**CHAPTER THREE**

**SYSTEM DESIGN**

1. **Introduction**

System design is the transformation of the analysis model into a system design model. System design is the first part to get into the solution domain in a software development. This chapter focuses on transforming the analysis model into the design model that takes into account the non-functional requirements and constraints described in the problem statement and requirement analysis sections discussed earlier.

The purpose of designing is to show the direction how the system is built and to obtain clear and enough information needed to drive the actual implementation of the system.

* 1. **Design goals**

The objectives of design are to model the system with high quality. The design goals are derived from non-functional requirements that means non-functional requirement is the description of the feature characteristics and attribute of the system as well as any constraints that may limit the boundary of the proposed solution.

Design goals describe the qualities of the system that the developers should consider. Generally there are many design goals exist in the project. Some of them are:-

* **Availability:** The system should be available for any valid users at any time at any place.
* **Security:** WBCSIBGRSAB system should be secured, i.e., not allow other users or unauthorized users to access data that has no the right to access it.
* **Modifiability:** WBCSIBGRSAB system should be modifiable for further modification and enhancement of the application like government placement criteria, add new functionality, portable to different platforms.
* **Performance**: The system should respond fast with high throughput, i.e. it should perform the task quickly possible as possible such as allocating employees and customers, viewing customer’s information.
* **Dependence:** The system should have ability to avoid service failures in the presence of mistakes.
* **Cost**: The system should be developed with minimum cost possible. In reality there is always trade-offs or disadvantages and therefore from its previous experience the institution prefers to invest more on development cost than maintenance cost to minimize bugs which may appear at the later stage.
* **Usability**: - The system should have simple and understandable graphical user Interface such as forms and buttons, which have descriptive names. It should give reliable response for each user request at least before the session expires. All the interfaces, forms and buttons are written or designed in a simple language or common language so that the user can access it without any difficult.
  1. **Proposed System Architecture**

The proposed system is expected to replace the existing manual system by web based credit and saving system for Benishangul Gumuz regional state in Assosa branch which is the software architecture used for the system is repository architecture because subsystems access and modify data from a single data structure which is called the central repository. This architecture allows different user of the system access the data from center database server.

The central repository of the proposed system is MYSQL database server which is every data related with the system is stored.

The proposed subsystem will be implemented in **Client/Server architecture.** Wherever a user is as long as there is Internet connection he/she can browse the system page, fill the required inputs by the web page, and then submit it then the request of the user will be sent to the central server.

The server will give response based on the user request. Clients include customer service officer, customer and manager.

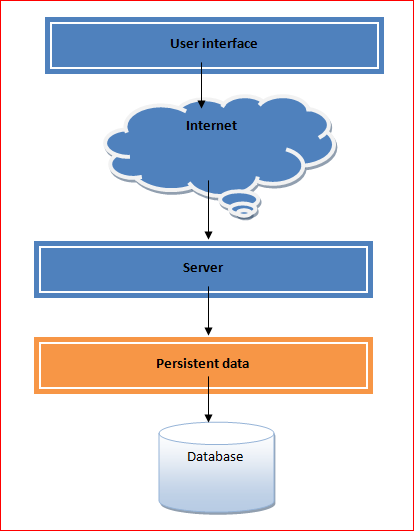


Fig 3.1 the overall architecture of the system

* + 1. **Subsystem Decomposition**

Subsystem decompositions will help us to reduce the complexity of the system. Subsystem decompositions will help reduce the complexity of the system. The subsystems can be considered as packages holding related classes/objects.

