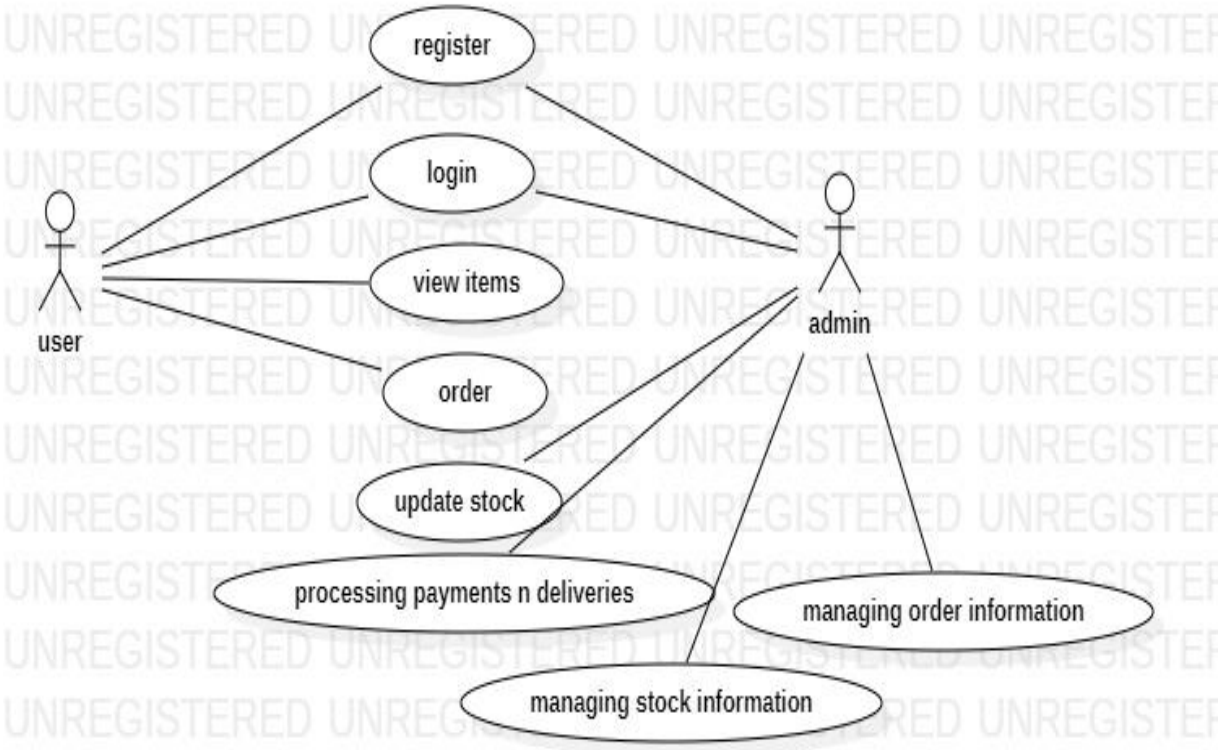
Print campaign summary check campaign budge

CONSTRUCTION OF USE CASE DIAGRAM AND FLOW OF EVENTS.

Use-case diagrams graphically represent system behavior. These diagrams present a high level view of how the system is used as viewed from an outsider's perspective.

* Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented.

USE CASE DIAGRAM FOR ONLINE VEG MART:



FLOW OF EVENTS

A flow of events is a sequence of transactions performed by the system. They typically contain very detailed information. Flow of events document is typically created in the elaboration phase.

Each use case is documented with flow of events

- A description of events needed to accomplish required behaviour
- Written in terms of what the system should do, NOT how it should do it
- Written in the domain language, not in terms of the implementation

A flow of events should include

- When and how the use case starts and ends
- What interaction the use case has with the actors
- What data is needed by the use case
- The description of any alternate or exceptional flows

The flow of events for a use case is contained in a document called the use case specification. Each project should use a standard template for the creation of the use case specification. Includes the following

1. Use case name – Brief Description
2. Flow of events –
    1. Basic flow
    2. Alternate flow
    3. Special requirements
    4. Pre conditions
    5. Post conditions
    6. Extension points

FLOW OF EVENTS FOR ONLINE ORGANIC MART:

FLOW OF EVENTS FOR REGISTRATION:

Name of the USECASE: Registration
Primary actor: User
Secondary actor: Admin
Pre-condition: User must provide valid credentials.
Basic Flow:

*User should browse website for registration/signup.
*In order to register user must provide their Username, Mobile number, Email id and finally he must set up a password for the created account.

Post-condition: User successfully registered/ signup to his/her account.
Alternative flow: User's should not provide fake credentials as it will be trouble in placing orders.

FLOW OF EVENTS FOR LOGIN:

Name of the usecase: Login
Primary actor: User
Secondary actor: Admin
Pre-condition: User must create an account in the website before they login.
Basic Flow: * User should click on the website and should choose login option on the navbar.

* User have to provide valid credentials that they provided while creating account.

Post-condition: User successfully got login to their account
Alternative flow: User should not use other credentials other than their own credentials. They should maintain internet connection while using this website.

FLOW OF EVENTS FOR ORDER:

Name of the usecase: Order
Primary actor: User
Secondary actor: Admin
Pre-condition: User must login to their accounts
Basic Flow:*After login to their account user chooses some items in the provided categories (Vegetables, Fruits).
* The selected items must be added to cart with the help of Add To Cart button.
* In the cart they should click on Checkout button in order to redirect for the payment page.
* They have to enter the required details to proceed for payment.
Post-condition: User have successfully places their order.
Alternative flow: User should not provide fake transaction details while processing payment.

FLOW OF EVENTS FOR MANAGING ORDER INFORMATION:

Name of the usecase: Managing order information.
Primary Actor: Admin
Secondary actor: Database server
Pre-condition: Admin must login with valid staff details
Basic Flow: * As admin first duty is he should update the stock information according to the quantity available in the store.
* Admin must store the vegetable and fruits information in stores database.
* Whenever admin receives an order he/she must confirm the order by checking the payment made by users.
* After confirming the order admin must transfer the data of the user to the delivery section staff and should update the database.
Post-condition: Admin has successfully maintained the order information in the database.
Alternative flow: Admin should not use another staff details to store the info in database and should maintain internet connection to complete the whole process.

BUILDING A BUSINESS PROCESS MODEL USING UML ACTIVITY DIAGRAM

An Activity diagram is a variation of a special case of a state machine, in which the states are activities representing the performance of operations and the transitions are triggered by the completion of the operations. The purpose of Activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. You can also use activity diagrams to model code-specific information such as a class operation. Activity diagrams are very similar to a flowchart because you can model a workflow from activity to activity. An activity diagram is basically a special case of a state machine in which most of the states are activities and most of the transitions are implicitly triggered by completion of the actions in the source activities.

- Activity Diagrams also may be created at this stage in the life cycle. These diagrams represent the dynamics of the system. They are flow charts that are used to show the workflow of a system; that is, they show the flow of control from activity to activity in the system, what activities can be done in parallel, and any alternate paths through the flow.
- At this point in the life cycle, activity diagrams may be created to represent the flow across use cases or they may be created to represent the flow within a particular use case.
- Later in the life cycle, activity diagrams may be created to show the workflow for an operation.

The following tools are used on the activity diagram toolbox to model activity diagrams:

Activities:

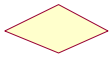
An activity represents the performance of some behavior in the workflow.

Transitions:

Transitions are used to show the passing of the flow of control from activity to activity. They are typically triggered by the completion of the behavior in the originating activity.

Decision Points:

When modeling the workflow of a system it is often necessary to show where the flow of control branches based on a decision point. The transitions from a decision point contain a guard condition, which is used to determine which path from the decision point is taken. Decisions along with their guard conditions allow you to show alternate paths through a work flow.



Decision point

Start state:

A start state explicitly shows the beginning of a workflow on an activity diagram or the beginning of the execution of a state machine on a state chart diagram.

End state:

An End state represents a final or terminal state on an activity diagram or state chart diagram. Place an end state when you want to explicitly show the end of a workflow on an activity diagram or the end of a state chart diagram. Transitions can only occur into an end state; however, there can be any number of end states per context.

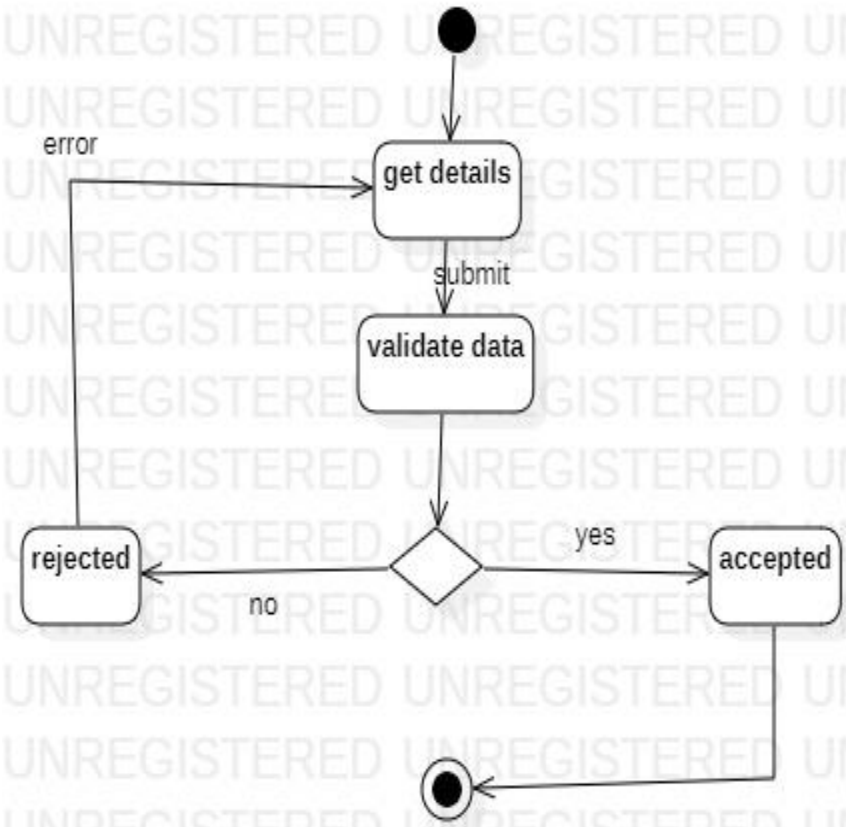
End state

Swim Lanes:

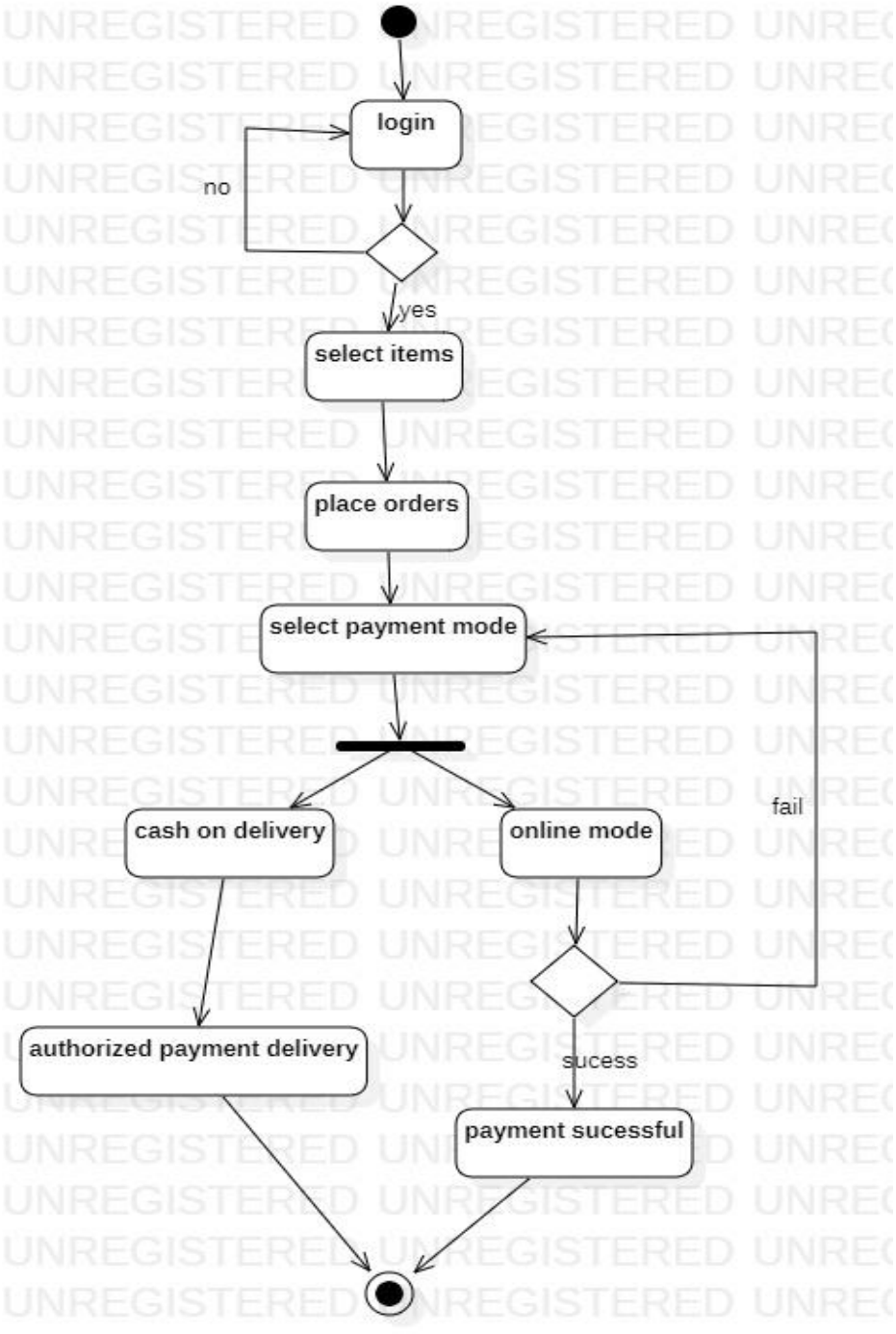
Swim lanes may be used to partition an activity diagram. This typically is done to show what person or organization is responsible for the activities contained in the swim lane.

- Horizontalsynchronization
- Vertical synchronization.

