Name:-Atharva Gangrade
Enrollment No: EN19CS305014

Software Engineering

Course Code:-
CS3CO26

BY
Atharva Gangrade
EN19CS305014

Submitted to
Mr. Preetesh Purohit

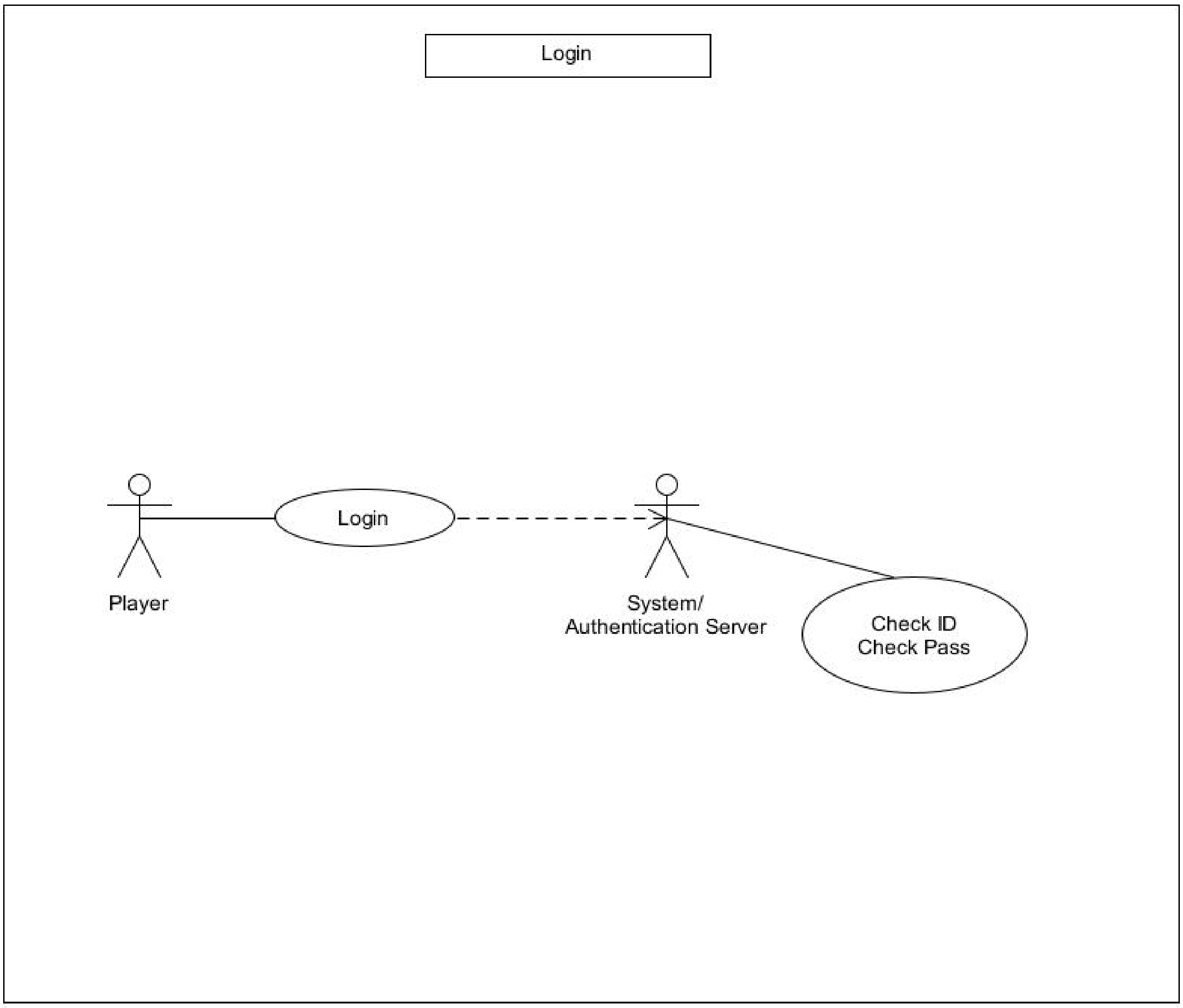
Department of Computer Science & Engineering Faculty of Engineering
MEDI-CAPS UNIVERSITY, INDORE-453331

