Project

CS445

E-Security Camera Store

Work by:

Rima Sukhadia

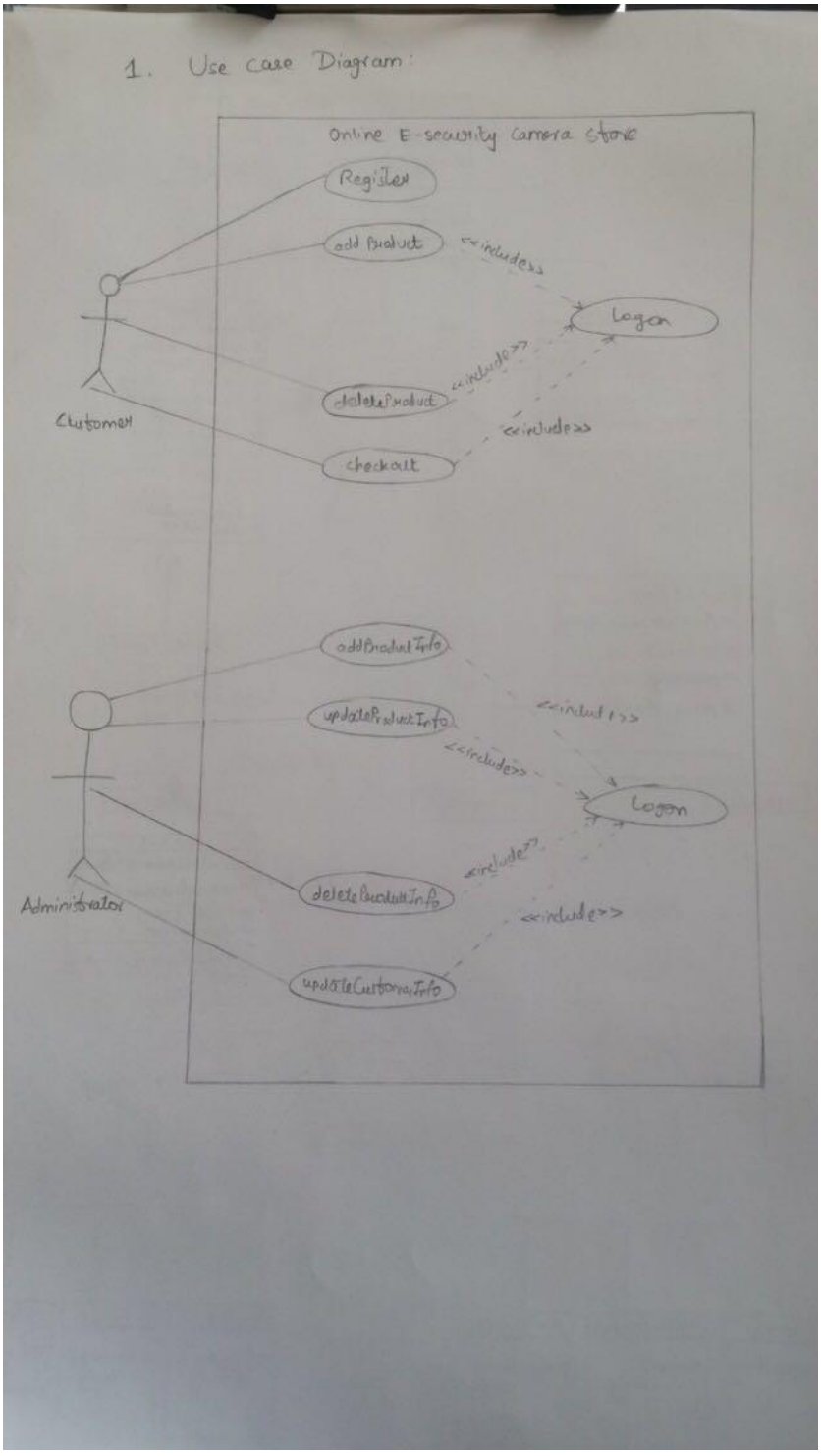
Individual: A20358993

Contact: +1 708 980 4529

Phase-I

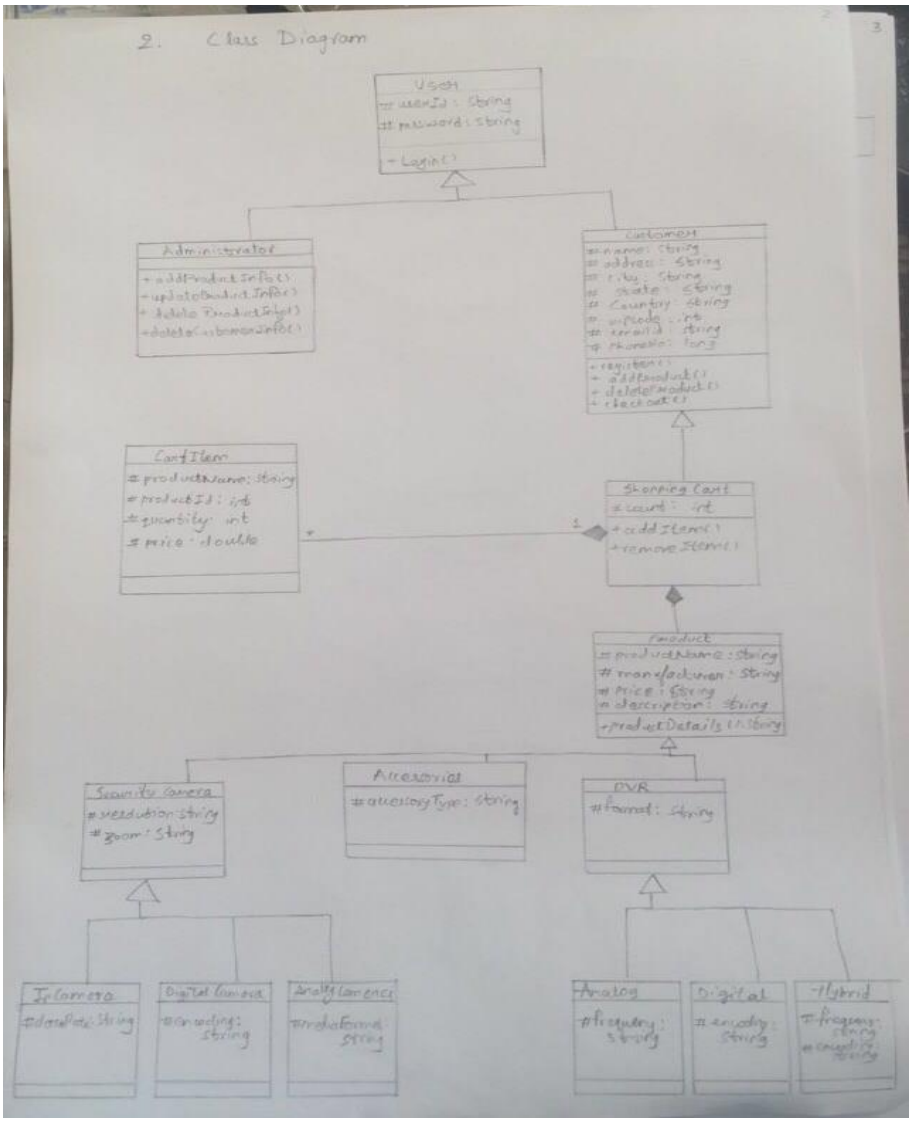
Analysis and Design

1. Use case Diagram:



A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. The following diagram is the use case diagram of our project. There are two actors Customer and Administrator for the security store

