# Software Design Specification GiGe

19BCE0436 Atul Agarwal
19BCE0421 Tanmay Bansal
19BCE0866 Aaditya Pareek

## Table of Contents

# 1. Introduction

## *1.1. Document Description*

Here is the description of the contents (by section and subsection) of the proposed template for software design specifications:

### 1.1.1. Introduction

A brief overview of this Software Design Document:

- The purpose of the document to clarify and specify all the development design models used to make the implementation and development of the project simpler. The diagrams and models provide a deeper understanding of the project while also bearing the technical applications in mind. The document intents to be direct and apt in its description for models of designing and constructing the components of the project.

- The scope of the document is to enclose all Unified Modelling Languages, the Architecture of the project's final product and the Data Models for the project.

- The SRS is intended for different types of readers such as:

1. Developers
2. Designers
3. Project Managers
4. Product Managers
5. Marketing Managers
6. Users
7. Copywriters

### 1.1.2. System Overview

GiGe is a platform for sharing goods and services among students using a Coin system. Gicoins can be purchased as well as earned. Users can upload "gives" and "gets" which are the way of changing points. In "get", you spend points by getting some items, and in "give", you earn points by giving your items to students. That item can be almost anything like books, video games, electronics, projects, cycles, sports goods, etc. The giver has to upload details about the service.

It is basically an ecosystem that thrives on people helping each other. The coins are used to measure how much one is helping others. Students who "give" earn coins and can spend those coins in the app to "get" things they need.

Main Features of the Site:

1. Login/Logout/Account creation
2. Product Sell/Give Mode
3. Product Buy/Get Mode
4. User Reviews and Ratings for Products and services
5. Searching for products and services
6. Transaction Maintenance(Gicoin transactions made safe by Cryptography)
7. Account View
    a. Wallet : GiCoins left and spent
    b. Product Selling History
    c. Product Purchase History

# 2. Design Considerations

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution.

## *2.1. Assumptions and Dependencies*

GiGe, as a project, was developed while keeping a few assumptions in mind about the technical skills of the use and the hardware present for use. These assumptions are specified below:

Describe any assumptions or dependencies regarding the software and its use. These may concern such issues as:

- Minimum Disk Space: 512 MB
- Minimum Memory: 256 MB
- Bandwidth greater than 50 KBps (400 kbps)
- Minimum Display: Super VGA with a resolution of 1024 x 768
- Required Software: W3C Compliant Web Browser
- User is expected to know and perceive the navigation and visual prompts provided by the site.

## *2.2. Goals and Guidelines*

The goals and principles followed by the design implementation of the project include:

1. Consistency: GiGe aims to provide a consistent user interface with all visual packs utilized by the website being consistent throughout.
2. Guidance: A well defined and developed navigation bar to be present in all pages of the site with links to all important and the most probable choice of the user provided.
3. Error Prevention: Being a user-oriented website GiGe, will always aim to provide propped visual cues for users to ensure minimal error occurrence.
4. Minimal Surprises: The user will always be in control. The complete response set of the site to every action is expected and only on call.
5. Recognize rather than Recall: The site aims to minimize the memory load on the user and hence aims to use a recognition model rather than asking the user to recall their query. Although a search option is provided for when the user wants to use it.

## *2.3. Development Methods*

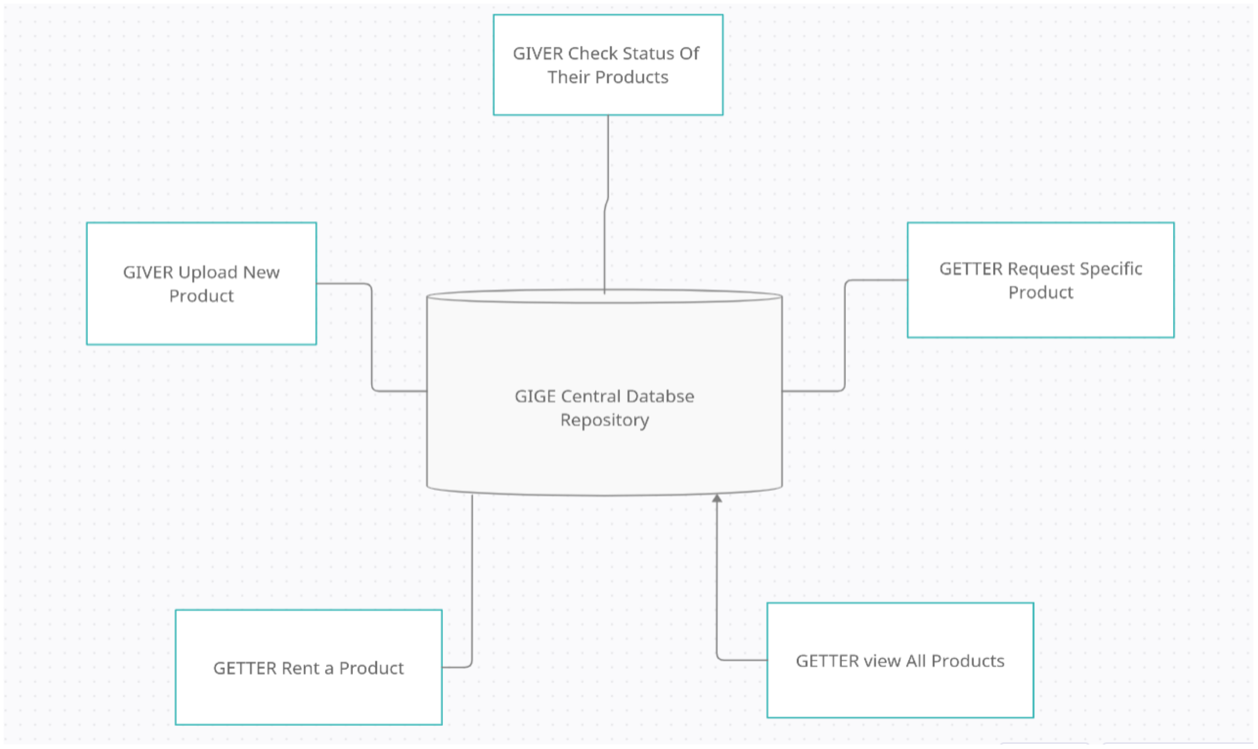
The Design Development methods used are the Unified Modelling Language diagrams. These diagrams or designing strategies include:

1. Use Case Diagram: A use case diagram is necessary to understand the functionalities of the systems and their interactions with the users of the site. They are important to gather the requirements of the system.
2. Class Diagram: A diagram that is not only used for the analysis and the design of the system but also for drawing up the executable code for the system. It is important for understand and visualizing all the components for the system.
3. Sequence Diagram: The sequence diagram identifies the user actions and the corresponding reactions from the system. It showcases the dynamic behavior of the system and also the message flow for the system.
4. State Chart Diagram: The visualization of all the states that are achievable by the system and how to reach them. These diagrams are necessary to map the behavior of the system.
5. Activity Chart: Another system that maps the actions of the system, used to visualize the sequence from one activity to another.
6. Object Diagram: Derived from class diagrams these diagrams implement the most important instances of the system which represent the most important functionalities of the system.
7. Component Diagram: These diagrams model the physical aspects of the system.
8. Deployment Diagram: These map the hardware topology of the system, and model all the hardware included in the user and server side.
9. Data Flow Diagram: Used to implement the flow of data in and out of the system.

## 3. System Architecture

The Structural Architectural Model followed by the system is a Repository Model, which emphasizes on a central database for all systems to follow and access. A central database becomes critical when implementing a web-based project. This model provides a concurrent maintenance of the system while also providing easy access for the technical team.

With GiGe, there is a central database and all subsystems such are described below:

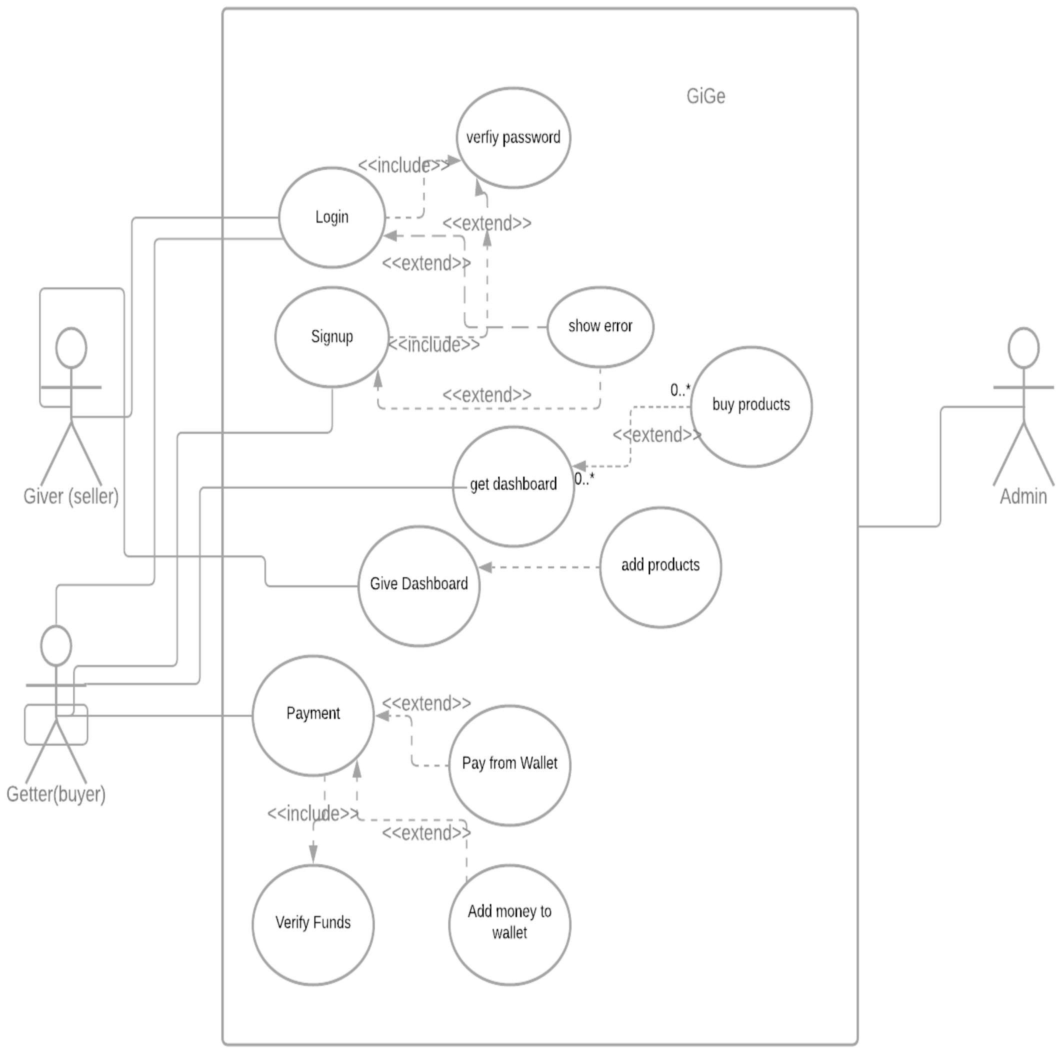

GIVER Check Status Of Their Products
GIVER Upload New Product
GETTER Request Specific Product
GIGE Central Databse Repository
GETTER Rent a Product
GETTER view All Products

The data for the website is stored on a centralized databased where well defined schemas exist for all backend functionality requirements. The various process of the site all access and request data from the central database and interpret it for the user interface application.

## 4. Detailed System Design

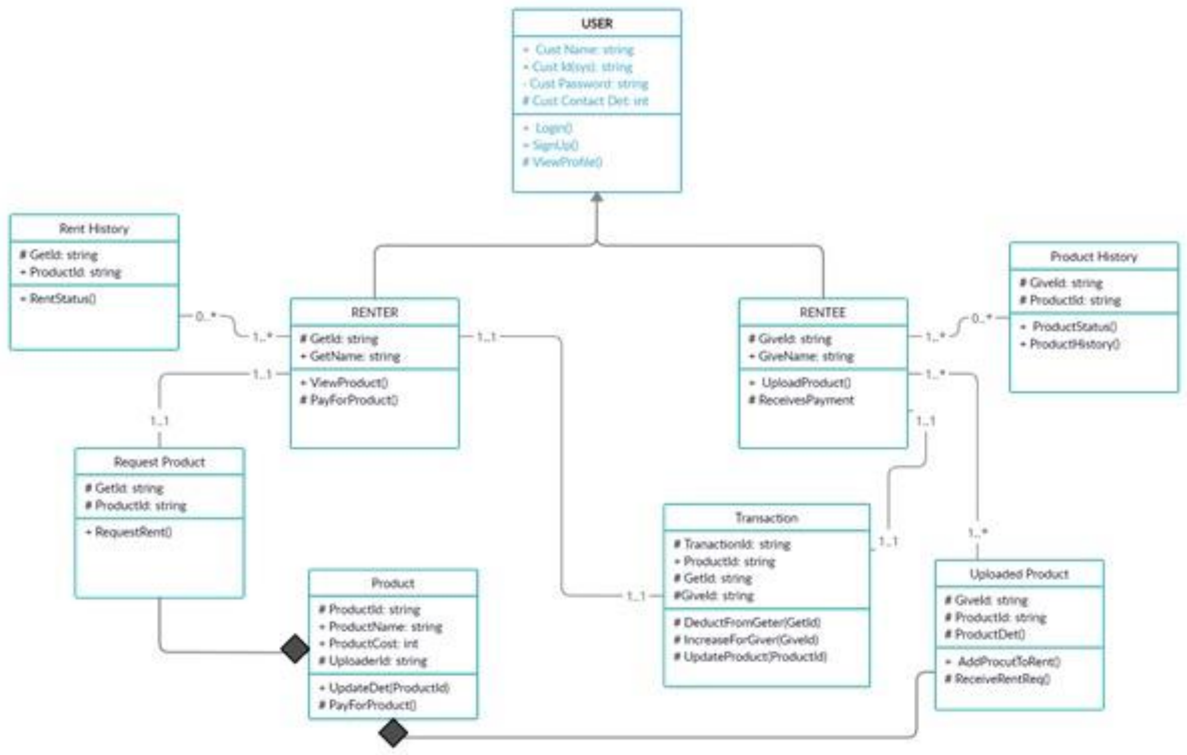
The UML Models used for the implementation and development of the systems are explained below:

1. USE CASE DIAGRAM:



This use case diagram aims to provide a clear understanding on how our site and system function. It has all the major functionalities of the system and also how each kind of intended user will interact with them. This diagram will be useful for anyone aiming to gain an insight into the system and also help the development team in the proper implementation of the system.