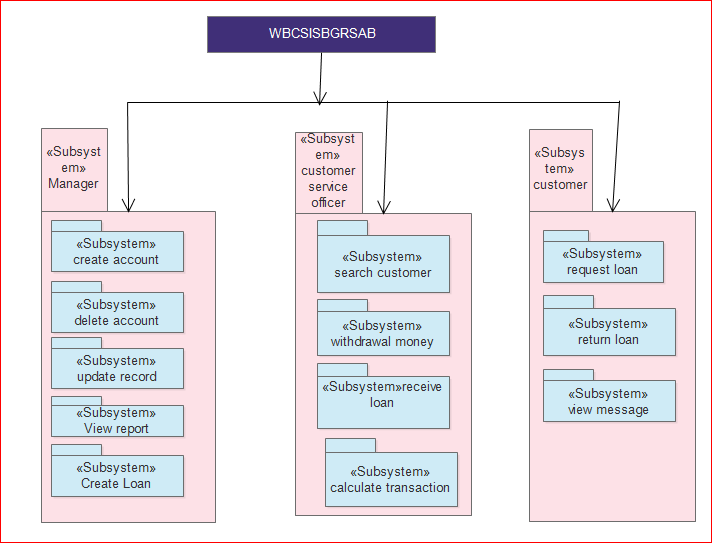
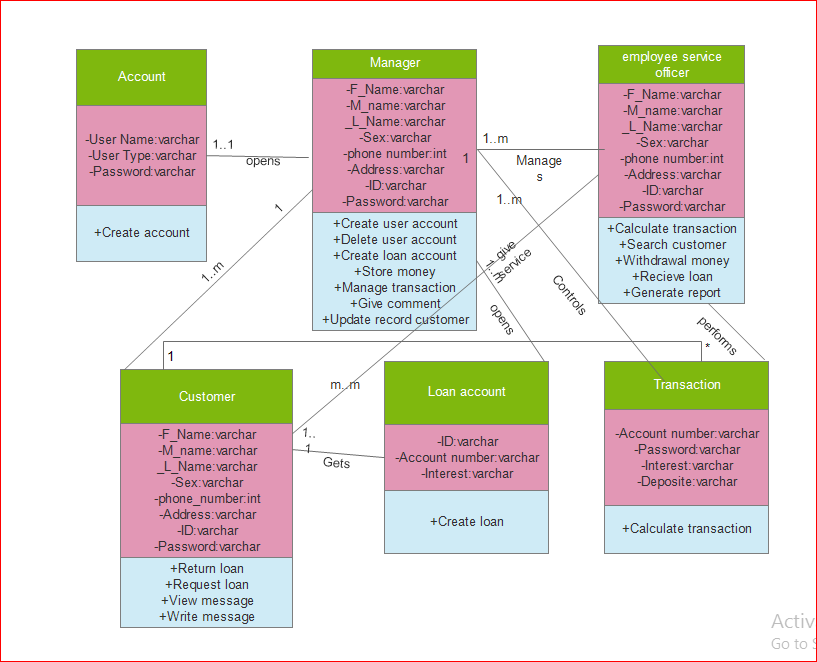


Figure3.2. Subsystem decomposition system

* 1. **System Class Diagram**

The class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The classes diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams which can be mapped directly with object oriented languages.

The class diagram shows a collection of classes, interfaces, associations, collaborations and constraints. Class diagram represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.Figure 3.3 System class diagram

* 1. **State Chart Modeling**

A state chart diagram is a view of a state machine that models the changing behavior of a state. State chart diagrams show the various states that an object goes through, as well as the events that cause a transition from one state to another.

The common model elements that state chart diagrams contain are:

* States
* Start and end states
* Transitions

A state represents a condition during the life of an object during which it satisfies some condition or waits for some event. Start and end states represent the beginning or ending of a process.

State chart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

States are represented as a rounded rectangle with the name of the state show in the system.

Connecting state together are transitions. These represent the events that cause the object to change from one state to another.