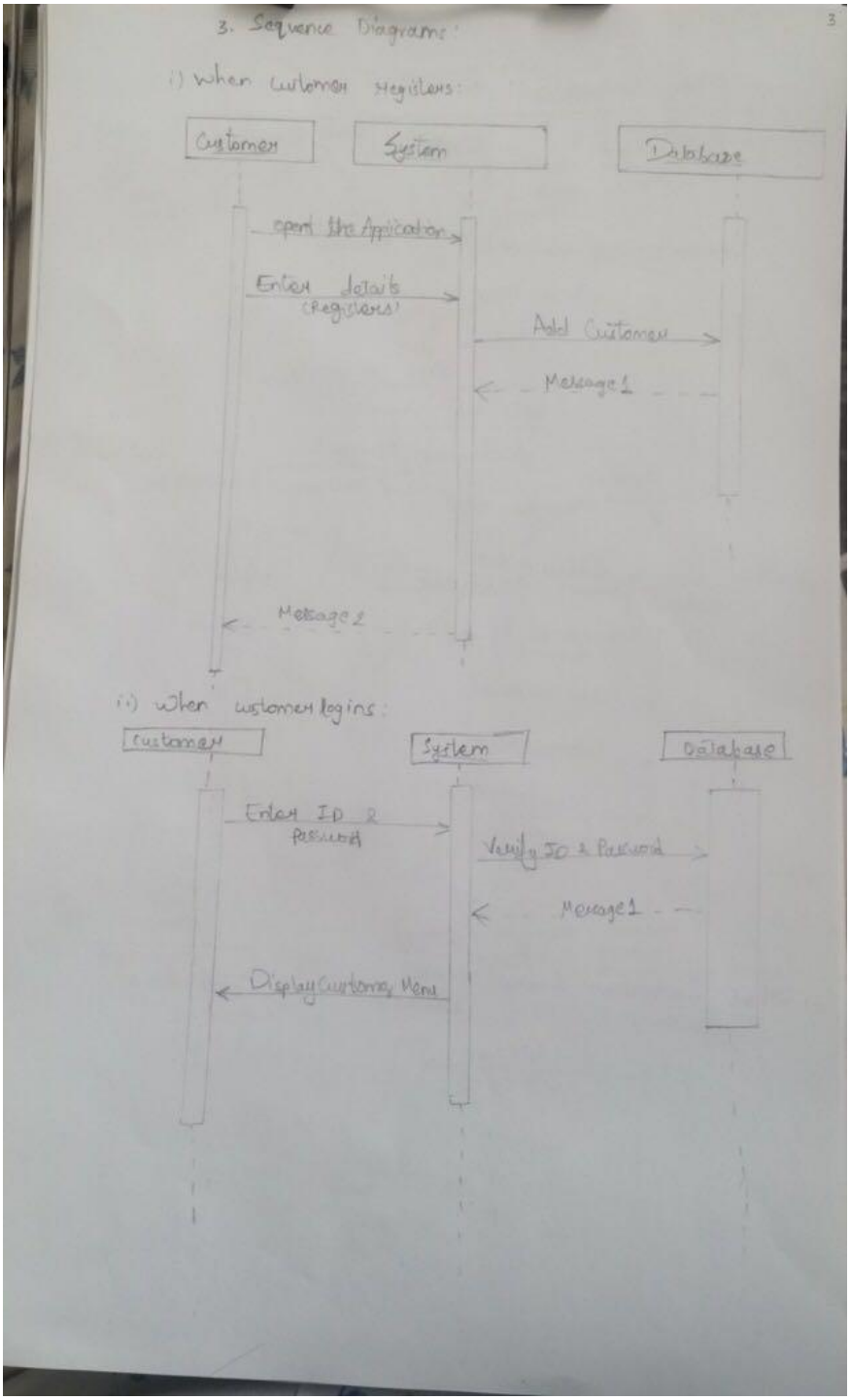
2. Class Diagram:



Class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

3. The following is a class diagram fro the surveillance camera store project and contains all the methods and classes.

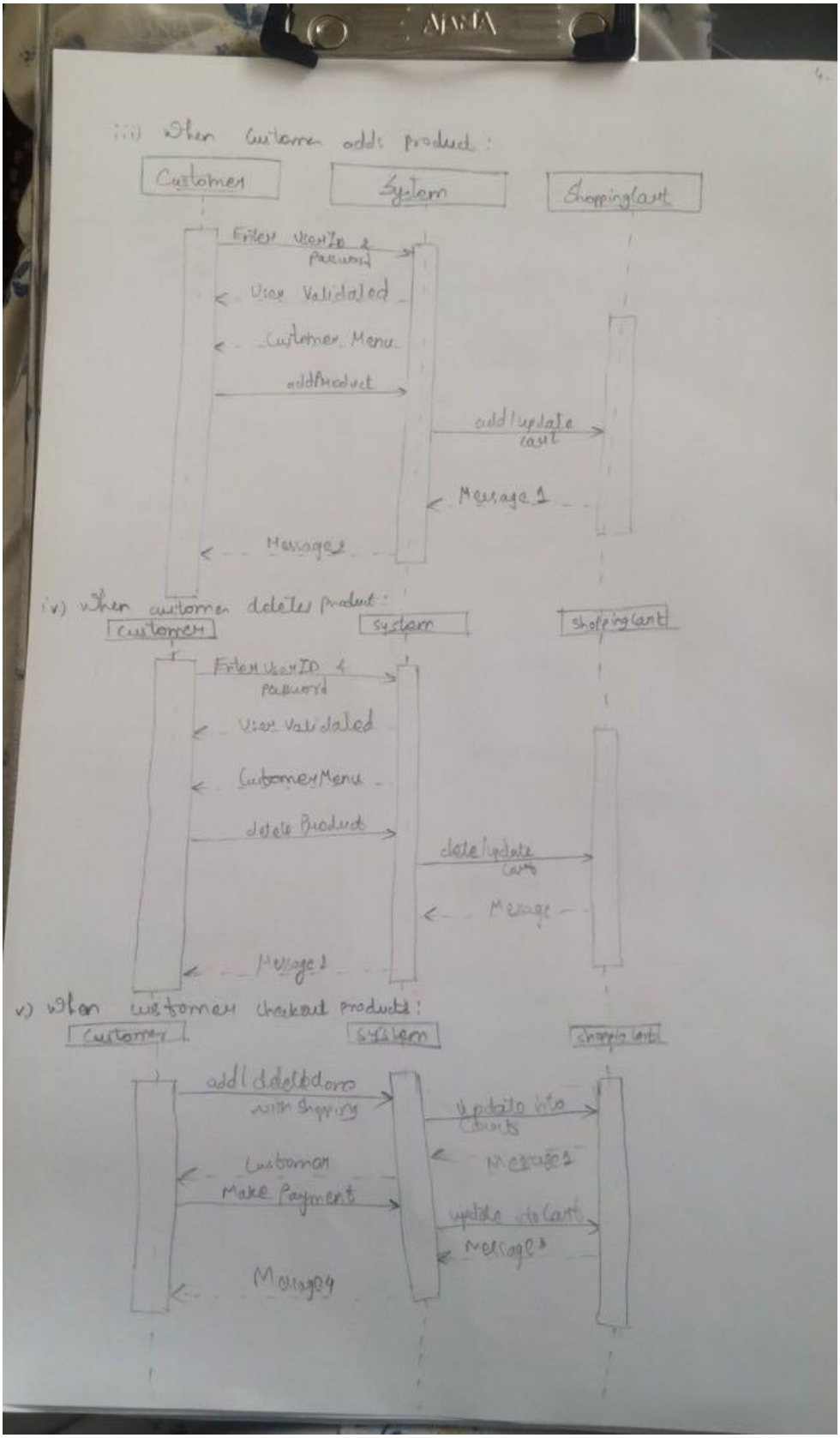
4. Sequence Diagrams:

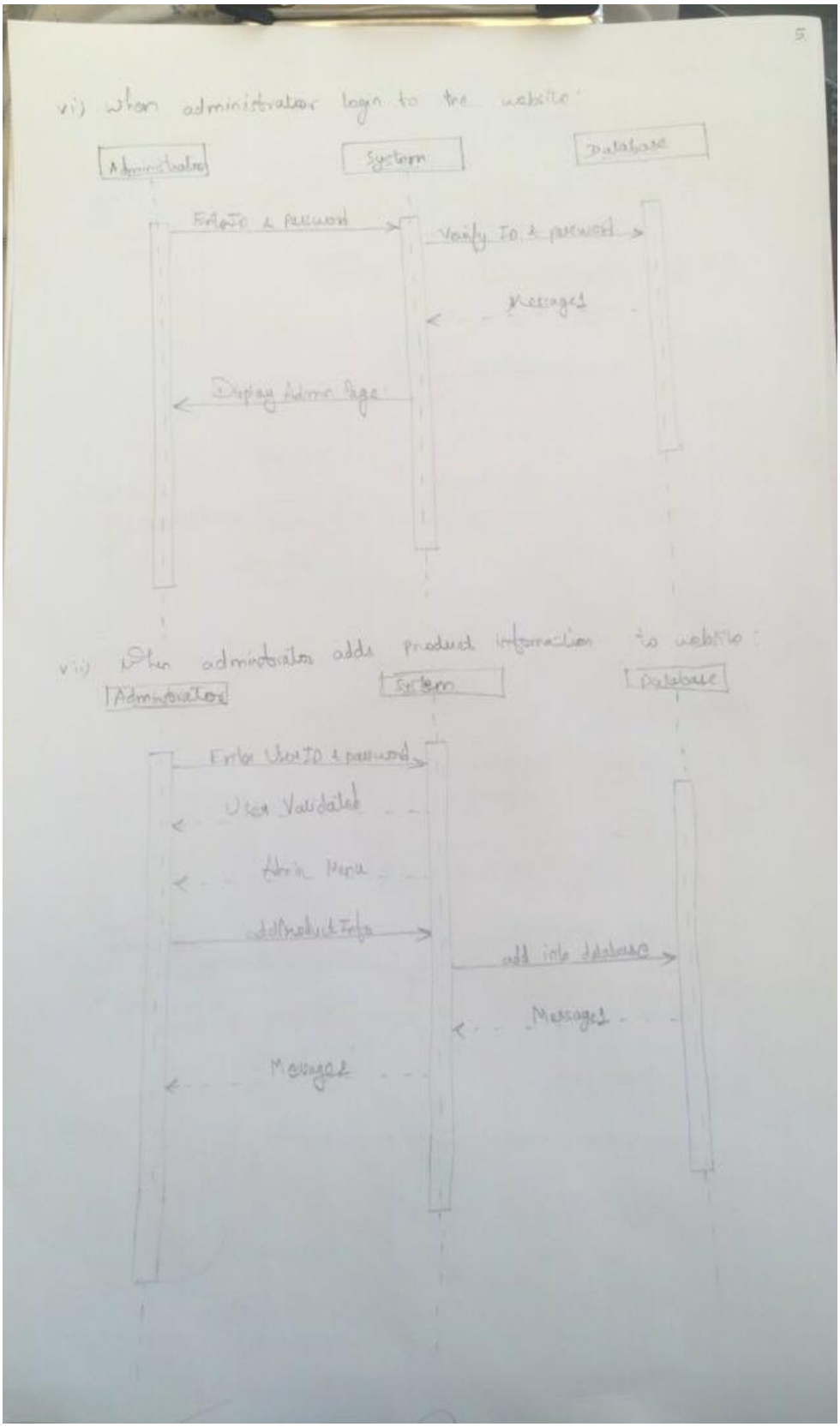


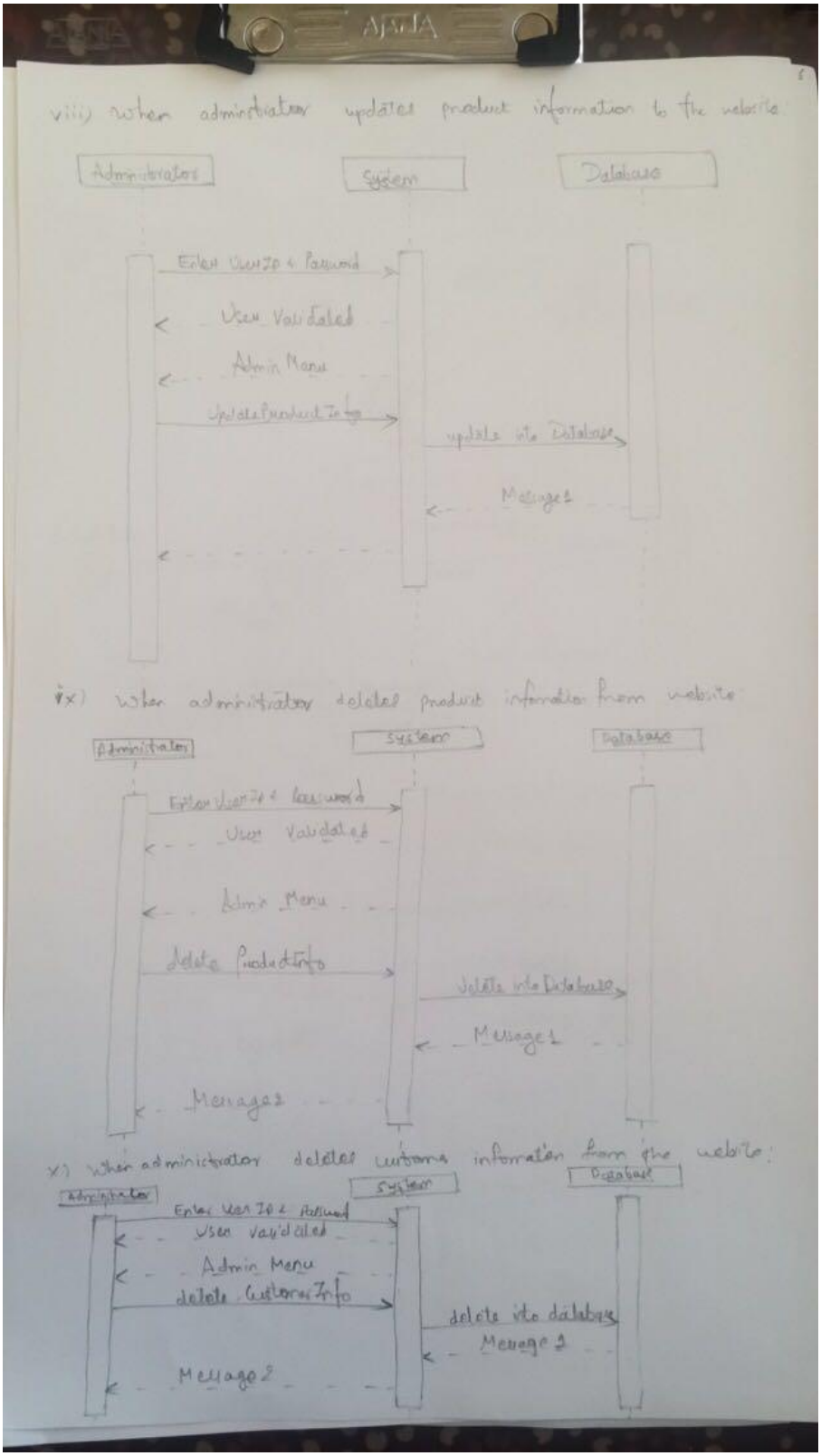
A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order.