ACTIVITY DIAGRAM FOR REGISTRATION:



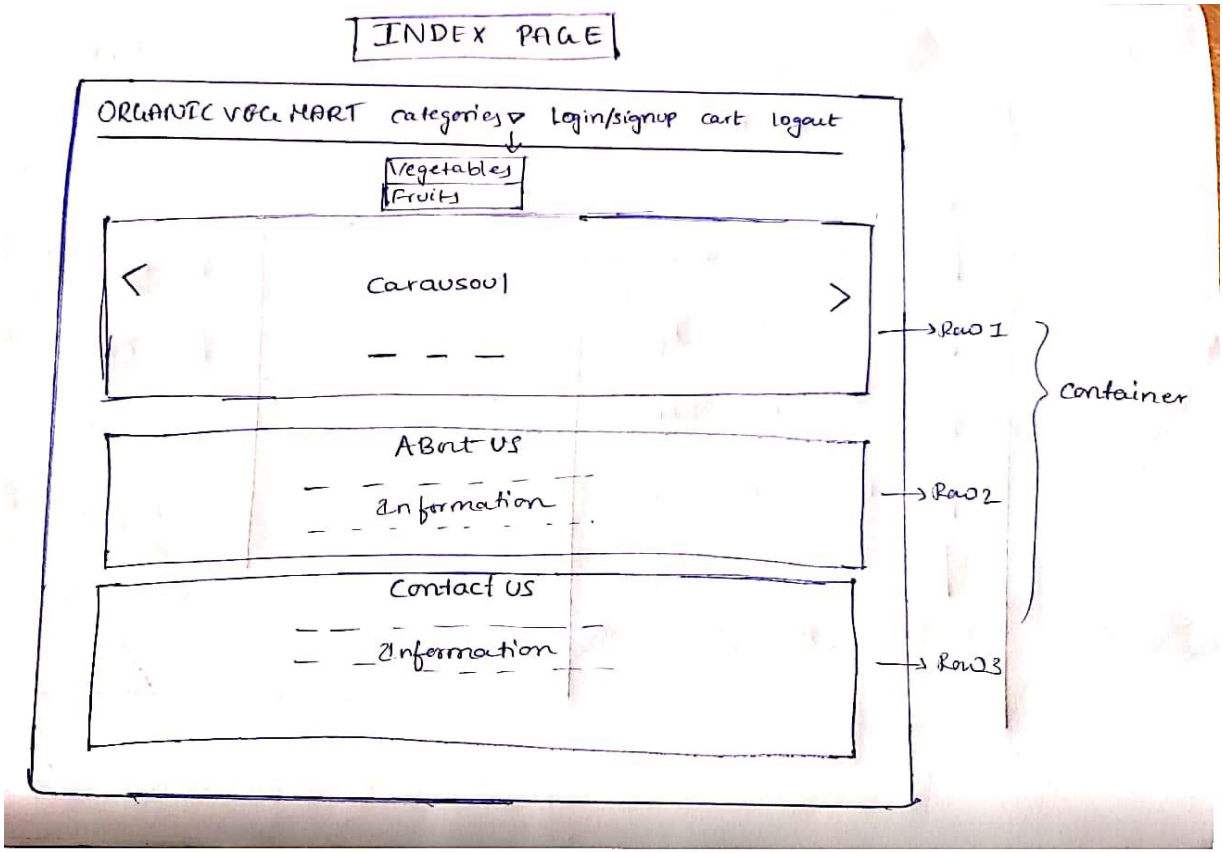
ACTIVITY DIAGRAM TO ORDER ITEMS:



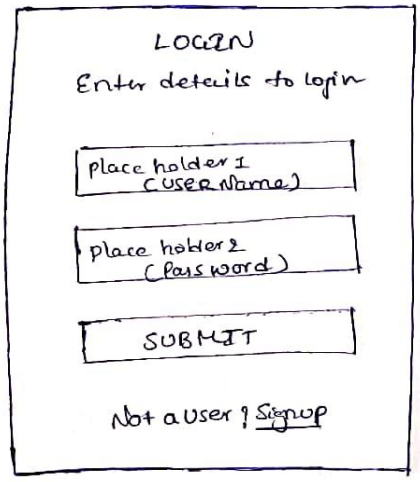
CONSTRUCTION OF PROTOTYPES

As the requirements for a system emerge in the form of use cases, it is sometimes helpful to build simple prototypes of how some of the use cases will work. A prototype is a working model of part of the system usually a program with limited functionality that is built to test out some aspect of how the system will work. Prototypes can be used to help elicit requirements. Showing users how the system might provide some of the use cases often produces a stronger reaction than showing them a series of abstract diagrams. Their reaction may contain useful information about requirements.

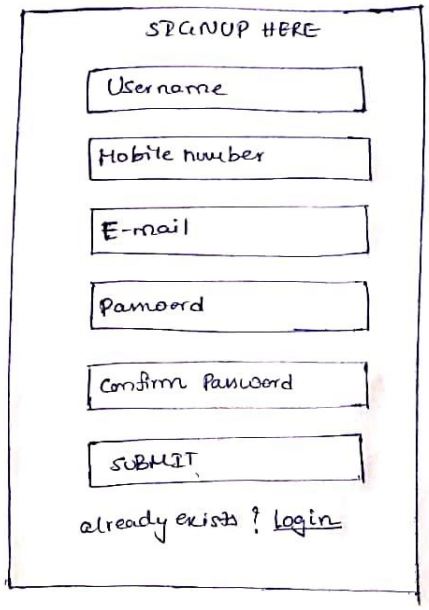
FOLLOWING ARE THE PROTOTYPES FOR ORGO-V-MART:



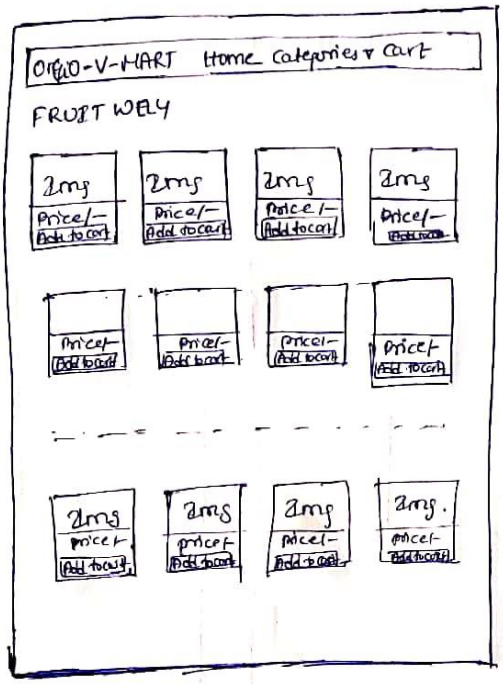
LOGIN

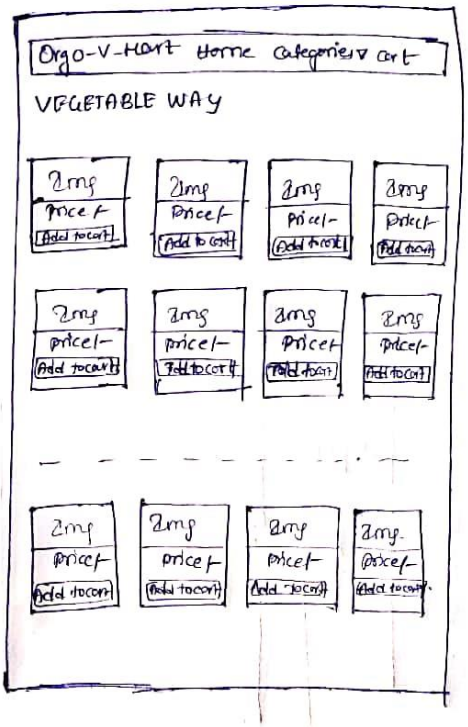


SIGNUP

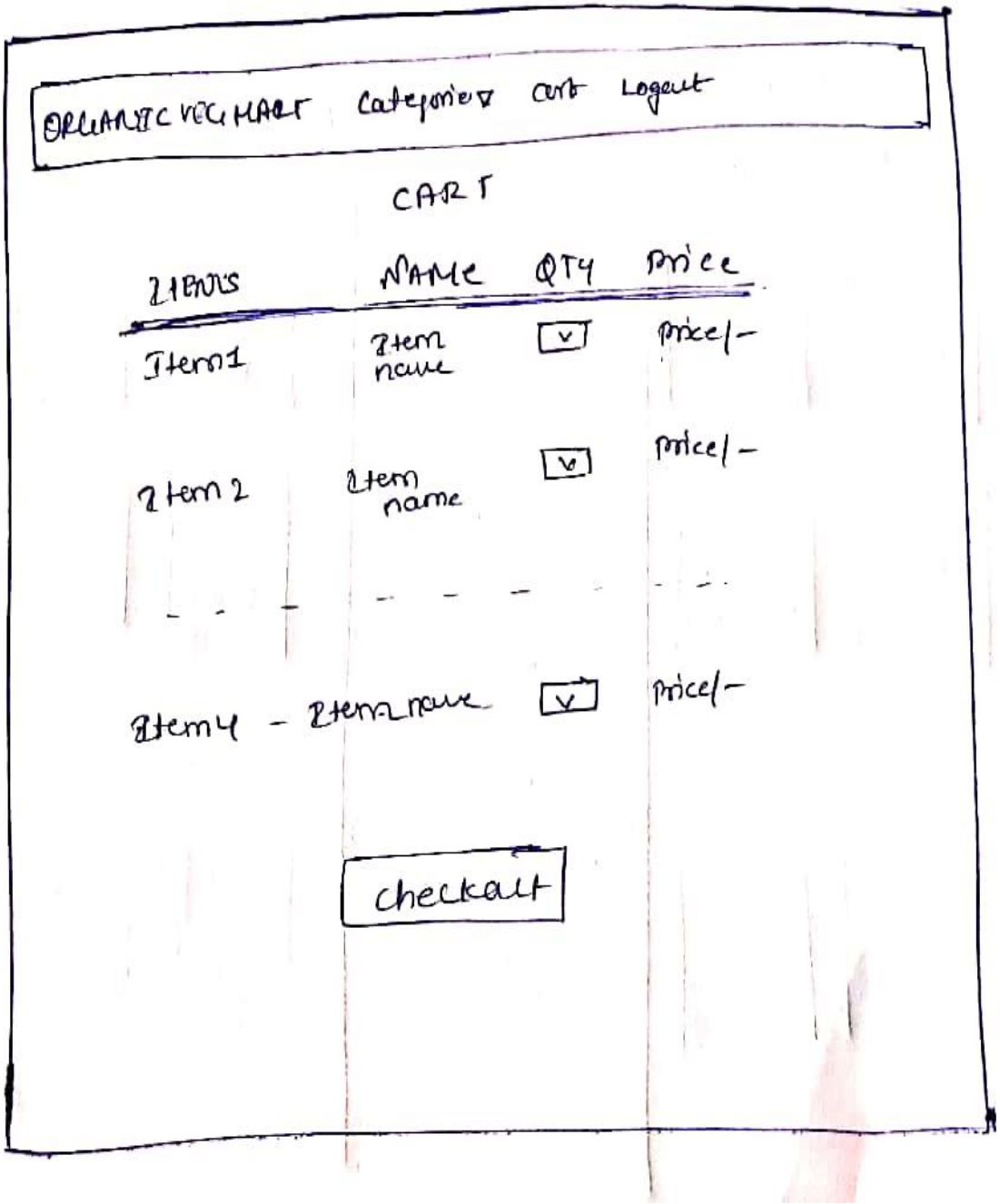


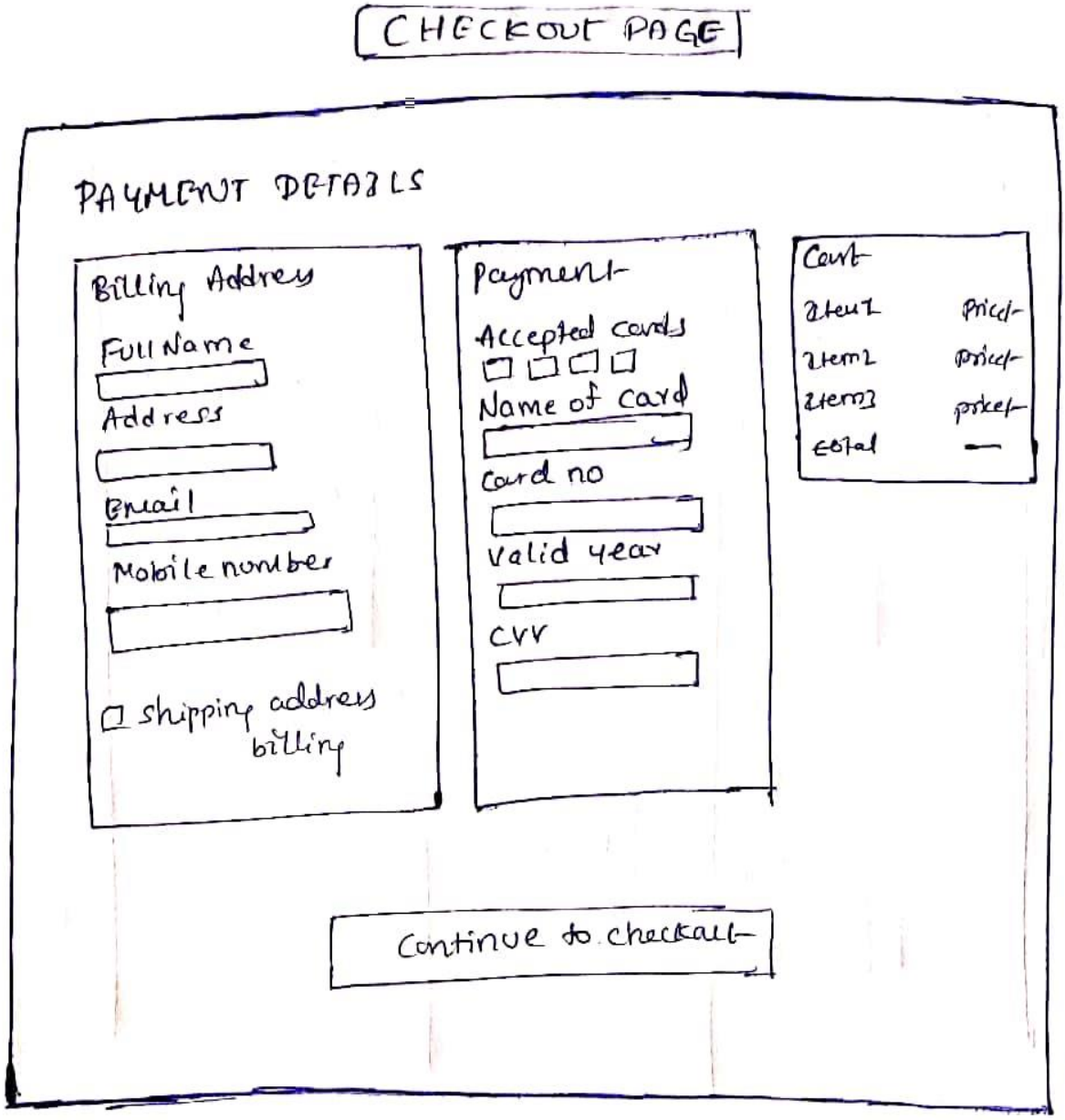
Categories Page





CART PAGE


ORGANIC VEGMART
Categories
cart
Logout
CART
ITEMS
NAME
QTY
Price
Item1
Item name
Price/-
Item 2
Item name
Price/-
Item4 - Item name
Price/-
checkout


CHECKOUT PAGE
PAYMENT DETAILS
Billing Address
Full Name
Address
Email
Mobile number
shipping address
billing
Payment-
Accepted cards
Name of card
card no
Valid year
CVV
Cart
Item1
Price-
Item2
Price-
Item3
price-
Total
—
Continue to checkout-

CONSTRUCTION OF SEQUENCE DIAGRAMS.