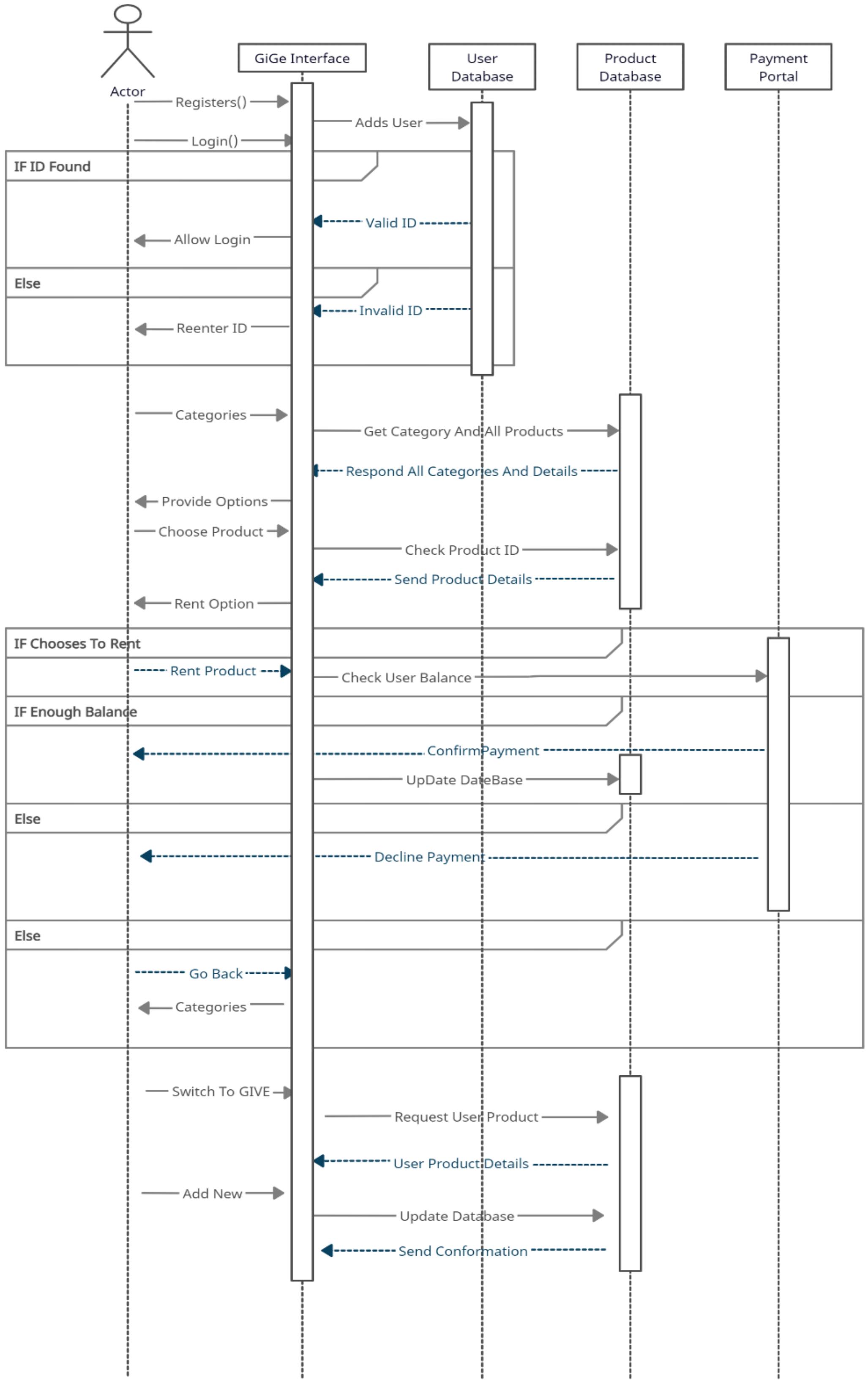
## 2. CLASS DIAGRAM:



The Class Diagram here showcases all the important classes of the system. It each class can be described as:

1. USER is a superclass. It's subclasses are RENTER and RENTEE. It contains all the details for a user and provides the methods for a new user to register on the site.
2. RENTER: It is the first kind of user for the site, it provides associations with the classes Rent History and Purchase Product Classes. A renter can have no purchase history or more than that. Similarly a renter can request to purchase one product at a time from the site.
3. RENTEE: A rentee can have one or more uploaded products by them, in the site. Also like the renter, they to have a History class with the same association classification.
4. PRODUCT: The Product class has methods for a product to be added or sold. Without the product class the Request Product Class, through which the renter can purchase a product cannot exist. And neither can a uploaded product for the rentee exist.
5. TRANSACTION: The transaction class is used to implement the transactions between the GIVER and GETTER.
6. HISTORY: Each mode of user(GIVER and GETTER) can access their transaction history, if any. These classes are separate for each mode.