The following are the sequence diagrams related to our surveillance DVR and security camera store.

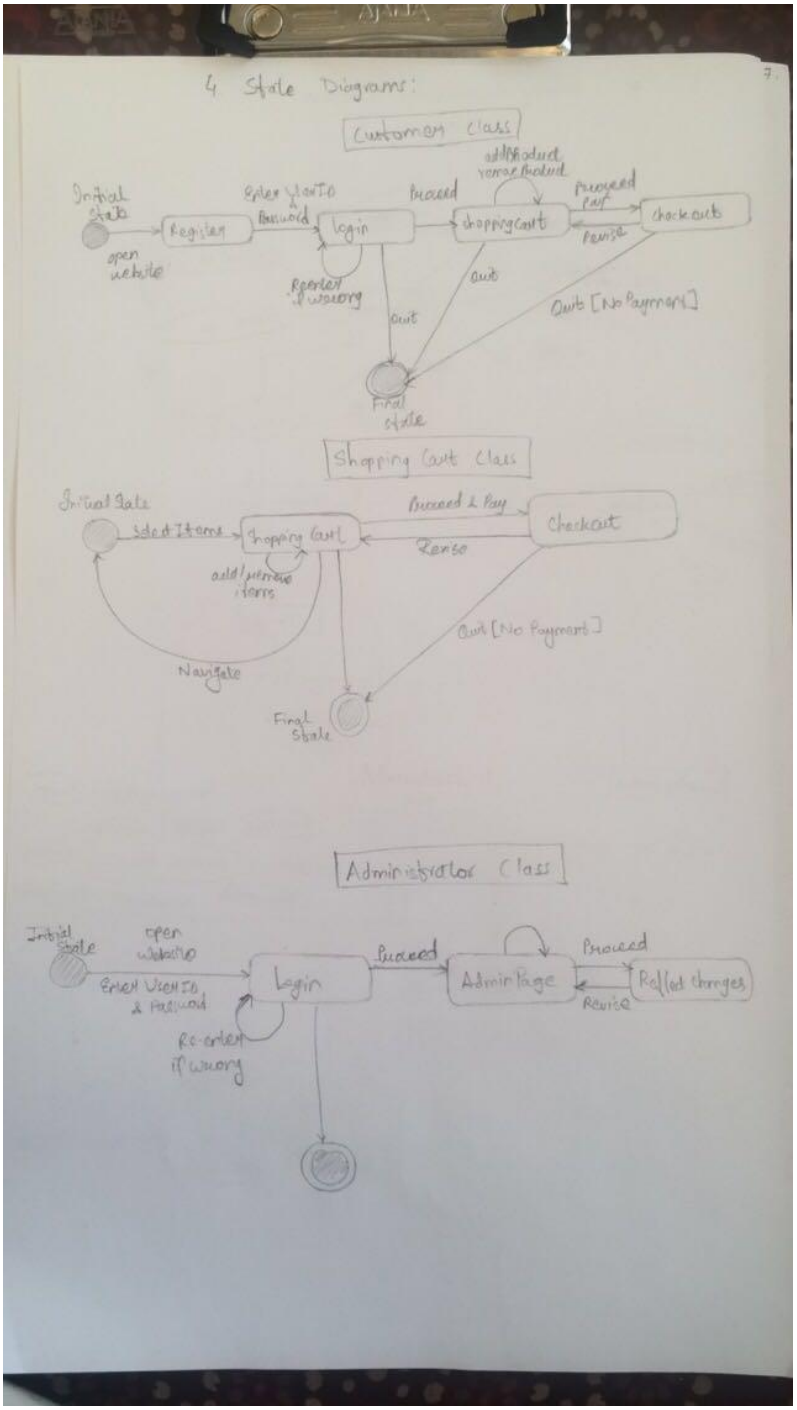
This is the sequence diagram for use case when customer registers to the site.


iii) When Customer adds product :
Customer
System
ShoppingCart
Enter UserID & Password
User Validated
Customer Menu
addProduct
add/update cart
Message 1
Message 2
iv) When customer deletes product :
Customer
System
ShoppingCart
Enter UserID & Password
User Validated
Customer Menu
delete Product
delete/update cart
Message
Message 3
v) When customer checkout products :
Customer
System
ShoppingCart
add/delete/done with Shopping
Update info Clients
Message 2
Customer Make Payment
update to Cart
Message 3
Message 4


5
vi) when administrator login to the website:
Administrator
System
Database
Enter ID & password
Verify ID & password
Message
Display Admin Page
vii) When administrator adds product information to website:
Administrator
System
Database
Enter UserID & password
User Validated
Admin Menu
Add Product Info
add into database
Message
Message

