# Chapter 3: Design

# Design:

Design is the plan or decorative pattern that show the functions, objects, methods of the software before it is developed. It provides the initial view of the proposed software such as how it will look, works.

Since I’m using object-oriented programming, this process helps me to divide the tasks into classes, how these classes will interact, what data each will contain and what action each will perform. It will make easier to understand the requirement easier and prevent redundancy. Risk will be very low if we use this process. It projects initial vision of the proposed software.

Types of software design level:

1. **Architectural Design:** It recognized the software as a system with multiple components associated with each other.
2. **High-level Design:** It focuses into how the system components can be implemented in forms of modules.
3. **Detailed Design:** It deals with the implementation of modules and the interactions between them.

# Structural Modelling:

Structural model projects the concept of the system and illustrate the relationship between system’s components. This model is independent of time.

# Final class diagram:

A class diagram is illustration of the architecture of a system which maps out the system’s classes, their attributes, methods and relationship between them. It provides the static view of the system which can be used as a blueprint for the final product.

**Justification of approach:**

* Forward and reverse engineering process.
* Analysis and design of the static view of a system.
* Provide basic notation for other modelling in UML.
* Construction of system using object-oriented programming (OOP)

**Notation used:**

|  |  |  |
| --- | --- | --- |
| **Notation** | **Explanation** | **Remarks** |
| + | This sign represents the class is public and is accessible within the namespace |  |
| - | This sign represents the class is private and is not accessible to other class |  |
| # | This sign represents the class is protected and only accessible within the solution inside a single namespace |  |
| Generalization | Generalization: -  This shows the inheritance from one class to another |  |
| Aggregation | Used to show relationship between class that have relation among their instances |  |
| Composition | Used to show the child class that is fully dependent to parent class |  |
|  | Used to show classes with attribute and function |  |

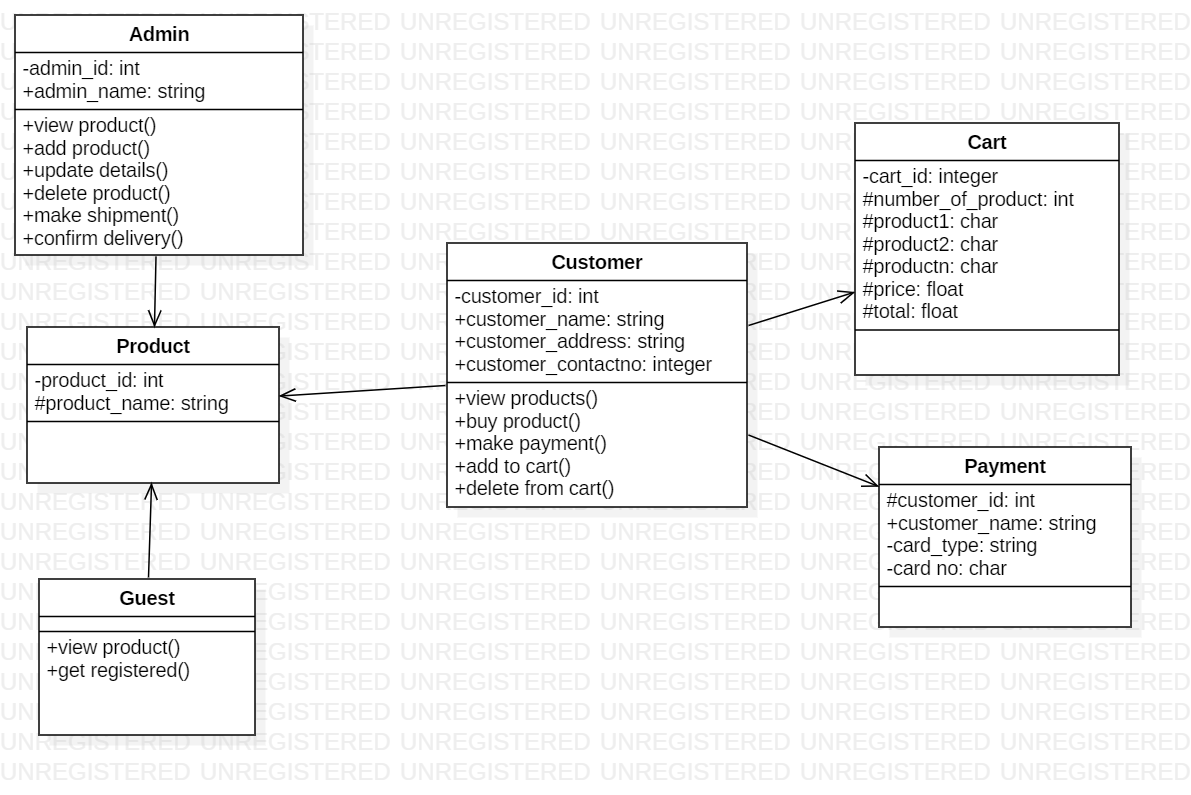


Figure 1: final class diagram

# Data flow diagram:

Data flow is used to show the flow of data in the system. DFD describes the process that are involved in a system to transfer data from the input to the file storage and reports generation.