A sequence diagram is a graphical view of a scenario that shows object interaction in a time based sequence--what happens first, what happens next… Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.

A sequence diagram has two dimensions: the vertical dimension represents time; the horizontal dimension represents different objects. The vertical line is called the object's lifeline. The lifeline represents the object's existence during the interaction.

Steps:

* An object is shown as a box at the top of a dashed vertical line. Object names can be specific (e.g., Algebra 101, Section 1) or they can be general (e.g., a course offering). Often, an anonymous object (class name may be used to represent any object in the class.)

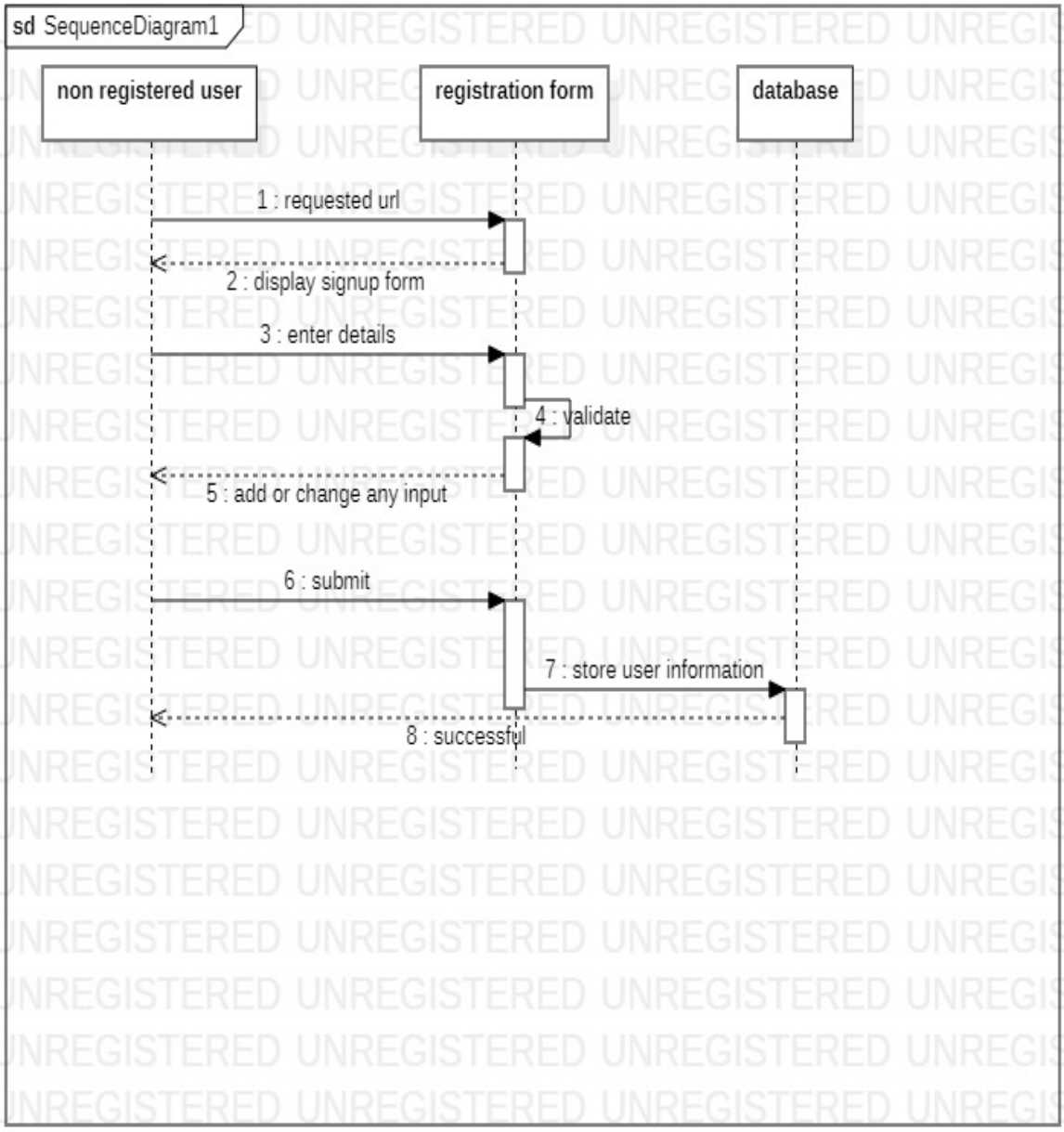
* Each message is represented by an Arrow between the lifelines of two objects. The order in which these messages occur is shown top to bottom on the page. Each message is labeled with the message name.

There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other. A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

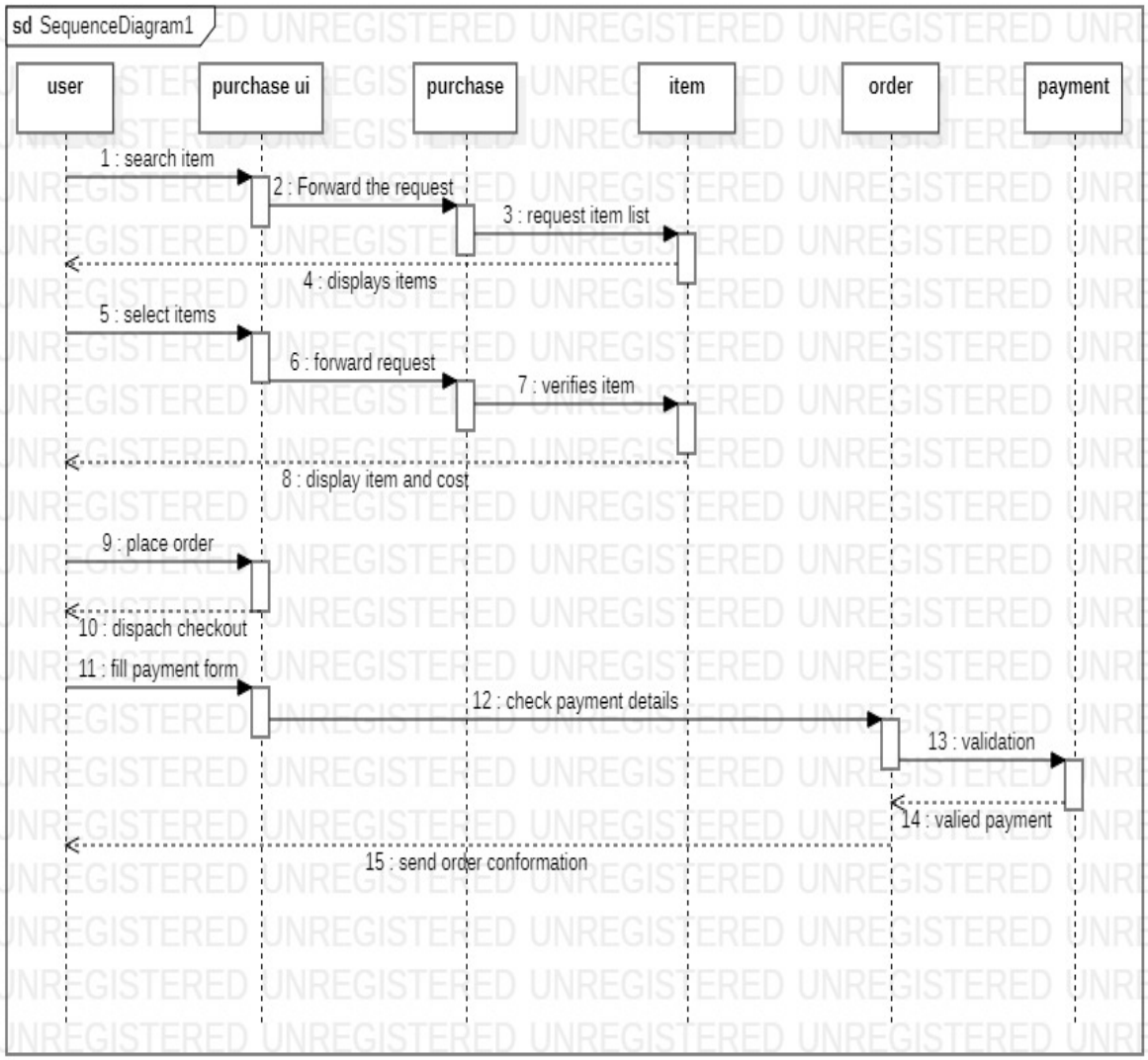
ELEMENTS OF SEQUENCE DIAGRAM:

- Objects
- Links
- Messages
- Focus of control
- Object life line

SEQUENCE DIAGRAM FOR REGISTRATION:



SEQUENCE DIAGRAM FOR ORDERING ITEMS:



CONSTRUCTION OF COLLABORATION DIAGRAMS

Collaboration diagrams are the second kind of interaction diagram in the UML diagrams. They are used to represent the collaboration that realizes a use case. The most significant difference between the two types of interaction diagram is that a collaboration diagram explicitly shows the links between the objects that participate in a collaboration as in sequence diagrams, there is no explicit time dimension.

Message labels in collaboration diagrams:

Messages on a collaboration diagram are represented by a set of symbols that are the same as those used in a sequence diagram, but with some additional elements to show sequencing and recurrence as these cannot be inferred from the structure of the diagram. Each message label includes the message signature and also a sequence number that reflects call nesting, iteration, branching, concurrency and synchronization within the interaction.

The formal message label syntax is as follows:
[predecessor] [guard-condition] sequence-expression [return-value ':='] message-name' (' [argument-list] ')'

A *predecessor* is a list of sequence numbers of the messages that must occur before the current message can be enabled. This permits the detailed specification of branching pathways. The message with the immediately preceding sequence number is assumed to be the predecessor by default, so if an interaction has no alternative pathways the predecessor list may be omitted without any ambiguity.
The syntax for a predecessor is as follows:

sequence-number { ',' sequence-number} *'I'*

The *'I'* at the end of this expression indicates the end of the list and is only included when an explicit predecessor is shown.

Guard conditions are written in Object Constraint Language (OCL), and are only shown where the enabling of a message is subject to the defined condition. A guard condition may be used to represent the synchronization of different threads of control.

A *sequence-expression* is a list of integers separated by dots ('.') optionally followed by a *name* (a single letter), optionally followed by a *recurrence* term and terminated by a colon.
A sequence-expression has the following syntax:

integer { '.' integer } [name] [recurrence] ':'

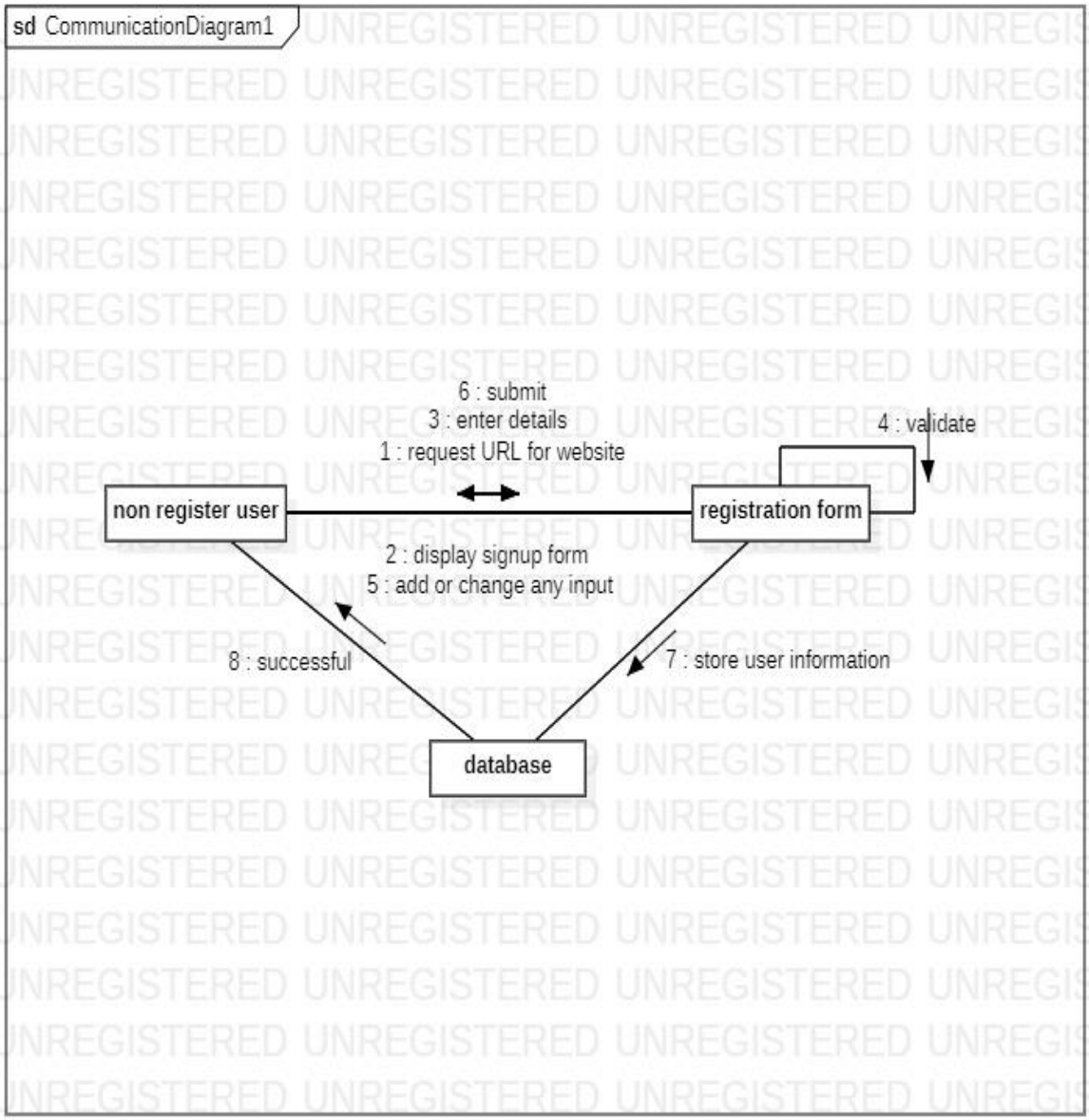
In this expression *integer* represents the sequential order of the message. This may be nested within a loop or a branch construct, so that, for example, message 5.1 occurs after message 5.2 and both are contained within the activation of message 5.

