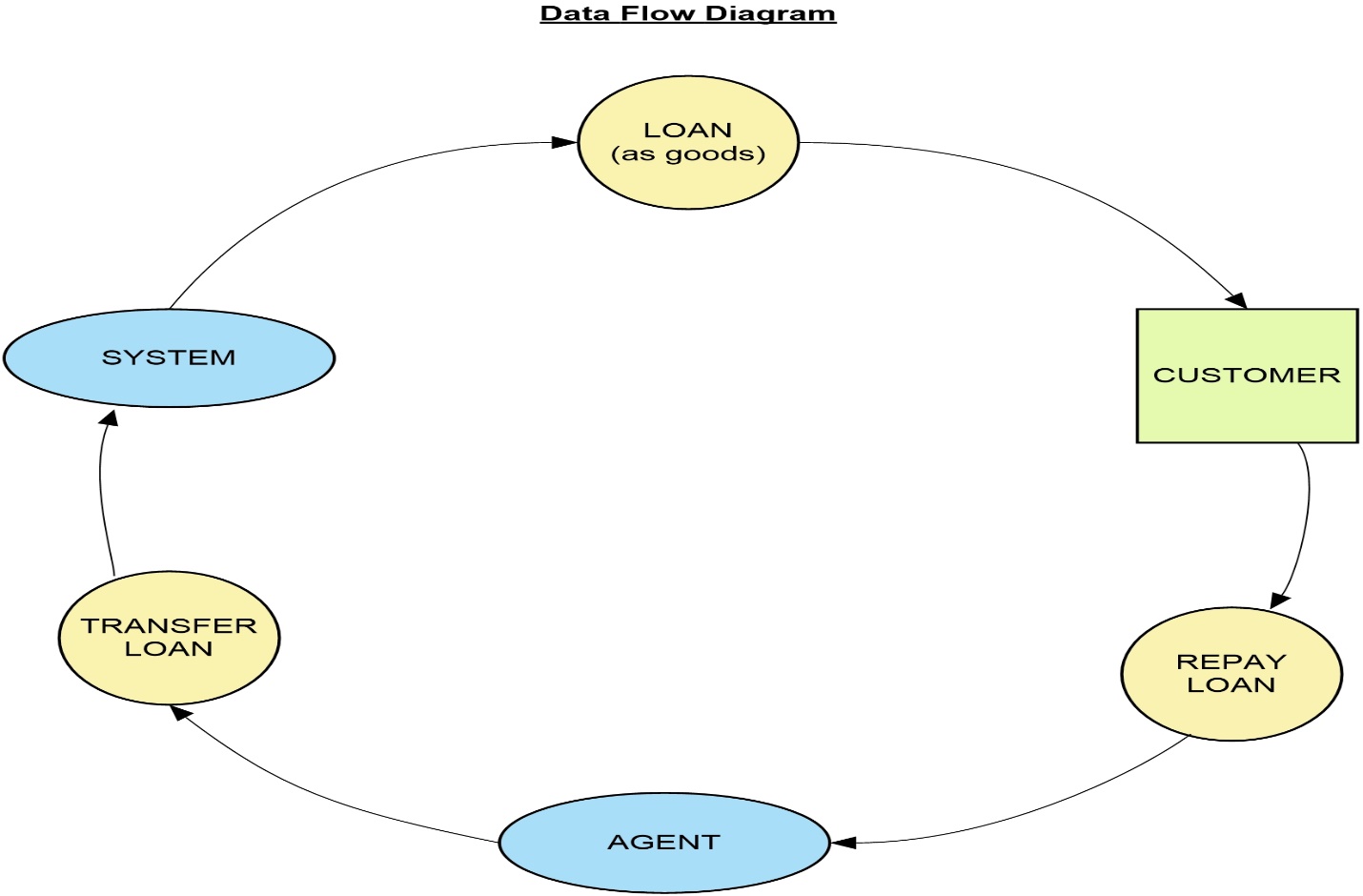
**SYSTEM ANALYSIS AND DESIGN OF OJAPAY EMBEDDED FINANCE SYSTEM**

This will focus on the practical approach to develop the application and plans to satisfy all mandatory objectives that includes software analysis methodologies before progressing to the final stage. It will also define the structure of the system, the development, and testing and integration of the new system.