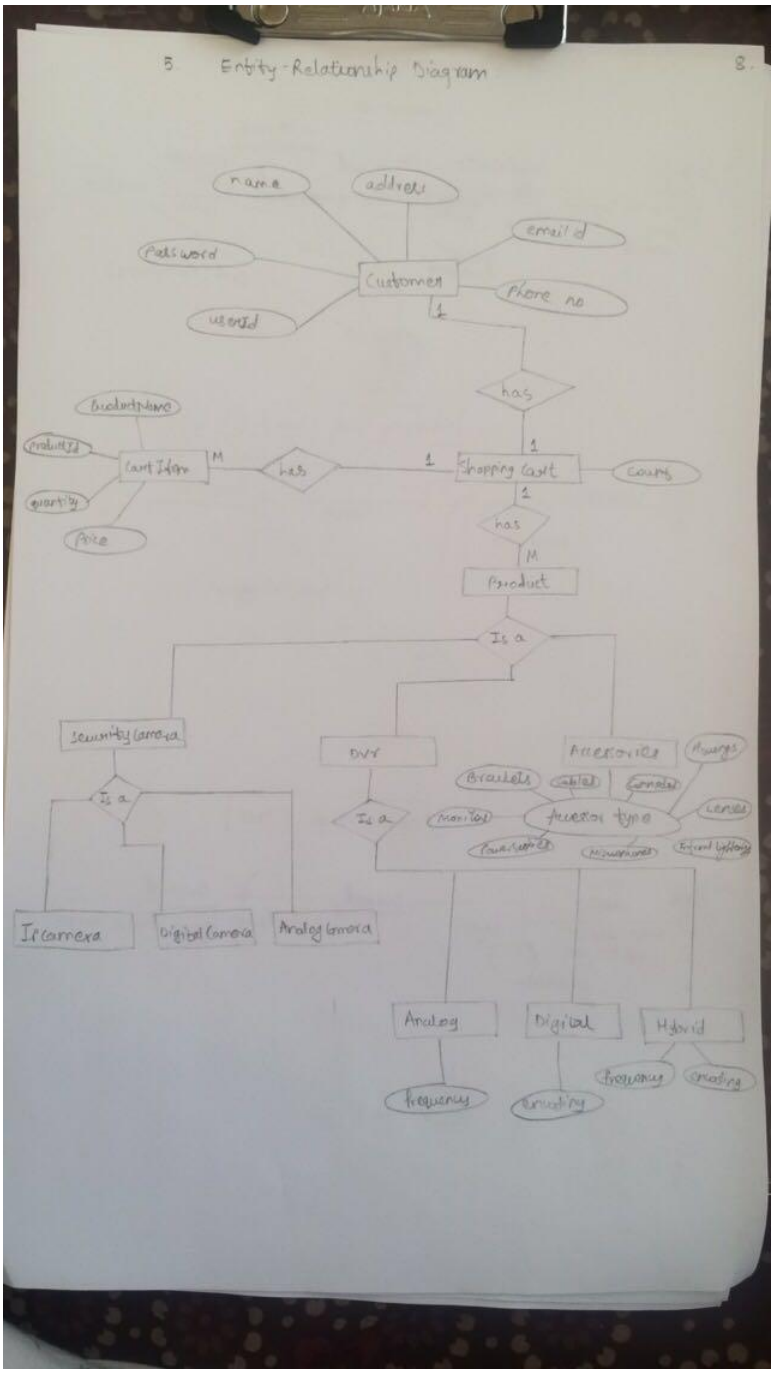

viii) When administrator updates product information to the website:
Administrator
System
Database
Enter User Id & Password
User Validated
Admin Menu
Update Product Info
update into Database
Message
ix) When administrator deletes product information from website:
Administrator
System
Database
Enter User Id & password
User Validated
Admin Menu
delete Product info
delete into Database
Message
Message
x) When administrator deletes customs information from the website:
Administrator
System
Database
Enter User Id & Password
User Validated
Admin Menu
delete Customs Info
delete into database
Message
Message

5. State Diagrams:



A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. The following are the state diagram for customer, administrator and shopping cart.

6. Entity-Relationship Diagram:



An Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. The following is an entity relationship diagram related to our project. ER diagram gives the representation of data related to our project.

Phase-II:

Implementation and Deployment

Implementation Models

Unit tests:

A *unit test* is a piece of code written by a developer that executes a specific functionality in the code to be tested and asserts a certain behavior or state.

The percentage of code which is tested by unit tests is typically called *test coverage*.

A unit test targets a small unit of code, e.g., a method or a class. External dependencies should be removed from unit tests, e.g., by replacing the dependency with a test implementation or a (mock) object created by a test framework.

Unit tests are not suitable for testing complex user interface or component interaction. For this, you should develop integration tests.

Verification and Validation:

Test Plans

As part of the test verification process it is necessary to write test cases for all functions and methods to be able to trace changes due to regression throughout the code. All methods relating to individual classes are listed and parameters assigned, including those to test exceptions. Valid and invalid criteria are normally set to validate the results.

The purpose of the unit tests for the Shopping Cart involves checking the functionality of the main methods which involves adding items to Cart and returning details of the respected products.

In the example used, the following parameters and associated input values were determined to generate testing criteria for a test plan, see table 1.1 below;

Table 1.1 Test Plan

Test ID	Parameter	Expected	Input value	Output	Result
1.1	Add Item	Menu displayed	Text	Item Name	
1.2	Enter Price	Menu displayed	Number	Value	
1.3	Name	Menu Displayed	Number	Quantity	
1.4	Description	Request?	Y/N	Proceeds	

Detailed Unit Tests

The IDE provides a unit testing framework (JUnit 3/4) to facilitate the production of JUnit test classes for User.java and Administrator.java. This exercise uses the JUnit3 to test Products and details.

The IDE has been used to auto-generate the unit test cases and are stored in the compressed zip file supplied. The method body of each of the generated tests provides a default framework.

Common Errors and Exceptions

A Syntax error caused by omitting a semicolon (;) at the end of a statement will return;

```
RunProject.java:14: `;' expected.
```

A semantic error generates a warning when a variable has not been initialized and returns;

```
RunProject.java:13: Variable count may not have been initialized.
count++

RunProject.java:14: Variable count may not have been initialized.
System.out.println("Input has " + count + " chars.");
```

In each case, the program will not successfully compile and the compiler will not create a .class file unless the error is resolved.

In the following example a runtime error is generated where Java cannot find the byte code file, User.class.

Exception in thread "main" java.lang.NoClassDefFoundError: User

This error can be rectified by assigning the .class file to the current directory.

A Java program needs to have a main method associated with the class and/o package to initiate the application else the following exception is generated;

Exception in thread "main" java.lang.NoSuchMethodError: main

Testing Class Structure and Associations

The class structure and associations for the online store were validated using the reverse engineering function of Eclipse IDE. Although the classes were selected alphabetically to auto generate the Class Diagram, the logical structure, dataflow and framework of nested packages used to model the Class Diagram were preserved.

Add Item:

Here when the user details goes invalid, the server or database is unable to find the actual products so it shows the exception about incorrect order or not finding it.

Delete Item:

The already deleted items or the products which are not available in the store are dumped as an error conditions.

Update Item:

the item updating is not done in the case where there is nothing to update with the new information is termed as an error or exception in such cases while testing with unit test cases with JUnit.

Remove Product:

the products which are already checked out or deleted are not used by the system and while testing those conditions will provide with the errors of using it without removing with certain exceptions.

Code:

The source code of the entire project is kept in the folder named CS445_Project and it is run by running RunProject.java file as mentioned in the CS445_Rima_ProjectReadMe.txt read me file.

Deployment Models (elaborated and complete)

User manual:

Introduction

The purpose of this document is to formalize all the requirements of the Products and Services Tracking system that needs to be developed for a E-security Camera store.
1.1 Goals and Objectives
The goal is to develop a web based software solution that can be used by any logged in user to efficiently track product and service availability and manage customer order creation as customers' order products or services. The software should be simple and easy to use to ensure smooth adoptability and easy trainability of all intended users of the system

This entire project is implemented in Eclipse IDE as java source code using the Graphical User Interfaces creating a well-established Desktop application making it more reliable and flexible for the user to interact with the system and make a purchase for different types of products using the information provided by the system and user is allowed to add, update and delete the purchases as well from the carts along with updating the their personal details, so using various UI components this project is made completely following all the functionalities using Swing.

1.2 Statement of Scope

The E-Security Store shall allow its customers to browse and order security cameras, DVRs and Accessories through the internet.

The main functionalities of the system provides information about the products it carries to help customers make purchases decisions; handle customer registration, order processing and shipping.

It provides means for supporting the management of the system such as adding, deleting and updating security cameras, Surveillance DVRs, updating customer information etc.

The scope of the project is to develop an inventory management for surveillance security camera shop. From inventory we mean tracking the products that are available in the shop. There are three major products besides services that are to be tracked along with Vendor Information. They are camera, DVR and accessories. The user should obtain a complete report of the current state on the quantity of the products to be listed. We also give options to add, delete and update and these options can be performed by a logged in user. Customer orders can be generated provided the quantity being purchased is available in stock. For the security level we are providing only a single level of security to the project. All data input/update/delete operations can be performed by all and any logged in user. The 6 available options are Security Camera Tracking, DVR System Tracking, Accessory Tracking, Service Tracking, Customer Order Tracking, and

Reporting. By generating the reports, we get the current state of the products.
The below table provides a list of all requirements that the system being developed shall meet. The descriptions are intended to illustrate system details to end users who would actually be using the system in a simple and clear manner so as to obtain their approval and sign off. To meet this end technical and other implementation details that are irrelevant to this purpose are avoided.

R1: The Main page shall prompt the user to enter a username and password. All users need to login to access the site.
Functional. Critical
R2: There shall be no role based permissions and all users who login shall access all functionalities.
Non Functional. Critical
R3: If a valid user enters an incorrect password three times, the user shall be locked out for 30 minutes.
Functional. High
R4: If a user forgets the password, he/she shall request a new password by entering the answer to his/her security question. Functional. Future
R5: New Users shall register to the site by entering the below information. Functional. High

- First Name
- address
- city
- state
- country
- zipcode
- contact
- email-id

R6: Once logged in, a main page with the below 6 options (as links) shall be displayed. Non Functional. Critical

- Security Camera Tracking
- DVR System Tracking
- Accessory Tracking
- Service Tracking
- Customer Order Tracking
- Reporting

R7: Security Camera Tracking shall be accessed by clicking on the “Security Camera Tracking” link on the main page.