Name:-Atharva Gangrade
Enrollment No: EN19CS305014

INDEX

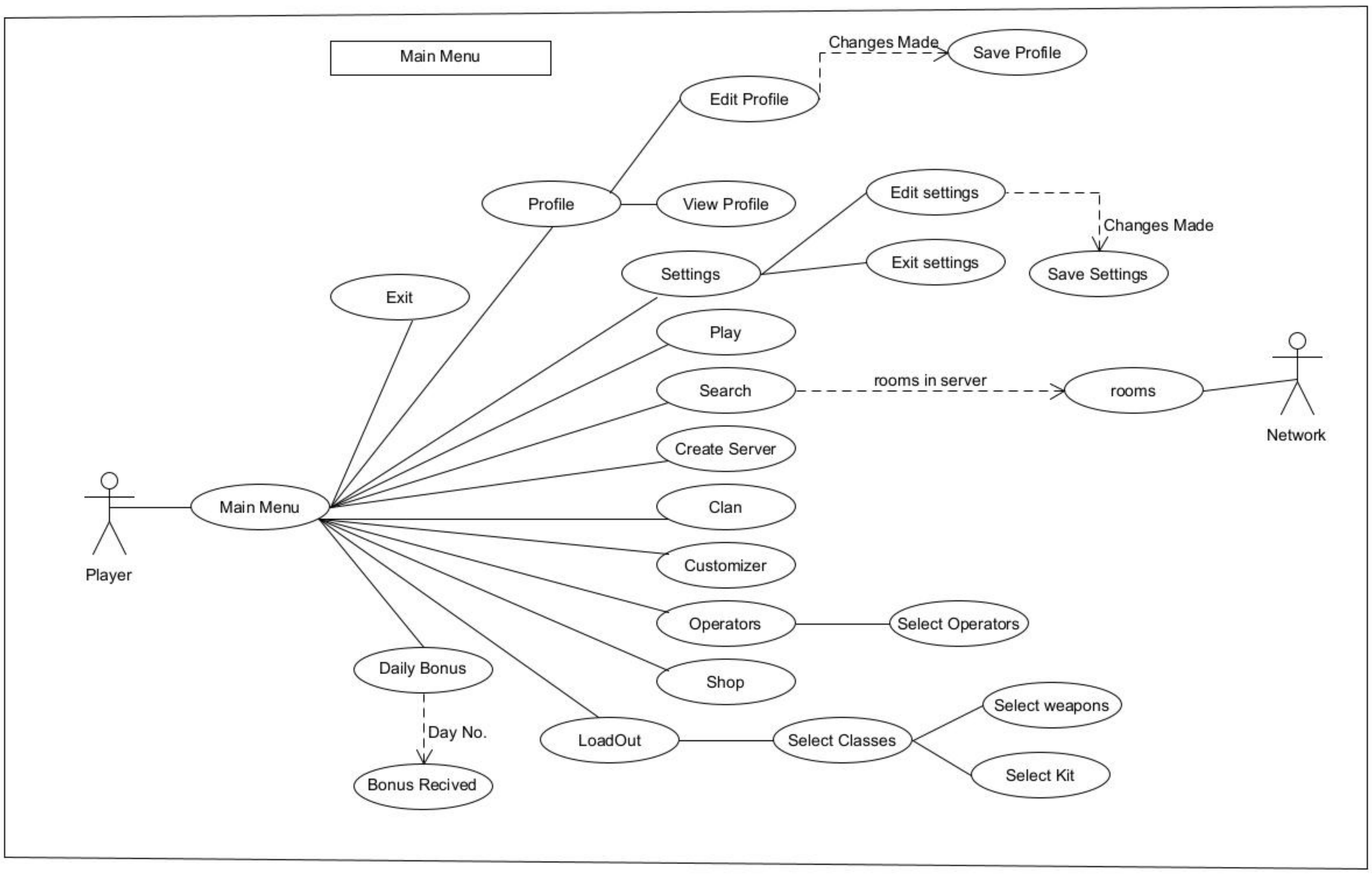
Name:-Atharva Gangrade
Enrollment No: EN19CS305014

Use Cases
Use Case Diagrams are used to illustrate the functioning of a system or a component of a system. They are commonly used to demonstrate the system's functional needs and interaction with external agents (actors). A use case is essentially a graphic that depicts several scenarios in which the system may be utilised. A use case diagram provides a high-level overview of what the system or a component of the system performs without delving into implementation specifics.
Login:



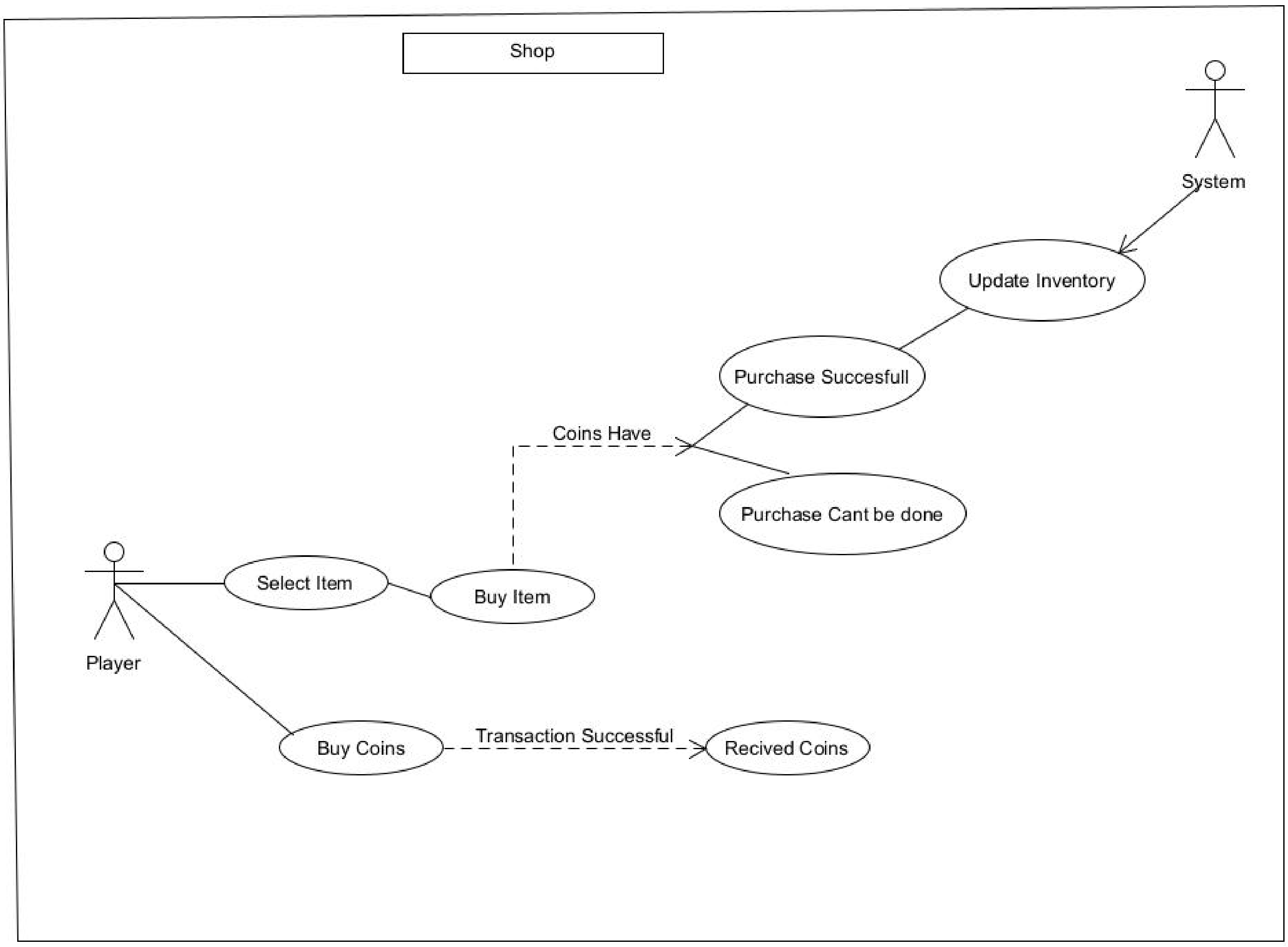
Name:-Atharva Gangrade
Enrollment No: EN19CS305014