1. CRITICAL ANALYSIS OF THE SYSTEM

This will be centered on specific components such as the Customers, Agents and the System itself and how they interact with each other. Some of the objectives that must be satisfied in the exploration of the new system are as follows

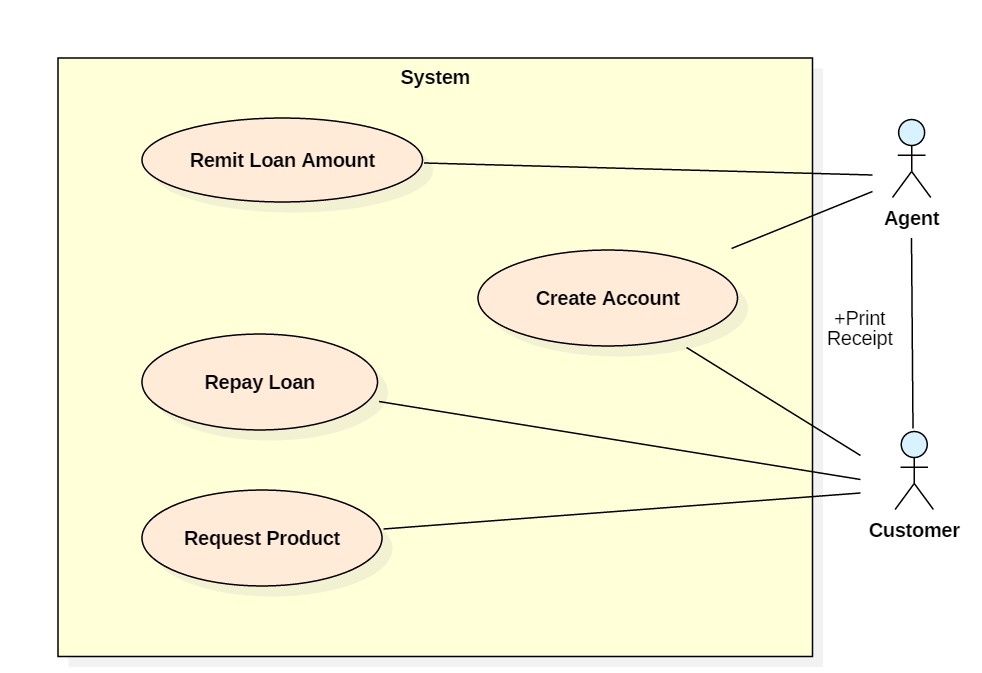
1. Customer
2. Creates an account on the System
3. Requests for loan facility on the System
4. Makes loan repayment on the System
5. Agent
6. Creates account on the System
7. Funds account on the System
8. Deposit customer repayment by crediting company account on the System
9. System
10. Limit transactions to amount in agent account
11. On Customer repayment of loan, System debits Agent account and reduce Customer debt by the credit receive and transfer credit to Company account.
12. Issue receipt of payment on repayment of loan to Customer.

****

*Figure 1.1 Context Diagram of the Flow of process for the Embedded Finance Project*

The figure 1.1 above shows a basic overview of the system. This is a high-level process showing the relationship between the company, agents, customers and processes (e.g. Loan, payment, repayment and funds remittance).