Non Functional. High
R8: This page shall allow the user to enter a new Security Camera with the below information:

- Camera Type : Text Field
- IR LED: Text Field
- IR LED working distance: Text Field
- Water resistance: Yes/No field
- Lens: Text Field
- Voltage: Text Field
- Dimension: Text Field
- Weight: Text Field
- Quantity : Numeric Field
- Price: Numeric Field

Functional. Critical

R9: An auto generated Camera ID shall be generated by the system for each newly added Security Camera.
Functional. High
R10: Security Camera information shall be updated and deleted by any logged in user.
Functional. Critical
R11: The last user who inserted or updated information on a Security Camera shall be recorded by the system along with the time the operation was done.
Functional. High
R12: DVR System Tracking shall be accessed by clicking on the "DVR System Tracking" link on the main page.
Non Functional. High
R13: This page shall allow the user to enter a new DVR System with the below information:

- DVR type: Text Field
- Compression Format : Text Field
- Resolution : Text Field
- Playback Resolution: Text Field
- Real time recording support: Text Field
- USB port: Numeric Field [default value of 1]
- Weigh: Text Field
- Quantity : Numeric field
- Price: Numeric Field

Functional. Critical

R14: An auto generated DVR System ID shall be generated by the system for each newly added DVR System.
Functional. High
R15: DVR System information shall be updated and deleted by any logged in user.
Functional. Critical

R16: The last user who inserted or updated information on a DEV system shall be recorded by the system along with the time the operation was done.
Functional. High
R17: Accessory Tracking shall be accessed by clicking on the "Accessory Tracking" link on the main page.
Non Functional. High
R18: This page shall allow the user to enter a new Accessory with the below information:

- Name: Text Field
- Description : Text Field
- Quantity : Numeric field
- Price: Numeric Field

Functional. Critical
R19: An auto generated Accessory ID shall be generated by the system for each newly added Accessory.
Functional. High
R20: Accessory information shall be updated and deleted by any logged in user. Functional. Critical

R21: The last user who inserted or updated information on an Accessory shall be recorded by the system along with the time the operation was done. Functional. High
R22: Service Tracking shall be accessed by clicking on the "Service Tracking" link on the main page.
Non Functional. High
R23: This page shall allow the user to enter a new Service with the below information:

- Name: Text Field
- Description : Text Field
- Quantity : Numeric field
- Price: Numeric Field

Functional. Critical

R24
An auto generated Service ID shall be generated by the system for each newly added Service.
Functional
High
R25
Service information shall be updated and deleted by any logged in user.
Functional
Critical
R26
The last user who inserted or updated information on a service shall be recorded by the system along with the time the operation was done.

Functional
High
R27
Customer Order Tracking shall be accessed by clicking on the “Customer Order Tracking” link on the main page.
Non Functional
High
R28
This page shall allow the user to enter a new Customer Order with the below information:

- • Item Type : Pick list with values Security Camera, DVR System, Accessory and Service
- • Product/Service: Pick list with the products or services with the Item type
- Contact Name: Text Field
- • Business Name: Text Field
- • Business Address: Text Field
- • Business URL: Text Field
- • Email address: Text Field
- • Phone Number : Text Field
- • Quantity Requested: Numeric field

Functional
Critical
R29
An auto generated Customer Order ID shall be generated by the system for each newly added Customer Order.
Functional
High
R30
Customer Order information shall not be updated or deleted once entered.
Non Functional
Critical
R31
The customer order shall be entered only if the “quantity requested” entered for the product or service is less than or equal to the quantity available.
Functional
Critical
R32
On entry of new order information, the “quantity requested” for the product/service shall be deducted from the corresponding quantity of available products/services. Functional. High

R33
The user who inserted a customer order shall be recorded by the system along with the time the operation was done.
Functional. High
R34

The Security Camera Report shall provide a list of all Security Camera types available along with quantity. Functional. High

R35

The Security DVR Report shall provide a list of all Security DVR types available along with quantityFunctional. High

R36

The Accessory Report shall provide a list of all accessories available along with quantity and vendor information. Functional. High

R37

All reports shall be printable

Functional

Future

R38

The system shall respond to all requests within 10 seconds

Non Functional

High

R39

The system shall send out an alert when the quantity available for a product is 0 (or a configured value close to 0)

Functional

Future

1.3 Software Context

The software consists of three tier architecture. They are User browser, the application and the database. Here the user starts the browser and uses the application and stores the data in the database. Data backup and the database maintenance will be provided by the company.

1.4 Major Constraints

Our website can be affected by various changes in the company. Listed below are potential conditions in the firm that may change our system.

- Employee job roles constraints: Any change in job positions will change the system as the system has only three main users i.e. Administrator, Employers and Customers.
- Employer job responsibility constraints: This is the core of the system, as the whole system will change if each job role's responsibility changes.
- Hardware Constraints: The hardware system must have Pentium-IV MHz processor, minimum of 128 MB RAM is required and hard disk of 10GB size.
- Operating System Constraint: System should have Windows 7, Vista, and XP as an operating system.

2.0 Architectural and component-level design.

A description of the program architecture is presented in this section.
Architecture diagram
E-security camera store
Admin_Login
Customer_Info
Employee_Login
Employee_Info
Add Customer
Delete Employee
Add Employee
View Order
Online Payments
Delete Customer
Place order
View Order

2.1 System Structure

Because the nature of the software primarily involves fetching and storing data, a data-centric architecture will form the basis of the architecture of our software. All components will fetch data from or store data to the database as shown below:
The interface to the database server component will consist of a function that can be easily modified for Server access change or be improved. Should the interface to the Server change, only the function will have to be modified to accommodate the change.
List of Components-
List of the components used in our application are given below.
Data Store Responsibility
Customer Info
Order Details
Login
Emp Info
Product Info
Return Info
Vendor Info
Camera Types and DVR's
Ship From
Ship To

Accessories

1. 1) Login
2. 2) Customer
3. 3) Employee

Description of each component is given below.
Processing narrative for component Login Validation
In order to make any changes in the database, the employee and admin needs to login into the system. The Login Validation component is responsible for prompting for the user's login information (login ID and password), accepting the user's login ID and password, checking if it is valid, by looking in the project repository database for a match, and then allowing the user into the proper system, or asking for re-entry (if there is no match). Login validation will also be responsible for the Sign-Out function of the system, which will allow the user to leave the system and lock it, until another user enters a valid ID and password.
Component Login Validation interface description.
The input will be the login ID and password. In this component, the user is authenticated using the login_id and the password. The output to the system will be the type of user the ID and password match with. The output will be log onto the corresponding page. The input for Sign-Out will be if the user clicks the "Sign Out" button, the output will be to end that user's session and lock the local application.

- Login
- Login_id
- Password
- User Type
- Assignaccess()
- Displaypage()
- Assignaccess()
- Displaypage()

2.1.1 Component Login Validation processing detail

- Design Class hierarchy for component Login Validation
Restrictions/limitations for component Login Validation
This component is restricted by the user input and a direct connection to the project repository.

- Login and password, both must be match.
- Login
- Login_id
- Password
- User Type
- Assignaccess()
- Displaypage()
- Assignaccess()
- Displaypage()
- CUSTOMER_INFO
- Customer_Id
- Customer_First_Name
- Customer_Last_Name

- Company_Name
- Address
- City
- State
- Zipcode
- Phone_No
- Email
- +AddCustInfo( )
- +DeleteCustInfo( )
- +UpdateCustInfo( )
- +SearchCustInfo( )
- EMPLOYEE_INFO
- Employee_Id
- Employee_Name
- Employee_Address
- Employee_City
- Employee_State
- Employee_Zipcode
- Employee_Email
- Employee_Phone
- +AddEmpInfo( )
- +DeleteEmpInfo( )
- +UpdateEmpInfo( )
- +ViewEmpInfo( )

- Password should not exceed 10 characters.