**Justification of the approach:**

* logical information flow of the system
* determination of physical system construction requirements
* simplicity of the notation
* establishment of manual and automated system requirements

**Notation used:**

|  |  |  |
| --- | --- | --- |
| **Notation** | **Explanation** | **Remarks** |
| External entity | Represent human, system or subsystem. It is where data come from. |  |
| Process | It is function or activity where data is manipulated. |  |
| Data store | Data are store here. Simply, database |  |
| Data flow | Show the direction of flow of information. |  |

**For admin**

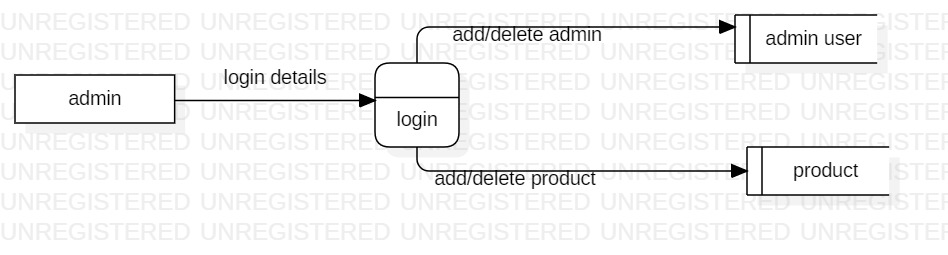


Figure 2: DFD of admin

**DFD explanation for admin:**

1. Admin provide login details
2. Login leads admin to admin panel
3. Admin can add/delete admin user and product

**For customer**

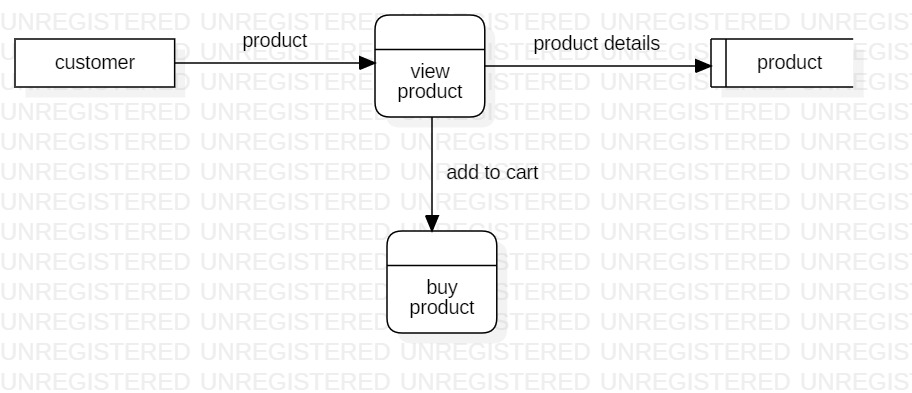


Figure 3: DFD of customer

**DFD explanation for customer:**

1. Customer request to view the product
2. Product details are retrieved from database and display to customer
3. Customer add the product in cart to buy

**For login and registration**

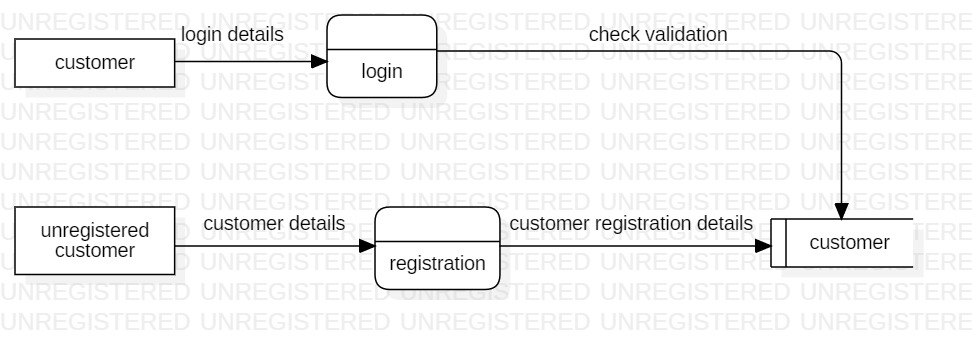


Figure 4: DFD of login

**DFD explanation:**

1. Customer provides their details for registration
2. Details are store in database
3. Customer provide details to login
4. System check the details with the database
5. Give access if details are valid otherwise deny

**For validation**

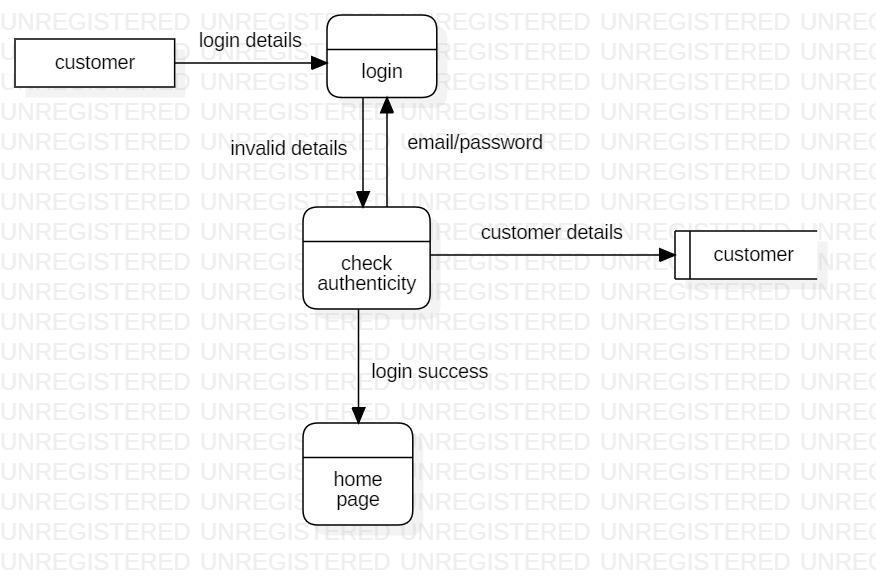


Figure 5: DFD of verification details

**DFD explanation for validation:**

1. Customer provide details to system for login
2. Details are checked with the details in the database
3. Give access and open home page if details are correct
4. Return back to login if details are wrong