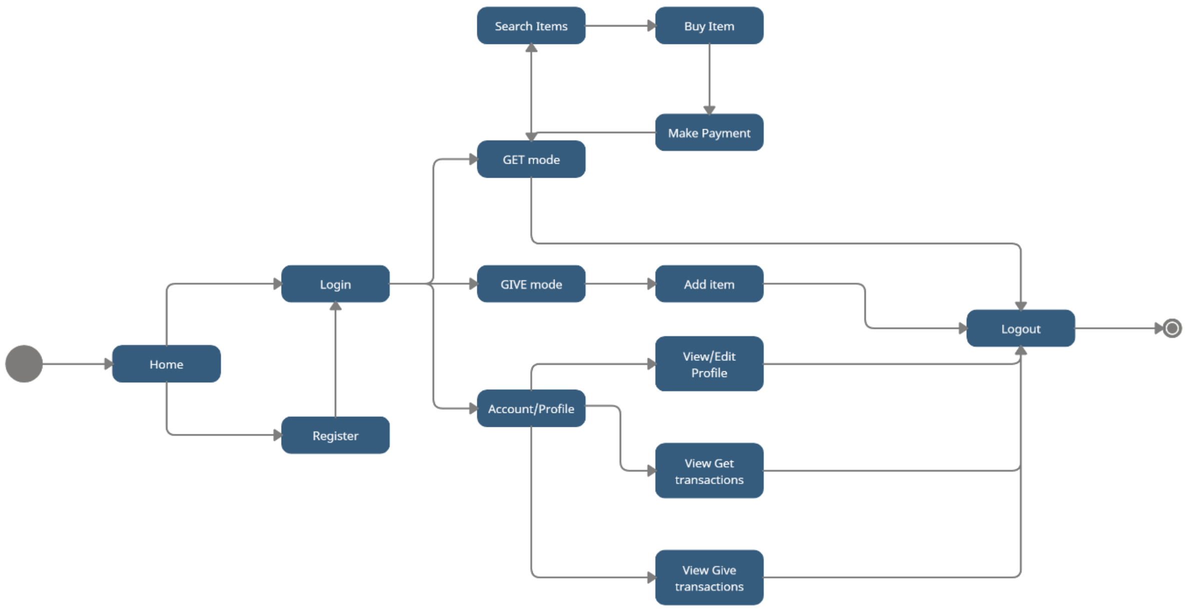
## 3. SEQUENCE DIAGRAM:



The sequence diagram here aims to properly demonstrate the dynamic behaviour of the system. It is important to visualize the actions of the user at the interface and the possible response by the site after connecting with the respective table in the database.

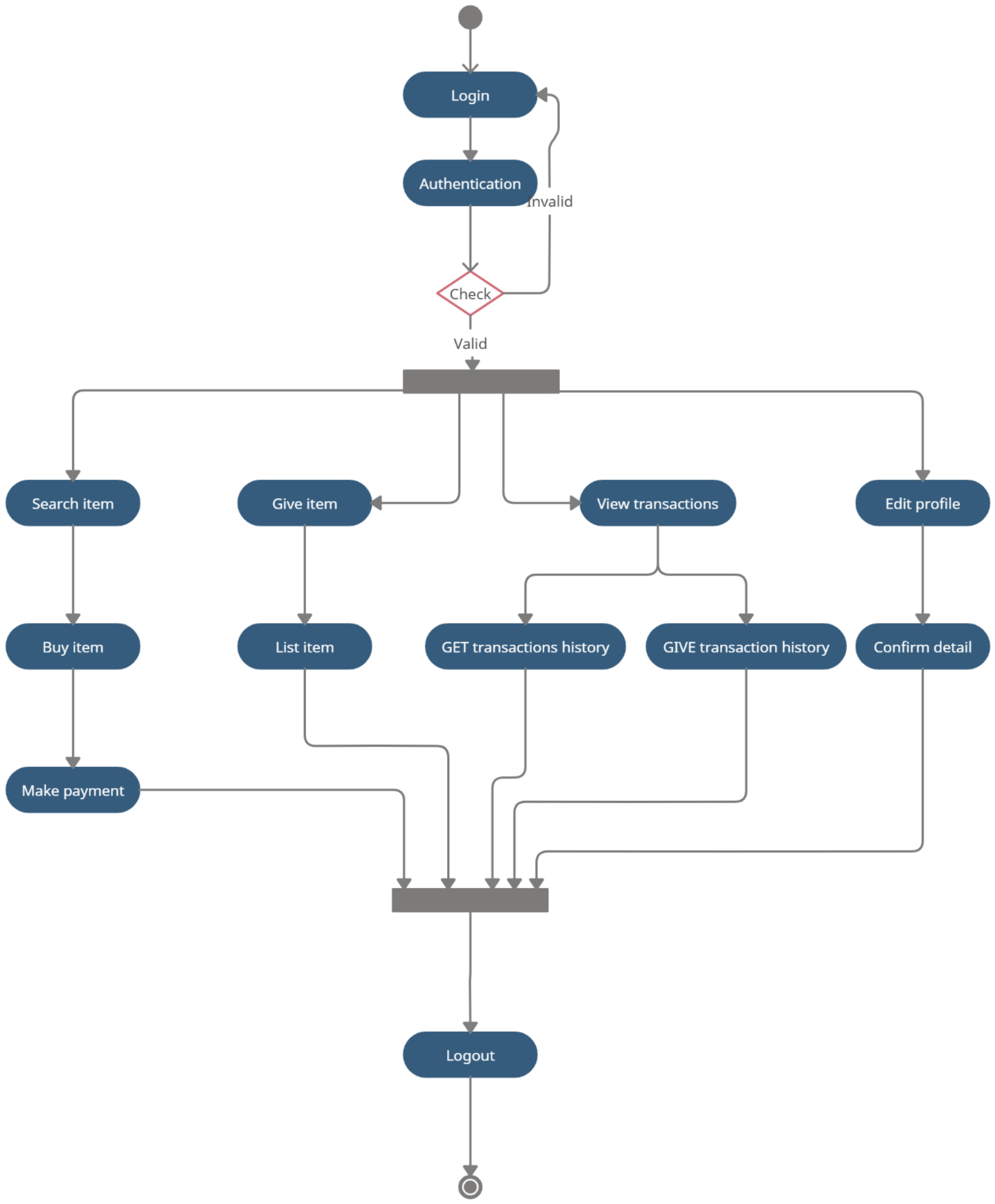
For every user action the interface connects with the backend server which uses the correct schema to lookup and reply with the data set corresponding to the request. The schema also showcases the handling of error in separate scenarios and how the system deals with them.