Main Menu:



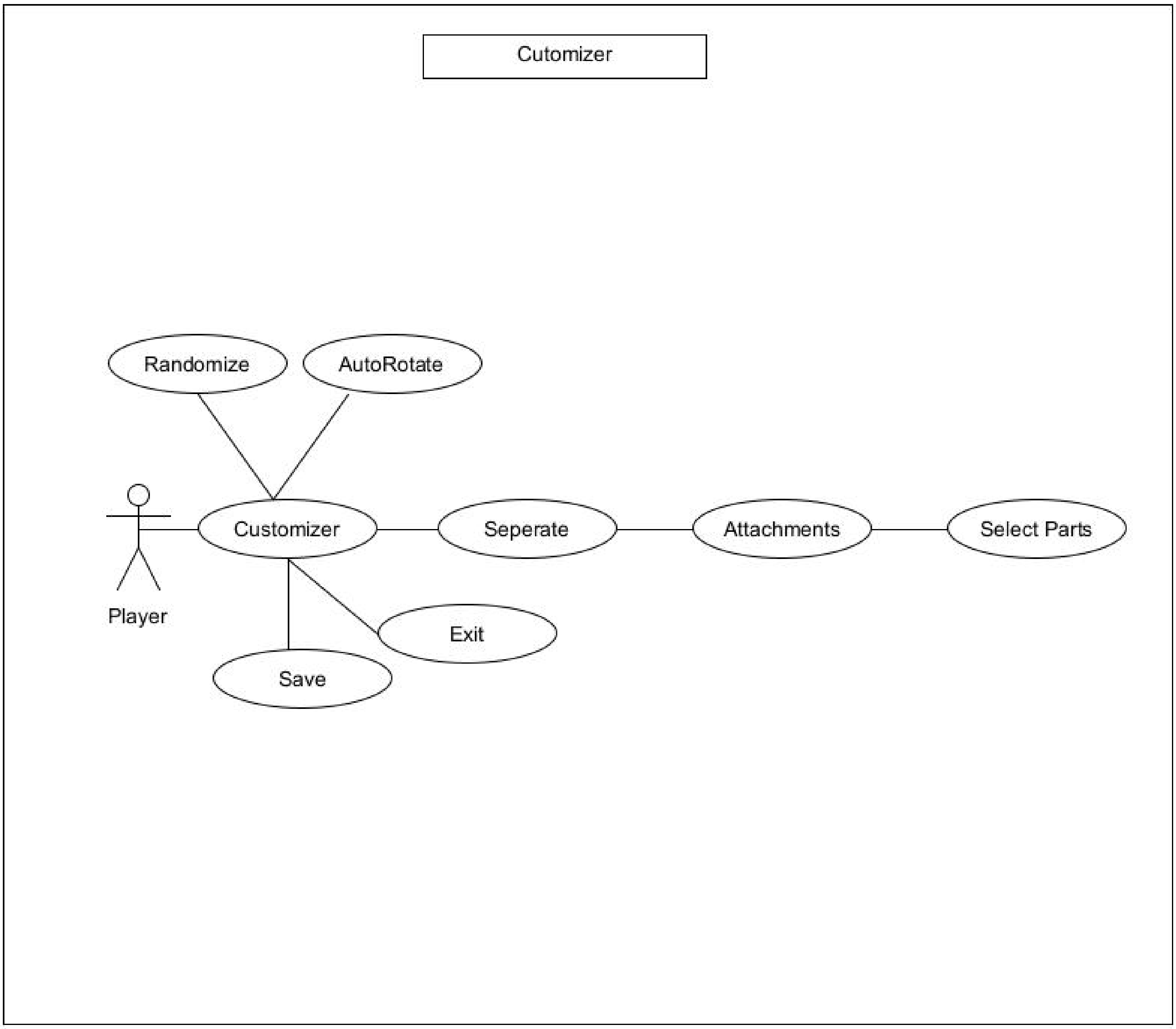
Name:-Atharva Gangrade
Enrollment No: EN19CS305014

Shop:



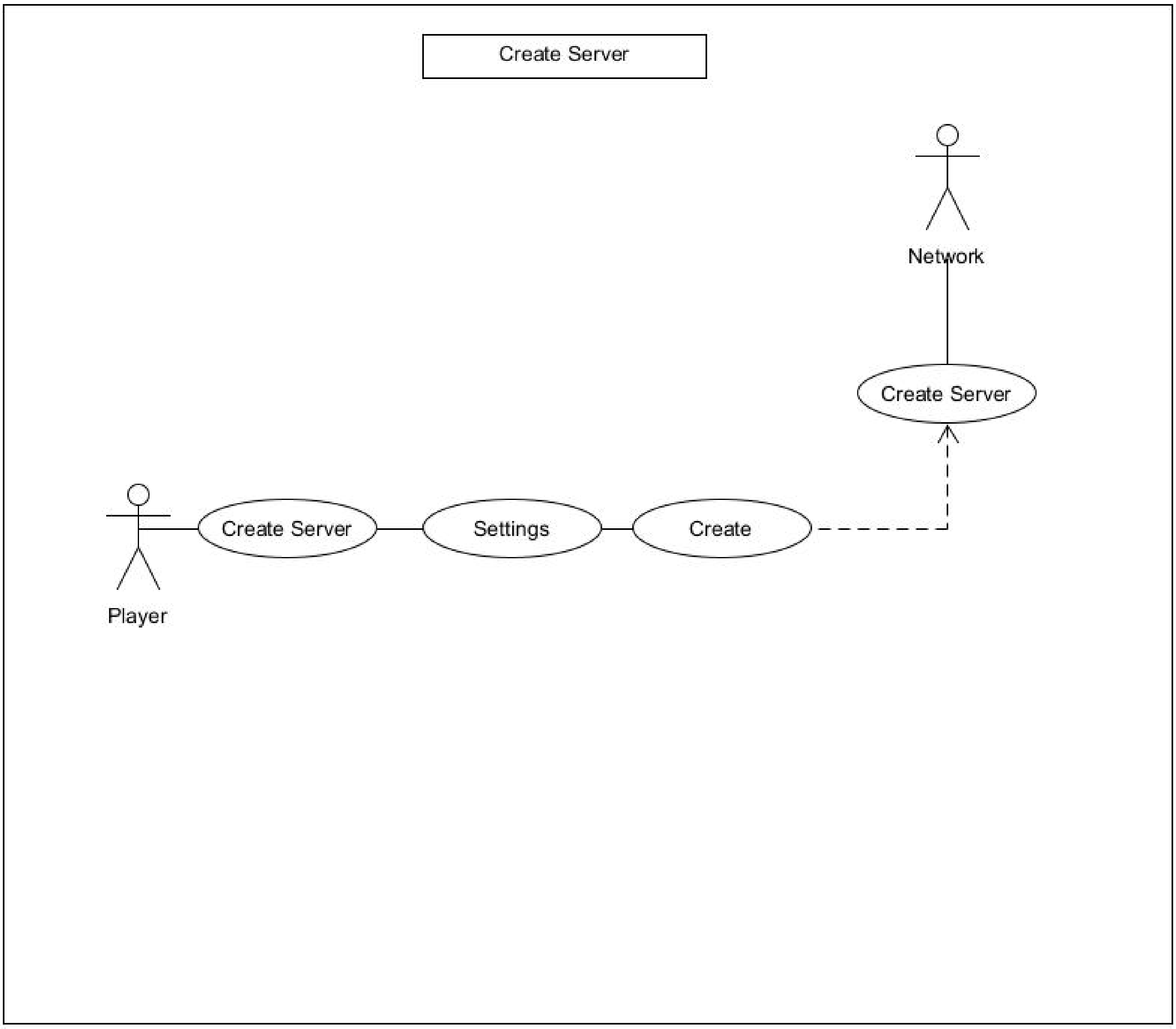
Name:-Atharva Gangrade
Enrollment No: EN19CS305014

Customizer:



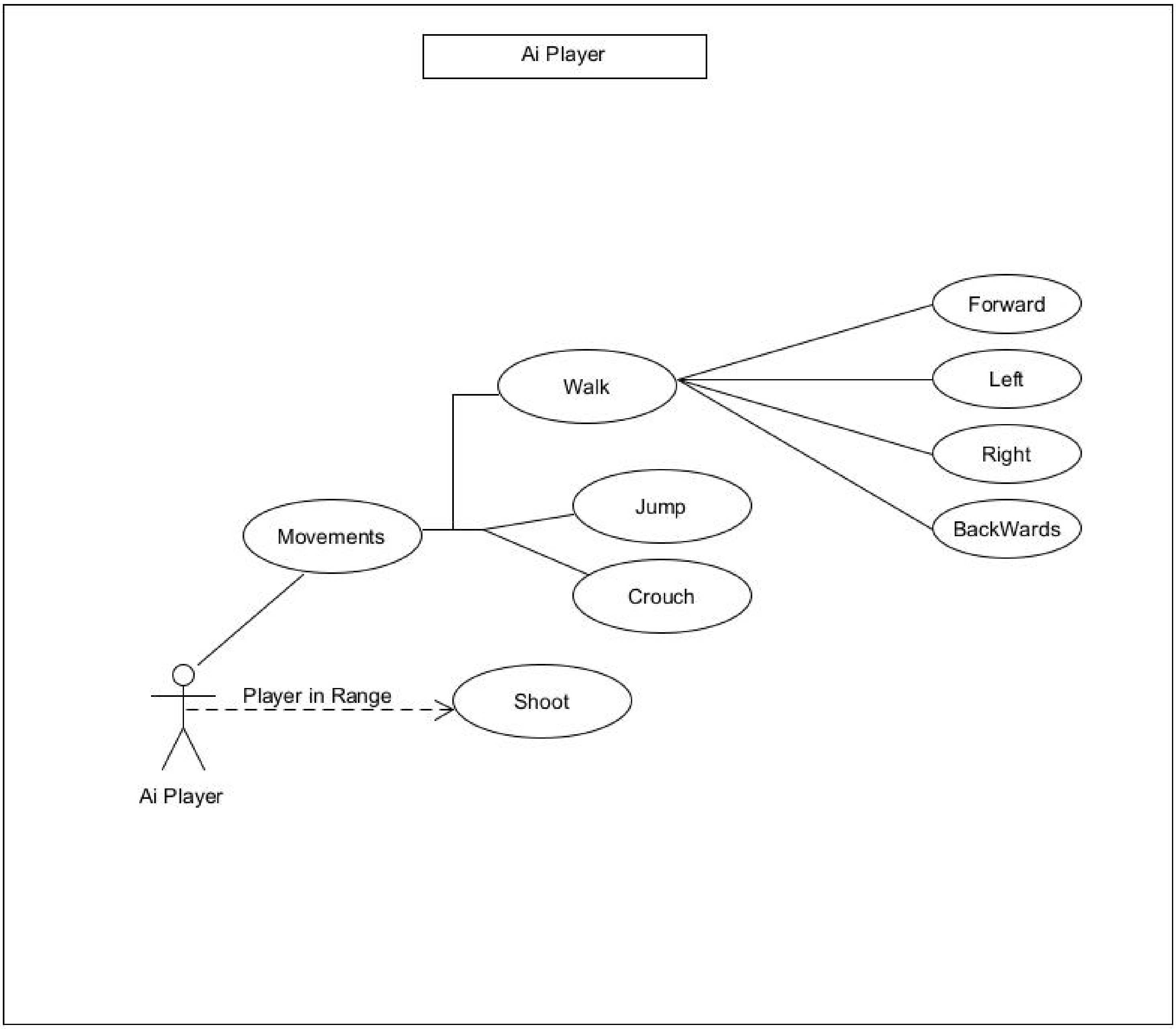
Name:-Atharva Gangrade
Enrollment No: EN19CS305014