The *name* of a sequence-expression is used to differentiate two concurrent messages since these are given the same sequence number. For example, messages 3.2.1a and 3.2.1b are concurrent within the activation of message 3.2.
Recurrence reflects either iterative or conditional execution and its syntax is as follows:
Branching: '['condition-clause‘] ,
Iteration: ‘ * ‘ ‘ [‘ iteration-clause ‘] '

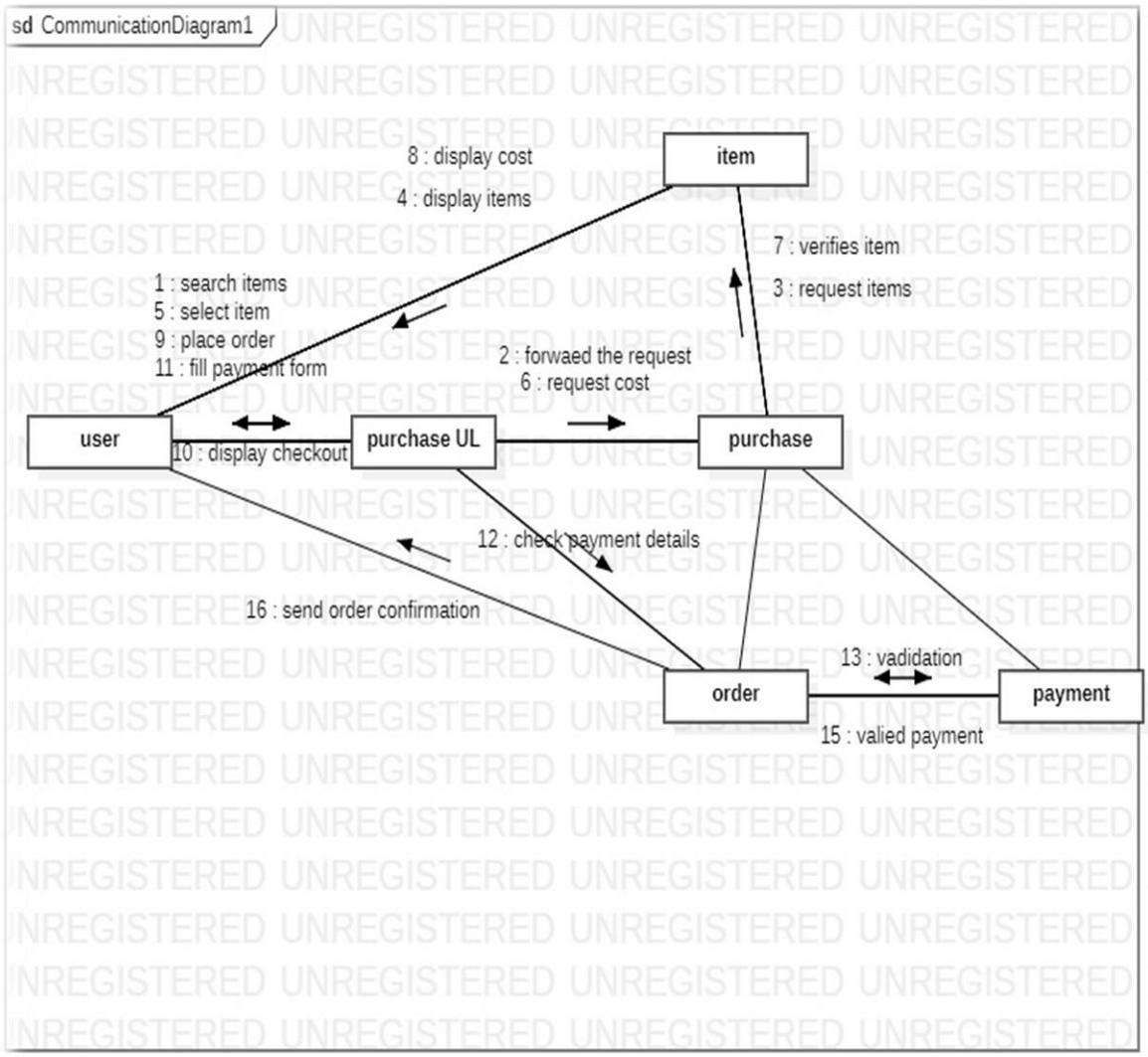
Elements:

•Objects, Messages, Path, Sequence Numbers, Links

COLLABORATION DIAGRAMS FOR REGISTRATION:



COLLABORATION DIAGRAM FOR ORDERING ITEMS:



Construction of UML static class diagram

Class diagrams contain icons representing classes, packages, interfaces, and their relationships. You can create one or more class diagrams to depict the classes at the top level of the current model; such class diagrams are themselves contained by the top level of the current model.

Class:

A Class a description of a group of objects with common properties (attributes), common behavior (operations), common relationships to other objects, and common semantics.

Thus, a class is a template to create objects. Each object is an instance of some class and objects cannot be instances of more than one class.

Classes should be named using the vocabulary of the domain.

For example, the CourseOffering class may be defined with the following characteristics:

Attributes - location, time offered

Operations - retrieve location, retrieve time of day, add a student to the offering .

Each object would have a value for the attributes and access to the operations specified by the CourseOffering class.

In the UML, classes are represented as compartmentalized rectangles.

The top compartment contains the name of the class.

The middle compartment contains the structure of the class (attributes).

The bottom compartment contains the behavior of the class (operations) as shown below.

OBJECT :

- AN OBJECT IS a representation of an entity, either real-world or conceptual.
- An object is a concept, abstraction, or thing with well defined boundaries andmeaning for an application.
- Each object in a system has three characteristics: state, behavior, and identity.

STATE : THE STATE OF an object is one of the possible conditions in which it may exist. The state of an object typically changes over time, and is defined by a set of properties (called attributes), with the values of the properties, plus the relationships the object may have with other objects.
For example, a course offering object in the registration system may be in one of two states: *open* and *closed.* It is available in the open state if value is < 10 otherwise closed.

Behavior :

- Behavior determines how an object responds to requests from other objects .

- Behavior is implemented by the set of operations for the object.

For example , In the registration system, a course offering could have the behaviors *add* a *student* and *delete* a *student.*

Identity :

- Identity means that each object is unique even if its state is identical to that of another object.

Attributes

Attributes are part of the essential description of a class. They belong to the class, unlike objects, which instantiate the class. Attributes are the common structure of what a member of the class can 'know'. Each object will have its own, possibly unique, value for each attribute.
Guidelines for identifying attributes of classes are as follows:

- Attributes usually correspond to nouns followed by prepositional phrases.
- Keep the class simple; state only enough attribute to define object state.
- Attributes are less likely to be fully described in the problem statement.
- Omit derived attributes.
- Do not carry discovery attributes to excess.

STEREOTYPES AND CLASSES :

As like stereotypes for relationships in use case diagrams. Classes can also have stereotypes. Here a stereotype provides the capability to create a new kind of modeling element. Here, we can create new kinds of classes. Some common stereotypes for a class are entity Class, boundary Class, control class, and exception.

Entity Classes

- An **entity class** models information and associated behavior that is generally long lived.
- This type of class may reflect a real-world entity or it may be needed to perform tasks internal to the system.
- They are typically independent of their surroundings; that is, they are not sensitive to how the surroundings communicate with the system.

Boundary Classes :

Boundary classes handle the communication between the system surroundings and the inside of the system. They can provide the interface to a user or another system (i.e., the interface to an actor). They constitute the surroundings dependent part of the system. Boundary classes are used to model the system interfaces.

Boundary classes are also added to facilitate communication with other systems. During design phase, these classes are refined to take into consideration the chosen communication protocols.

Control Classes

- Control classes model sequencing behavior specific to one or more use cases.
- Control classes coordinate the events needed to realize the behavior specified in the use case.
- Control classes typically are application-dependent classes.

  In the early stages of the Elaboration Phase, a control class is added for each actor/use case pair. The control class is responsible for the flow of events in the use case.

In the early stages of the Elaboration Phase, a control class is added for each actor/use case pair. The control class is responsible for the flow of events in the use case.

NEED FOR RELATIONSHIPS AMONG CLASSES:

All systems are made up of many classes and objects. System behaviour is achieved through the collaborations of the objects in the system.
Two types of relationships in CLASS diagram are:

1. Association Relationship
2. Aggregation Relationship

1. Association Relationship:

An association is a bidirectional semantic connection between classes. It is not a data flow as defined in structured analysis and design data may flow in either direction across the association. An association between classes means that there is a link between objects in the associated classes.

2. Aggregation Relationship:

An aggregation relationship is a specialized form of association in which a whole is related to its part(s). Aggregation is known as a "part-of" or containment relationship. The UML notation for an aggregation relationship is an association with a diamond next to the class denoting the aggregate(whole).

3. Super-sub structure (Generalization Hierarchy):

These allow objects to be build from other objects. The super-sub class hierarchy is a relationship between classes, where one class is the parent class of another class.

NAMING RELATIONSHIP:

An association may be named. Usually the name is an active verb or verb phrase that communicates the meaning of the relationship. Since the verb phrase typically implies a reading direction, it is desirable to name the association so it reads correctly from left to right or top to bottom. The words may have to be changed to read the association in the other direction (e.g., Buses are allotted to Routes). It is important to note that the name of the association is optional.

ROLE NAMES:

The end of an association where it connects to a class is called an association role. Role names can be used instead of association names.

A role name is a noun that denotes how one class associates with another. The role name is placed on the association near the class that it modifies, and may be placed on one or both ends of an association line.

- It is not necessary to have both a role name and an association name.
- Associations are named or role names are used only when the names are needed for clarity.

MULTIPLICITY INDICATORS:

Although multiplicity is specified for classes, it defines the number of objects that participate in a relationship. Multiplicity defines the number of objects that are linked to one another. There are two multiplicity indicators for each association or aggregation one at each end of the line. Some common multiplicity indicators are

1	Exactly one
0... *	Zero or more
1... *	One or more
0... 1	Zero or one
5... 8	Specific range (5, 6, 7, or 8)
4... 7, 9	Combination (4, 5, 6, 7, or 9)

Analyzing the object behavior by constructing the UML STATE CHART DIAGRAM

Use cases and scenarios provide a way to describe system behavior; in the form of interaction between objects in the system. Sometimes it is necessary to consider inside behavior of an object.

A state chart diagram shows the **states** of a single object, the events or messages that cause a **transition** from one state to another, and the **actions** that result from a state change. As in Activity diagram , state chart diagram also contains special symbols for start state and stop state.

State chart diagram cannot be created for every class in the system , it is only for those class objects with significant behavior.

State chart diagrams are closely related to activity diagrams. The main difference between the two diagrams is state chart diagrams are state centric, while activity diagrams are activity centric. A state chart diagram is typically used to model the discrete stages of an object's lifetime, whereas an activity diagram is better suited to model the sequence of activities in a process.

STATE:

A state represents a condition or situation during the life of an object during which it satisfies some condition, performs some action or waits for some event.

UML notation for STATE is

To identify the states for an object its better to concentrate on sequence diagram. In an ESU the object for Course Offering may have in the following states, initialization, open and closed state. These states are obtained from the attribute and links defined for the object. Each state also contains a compartment for actions.

Actions:

Actions on states can occur at one of four times:

- on entry
- on exit
- do
- on event.

on entry : What type of action that object has to perform after entering into the state.

on exit : What type of action that object has to perform after exiting from the state.

Do : The task to be performed when object is in this state, and must to continue until it leaves the state.

on event : An on event action is similar to a state transition label with the following

syntax: event(args)[condition] : the Action

State Transition:

A state transition indicates that an object in the source state will perform certain specified actions and enter the destination state when a specified event occurs or when certain conditions are satisfied. A state transition is a relationship between two states, two activities, or between an activity and a state. You can show one or more state transitions from a state as long as each transition is unique. Transitions originating from a state cannot have the same event, unless there are conditions on the event.

Transitions are labeled with the following syntax:

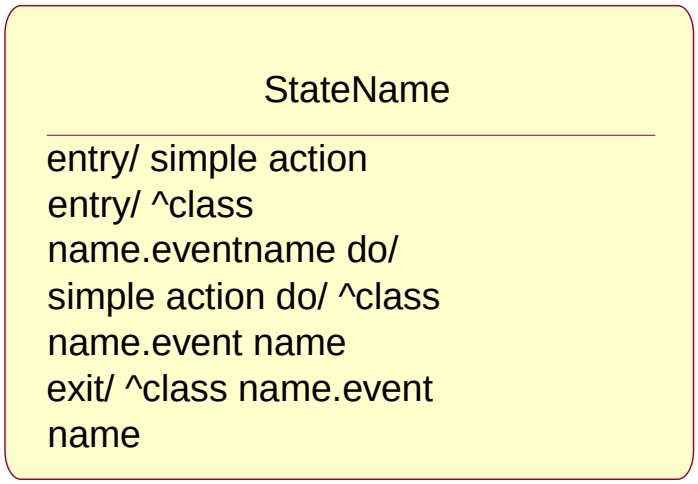
event (arguments) [condition] / action ^ target. send Event (arguments) Only one event is allowed per transition, and one action per event.

State Details :

Actions that accompany all state transitions into a state may be placed as an entry action within the state. Like wise that accompany all state transitions out
of a state may be placed as exit actions within the state. Behavior that occurs within the state is called an activity.

An activity starts when the state is entered and either completes or is interrupted by an outgoing state transition. The behavior may be a simple action or it may be an event sent to another object.

UML notation for State Details:



Purpose of State chart diagram:

- State chart diagrams are used to model dynamic view of a system.
- State chart diagrams are used to modelling lifetime of an object.
- State chart diagrams are used to focus on the changing state of a system driven by events.
- It will also be used when showing the behavior of a class over several use cases.

CONSTRUCTION OF IMPLEMENTATION DIAGRAMS

Component diagrams:

In a large project there will be many files that make up the system. These files will have dependencies on one another. The nature of these dependencies will depend on the language or languages used for the development and may exist at compile-time, at link-time or at run-time. There are also dependencies between source code files and the executable files or byte code files that are derived from them by compilation. Component diagrams are one of the two types of implementation diagram in UML. Component diagrams show these dependencies between software components in the system. Stereotypes can be used to show dependencies that are specific to particular languages also.

A component diagram shows the allocation of classes and objects to components in the physical design of a system. A component diagram may represent all or part of the component architecture of a system along with dependency relationships.