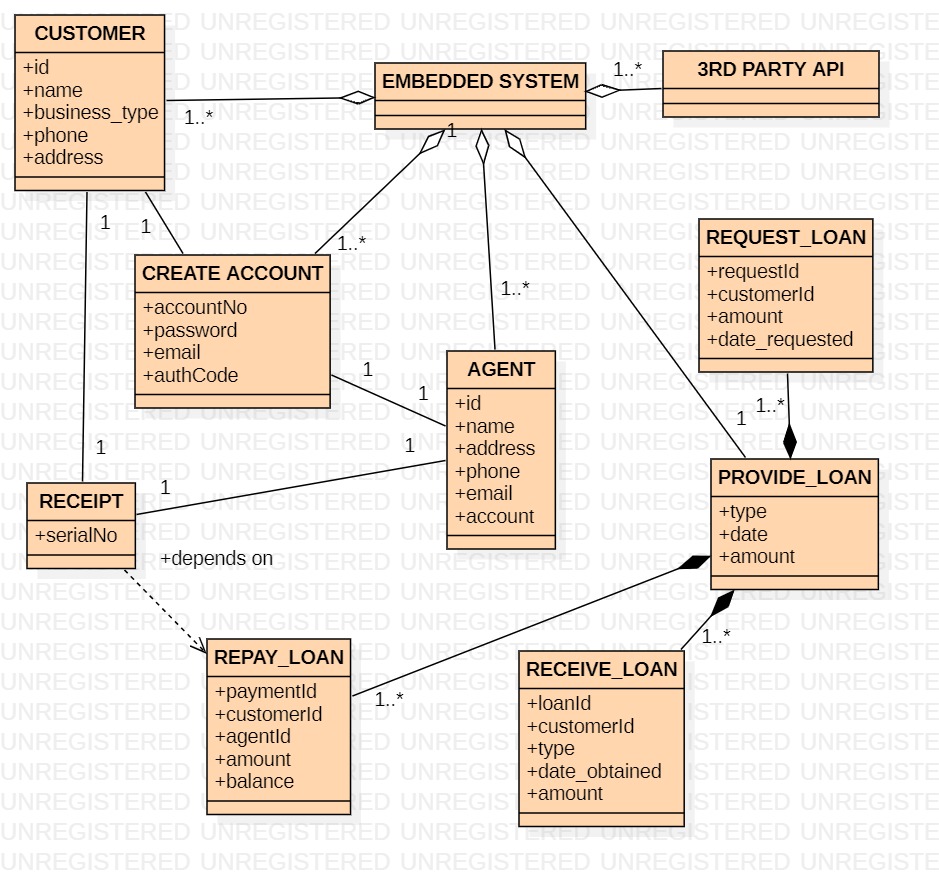
USE CASE DIAGRAM



*Figure 1.2. Use case diagram to illustrate interaction between System components*

In the figure 1.2 above, the system encapsulates the function and features of the application backend integration with third party APIs and financial institutions, and focuses on the interaction of the customer (end users) and agents with the system. This specifically narrows the functions of the system to loan acquisition and repayment processes.