**For manipulate information**

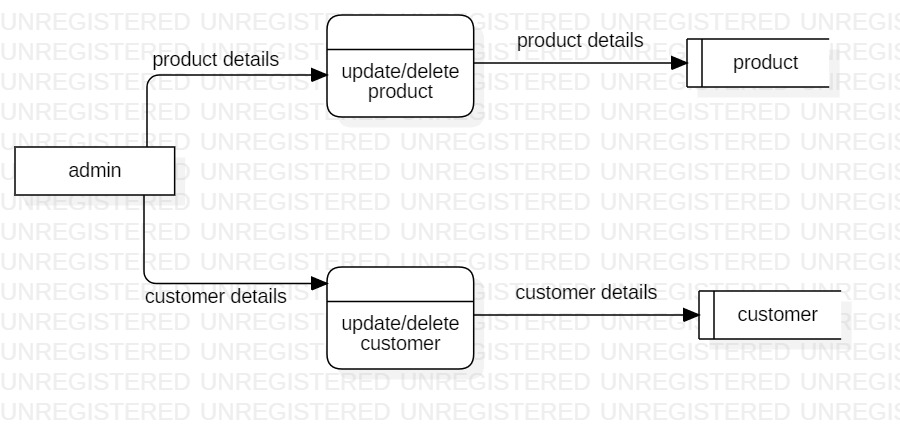


Figure 6: DFD of admin

**DFD explanation:**

1. Admin either manipulate customer or product details
2. Admin add/delete either customer or product
3. Products details are stored in database after manipulation

# Behavior Modelling:

It captures the varieties of interaction and instantaneous states within a model as it executes over time; tracking how the system will act in a real-world environment, and observing the effects of an operation or event, including its results.

# Activity diagram:

Activity diagram is graphical representation of flow of activity from one to another. This describe the dynamic aspect of the system. It is essentially an advanced version of flow chart that modeling the flow from one activity to another.

**Justification of approach:**

* Describe the sequence from one activity to another.
* Demonstrate the logic of an algorithm.
* Illustrate the various steps involved in a UML use case

**Notation used:**

|  |  |  |
| --- | --- | --- |
| **Notation** | **Explanation** | **Remarks** |
| Start | This symbol represents the start point of the system |  |
| Action/activity | This symbol represents the action or activity |  |
| Action flow | This symbol represents the flow of action from one to another |  |
| Decision | This shows where decision is made of the action |  |
| Merge |  |  |
| Join node | It joins the multiple activities into single activity |  |
| Fork node | This splits one activity into multiple activities |  |
| Swim lane |  |  |
| End point | This symbol represents the end point of the activity |  |