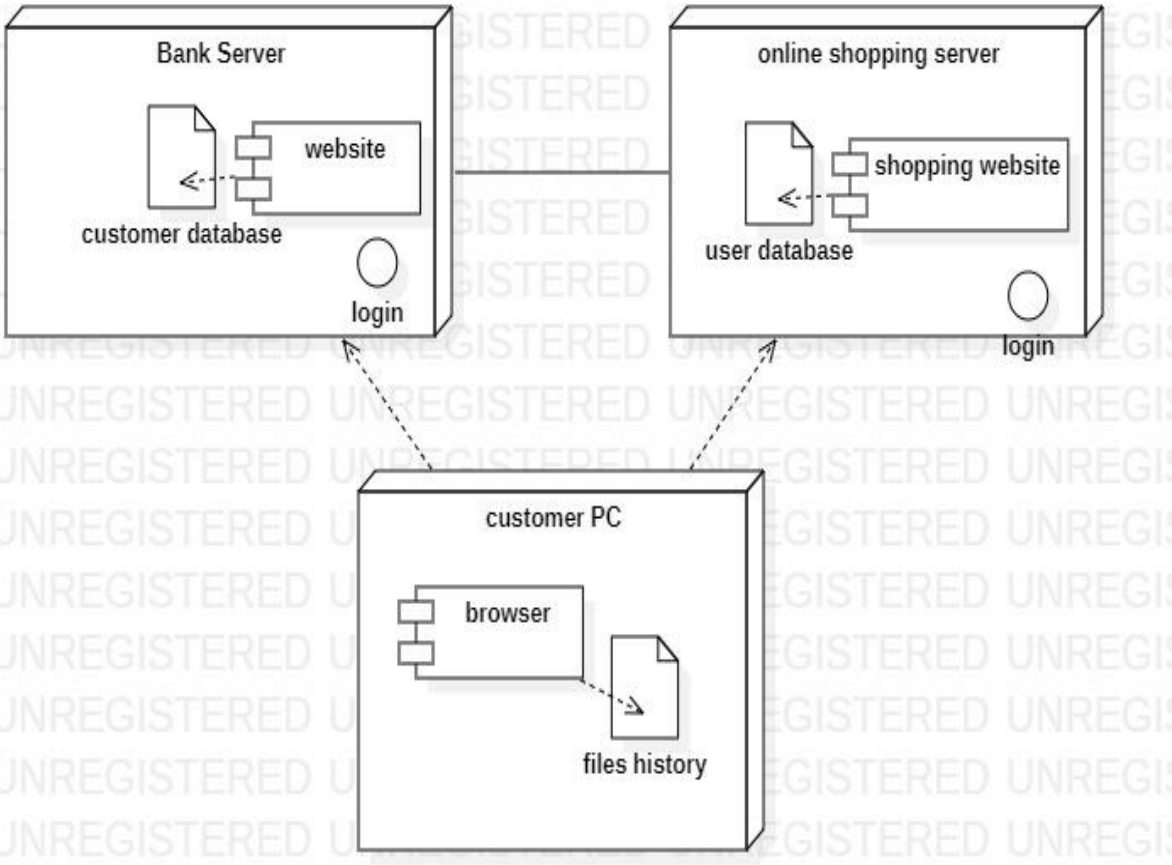
DEPLOYMENT DIAGRAMS:

The second type of implementation diagram provided by UML is the deployment diagram. Deployment diagrams are used to show the configuration of run-time processing elements and the software components and processes that are located on them.

Deployment diagrams are made up of nodes and communication associations. Nodes are typically used to show computers and the communication associations show the network and protocols that are used to communicate between nodes. Nodes can be used to show other processing resources such as people or mechanical resources.

Nodes are drawn as 3D views of cubes or rectangular prisms, and the following figure shows a simplest deployment diagram where the nodes connected by communication associations.

DEPLOYEMENT DIAGRAM FOR ONLINE VEG MART



SAMPLE APPLICATION CODE AND DATABASE

CODE FOR USER REGISTRATION/SIGNUP :

```
<html>
<head>
 <title>Signup Here </title>
<style type="text/css">
 @import url(https://fonts.googleapis.com/css?family=Roboto:300);
header .header{
  background-color: #fff;
  height: 45px;
}
header a img{
  width: 134px;
margin-top: 4px;
}
.login-page {
  width: 360px;
  padding: 3% 0 0;
  margin: auto;
}
.form {
  position: relative;
  z-index: 1;
  background: #FFFFFF;
  max-width: 360px;
  margin: 0 auto 100px;
  padding: 45px;
  text-align: center;
  box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);
}
.form input {
  font-family: "Roboto", sans-serif;
  outline: 0;
  background: #f2f2f2;
  width: 100%;
  border: 0;
  margin: 0 0 15px;
  padding: 15px;
  box-sizing: border-box;
  font-size: 14px;
}
```

```
.form button {
  font-family: "Roboto", sans-serif;
  text-transform: uppercase;
  outline: 0;
  background-color: #328f8a;
  background-image: linear-gradient(45deg,#328f8a,#08ac4b);
  width: 100%;
  border: 0;
  padding: 15px;
  color: #FFFFFF;
  font-size: 14px;
  -webkit-transition: all 0.3 ease;
  transition: all 0.3 ease;
  cursor: pointer;
}
.form .message {
  margin: 15px 0 0;
  color: #b3b3b3;
  font-size: 12px;
  text-align:justify;
}
.form .message a {
  color: #4CAF50;
  text-decoration: none;
}

.container {
  position: relative;
  z-index: 1;
  max-width: 200px;
  margin: 0 auto;
}
```

```
body {
 background-image: url("https://pr1.nicelocal.in/O8jsnUt-eKkrAcRZCqL2Xw/640x440,fit,q85/4px-BW84_n2aGZJPv2Q_uysuQlRKXWAtkzmuAfqDQI
 background-color: #cccccc;
}
</style>
</head>
<body>
  <body>
    <div class="login-page">
      <div class="form">
        <div class="login">
          <div class="login-header">
            <h3>SIGNUP HERE</h3>
          </div>
        </div>
        <form class="login-form" action="signup.php">
          <input type="text" placeholder="Username" name="Username"/>
          <input type="text" placeholder="Mobile number" name="Mobilenumber"/>
          <input type="text" placeholder="E-mail" name="E_mail"/>
          <input type="password" placeholder="Password" name="Password"/>
          <input type="password" placeholder="Confirm password" name="Confirmpassword"/>
            <div class="signup">
          <button type= "submit" class="btn" name="submit">submit</button>
          </div>
          <p>
          already exists? <a href="loginpage.html">Login</a>
          </p>
          </form>
        </div>
      </div>
</body>
</body>
</html>
```

SIGNUP FORM PHP:

```php
<?php

$link = mysqli_connect("localhost", "root", "", "test");

// Check connection
if($link === false){
    die("ERROR: Could not connect. " . mysqli_connect_error());
}

// Escape user inputs for security
$User_name = mysqli_real_escape_string($link, $_REQUEST['Username']);
$mbno = mysqli_real_escape_string($link, $_REQUEST['Mobilenumber']);
$email = mysqli_real_escape_string($link, $_REQUEST['E_mail']);
$pwd = mysqli_real_escape_string($link, $_REQUEST['Password']);
$conpwd = mysqli_real_escape_string($link, $_REQUEST['Confirmpassword']);


 echo $Username;

// Attempt insert query execution
$sql = "INSERT INTO signupform VALUES ('$User_name','$mbno ','$email ', '$pwd','$conpwd')";
if(mysqli_query($link, $sql)){
    echo "read successfully";
} else{
    echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
}

// Close connection
mysqli_close($link);
?>
```

LOGIN FORM CODE:

```
<!doctype html>
<html>
<head>
<title>Login form </title>
<style type="text/css">
@import url(https://fonts.googleapis.com/css?family=Roboto:300);
header .header{
  background-color: #fff;
  height: 45px;
}
header a img{
  width: 134px;
margin-top: 4px;
}
.login-page {
  width: 360px;
  padding: 8% 0 0;
  margin: auto;
}
.login-page .form .login{
  margin-top: -31px;
margin-bottom: 26px;
}
.form {
  position: relative;
  z-index: 1;
  background: #FFFFFF;
  max-width: 360px;
  margin: 0 auto 100px;
  padding: 45px;
  text-align: center;
  box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);
}
.form input {
  font-family: "Roboto", sans-serif;
  outline: 0;
  background: #f2f2f2;
  width: 100%;
  border: 0;
  margin: 0 0 15px;
  padding: 15px;
  box-sizing: border-box;
  font-size: 14px;
```

```
}
.form .message a {
  color: #4CAF50;
  text-decoration: none;
}

.container {
  position: relative;
  z-index: 1;
  max-width: 200px;
  margin: 0 auto;
}
body {
 background-image: url("https://pr1.nicelocal.in/O8jsnUt-eKkrAcRZCqL2Xw/640x440,fit,q85/4px-BW84_n2aGZJPv2Q_uysuQlRKXWAtkzmuAfqDQITcuO7_FYe_adbVn_ntvR8
 background-color: #cccccc;
}
</style>
</head>
 <body>
    <div class="login-page">
      <div class="form">
        <div class="login">
          <div class="login-header">
            <h3>LOGIN</h3>
            <p>Please enter your credentials to login.</p>
          </div>
        </div>
        <form class="login-form" action="fruits.html">
          <input type="text" placeholder="username" name="Username"/>
          <input type="password" placeholder="password"/ name="Password">
           <div class="signin">
        <button type= "submit" class="btn" name="submit">submit</button>
        </div>
        <p>
        Not a user? <a href="signupform.html">Signup</a>
        </p>
        </form>
      </div>
    </div>
</body>
</html>
```

INDEX PAGE CODE:

```
<html>
<head>
<title>ORGOMART</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iM
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-/bQdsTh/da6pkI1MST/rWKFNjaCP5gBSY4sEBT38Q/9RBh9AH
<style>
body {background-color: #c9ced6;}
</style>
</head>
<body>
<div class="container-fluid">
<div class="row">
    <div class="col-md-12">
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand text-success"  href="#"><h3>ORGANIC VEG MART</h3></a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavDropdown">
      <ul class="navbar-nav">
        <li class="nav-item dropdown">
          <a class="nav-link active dropdown-toggle" href="#" id="navbarDropdownMenuLink" role="button" data-bs-toggle="dropdown" aria-expanded="false">
            Categories
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
            <li><a class="dropdown-item" href="vegetable.html">VEGITABLES</a></li>
            <li><a class="dropdown-item" href="fruits.html">FRUITS</a></li>
          </ul>
        </li>
<li class="nav-item">
          <a class="nav-link active" aria-current="page" href="loginpage.html">Login/Signup</a>
        </li>
<li class="nav-item">
          <a class="nav-link active" href="#">Cart</a>
        </li>
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="loginpage.html">Logout</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

```
</nav>
</div>
</div>
<div class="row">
    <div class="col-12">
      <div id="carouselExampleDark" class="carousel carousel-dark slide" data-bs-ride="carousel">
  <div class="carousel-indicators">
    <button type="button" data-bs-target="#carouselExampleDark" data-bs-slide-to="0" class="active" aria-current="true" aria-label="Slide 1"></button>
    <button type="button" data-bs-target="#carouselExampleDark" data-bs-slide-to="1" aria-label="Slide 2"></button>
    <button type="button" data-bs-target="#carouselExampleDark" data-bs-slide-to="2" aria-label="Slide 3"></button>
  </div>
  <div class="carousel-inner">
   <div class="carousel-item active" data-bs-interval="2000">
     <img src="https://www.kosher.com/resized/article_view/s/h/shutterstock_417089731.jpg" class="d-block w-100" alt="...">
     <div class="carousel-caption d-none d-md-block">
       <h1 class="text-justify text-white bg-dark">WELCOME TO ORGANIC VEG MART</h1>
     </div>
    </div>
    <div class="carousel-item" data-bs-interval="1000">
     <img src="https://images-na.ssl-images-amazon.com/images/G/01/projectmendel/factoria/desktop/Selected-With-Care_Header_DESKTOP.png" class="d-block w
     <div class="carousel-caption d-none d-md-block">
       <h1></h1>
     </div>
    </div>
    <div class="carousel-item">
     <img src="https://www.jaiviksetu.com/images/slider/jaivik-setu-banner-1.jpg" class="d-block w-100" alt="...">
     <div class="carousel-caption d-none d-md-block">
       <h1></h1>
     </div>
    </div>
    <button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleDark" data-bs-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleDark" data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
    </div>
</div>
</div> <!--container fluid ends here-->
<div class="container">
```

```
<div class="row mt-4 mb-4">
<div class="col-12">
<h2 class="text-center text-success" >ABOUT US</h2>
<p class="text-center">
ORGANIC VEG MART strives to create a community with sustainable and meaningful values by providing healthy organic food. We work with
All ORGANIC VEG MART products support health and True Wellness and are made with loving care. Each product is one link in a chain of
Our mart is committed to being a trustworthy and innovative global leader by providing genuine True Wellness products. Our advanced p
</p>
</div>
</div>
</div>
<div class="row mt-3">
<div class="col-12 "><h2 class=" text-success text-center">CONTACT US</h2></div>
<div class='text-center p-3 mt-1'>
<p><h4>Mart Location:<h4></p>
2/4 Brodipet, Guntur, AndhraPradesh-522233
<p><h4>You can also make order using follow details:</h4></p>
<h4>contact:7689342341</h4>
<h4> Drop you doubts and reviews in:</h4>
<h4>Mail: Orgovmart@gmail.com </h4>
</div>
</div>
<div class="container-fluid">
<div class="row">
      <div class="col-12 text-center bg-dark text-white"> <h6 class="pt-2 pb-2"> copy right @2021</h6></div>
</div>
</div>
</div>
</body>
</html>
</body>
</html>
```

CATEGORIES CODE:

```
<!doctype html>
<html>
<head>
<title> Organic Fruits Store </title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-KyZXEAg3QhqLMpG8r+8fhAXLRk2vvoC2f
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-U1DAWAznBHeqEIlVSCgzq+c9gqGAJn5c/t99JyeKa9xx
<style>
body {background-color: #c9ced6;}
</style>
</head>
<body>
<div class="container">
<div class="row mt-4 mb-2">
<div class="col-12">
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand text-success"  href="#">ORGANIC VEG MART</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-exp
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavDropdown">
      <ul class="navbar-nav">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="Index.html">Home</a>
        </li>
        <li class="nav-item dropdown">
          <a class="nav-link active dropdown-toggle" href="#" id="navbarDropdownMenuLink" role="button" data-bs-toggle="dropdown" aria-expanded="false">
            Categories
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
            <li><a class="dropdown-item" href="vegetable.html">VEGITABLES</a></li>
            <li><a class="dropdown-item" href="fruits.html">FRUITS</a></li>
          </ul>
        </li>
<li class="nav-item">
          <a class="nav-link active" href="trail.html">Cart</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
</div>
</div>
```

```
<div class="row">
    <div class="col-12 "><h2 class=" text-success mt-4"> FRUIT WAY</h2></div>
</div>
<div class="row">
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
  <img src="https://5.imimg.com/data5/SL/DA/GLADMIN-9523458/imam-pasand-mango-fruit-500x500.jpg" class="card-img-top" alt="some image">
  <div class="card-body">
    <h5 class="card-title text-center">Mango</h5>
    <p class="shop-item-price">&#x20B9; 60 perkg</p>
    <a href="#" class="btn btn-success">Add to cart</a>
  </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
  <img src="https://c.ndtvimg.com/2019-04/d19g6gn8_bananas_625x300_11_April_19.jpg" class="card-img-top" alt="some image">
  <div class="card-body">
    <h5 class="card-title text-center">Banana</h5>
    <p class="shop-item-price">&#x20B9; 45 per dozen</p>
    <a href="#" class="btn btn-success">add to cart</a>
  </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
  <img src="https://ak.picdn.net/shutterstock/videos/5308121/thumb/1.jpg" class="card-img-top" alt="some image">
  <div class="card-body">
    <h5 class="card-title text-center">Oranges</h5>
    <p class="shop-item-price">&#x20B9; 80 per dozen</p>
    <a href="#" class="btn btn-success">add to cart</a>
  </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
  <img src="https://m.media-amazon.com/images/I/815mO326PQL._SX679_.jpg" class="card-img-top" alt="some image">
  <div class="card-body">
    <h5 class="card-title text-center">Pomegranates</h5>
    <p class="shop-item-price">&#x20B9; 70 per dozen</p>
    <a href="#" class="btn btn-success">add to cart</a>
  </div>
```