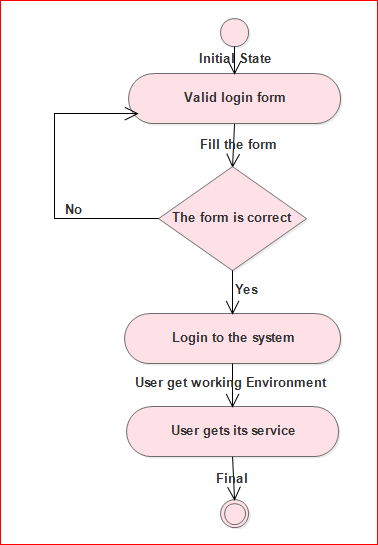
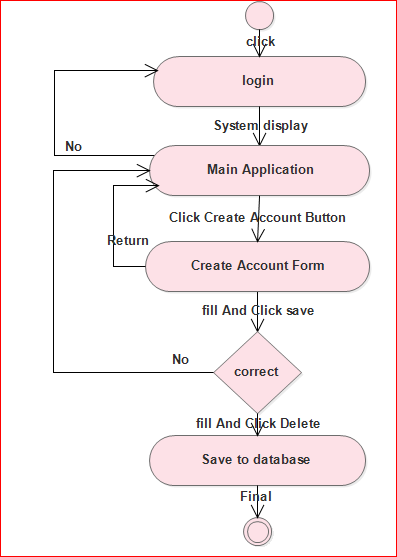


Figure 3.4 State chart diagrams for login

Figure 3.5 State chart diagrams for create account

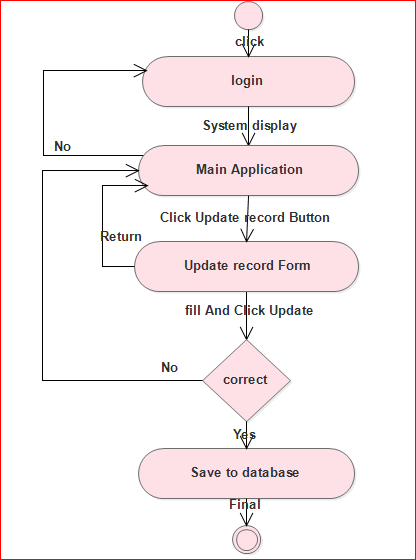


Figure 3.6 State chart diagrams for create loan account

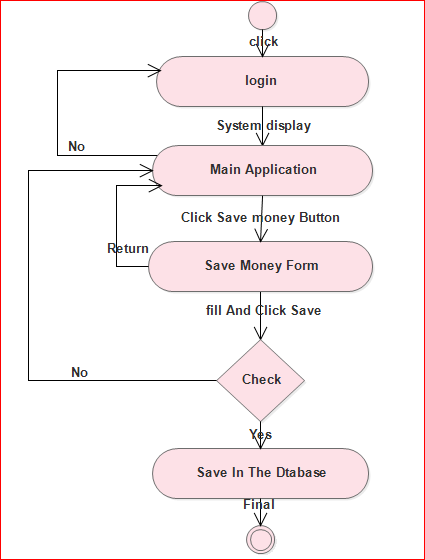


Figure 3.7 State chart diagrams for save money

**3.6. Collaboration Diagram**

The collaboration diagram is an interaction diagram emphasized the structural organization of the object that participates in an interaction.

It consists of collection of objects that work together to perform a task. In the class diagram modeling the static nature of the system shown .Whereas the collaboration diagram shows the message flow between objects in an oriented are developed in the following manner.

* The rectangle represents the various objects involved that make up the application.
* The line between the classes represent the relationship(association, aggregation , composition dependency between them )
* The text along the line is for invoking message on the collaboration diagram.

The collaboration diagrams are drawing completely based on the sequence diagrams in the analysis phase.