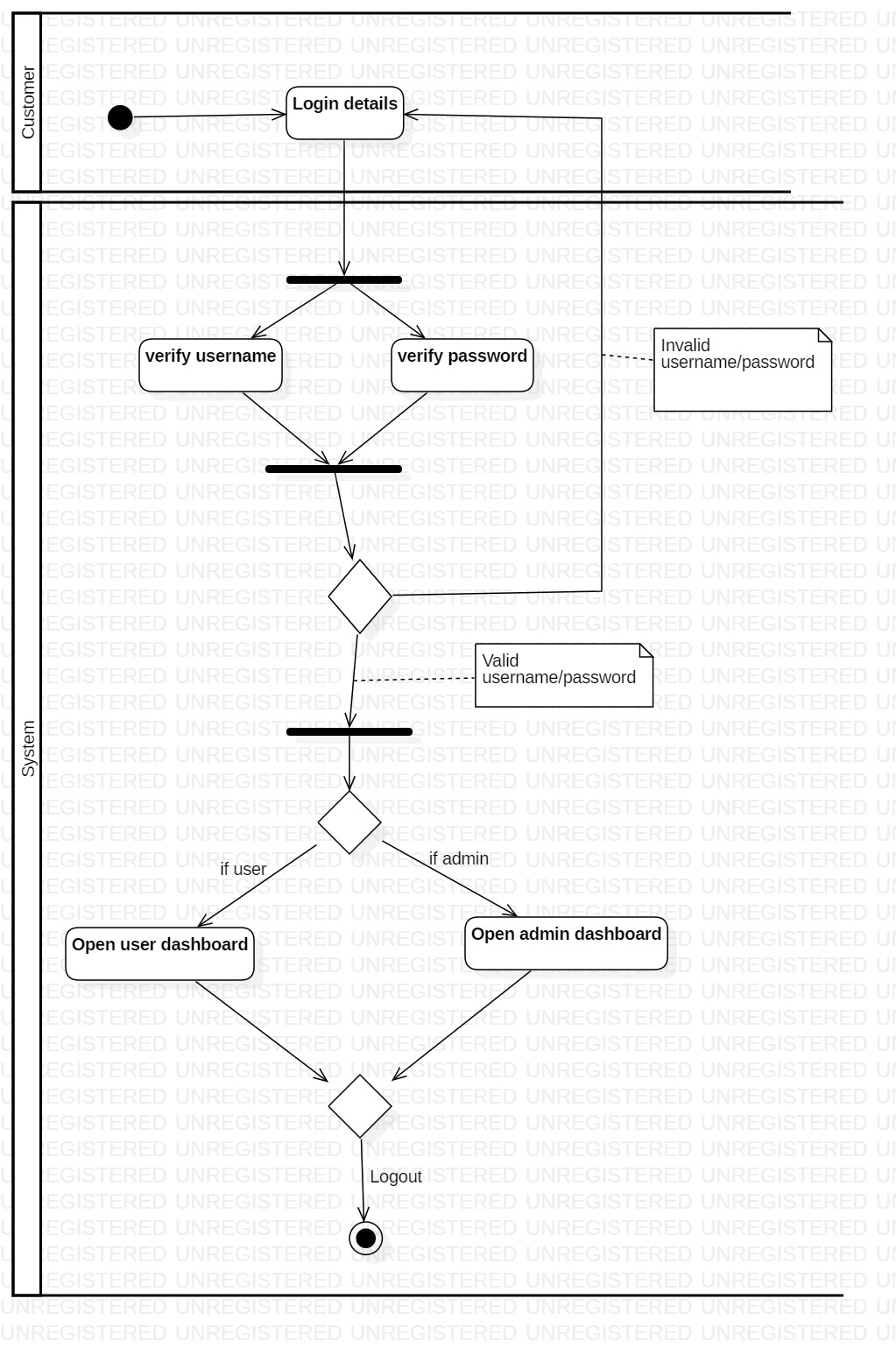


Figure 7: activity diagram for login

**Activity diagram explanation:**

1. Customer provide login details to access the system
2. System check the details
3. If details are valid, dashboard is opened otherwise it moves back to login
4. User logout from the system