Create Server:

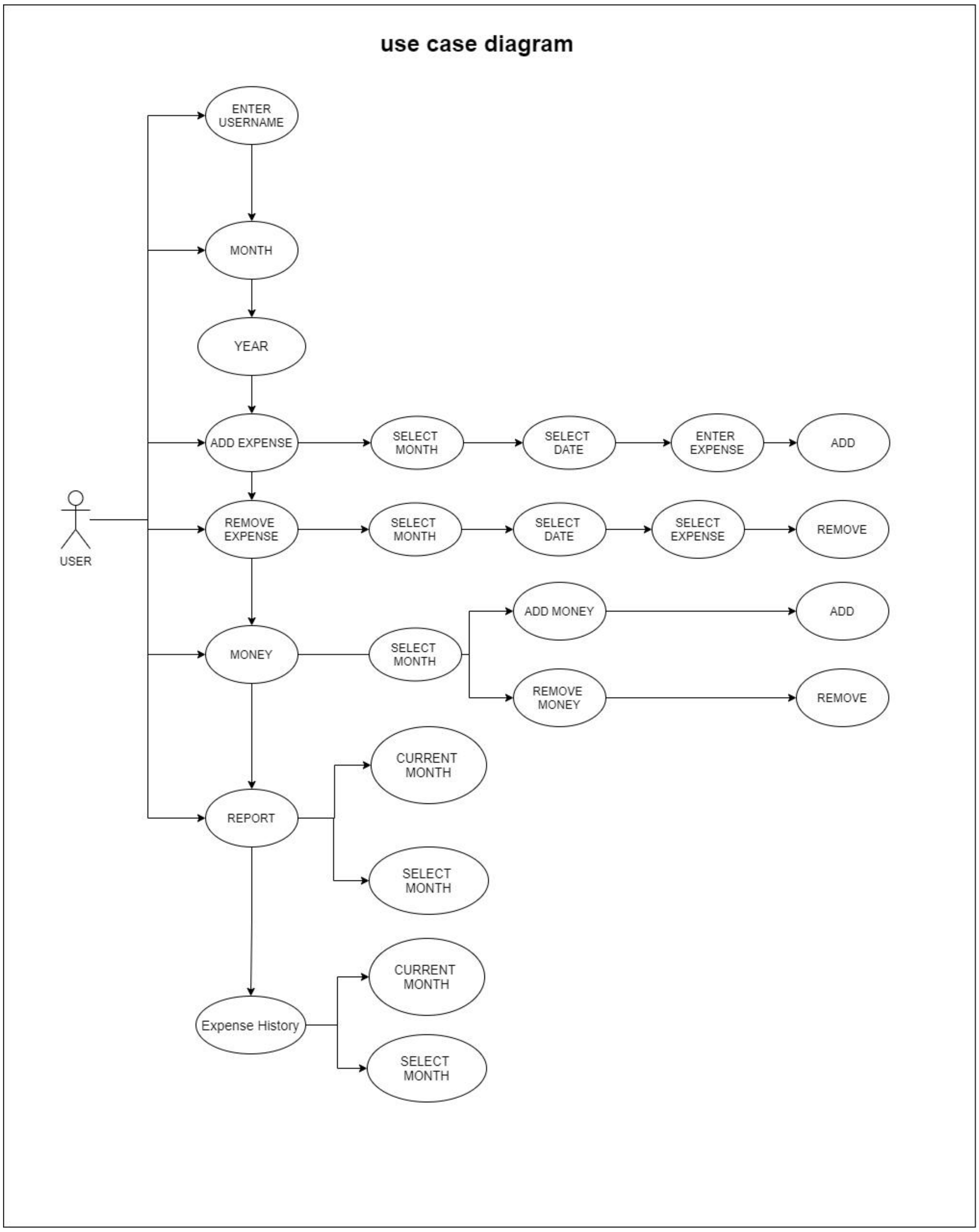


Name:-Atharva Gangrade
Enrollment No: EN19CS305014

AI Player:



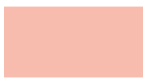
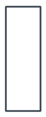
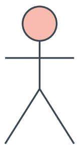
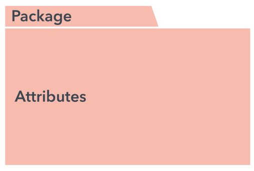
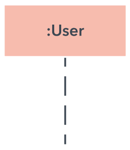
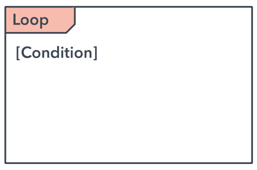
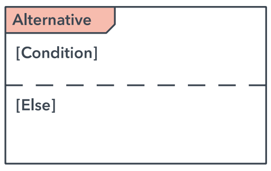
Name:-Atharva Gangrade
Enrollment No: EN19CS305014



Name:-Atharva Gangrade
Enrollment No: EN19CS305014

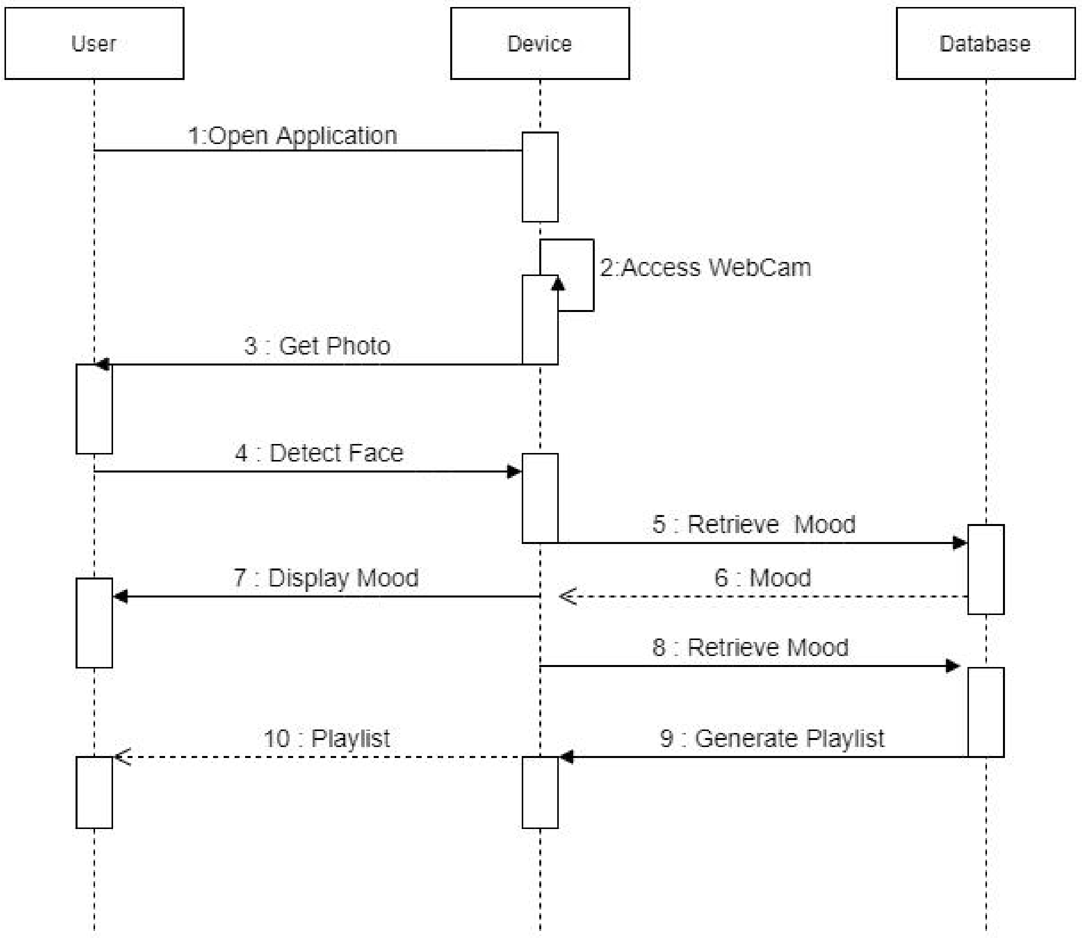
Sequence Diagram:

A sequence diagram simply illustrates the interaction of objects in a sequential manner, i.e. the order in which these interactions occur. A sequence diagram can also be referred to using the words event diagrams or event scenarios. Sequence diagrams show how and in what sequence the objects in a system work. Business people and software engineers frequently use these diagrams to describe and understand requirements for new and current systems.

Symbol	Name	Description
	Object symbol	Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape.
	Activation box	Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.
	Actor symbol	Shows entities that interact with or are external to the system.
	Package symbol	Used in UML 2.0 notation to contain interactive elements of the diagram. Also known as a frame, this rectangular shape has a small inner rectangle for labeling the diagram.
	Lifeline symbol	Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labeled rectangle shape or an actor symbol.
	Option loop symbol	Used to model if/then scenarios, i.e., a circumstance that will only occur under certain conditions.
	Alternative symbol	Symbolizes a choice (that is usually mutually exclusive) between two or more message sequences. To represent alternatives, use the labeled rectangle shape with a dashed line inside.

