**Q1. Draw a Use Case Diagram**

A customer can make a payment either by Card or by Wallet or by Cash or by Net banking

**Q2. Derive Boundary Classes, Controller classes, Entity Classes.**

**3-Tier Architecture:**

3-Tier Architecture there will be Business Logic written in Business logic layer, having constraint to proceed input depend upon logic and gives output. From each and every Use case, going to extract classes. For ex: In ATM, user can only some amount of cash, this logic has been written in the business logic layer. After UCD+UCDD, team will start building Software and For that MVC Architecture will going to use

MVC Architecture:

1. **Boundary class**:

Actor Who has involved in this Use case, He may perform some **action** or support some action. The Kind of Action user will perform or initiate some action will be represented by **Presentation layer or Application Layer** of an application and known as **Boundary class**.

Symbol:

Boundary Class

1. **Controller Class:**

The specific **Actor** will perform some Action Based on some functionality like Login, Registration or Withdraw cash are the functionalities will be represented in **Controller layer** or in **Business Logic Layer** in middle of the 3 tier architecture or known as **Controller Class.**

Symbol**:**

****

1. **Entity Class**:

In the Use Case perspective, **As an Info of an** **Actor** for person or user his name, profession, person\_id, Mobiile\_num this information has to be recorded by means of Modelling. The Model object knows all about the data that need to be displayed and represented by **Entity Class or Form**

**Class**. **Details of the particular actor who has involved.** It will be in **Data Layer.**

Symbol:

1. **Boundary Classes**

**PayByCashBC()**

**PayByNBBC ()**

**PayByCardBC()**

**AgntMkPayBC()**

1. **Controller Class:**

****

**PayCC()**

1. **Entity Class**:

**CashEC ()**

**NBEC()**

**CardEC()**

**AgentEC()**

**Q3. Place these classes on a three tier Architecture.**

There will be the guideline to put all classes in 3 tier architecture

1. Place all entity classes in DB Layer
2. Place Primary Actor associated Boundary Class in Application Layer
3. Place controller class in Application Layer
4. If Governing Body influence or reusability is there with any of remaining Boundary classes, place them in Business logic Layer else place them in Application layer

**PayByNBBC ()**

**PayByCardBC()**

**PayByCashBC()**

**AgntMkPayBC()**

Application Layer

****

**PayCC()**

Business Logic Layer

**CashEC ()**

**NBEC()**

**CardEC()**

**AgentEC()**

Data Layer