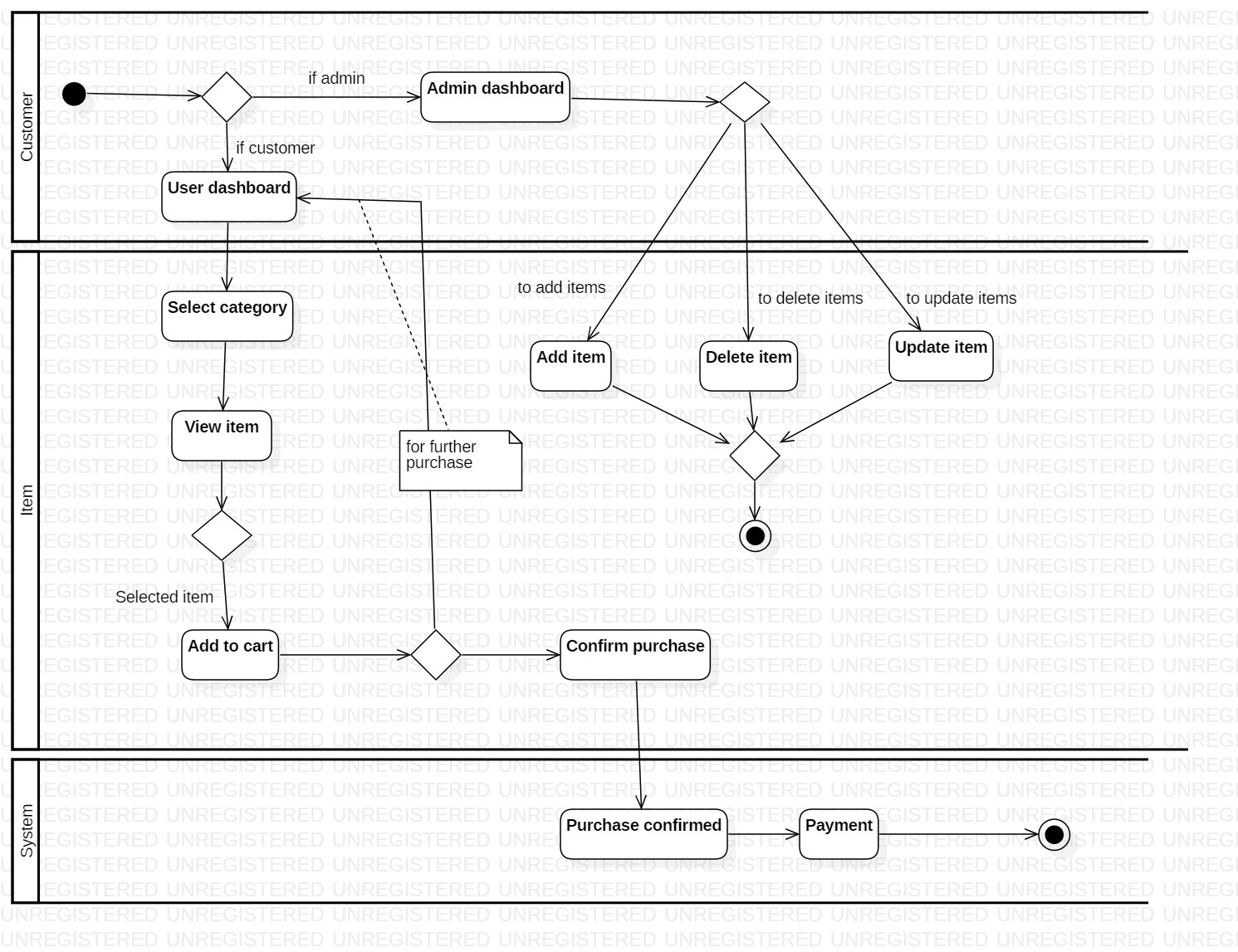


Figure 8: activity diagram to purchase item

**Activity diagram explanation:**

1. User login the system
2. Open dashboard => if admin – admin dashboard; if customer – customer panel
3. Admin make decision either to add, delete or update item and logout
4. Customer select the item category, view items and make decision what to add to cart.
5. After that customer make decision either to continue or confirm purchase
6. Customer make payment for confirmed purchase and logout

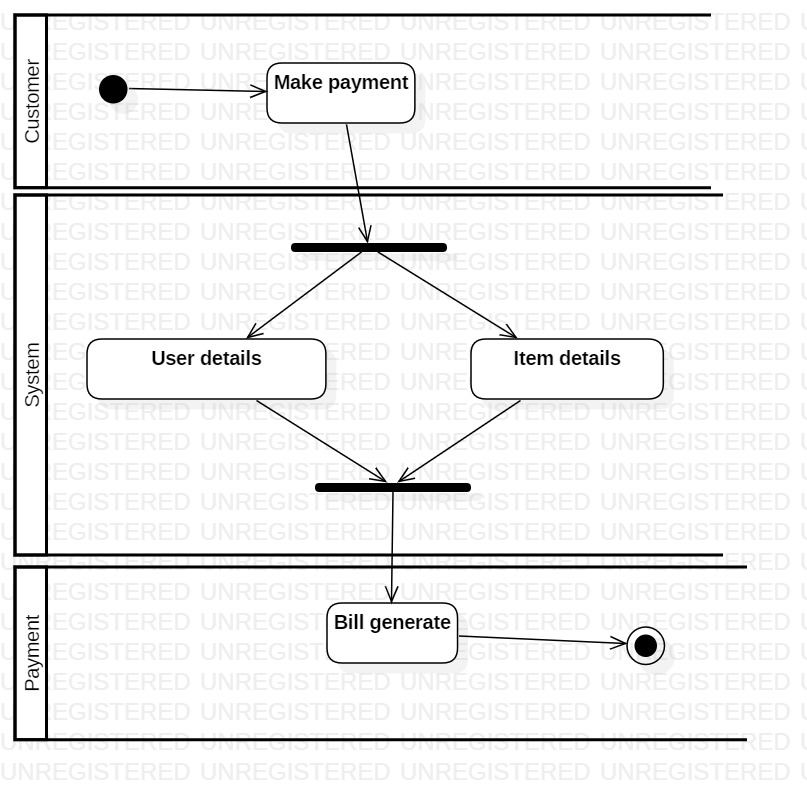


Figure 9: activity diagram for payment

**Activity diagram explanation:**

1. Customer want to make payment
2. System gain the customer and item details
3. Bill is generated and customer logout

# Sequence diagram:

This diagram describes interaction of objects in sequential order. Sequence diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what message are sent and when.

**Justification of approach:**

* Used to understand the detailed functionality of current or future systems
* Visualize how message and tasks move between objects in the system
* Analyze and visualize the logic behind function, operation or procedure

**Notation used:**

|  |  |  |
| --- | --- | --- |
| **Notation** | **Explanation** | **Remarks** |
| Lifeline | Used to represent the time duration. |  |
| Message | Used to share message between classes. |  |
| Reply message | Response to the message |  |
| Self-message | Process or method arise within the lifeline operation |  |
| Alternate | Used to show the choices |  |