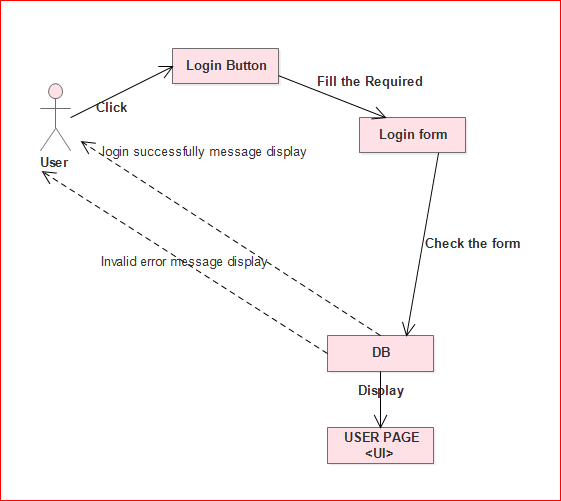


Fig 3.8 collaboration diagram for login

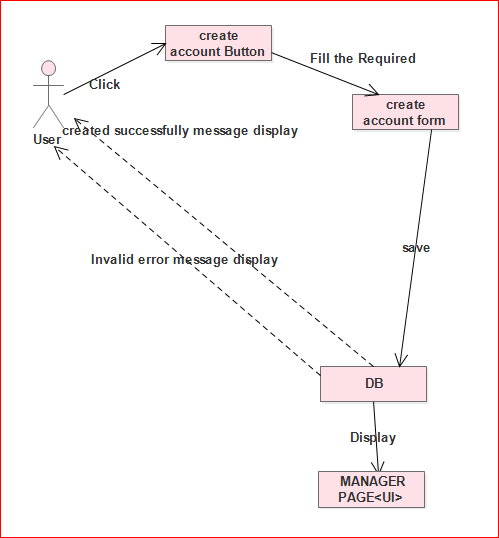


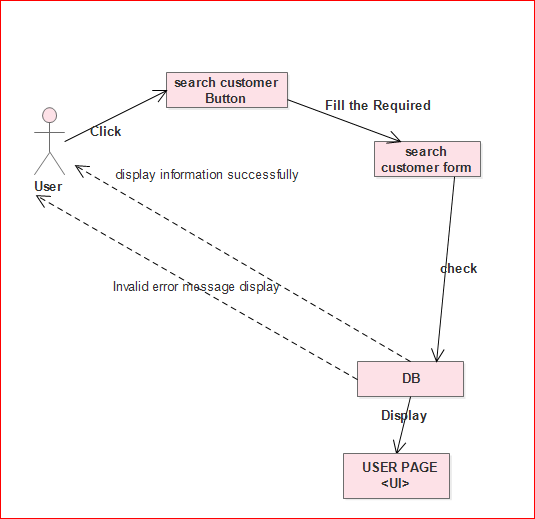
Fig 3.9 collaboration diagram for create account

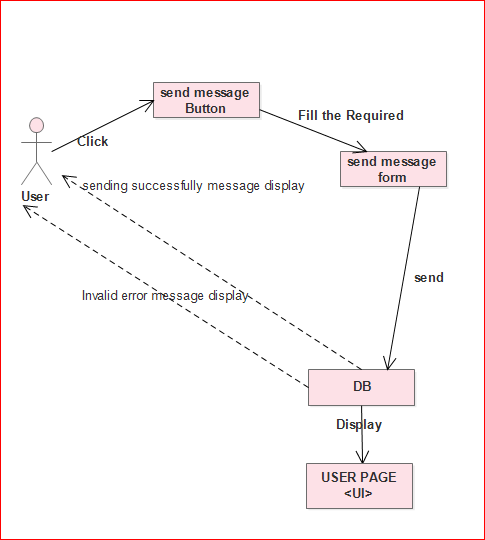
Fig3.10 collaboration diagram for search customer

Fig 3.11 collaboration diagram for send message

3.7. **Entity Relationship Diagram (ERD)**

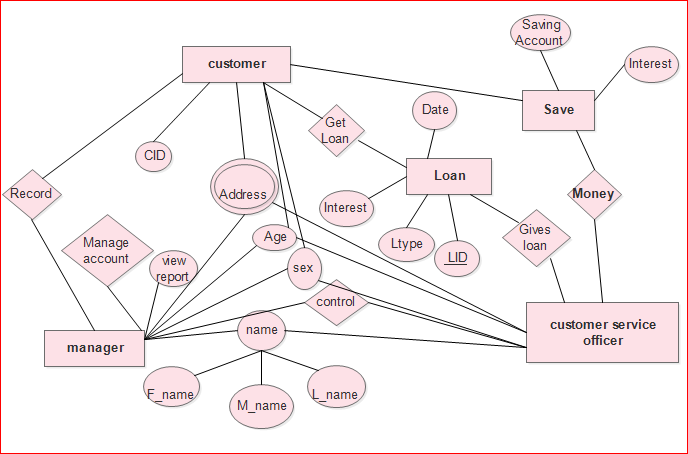
****

Figure 3.22 Entity relationship diagram

**3.8. Persistent Data Management**

In this section the team describes how the persistent data stored by the system and the data management infrastructure required for it. The system will use the **MY SQL** database server for storing data. This will allow the database to be easily integrated with and accessed by the rest of the system. The database will contain customer information (name, password etc.), and also retain configuration data such as authorized administrator. Each of these items will be store in a separate table.