CLASS MODELLING OF SYSTEM

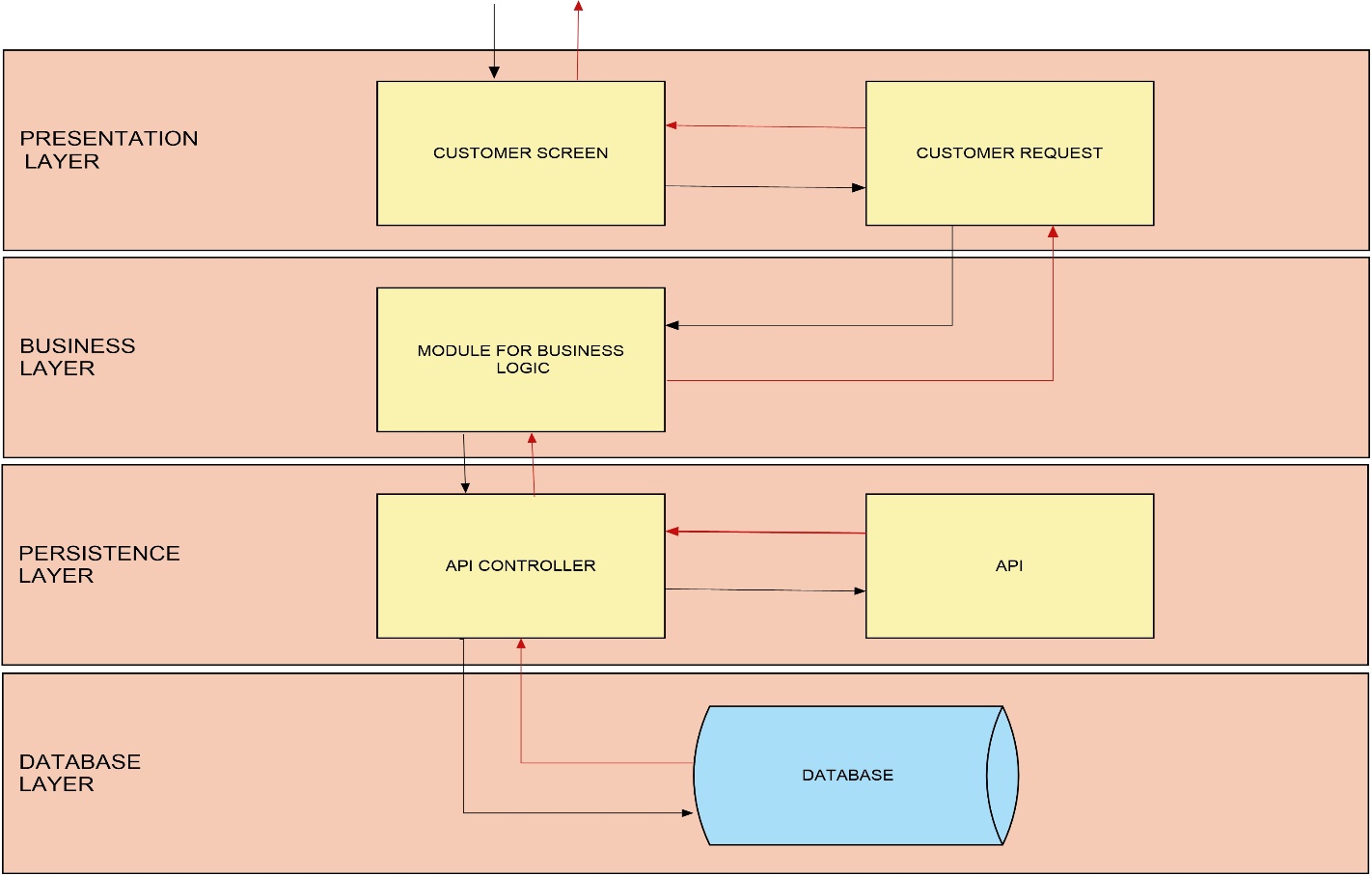


*Figure 1.3. The class modelling illustrates shows the relationship with all the components in the system*

This modelling gives more clarity to the interaction between the different components of the system. The “RECEIVE\_LOAN”, “REPAY\_LOAN” and “REQUEST LOAN” components are composite part of the “PROVIDE LOAN” meaning that their existence is dependent on the provision of loan facility in the system. Receipt processing will only be required when there is repayment of loan. The AGENT component interacts with the system by creating an account and depositing money to AGENT account through the 3RD PARTY API subsystem. The CUSTOMER makes repayment of loan through the AGENT. The AGENT then remits loan repayment through the subsystem into the EMBEDDED SYSTEM.

1. SYSTEM DESIGN

The design for the embedded finance system borders on the architecture (MVC – model, view, controller architecture or MVT – model, view, template architecture), modules (classes and functions) and components (libraries), and the different interfaces (APIs, REST, client/server paradigm) of those components and the data (database information – MySQL/postgreSQL) that goes through the system.