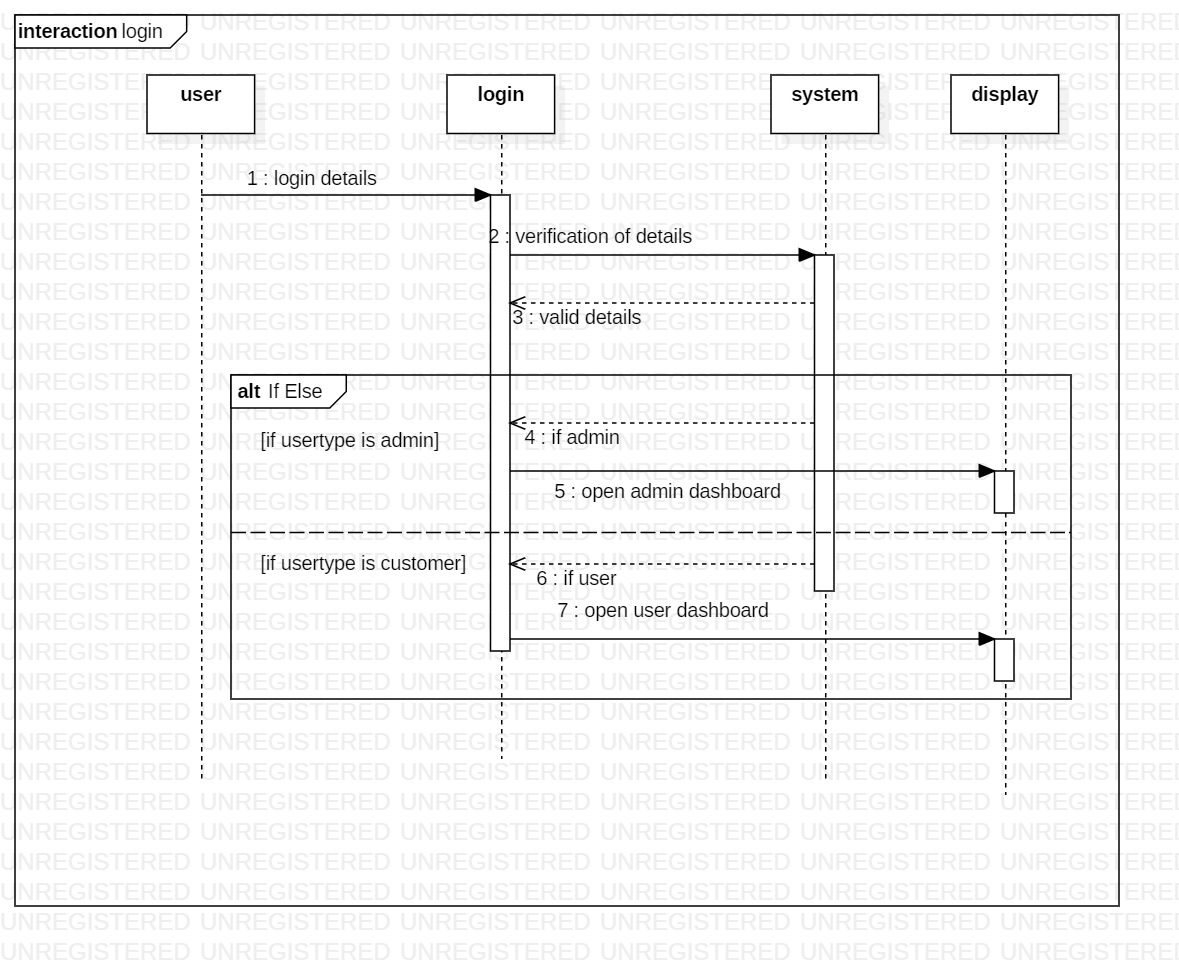


Figure 10: login sequence

**Sequence diagram explanation:**

1. User provide details for login
2. System check the details and reply message
3. If valid open respective dashboard for admin and customer

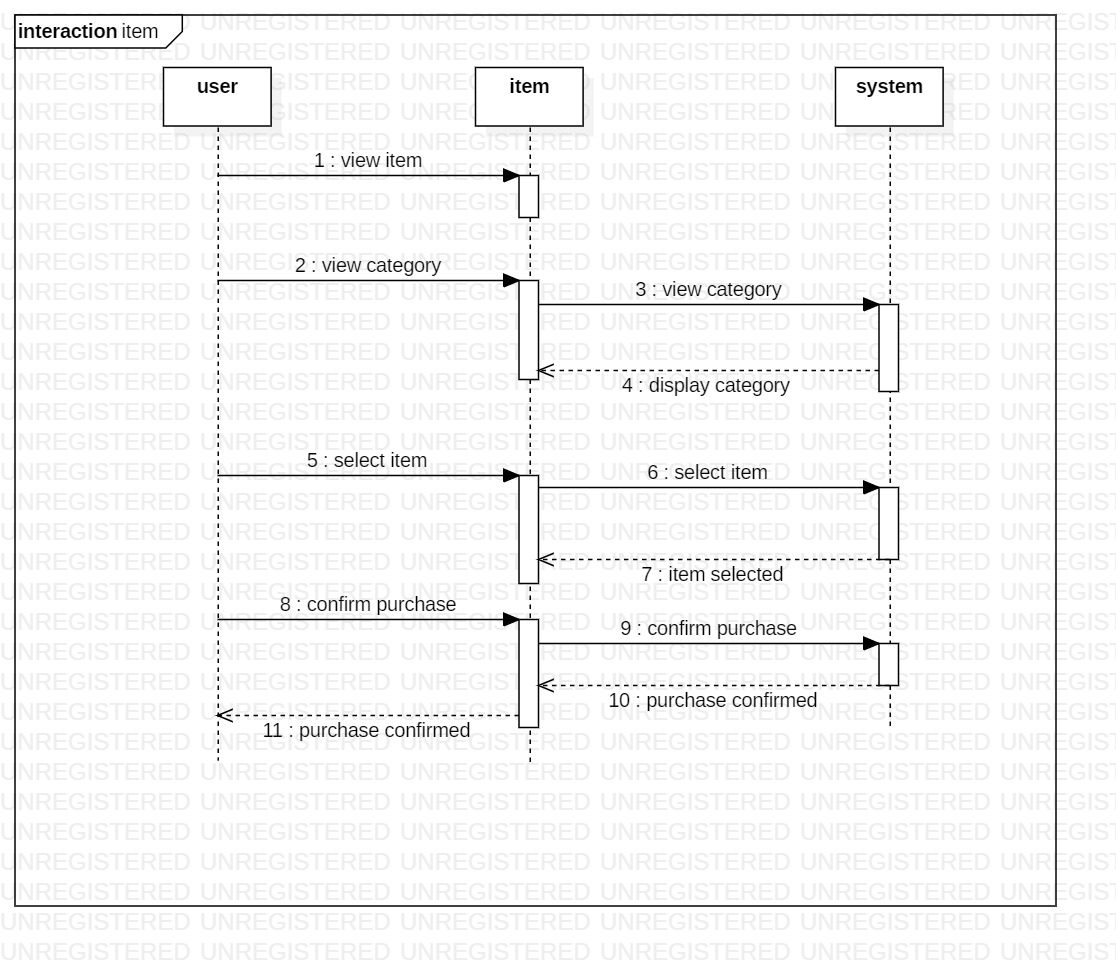


Figure 11: item sequence

**Sequence diagram explanation:**

1. User selects items by category
2. System display the category
3. User select the items
4. System verify the items are selected
5. Customer confirm purchase and the system reply for it