As described in the system contain seven tables which are stored in MYSQL server. These tables are:-

Customer table: a table which store customer’s information.

Create user account table: a table which store user account.

Manager table: a table which contain manager’s information.

Customer service officer: a table which contain customer service relation information.

Transaction table: a table which contain transaction information.

Save table: a table which contains information about saving money.

Loan table: a table which contain information about loan.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field\_Name** | **Data type** | **visibility** | **Description** | **Key** |
| Manager\_id | Vrachar | **+** | Not null | Primary |
| First name | Varchar | **-** | Not Null |  |
| Middle name | Varchar | **-** | Not Null |  |
| Last name | Varchar | **-** | Not Null |  |
| Age | Int | **-** | Not Null |  |
| Sex | Varchar | **-** | Not Null |  |
| Address | Varchar | **-** | Null |  |
| Phone number | Int | **+** | Null |  |

Table: 3.3 manager table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field\_Name** | **Data type** | **Visibility** | **Description** | **Key** |
| Customer service officer\_id | Vrachar | **+** | Not null | Primary |
| First name | Varchar | **-** | Not Null |  |
| Middle name | Varchar | **-** | Not Null |  |
| Last name | Varchar | **-** | Not Null |  |
| Age | Int | **-** | Not Null |  |
| Sex | Varchar | **-** | Not Null |  |
| Address | Varchar | **-** | Not Null |  |
| Phone number | Int | **+** | Null |  |
| Manager\_id | varchar | **+** | Not null | Foreign key |

Table: 3.3 customer service table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field\_Name** | **Data type** | **Visibility** | **Description** | **Key** |
| Customer\_id | Vrachar | **+** | Not null | Primary |
| First name | Varchar | **-** | Not Null |  |
| Middle name | Varchar | **-** | Not Null |  |
| Last name | Varchar | **-** | Not Null |  |
| Age | Int | **-** | Not Null |  |
| Sex | Varchar | **-** | Not Null |  |
| Address | Varchar | **-** | Not Null |  |
| Phone number | Int | **+** | Not Null |  |
| Customer service officer\_id | Varchar | **+** | Not null | Foreign key |

Table: 3.3 customer table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field\_Name** | **Data type** | **visibility** | **Description** | **Key** |
| Loan\_id | Vrachar | **+** | Not null | Primary |
| Loan type | Varchar | **-** | Not Null |  |
| Amount | Varchar | **-** | Not Null |  |
| Date | Date | **-** | Not Null |  |
| Customer service officer\_id | Varchar | **-** | Not Null | Foreign |

Table: 3.3 loan table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field\_Name** | **Data type** | **visibility** | **Description** | **Key** |
| Saving id | Vrachar | **+** | Not null | Primary |
| Saving amount | Varchar | **-** | Not Null |  |
| Date | Date | **-** | Not Null |  |
| Customer service officer\_id | Id | **-** | Not Null | Foreign |

Table: 3.3 saving table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field\_Name** | **Data type** | **visibility** | **Description** | **Key** |
| Transaction\_id | Vrachar | **+** | Not null | Primary |
| Deposit | Varchar | **-** | Not Null |  |
| Interest | Varchar | **-** | Null |  |
| Customer service officer\_id | Varchar | **-** | Not Null | Foreign |

Table: 3.3 transaction table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field\_Name** | **Data type** | **visibility** | **Description** | **Key** |
| User type | Vrachar | **+** | Not null | Primary |
| User name | Varchar | **-** | Not Null |  |
| Password | Varchar | **-** | Not Null |  |

Table: 3.3 create account table

**3.9. Component Diagram**

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

So from that point component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams are used to represent the whole.