Performance issues for component Login Validation
This component may be slowed if the project repository becomes excessively large and many records need to be searched to find the correct one.

- The Login interface needs to fetch the username and password from the database and hence the delay is dependent on network configuration and load on the database server.
- Database should not overflow or become out of control.

Design constraints for component Login Validation
The component should manage its tasks with minimal dependencies on other components, in order to keep the project repository model at a perfect level of data fidelity.

- All the users have different access level according to their privilege level.
- Data members of the Login Interface will be implemented as follows:

- User_Name – Varchar2
- Password – Varchar2

- TypeOfUser – Varchar2

Processing detail for each operation of component Login Validation
Sign-In: prompt for the user's login information (login ID and password), accept the user's login ID and password, checking if it vaid, by looking in the project repository database for a match, and then allowing the user into the proper system, or asking for re-entry (if there is no match).
Sign-Out: click button, end session and lock system, until another user enters a valid ID and password.
Processing narrative for each operation
Processing narrative (PSPEC) for "AssignAccess()"

1. 1. User enters a username and password.
2. 2. This is checked in the database table Login depending access level.
3. 3. Depending on the access level provided to that user, the corresponding page is displayed.
4. 4. After successful login in, display the page corresponding assigned access level.
5. 5. Depending on the access level provided to that user, the corresponding page is displayed.

Processing narrative (PSPEC) for "DisplayPage()"
After successful login in, display the page corresponding assigned access level.
Depending on the access level provided to that user, the corresponding page is displayed.
Algorithmic model (e.g., PDL) for each operation
Algorithmic model (e.g., PDL) for "AssignAccess()"

1. 1. Enter username and password.
2. 2. Check for password and access level in the database table Login.

Algorithmic model (e.g., PDL) for "Displaypage()"

1. Call the function "SignIn()" to display the page.
2. Display the corresponding page depending access level.

Component Employee

Processing narrative (PSPEC) for component Employee
This component comes into use when the Admin wants to View order, view customer profile or update customer profile

EMPLOYEE_INFO

- Employee_Id
- Employee_Name

- Employee_Address
- Employee_City
- Employee_State
- Employee_Zipcode
- Employee_Email
- Employee_Phone
- Date_Time
- User_Name
- Password

- +AddEmpInfo( )

Component Employee interface description.
The inputs to the Employee component are the Employee Information,
2.2.1 Component Customer processing detail
- Design Class hierarchy for component Customer

EMPLOYEE_INFO

- Employee_Id
- Employee_Name
- Employee_Address
- Employee_City
- Employee_State
- Employee_Zipcode
- Employee_Email
- Employee_Phone
- Date_Time
- User_Name
- Password

- +AddEmpInfo( )
- +DeleteEmpInfo( )
- +UpdateEmpInfo( )
- +ViewEmpInfo( )
- Login

Login_id

- Password
- User Type

- Assignaccess()
- Displaypage()

Restrictions/limitations for component Employee
Admin needs to enter Employee data to get information while intending to view from Customer relationship management system database.
Performance issues for component Employee
For the best performance, all the field of Employee information should be filled properly.
If not provided, send back error message to user.
Design constraints for component Employee
Data members of the Employee Information will be implemented as follows:

- EmployeeId
- Number
- EmployeeName
- Varchar2
- EmployeeAddress
- Varchar2
- EmployeeCity
- Varchar2
- EmployeeState
- Varchar2
- EmployeeZipcode
- Number
- EmployeeEmail
- Varchar2
- EmployeePhone
- Number
- DateTime
- Datetime
- UserName
- Varchar2
- Password
- Varchar2

Processing detail for each operation of component Employee
Processing narrative (PSPEC) for each operation
Processing narrative (PSPEC) for "AddEmpInfo()"
The input is new Employee information which is added in the database in the table Employee_info.
If any errors have occurred, corresponding error message will be display Otherwise, record will be store in database table.
Processing narrative (PSPEC) for "UpdateEmpInfo()"

The input is the current Employee information which is retrieve from the database table employee_Info for update current employee information and save changes to the database table employee_Info.

If any errors have occurred, corresponding error message will be display Otherwise, record will be update to the database table.
Processing narrative (PSPEC) for “DeleteEmpInfo()”
The input is current Employee information list which is retrieve from the database table employee_Info and record to delete a particular entry from the database table employee_Info.
If any errors have occurred, corresponding error message will be display Otherwise, record will be delete from the database table.
Processing narrative (PSPEC) for “ViewEmpInfo()”

- The input for this operation is current employee information which is retrieve from the database table employee_Info and display list of employee as a report.
- Algorithmic model (e.g., PDL) for each operation
- Algorithmic model (e.g., PDL) for “AddEmpInfo()”
- Acquire new record
- Add records to database in the table Employee_Info.
- Display the message “Added Successful” if no errors otherwise display the corresponding message.
- Algorithmic model (e.g., PDL) for “UpdateEmpInfo()”
- Acquire the changes in the record
- Update records to the database in the table Employee_Info.
- Display the message “Update Successful” if no errors otherwise display the corresponding message.
- Algorithmic model (e.g., PDL) for “DeleteEmpInfo()”
- Acquire the command to delete a particular entry from the database.
- Delete that entry from the database from the table Employee_Info.
- Display the message “Deleted Successfully” if no errors otherwise display the corresponding message.
- Algorithmic model (e.g., PDL) for “ViewEmpInfo()”
- Acquire Employee information for display employee list.

Call the function “ViewEmpinfo( )” to display the employee records on page as a list.

2.3 Dynamic Behavior for Component

A description of the interaction of the classes is presented in this section.

2.3.1 Interaction Diagrams

Interaction diagram for “Login Failed”

- CLR
- Login Interface
- User
- Call function
- Call function
- Check Password and UserId

- False
- Login Failed

Interaction diagram for “Login success”

- CLR
- Login Interface
- User
- Call function
- Call function
- Check Password and UserId
- True
- Home Page

Interaction diagram for “Add Customer Information”
Interaction diagram for “View Customer Information”

- CLR
- Customer_Info
- Request to Customer_Info
- Retrieve Database
- Admin
- CLR
- Customer_Info
- Create & Validate Customer information
- Add to Database
- Customer

Interaction diagram for “Update Customer Information”
Interaction diagram for “Delete Customer Information”

- CLR
- Customer_Info
- Request to Customer Info
- Update Database
- Admin/Customer
- CLR
- Customer_Info
- Request to Customer Info
- Delete Data
- Admin

Interaction diagram for “View shipping Information”

- CLR
- Shipping_To
- Shipping_From
- Request For View Shipping_info
- Retrieve Shipping_Info From Database
- Admin/Customer

Interaction diagram for “Add Shipping Information”

- CLR
- Shipping_To
- Shipping_From
- Add And Validate ShippingTo _info
- Success Fully Insert ShippingTo_Info
- Add And Validate ShippingFrom _info
- Success Fully Insert ShippingFrom_Info
- Customer

Interaction diagram for “Update Shipping Information”

- CLR
- Shipping_To
- Shipping_From
- Request For Update
- ShippingTo _info
- Success Fully Update ShippingTo_Info
- Request For Update ShippingFrom _info
- Success Fully Update ShippingFrom_Info
- Admin/Customer

Interaction diagram for “Delete Shipping Information”

- CLR
- Shipping_To
- Shipping_From
- Request For Delete Shipping_info
- Success Shipping_Info Delete From Database
- Admin/Customer

Interaction diagram for “View Order Details”

- CLR
- Order
- Request For View Order
- Retrieve Order From Database
- Admin/Customer

Interaction diagram for “Add order Information”

- CLR
- Order
- Insert & Validate Info
- Success Fully Add Record
- Customer

Interaction diagram for “Update Order Information”

- CLR
- Order
- Request Update Order
- Success Fully Update Record
- Admin/Customer

Interaction diagram for “Delete Order Information”

3.0 User interface design:

The application has a Graphical User Interface, which is build completely in HTML. Among the variety of components available in HTML and more sophisticated techniques are available to change to appearances of those components if required in future.
The top of the page of application displays our company logo and name after that the line of the page displays the site global navigation bar.