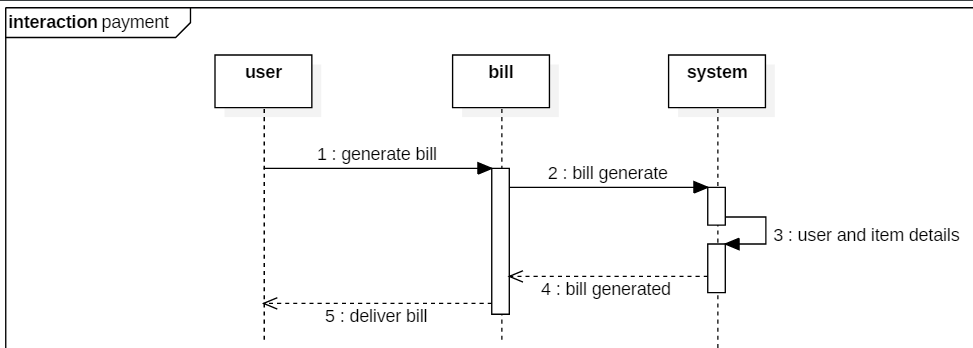


Figure 12: payment sequence

**Sequence diagram explanation:**

1. User ask to generate bill
2. System generate the bill and reply to user
3. User get the bill

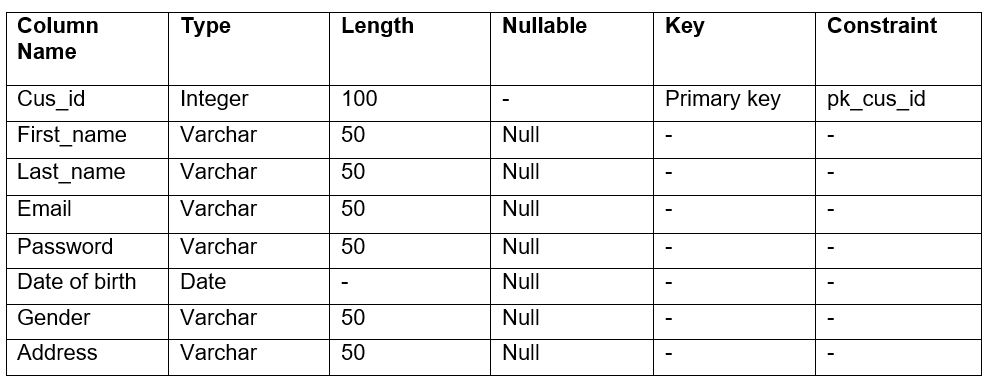
# Database Modelling:

Database is another important thing in system. Database modelling is the process of creating a database model for the data to be stored in it. Database modelling helps in the visual representation of data and enforces business rules, regulatory compliance.

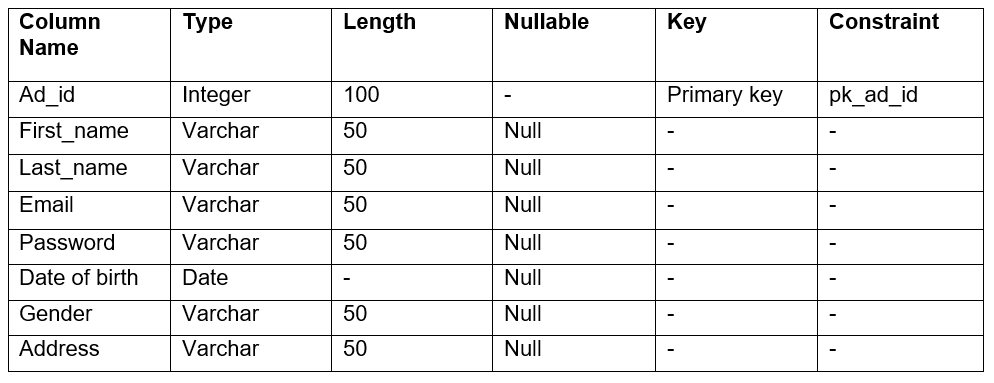
# Data dictionary:

The table which contain the overall details of data that define the principle of database design and work. It contains database metadata. It describes things such as access to data, where the database is physically located.

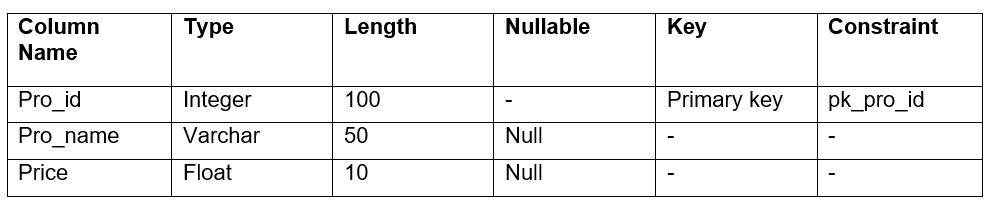
**Column Customer**

****

**Column Admin**

****

**Column Product**

****

# ER diagram:

This diagram project the relationship between main entities of the system by mapping out the attributes, methods of the entities. It is used in database designing. ER model is based on the notation of real- world entities and the relationship between them.