## 4. STATE MACHINE MODEL:

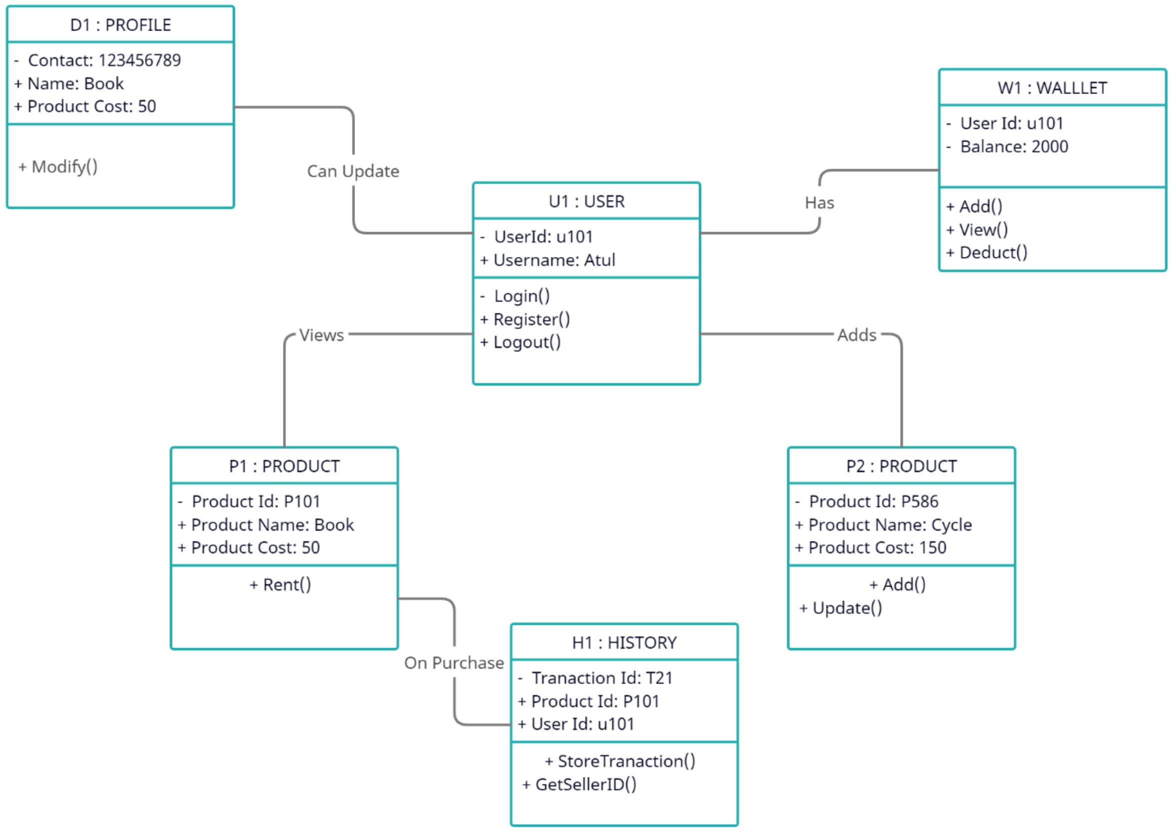


This UML implementation implements the states for the system. It has allowed us to visualize where the system would end for all user actions and the logical sequences for a stable exit from the system. It has showcased not only the function but also how the system attains these states.