So the purpose of the component diagram can be summarized as:

* Visualize the components of a system.
* Construct executables by using forward and reverse engineering.
* Describe the organization and relationships of the components.

Generally components are considered autonomous, encapsulated unites with in a system or subsystem things that will typically be implemented using replaceable models.

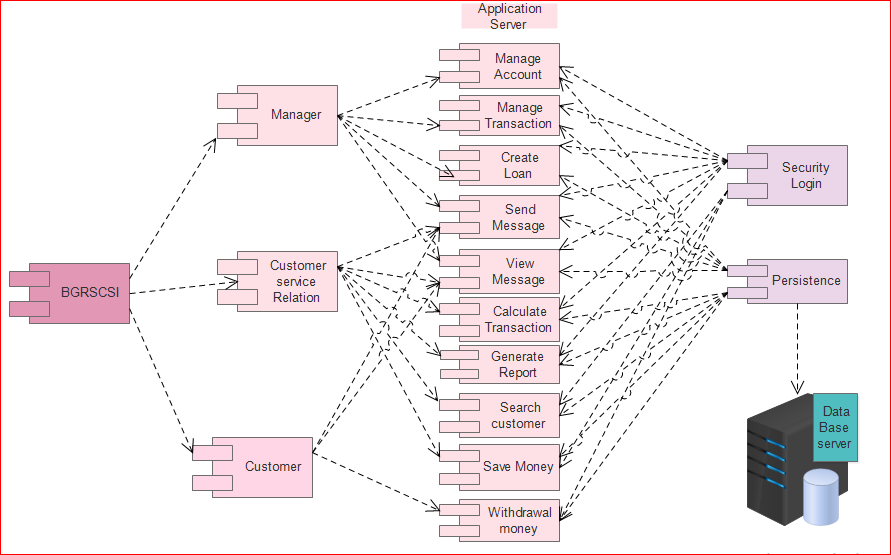


Figure 3.24 Component diagram

**3.10. Deployment Diagram (Hardware or Software Mapping)**

The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related.

Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.

UML is mainly designed to focus on software artifacts of a system. But these two diagrams are special diagrams used to focus on software components and hardware components.

So most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on hardware topology of a system. Deployment diagrams are used by the end users.

The purpose of deployment diagrams can be described as:

* Visualize hardware topology of a system.
* Describe the hardware components used to deploy software components.
* Describe runtime processing nodes.

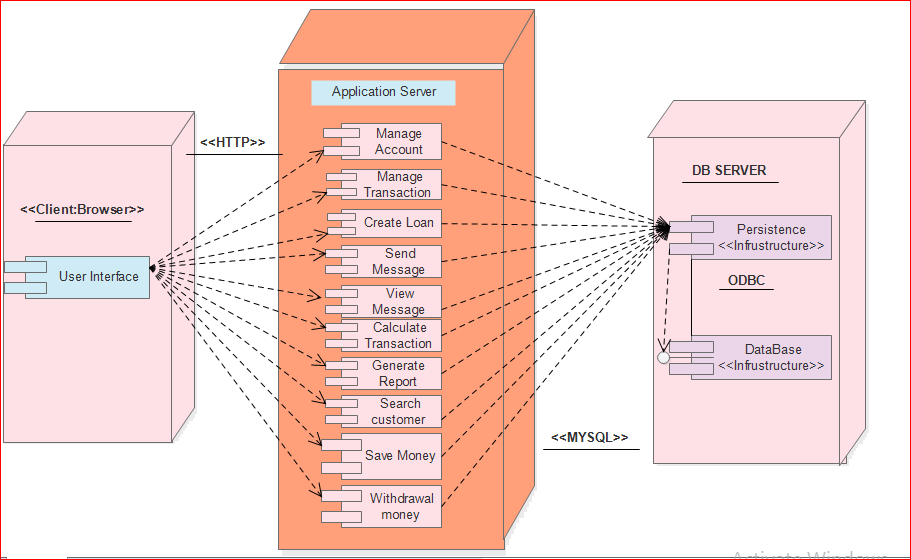


Figure 3.25-deployment diagram