It is used due to follow reason:

* Helps to define terms related to ER modeling
* Helps to describe entities, attributes, relationships
* Database designer gains better understanding of the information to be contained in the database.

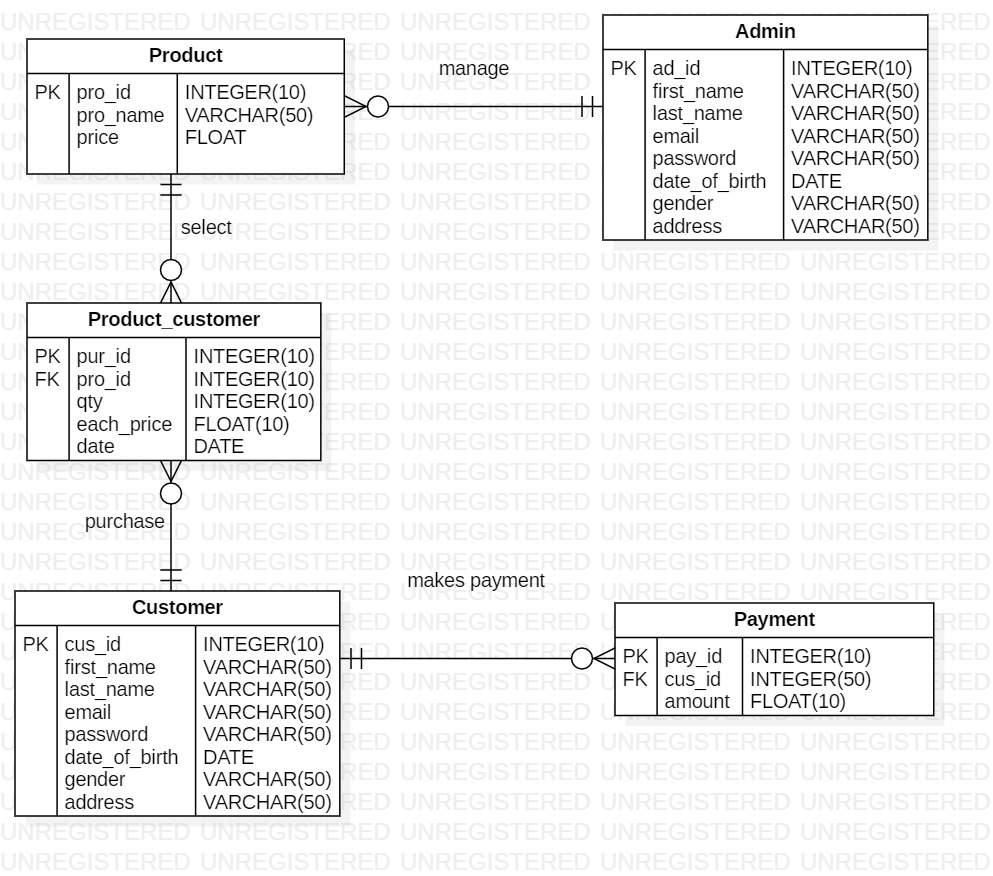


Figure 13: ER-diagram

**ER diagram explanation:**

1. Admin manages many items (one to many relationship)
2. One customer buys many items and many customers buy one item (many to many relationship)
3. Linking table is crated between products and customer
4. Customer makes payment of many products (one to many relationship)

# User Interface Modelling:

This model illustrates the way how human react with the computer or system to perform any task in the system. It focused in looks or style.

Great UI have:

* Common icon and well leveled
* Simple and clear interface

# Prototyping:

This is the representation of how the system will look after it’s completion. This makes easy in coding.

**Login Prototype**

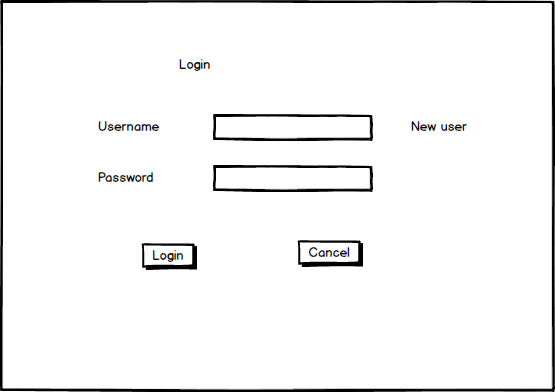


Figure 14: login UI

**Registration Prototype**

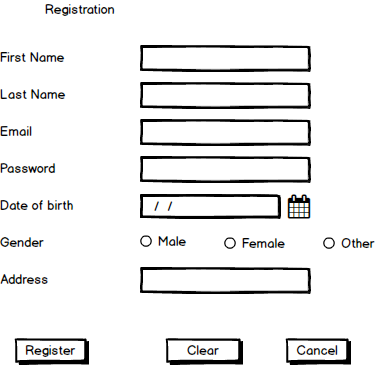


Figure 15: registration UI