```
  </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
   <img src="https://cdn.shopify.com/s/files/1/0271/6634/3237/products/greengrape.png?v=1610491027" class="card-img-top" alt="some image">
   <div class="card-body">
     <h5 class="card-title text-center">Grapes</h5>
     <p class="shop-item-price">&#x20B9; 100 per kg</p>
     <a href="#" class="btn btn-success">add to cart</a>
   </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
   <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQjvyz4BsxKDPUswZUPdDxmNx2bxAKghhx5ug&usqp=CAU" class="card-img-top" alt="some image">
   <div class="card-body">
     <h5 class="card-title text-center">Sapota</h5>
     <p class="shop-item-price">&#x20B9; 60 per dozen</p>
     <a href="#" class="btn btn-success">add to cart</a>
   </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
   <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTzJIvDlJI-zU1HK7eKYc0lFOXhjHU9FRoFUg&usqp=CAU" class="card-img-top" alt="some image">
   <div class="card-body">
     <h5 class="card-title text-center">Guva</h5>
     <p class="shop-item-price">&#x20B9; 60 per dozen</p>
     <a href="#" class="btn btn-success">add to cart</a>
   </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
   <img src="https://media.istockphoto.com/photos/papaya-isolated-on-white-background-picture-id1136327944?k=20&m=1136327944&s=612x612&w=0&h=wZBJ1umEXdHoHtdRtn
   <div class="card-body">
     <h5 class="card-title text-center">Papaya</h5>
     <p class="shop-item-price">&#x20B9; 25 per piece</p>
     <a href="#" class="btn btn-success">add to cart</a>
   </div>
</div>
    </div>
```

```
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
   <img src="https://hips.hearstapps.com/hbu.h-cdn.co/assets/15/31/2048x1024/landscape-1438283137-watermelon.jpg?resize=1200:*" class="card-img-top" alt="some
   <div class="card-body">
     <h5 class="card-title text-center">Watermelon</h5>
     <p class="shop-item-price">&#x20B9; 15 per kg</p>
     <a href="#" class="btn btn-success">add to cart</a>
   </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
   <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQpVGj_XLvIW9Lb8PYQ2WjZjfvPyKax4CTEZMlABOWJ3DCmLrpNEDBHZKFtAmtnFO62noI&usqp=CAU" class="card-
   <div class="card-body">
     <h5 class="card-title text-center">Kiwi</h5>
     <p class="shop-item-price">&#x20B9; 10 per piece</p>
     <a href="#" class="btn btn-success">add to cart</a>
   </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
   <img src="https://cdn.shopify.com/s/files/1/0054/9742/1937/products/white-dragon-fruit-fruit-fruits-n-rootz-350568_620x.jpg?v=1627496125" class="card-img-t
   <div class="card-body">
     <h5 class="card-title text-center">Dragon Fruit</h5>
     <p class="shop-item-price">&#x20B9; 50 per piece</p>
     <a href="#" class="btn btn-success">add to cart</a>
   </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
   <img src="https://static.india.com/wp-content/uploads/2018/02/Strawberries.jpg?impolicy=Medium_Resize&w=1200&h=800" class="card-img-top" alt="some image">
   <div class="card-body">
     <h5 class="card-title text-center">Strawberrie</h5>
     <p class="shop-item-price">&#x20B9; 70 per pack</p>
     <a href="#" class="btn btn-success">add to cart</a>
   </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
```

```
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
  <img src="https://5.imimg.com/data5/AD/ST/MY-27216245/fluxy-froottr-anjeer-figs-500x500.jpg" class="card-img-top" alt="some image">
  <div class="card-body">
    <h5 class="card-title text-center">Fig</h5>
    <p class="shop-item-price">&#x20B9; 75 per kg</p>
    <a href="#" class="btn btn-success">add to cart</a>
  </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
  <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcShMEB5GRyWpPvzQVSJ3EZdAv7HX3VDiVZdmg&usqp=CAU" class="card-img-top" alt="some image">
  <div class="card-body">
    <h5 class="card-title text-center">Pear</h5>
    <p class="shop-item-price">&#x20B9; 30 per piece</p>
    <a href="#" class="btn btn-success">add to cart</a>
  </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
  <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTq49aTYD7TLjGNzGq5_qjgldN5jsPuRCUacGgxaxy_LIENVTVRSfOayOJ2Wzrwheo5oM4&usqp=CAU" class="car
  <div class="card-body">
    <h5 class="card-title text-center">Cantaloupe</h5>
    <p class="shop-item-price">&#x20B9; 40 per kg</p>
    <a href="cart.html" class="btn btn-success">add to cart</a>
  </div>
</div>
    </div>
</div>
</div>
<div class="container-fluid">
<div class="row">
    <div class="col-12 text-center bg-dark text-white"><h6 class="pt-2 pb-2"> Copy right @2021 </h6></div>
</div>
</div>
</body>
</html>
```

```
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
  <img src="https://m.media-amazon.com/images/I/31g7BViB+jL.jpg" class="card-img-top" alt="some image">
  <div class="card-body">
    <h5 class="card-title text-center">Snakegourd</h5>
    <p class="price">Price: 15/- per kg</p>
    <a href="#" class="btn btn-success">Add to cart</a>
  </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
  <img src="https://cdn.xxl.thumbs.canstockphoto.com/ladys-finger-or-okra-okra-or-ladys-finger-or-bhindi-fresh-green-vegetable-arranged-in-a-baske
  <div class="card-body">
    <h5 class="card-title text-center">Ladies finger</h5>
    <p class="price">Price: 15/- perkg</p>
    <a href="#" class="btn btn-success">Add to cart</a>
  </div>
</div>
    </div>
    <div class="col-md-4">
    <div class="card" style="width: 100%;">
  <img src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRIlNyg0jWeJqnpxCto9pRs3gOS0MpRbtJ6Zg&usqp=CAU" class="card-img-top" alt="some im
  <div class="card-body">
    <h5 class="card-title text-center">Bottlegourd</h5>
    <p class="price">Price:20/- per kg</p>
    <a href="#" class="btn btn-success">Add to cart</a>
  </div>
</div>
    </div>


    <div class="col-md-4">
    <div class="card" style="width: 100%;">
  <img src="https://cdn.shopify.com/s/files/1/1380/2059/files/Intro-Bitter_Gourd_480x480.jpg?v=1622540276" class="card-img-top" alt="some image">
  <div class="card-body">
    <h5 class="card-title text-center">Bittergourd</h5>
    <p class="price">Price:35/- per kg</p>
    <a href="#" class="btn btn-success">Add to cart</a>
  </div>
```

CART CODE:

```
<!doctype html>
<html>
<head>
<title> Organic Vegetables Store </title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-KyZXEAg3QhqLMpG8r+8fhAXLRk2vvoC2f3B(
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-U1DAWAznBHeqEIlVSCgzq+c9gqGAJn5c/t99JyeKa9xxaY
<style>
table.center {
    margin-left:auto;
    margin-right:auto;
  }
body {background-color: #c9ced6;}
</style>
</head>
<body>
<div class="container-fluid">
<div class="row mt-4 mb-2">
<div class="col-12">
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand text-success"  href="#">ORGANIC VEG MART</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanc
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavDropdown">
      <ul class="navbar-nav">
        <li class="nav-item dropdown">
          <a class="nav-link active dropdown-toggle" href="#" id="navbarDropdownMenuLink" role="button" data-bs-toggle="dropdown" aria-expanded="false">
            Categories
          </a>
          <ul class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
            <li><a class="dropdown-item" href="vegetable.html">VEGITABLES</a></li>
            <li><a class="dropdown-item" href="fruits.html">FRUITS</a></li>
          </ul>
        </li>
<li class="nav-item">
          <a class="nav-link active" href="#">Cart</a>
        </li>
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="Index.html">Logout</a>
        </li>
      </ul>
    </div>
```

```
    </div>
  </div>
</nav>
</div>
</div>
<div class="row">
    <div class="col-12 "><h2 class=" text-success mt-4 text-center">CART</h2></div>
</div>
<div class="row mt-2 mb-2">
<div class="col-7">
<div class="table-responsive">
  <table class="table">
  <table class="center">
    <table class="table caption-top">
    <thead>
    <tr>
      <th scope="col"><h5 class="text-success text-center">ITEM'S</h5></th>
      <th scope="col"><h5 class="text-success text-center">NAME</h5></th>
      <th scope="col"><h5 class="text-success text-center">QTY</h5></th>
      <th scope="col"><h5 class="text-success text-center">PRICE</h5></th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row"><img src="https://ak.picdn.net/shutterstock/videos/5308121/thumb/1.jpg" class="card-img-top" alt="some image" width="10" height="250"></th>
      <td><h6 class="text-center">Oranges</h6></td>
      <td><select align="middle">
      <option>1</option>
      <option>2</option>
      <option>3</option>
      </select></td>
      <td><h5 class="text-center">&#x20B9; 80</h5></td>

    </tr>
    <tr>
      <th scope="row"><img src="https://inputi.market/wp-content/uploads/2021/05/beet-root-boro-f1.jpg" class="card-img-top" alt="some image" width="10" height="
          <td><h6 class="text-center">Beetroot</h6></td>
      <td><select align="center">
      <option>1</option>
      <option>2</option>
      <option>3</option>
      </select></td>
      <td><h5 class="text-center">&#x20B9; 35</h5></td>
```

t Markup Language file | length : 4,200 | lines : 108 | Ln : 36 | Col : 22 | Pos : 1,777 | Windows (CR LF)

```
      <th scope="row"><img src="https://ak.picdn.net/shutterstock/videos/5308121/thumb/1.jpg" class="card-img-top" alt="some image" width="10" height="250"></t
      <td><h6 class="text-center">Oranges</h6></td>
      <td><select align="middle">
      <option>1</option>
      <option>2</option>
      <option>3</option>
      </select></td>
      <td><h5 class="text-center">&#x20B9; 80</h5></td>

    </tr>
    <tr>
      <th scope="row"><img src="https://inputi.market/wp-content/uploads/2021/05/beet-root-boro-f1.jpg" class="card-img-top" alt="some image" width="10" height
          <td><h6 class="text-center">Beetroot</h6></td>
      <td><select align="center">
      <option>1</option>
      <option>2</option>
      <option>3</option>
      </select></td>
      <td><h5 class="text-center">&#x20B9; 35</h5></td>
    </tr>
    <tr>
      <th scope="row"><img src="https://www.askmepasal.com/wp-content/uploads/2021/09/Types-of-Cucumbers-FB.jpg" class="card-img-top" alt="some image"width="10
      <td><h6 class="text-center">Cucumber</h6></td>
      <td><select align="center">
      <option>1</option>
      <option>2</option>
      <option>3</option>
      </select></td>
      <td><h5 class="text-center">&#x20B9; 20</h5></td>
    </tr>
  </tbody>
</table>
  </table>
  </div>
  </div>
</div>
<div class="d-grid gap-2 col-2 mx-auto">
  <a href="trail2.html" class="text-white"><button class="btn btn-success" type="button">CHECKOUT</button>
</div>
</div>
</body>
</html>
```

CHECKOUT CODE:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<style>
body {
  font-family: Arial;
  font-size: 17px;
  padding: 8px;
  background-color: #c9ced6;

}
* {
  box-sizing: border-box;
}
.row {
  display: -ms-flexbox; /* IE10 */
  display: flex;
  -ms-flex-wrap: wrap; /* IE10 */
  flex-wrap: wrap;
  margin: 0 -16px;
}
.col-25 {
  -ms-flex: 25%; /* IE10 */
  flex: 25%;
}
.col-50 {
  -ms-flex: 50%; /* IE10 */
  flex: 50%;
}
.col-75 {
  -ms-flex: 75%; /* IE10 */
  flex: 75%;
}
.col-25,
.col-50,
.col-75 {
  padding: 0 16px;
}
.container {
  background-color: #f2f2f2;
  padding: 5px 20px 15px 20px;
```

```
  border-radius: 3px;
}
label {
  margin-bottom: 10px;
  display: block;
}
.icon-container {
  margin-bottom: 20px;
  padding: 7px 0;
  font-size: 24px;
}
.btn {
  background-color: #04AA6D;
  color: white;
  padding: 12px;
  margin: 10px 0;
  border: none;
  width: 100%;
  border-radius: 3px;
  cursor: pointer;
  font-size: 17px;
}
.btn:hover {
  background-color: #45a049;
}
a {
  color: #2196F3;
}
hr {
  border: 1px solid lightgrey;
}
span.price {
  float: right;
  color: grey;
}
/* Responsive layout - when the screen is less than 800px wide, make the two columns stack on top of each other instead of next to each other (also change the d
@media (max-width: 800px) {
  .row {
    flex-direction: column-reverse;
  }
  .col-25 {
    margin-bottom: 20px;
  }
```

```
  }
}
#txt{
color:blue;
}
</style>
</head>
<body>
<div class="row">
  <div class="col-75">
    <div class="container">
      <form action="Index.html">
    <div class="row">
    <div class="col-12"><h2>PAYMENT DETAILS</h2></div>
</div>
        <div class="row">
          <div class="col-50">
            <h3>Billing Address</h3>
            <label for="fname"><i class="fa fa-user"></i> Full Name</label>
            <input type="text" id="fname" name="firstname" placeholder="John M. Doe">
            <label for="email"><i class="fa fa-envelope"></i> Email</label>
            <input type="text" id="email" name="email" placeholder="john@example.com">
            <label for="adr"><i class="fa fa-address-card-o"></i> Address</label>
            <input type="text" id="adr" name="address" placeholder="542 W. 15th Street">
            <label for="city"><i class="fa fa-institution"></i> City</label>
            <input type="text" id="city" name="city" placeholder="New York">
            <div class="row">
              <div class="col-50">
                <label for="state">State</label>
                <input type="text" id="state" name="state" placeholder="NY">
              </div>
              <div class="col-50">
                <label for="zip">Zip</label>
                <input type="text" id="zip" name="zip" placeholder="10001">
              </div>
            </div>
          </div>
          <div class="col-50">
            <h3>Payment</h3>
            <label for="fname">Accepted Cards</label>
            <div class="icon-container">
              <i class="fa fa-cc-visa" style="color:navy;"></i>
              <i class="fa fa-cc-amex" style="color:blue;"></i>
```

```
            <label for="ccnum">Credit card number</label>
            <input type="text" id="ccnum" name="cardnumber" placeholder="1111-2222-3333-4444">
            <label for="expmonth">Exp Month</label>
            <input type="text" id="expmonth" name="expmonth" placeholder="September">
            <div class="row">
              <div class="col-50">
                <label for="expyear">Exp Year</label>
                <input type="text" id="expyear" name="expyear" placeholder="2018">
              </div>
              <div class="col-50">
                <label for="cvv">CVV</label>
                <input type="text" id="cvv" name="cvv" placeholder="352">
              </div>
            </div>
          </div>
        </div>
        <label>
          <input type="checkbox" checked="checked" name="sameadr"> Shipping address same as billing
        </label>
        <input type="submit" value="Continue to checkout" class="btn" onclick="test()">
        <script type="text/javascript">
function test(){
alert ("Payment successfull");
}
</script>
      </form>
    </div>
  </div>
  <div class="col-25">
    <div class="container">
      <h4>Cart <span class="price" style="color:black"><i class="fa fa-shopping-cart"></i> <b>3</b></span></h4>
      <p><a href="#">Oranges</a> <span class="price">&#x20B9; 80</span></p>
      <p><a href="#">Beetroot</a> <span class="price">&#x20B9; 35</span></p>
      <p><a href="#">Cucumber</a> <span class="price">&#x20B9; 20</span></p>
      <hr>
      <p>Total <span class="price" style="color:black"><b>&#x20B9;135</b></span></p>
    </div>
  </div>
</div>
</body>
</html>
```