SYSTEM ARCHITECTURE USING LAYERED ARCHITECTURE

*Figure 1.4. Illustrating the Layered architecture*

Using the layered architecture patterns, four major layers are used namely, presentation, business, persistence, and database layers. These layers are isolated from the other so that changes made in one layer of architecture does not affect components in other layers. The embedded finance system will be designed with this architecture as follows.

1. Presentation layer:

This is the user interface and browser logic for the system. This will cover the UI/UX design and front-end design logic that interfaces with the user and is concerned with only getting user data or displaying information in a specific front-end design format (Adobe XD, Figma, HTML, CSS, JavaScript, JQuery, Bootstrap, FontAwesome).

1. Business layer:

This layer is responsible for executing specific business rules associated with the user request (e.g. account registration and loan request). It focuses on getting data from the presentation layer, perform business logic against the data and pass the information to the persistence layer.

1. Persistence layer:

This layer is responsible for models that interact with the database directly and other external server requests (Application Programming Interface – API). Changes made in the persistence layer will impact both the business layer and the presentation layer.

1. Database layer:

The database layer architecture will be responsible for storing data into the database for the system. This also involves retrieving information from the database for computation or other processes before the result is finally sent to the presentation layer as output.

SOFTWARE DEVELOPMENT ARCHITECTURE

The software development architecture for the embedded finance system will be the MVC (Model, View and Controller) software architecture design pattern.

1. Model: This interfaces with the database and is responsible for maintaining data. It is the logical data structure behind the application.
2. View: This is used for the UI (User Interface) logic of the application displayed in a web browser.
3. Controller: This is the interface between the Model and the View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the View component to render the final output.

In this system, we will be using Django, a web framework of Python Software programming language that follows the MVC architecture. In Django, the Controller component is inbuilt and only the Model and View is required when developing a software application.

SYSTEM TESTING AND INTEGRATION

There will be quality control testing of the system against expectations in order to assess the ability of the system to integrate with software related to the system such as databases and external libraries and APIs. The testing will be iterative in order to cater for concerns raised and to check new approaches for areas requiring further development. The two types of testing are User Acceptance Testing (UAT) and System Integration Testing (SIT).

1. User Integration Testing:

This will introduce the system to end users or clients or internal members for testing in order to accept the system before moving it to the production environment.

1. System Integration Testing:

When the embedded finance software system is completed, the whole system is tested by its component sub-systems (e.g. frontend, database, requests, and processes) in order to ascertain that all the component dependencies are functioning properly and that data integrity is maintained between these components of the system.

DEPLOYMENT PLAN

The various environment for deploying the software system within the different phases of completion include the following.

1. Local environment
2. Development environment
3. Staging environment
4. Live environment
5. Local Environment: This environment is independent of hosting as it is kept offline and runs on a local machine
6. Development Environment: This is a test server to test new features and changes. This can also be local environment.
7. Staging Environment: This is the last step before going into production and making it visible on the live site. This stage ensures everything is working fine and all bugs have been eliminated before going live.
8. Live Environment: this is the production environment. This final deployment is done when all stages are satisfactory and it is time to proceed to the live website.

DEPLOYMENT MANAGEMENT

1. GITHUB:

All software source codes will be deployed to Github services for version control using git, task management and continuous integration.

1. DigitalOcean:

DigitalOcean is a cloud computing infrastructure provider that is used to build, test, manage and scale software applications. This platform will be used in the software system deployment.

SOFTWARES

1. UI/UX – AbodeXD, Figma
2. Frontend development – (HTML5, CSS, Bootstrap, JavaScript, JQuery, FontAwesome)
3. Backend development – Django/Python
4. Database design and management – (MySQL, PostgreSQL)
5. WebOps – SSH, git, postman, filezilla, putty, Linux subsystem.