3.1 Interface design rules

A fundamental reality of application development is that the user interface is the system to the users. What users want is for developers to build applications that meet their needs and that are easy to use. The point is that a good user interface allows people who understand the

- CLR
- Order

- Delete Order
- Success Fully Deleted Record
- Admin/Customer

problem domain to work with the application without having to read the manuals or receive training.
User interface design is important for several reasons. First of all the more intuitive the user interface the easier it is to use, and the easier it is to use the cheaper it is. The better the user interface the easier it is to train people to use it. The better our user interface the less help people will need to use it. The better our user interface the more our users will like to use it, increasing their satisfaction with the work that we have done.
To improve the usability of an application it is important to have a well designed Interface.

3.2 Components available

These various Graphical User Interfaces are used in the application which simplifies the work for the user to fetch the corresponding details of each of the products using Swing JFrame and its following components.

Textbox
Label

Checkbox
Radio Button
Button
Link Button
Text Area

Drop down box

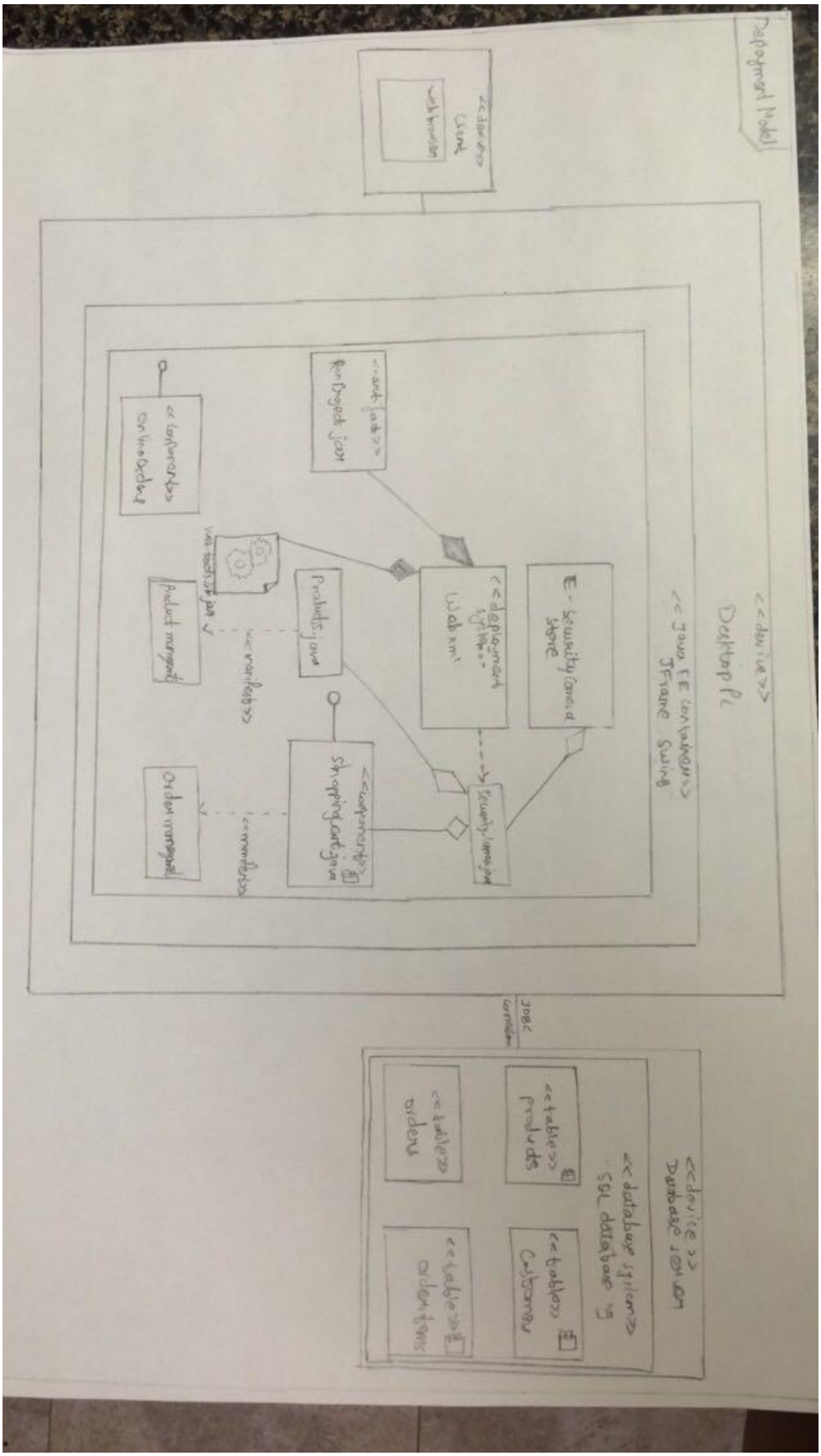
Deployment Diagram:

A deployment diagram can be used to model the physical architecture and (in the case of networks or distributed systems) the topology of the software system being developed. It describes the hardware devices (known as *nodes*), the software components that run on them (known as *artifacts*), and the communication links between the various nodes and artifacts. In the most trivial case, the "system" may consist of a single executable (*.exe*) file that runs on a stand-alone computer. The illustration below shows what such a scenario might look like as a deployment diagram. The *System.java* artifact represents a single instance of an executable program file, is deployed to the *DesktopPC* node, which represents a stand-alone desktop computer.

So in this case RunProject.java artifact is deployed where an artifact icon is drawn *inside* a node icon to show that the artifact is *deployed* on that node. The node itself thus becomes the *deployment target* for the artifact. Note that the node *DesktopPC* is labeled with the stereotype *«device»* (to indicate that this is a hardware node), and the name of the node. A node is represented as a cube, presumably because the use of a three-dimensional shape emphasizes the fact that the node represents some physical object in the real world (i.e. an item of hardware). Note, however, that a node can also be used to represent a software environment that hosts an artifact. Examples of hardware nodes include desktop computers, workstations, servers, disk drives, and embedded hardware such as microcontrollers.

So the Java EE application server used is the JFrame Swing package, to run the application in its own.

It is shown using the dependency <<deploy>> manifesting from Products.java to its management.


Deployment Model
<<device>>
Desktop PC
<<Java EE container>>
JFrame Swing
<<artifact>>
Project.jar
<<container>>
Tomcat Server
E-Commerce
Application
Web.xml
Product.jsp
Product.java
<<manifest>>
Product manager
<<manifest>>
Shopping Cart.java
<<component>>
Shopping Cart.jsp
<<manifest>>
Order manager
<<manifest>>
Security manager
E-Security Framework
JDBC
<<device>>
Database Server
<<database system>>
SQL database
<<table>>
Products
<<table>>
Customers
<<table>>
Orders
<<table>>
Order items