Database tables created in application

Database table structure for Signup/Registration:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Username	text	latin1_swedish_ci		No	None			Change Drop More
2	Mobilenumber	bigint(10)			No	None			Change Drop More
3	E_mail	varchar(25)	latin1_swedish_ci		No	None			Change Drop More
4	Password	varchar(10)	latin1_swedish_ci		No	None			Change Drop More
5	Confirmpassword	varchar(10)	latin1_swedish_ci		No	None			Change Drop More

After storing the data the database of signup looks as following picture

Username	Mobilenumber	E_mail	Password	Confirmpassword
Harshitha	7890656789	har@gmail.com	12345	12345
krishna	8790667890	k@gmail.com	67890	67890
qwd	3456	a@gmail.com	er	er
qq	34454565	g@gmail.com	345	345
lav	456788956	l@gmail.com	lav	lav

The structure of login is as follows:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Username	text	latin1_swedish_ci		No	*None*			Change Drop More
2	Password	varchar(10)	latin1_swedish_ci		No	*None*			Change Drop More

The stored data of user in login table is displyed below:

Username	Password
Harshitha	12345
Harshitha	12345
lav	lav

The structure of fruits list table is as follows:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Item	text	latin1_swedish_ci		No	*None*			Change Drop More
2	Price	int(11)			No	*None*			Change Drop More
3	Qty	int(11)			No	*None*			Change Drop More

The fuits list table after updating the data init is as follows:

Item	Price	Qty
Mango	60	1
Banana	45	1
Oranges	80	1
Pomegranates	70	1
Grapes	100	1
Sapota	60	1
Guva	60	1
Papaya	25	1
Watermelon	15	1
Kiwi	10	1
Dragonfruit	50	1
Strawberrie	70	1
Avacado	35	1
Apricot	45	1
Jackfruit	50	1
Pineapple	35	1
Apple	90	1
Plum	60	1
Cherries	65	1
Coconut	35	1
Custardapple	60	1

The structure of vegetable list table is as follows:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Item	text	latin1_swedish_ci		No	*None*			Change Drop More
2	Price	int(11)			No	*None*			Change Drop More
3	Qty	int(11)			No	*None*			Change Drop More

The vegetable list table after updating the data init is as follows:

Item	Price	Qty
Tomato	40	1
Potato	30	1
Carrot	35	1
Brinjal	25	1
Cucumber	20	1
Beetroot	35	1
Capcicum	25	1
Snakegourd	15	1
Bottlegourd	20	1
Beans	30	1
Drumstick	3	1
Bittergourd	35	1
Ridgegourd	40	1
Cabage	30	1
Greenchilli	15	1
Mushroom	50	1
Ladiesfinger	15	1
Gherkins	25	1
Onion	60	1
Garlic	45	1
Redchilli	30	1

TESTING

The main purpose of testing ORGO-V-MART is to ensure that all the activities and functionalities of this software run smoothly with no errors and it remains protected.

FOR ADMIN :

- Verify admin login with valid and invalid data
- Verify admin view profile
- Verify admin update profile with valid and invalid data
- Verify admin change password with valid and invalid data

FOR USER :

- Verify user registration with valid and invalid data.
- Verify user login with valid and invalid data
- Verify user view profile.
- Verify user update profile with valid and invalid data.

Test case no	Functionality to be checked	Actual input	Actual output	Expected output	Status
1	Verify admin login/verify user login	Given valid username & valid password	Login success	Login success	Pass
		Given valid username& invalid password	Deny login	Deny login	Pass
		Given invalid username& valid password	Deny login	Deny login	Pass
		Given invalid Username& invalid password	Deny login	Deny login	Pass
2	Verify admin update profile/verify user update profile	Given new email id, new mobile no	updated	updated	Pass

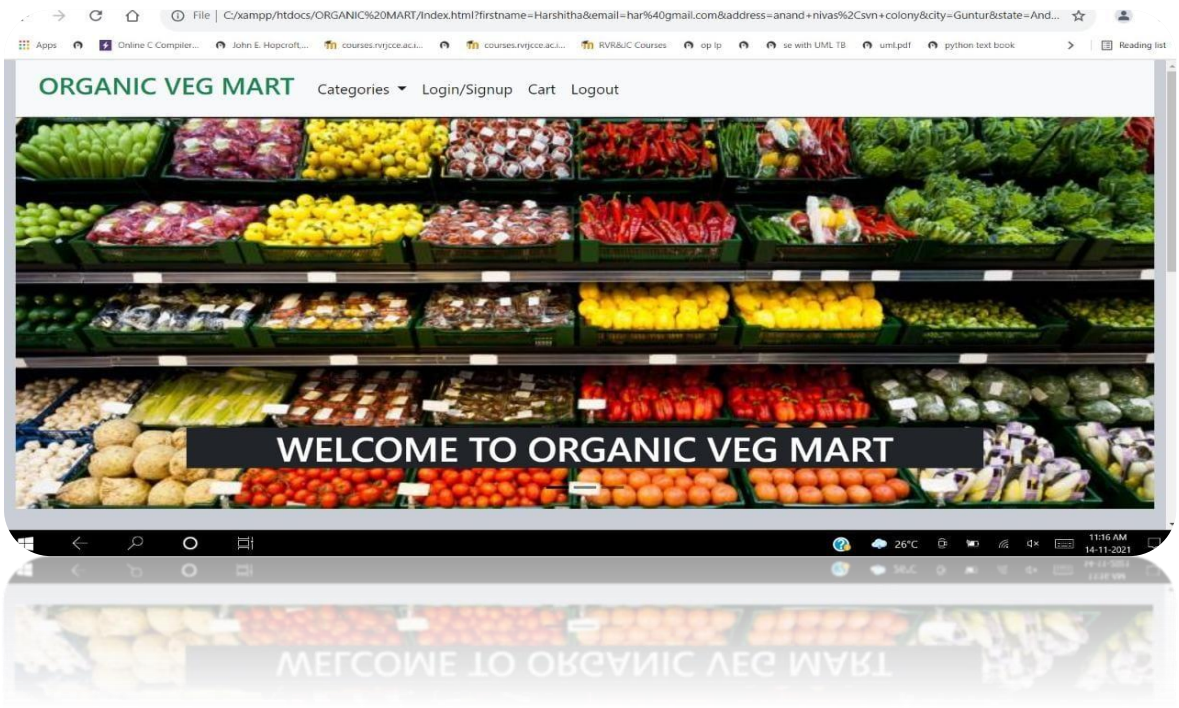
		Given only one field	updated	updated	Pass
		Given only two fields	updated	updated	Pass

IMPLEMENTATION SCREEN SHOTS

The following are runtime GUI developed in the application along with functionality. As we made a responsive website for selling online vegetable mart we can use this application in any type of devices such as Mobile, Tab and Desktop.

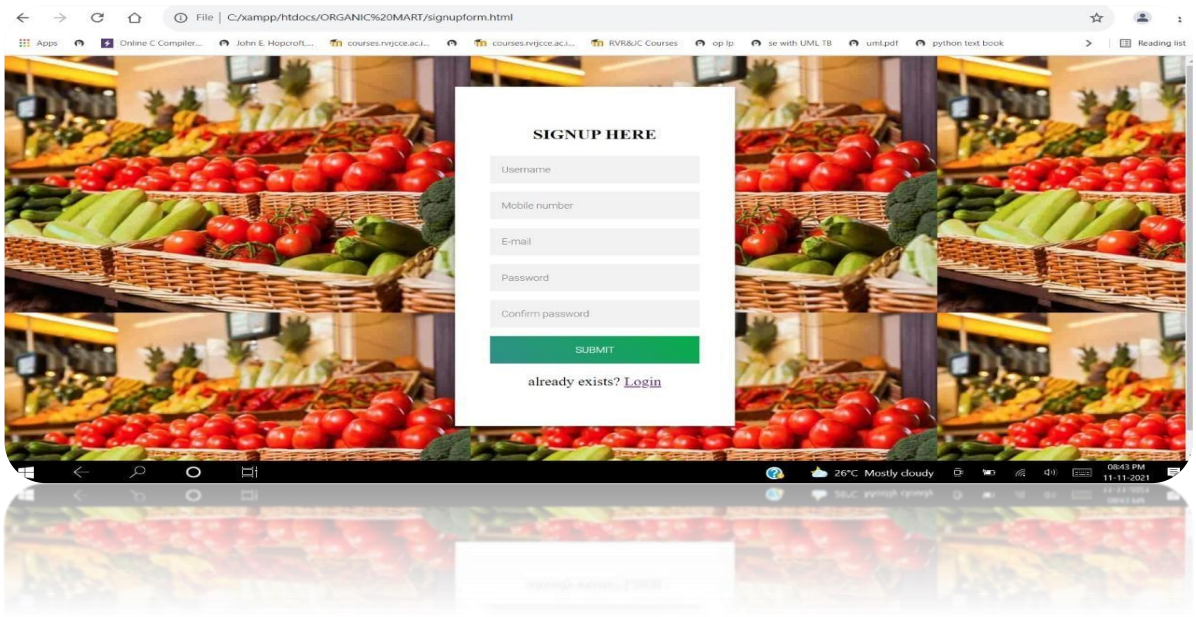
INDEX PAGE :

Desktop view:

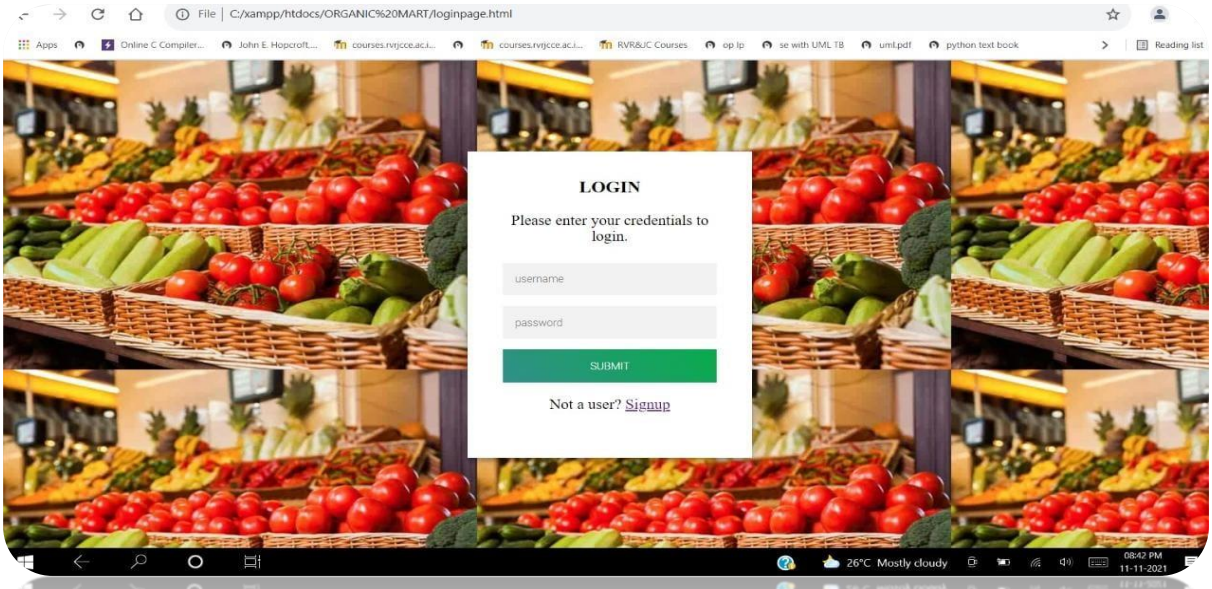


Index Mobile view:

USER REGISTRATION :



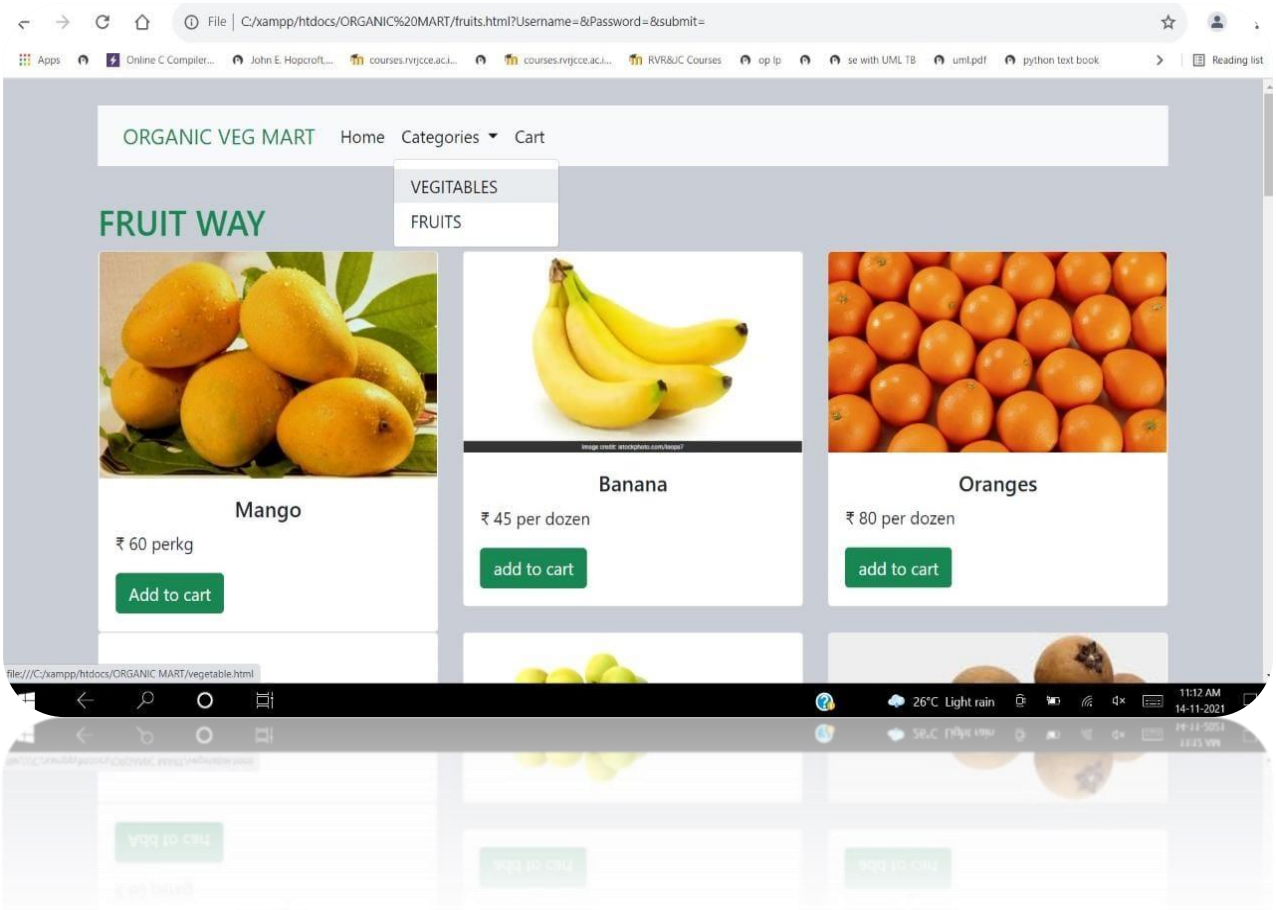
USER LOGIN :