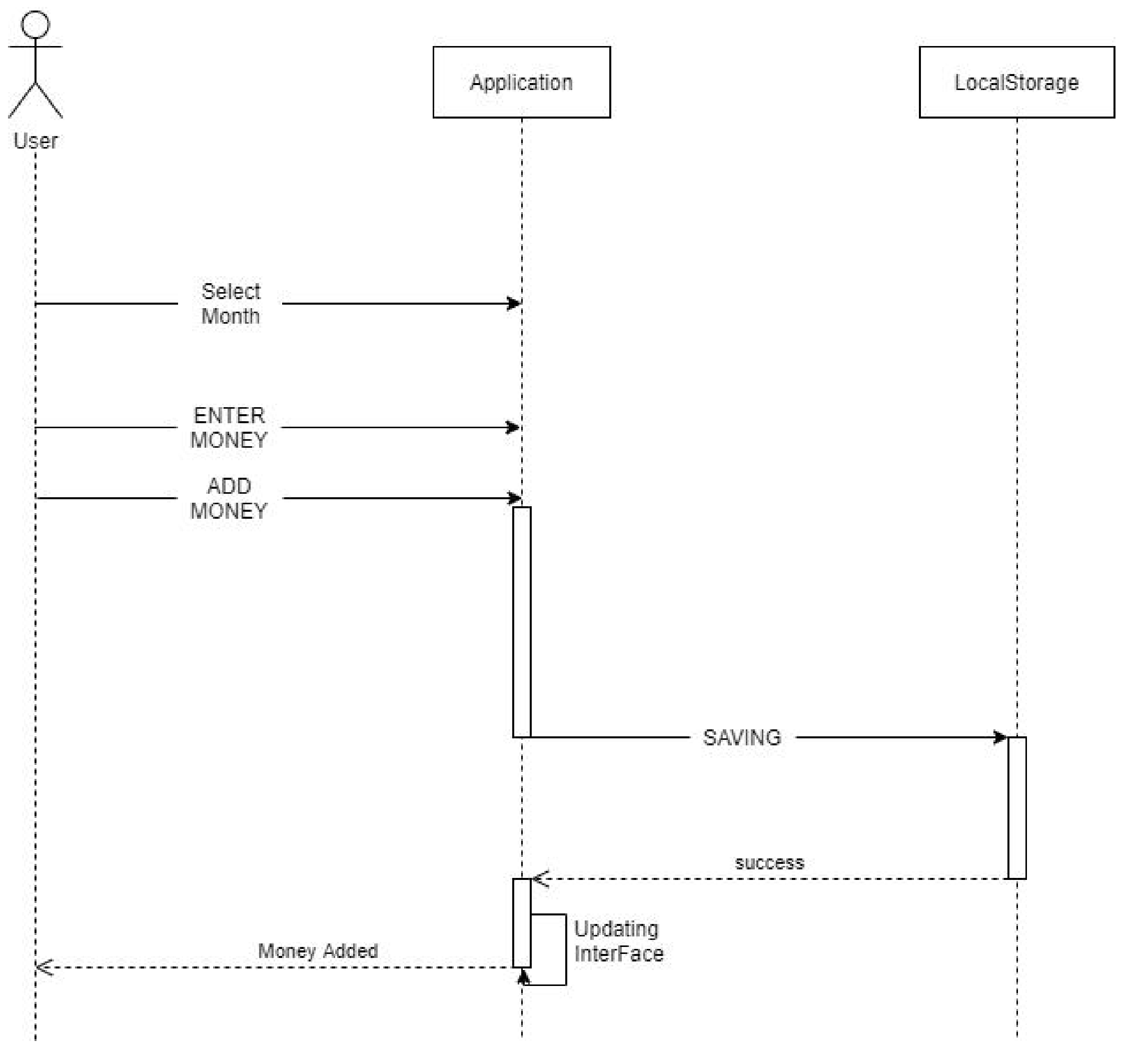
Name:-Atharva Gangrade
Enrollment No: EN19CS305014