## 5. ACTIVITY CHART:

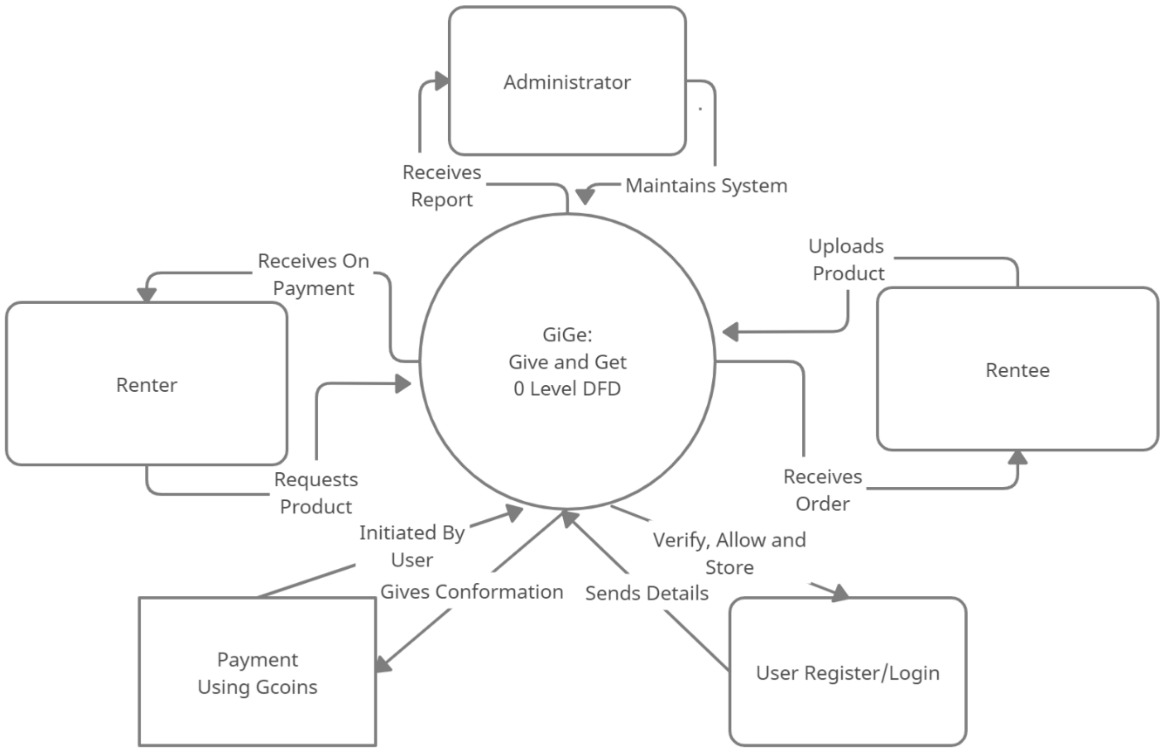


## 6. OBJECT DIAGRAM:



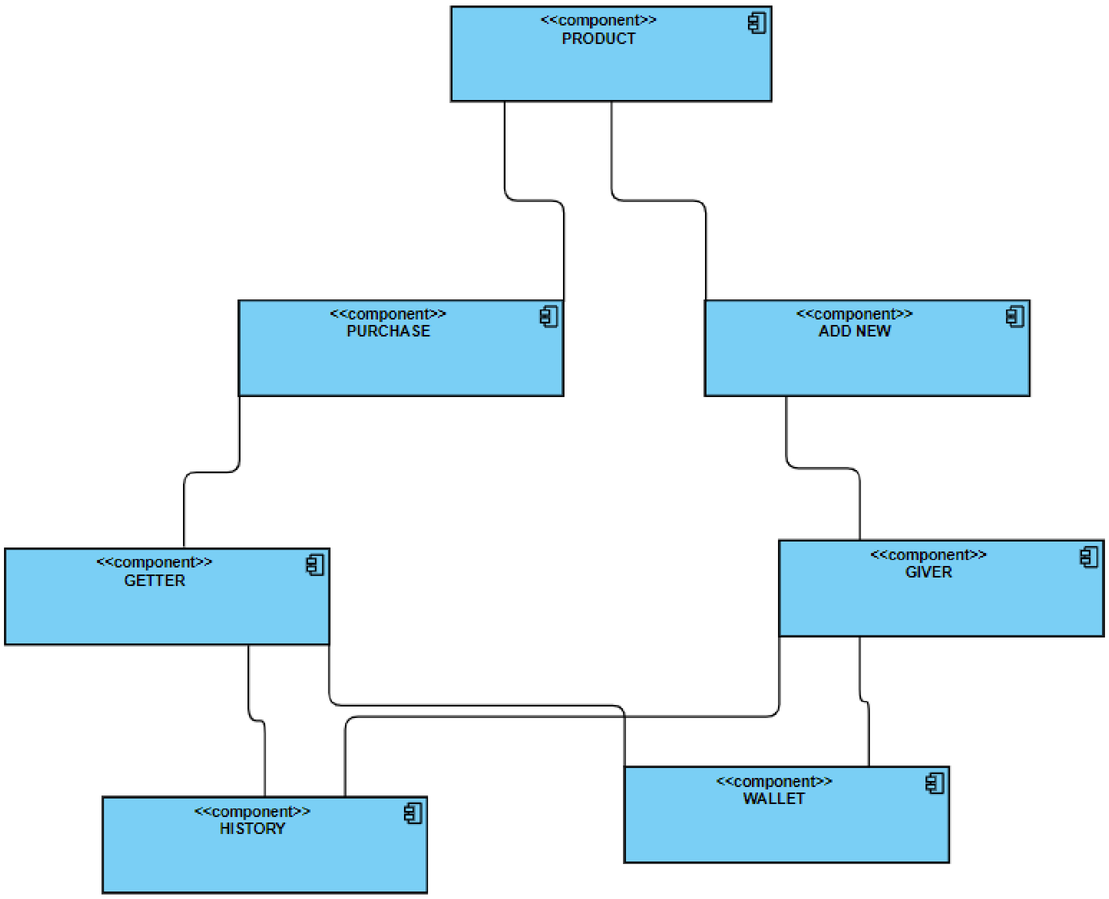
This object diagram has been derived from our class diagram. It contains all the important instances for our classes and the most important functionalities for them.

7. DATA FLOW DIAGRAM:

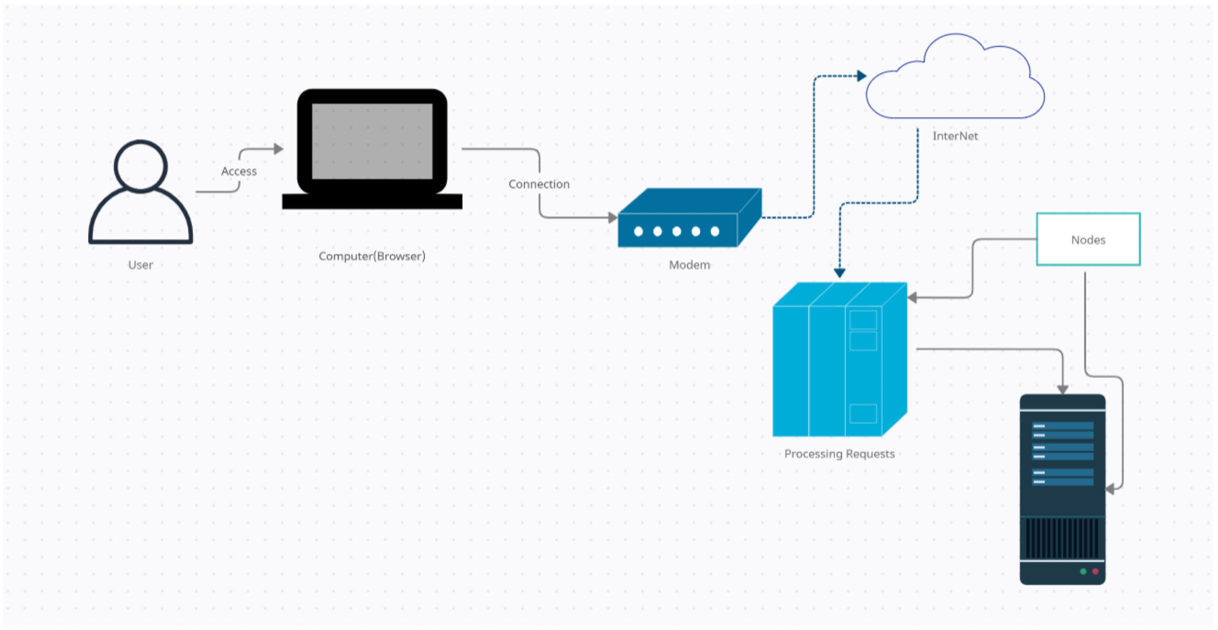


The data flow diagram for the system implements the data handling in and out of the system. It has the interactions between all roles that a user might take while using the system.

## 8. COMPONENT DIAGRAM:



9. DEPLOYMENT DIAGRAM:



## 5. Tools Used

Visual Paradigm: https://www.visual-paradigm.com/

Creately: https://creately.com/

Lucid Chart: https://www.lucidchart.com/pages/