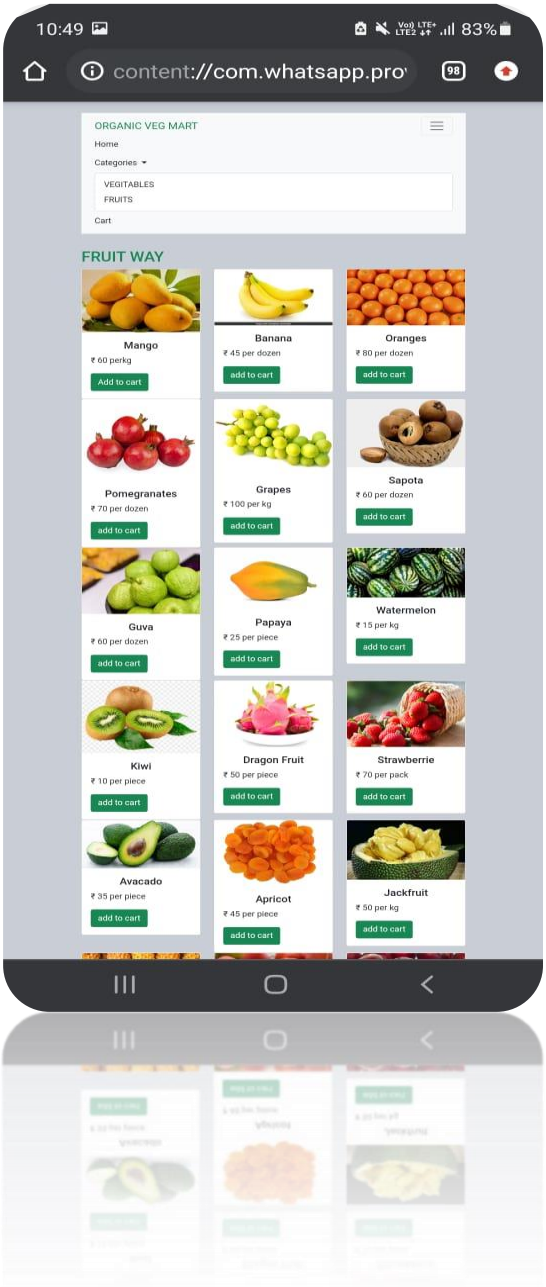
USER VIEW CATEGORIE PAGE :

FRUITS PAGE:

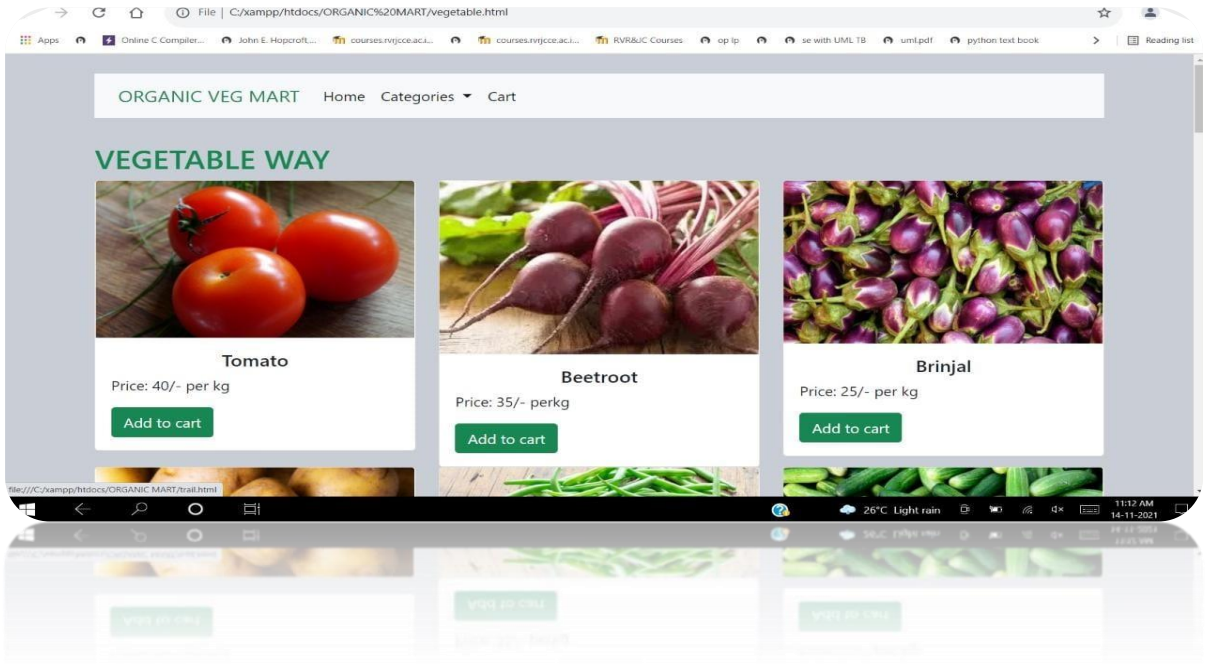
Laptop view:



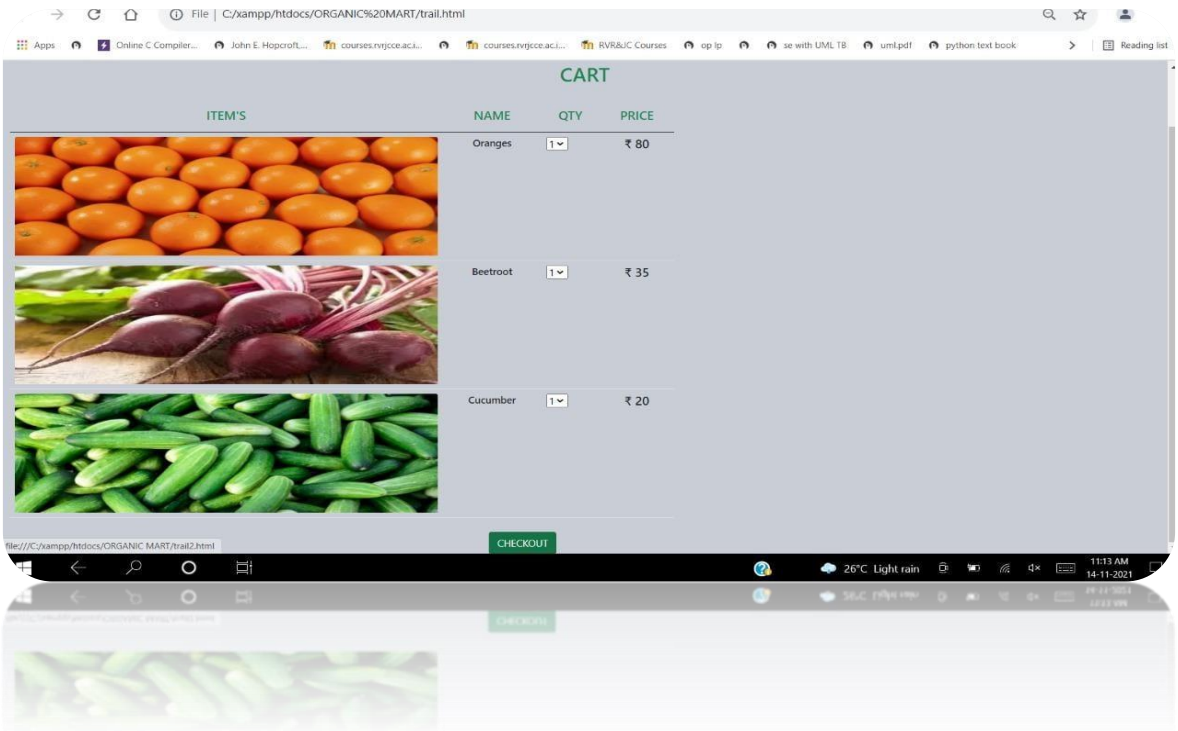
Mobile view:


10:49
content://com.whatsapp.pro
ORGANIC VEG MART
Home
Categories
VEGITABLES
FRUITS
Cart
FRUIT WAY
Mango
₹ 60 perkg
Add to cart
Banana
₹ 45 per dozen
add to cart
Oranges
₹ 80 per dozen
add to cart
Pomegranates
₹ 70 per dozen
add to cart
Grapes
₹ 100 per kg
add to cart
Sapota
₹ 60 per dozen
add to cart
Guva
₹ 60 per dozen
add to cart
Papaya
₹ 25 per piece
add to cart
Watermelon
₹ 15 per kg
add to cart
Kiwi
₹ 10 per piece
add to cart
Dragon Fruit
₹ 50 per piece
add to cart
Strawberrie
₹ 70 per pack
add to cart
Avacado
₹ 35 per piece
add to cart
Apricot
₹ 45 per piece
add to cart
Jackfruit
₹ 50 per kg
add to cart

VEGETABLES PAGE:

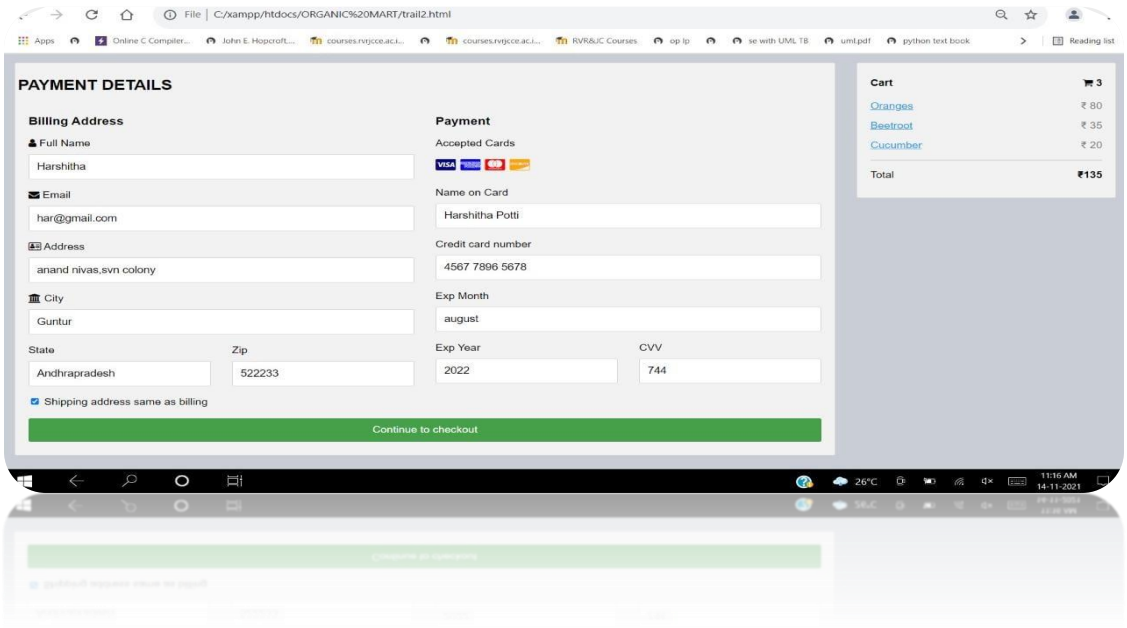


USER VIEW CART PAGE:

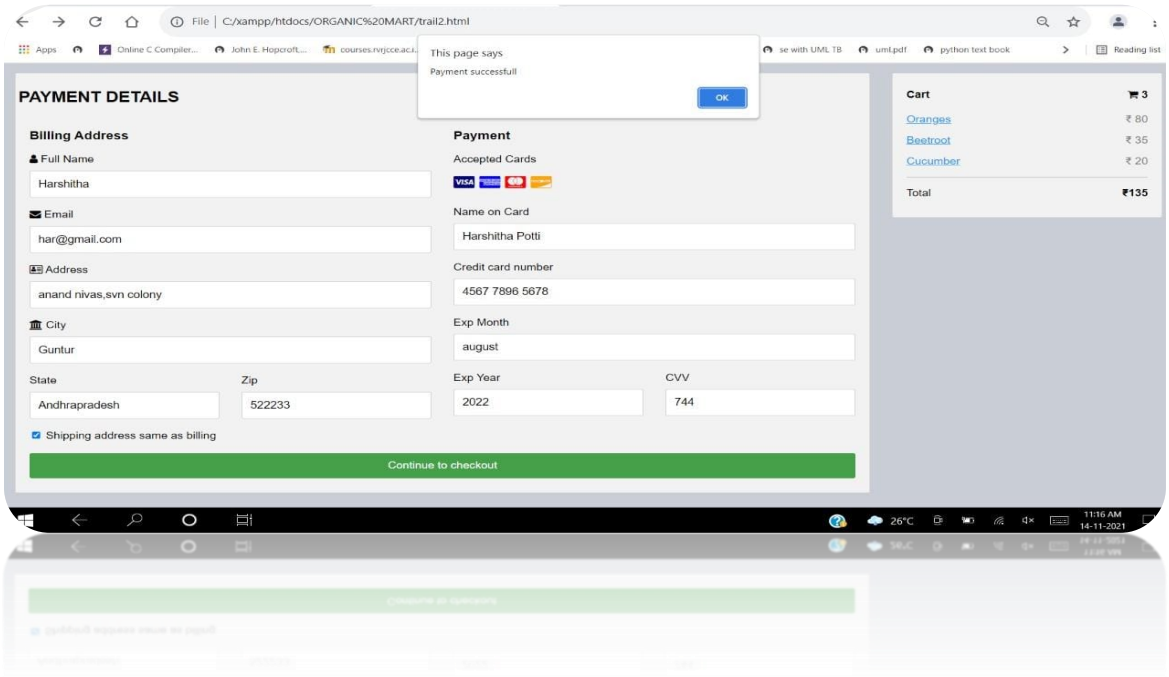


USER VIEW CHECK OUT PAGE:

Before Payment:



After Payment:



After payment it redirects to index page:

At last after logout it redirects to login page.....

CONCLUSION:

Organic veg mart is a project which aims to provide fresh and organic vegitables and fruits for it's users or customers. Making this store available in online helps customers to buy their items straight away from their place. This method decreases outside exposure of customers and make them to feel safe.

Here in this project the key roles are played by Admin and User because admin have to maintain all the data regarding stock available in mart and should update items in websites well. Along that admin should manage the data of the users. Here the user role is to view items, add those items to cart and ordering them through payment.