Music Player Based On Mood

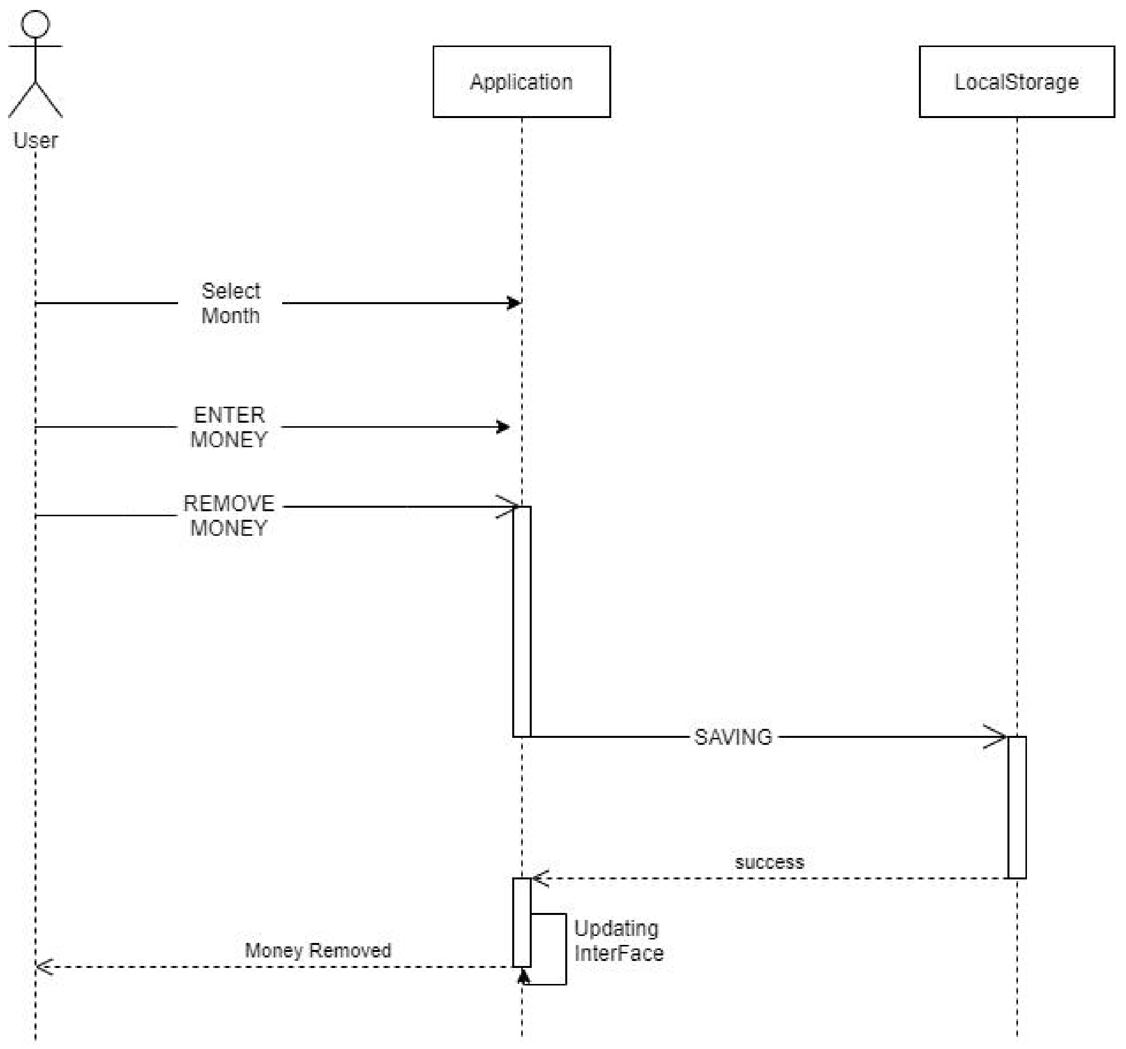


Name:-Atharva Gangrade
Enrollment No: EN19CS305014

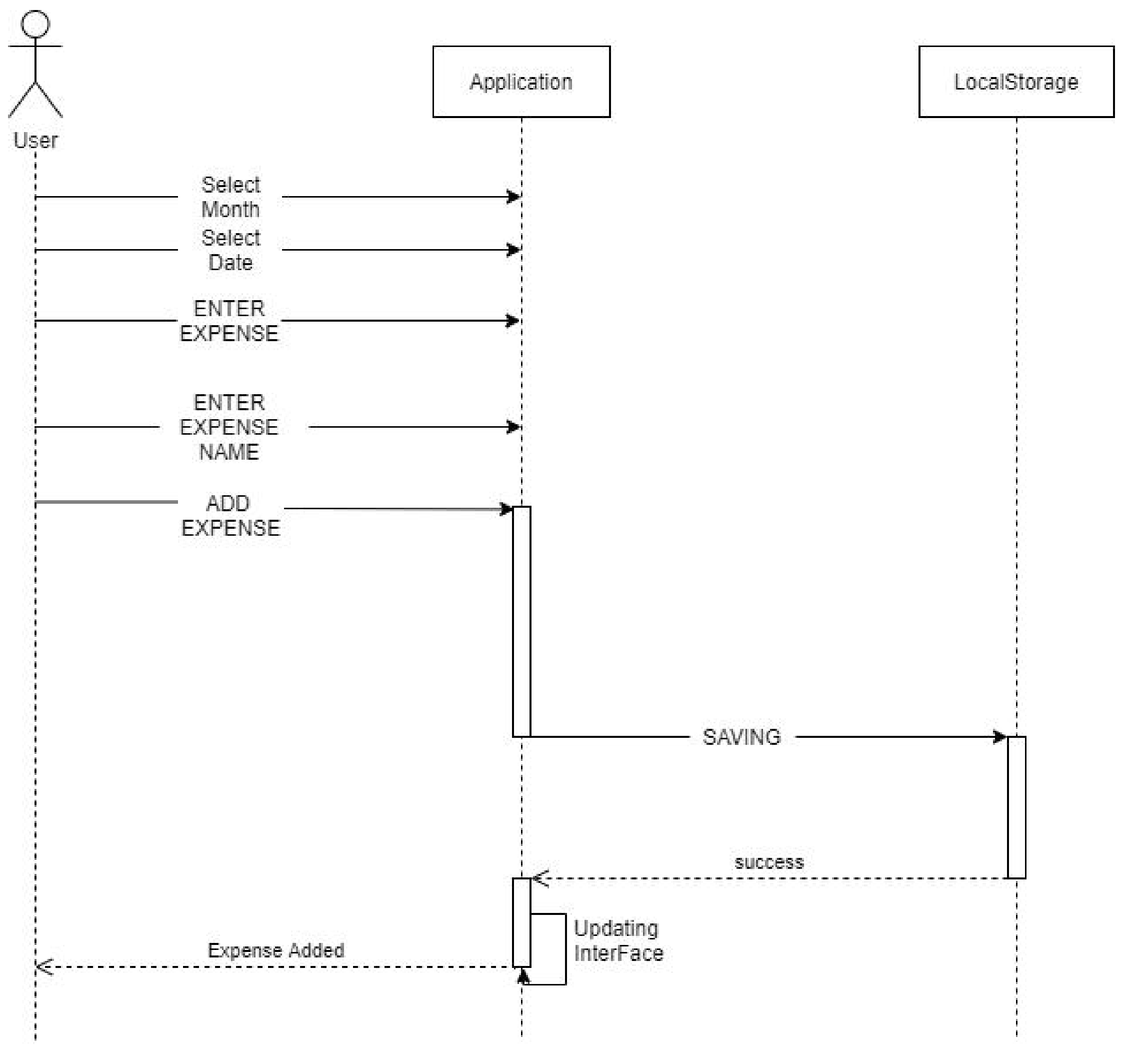
Sequence Diagram



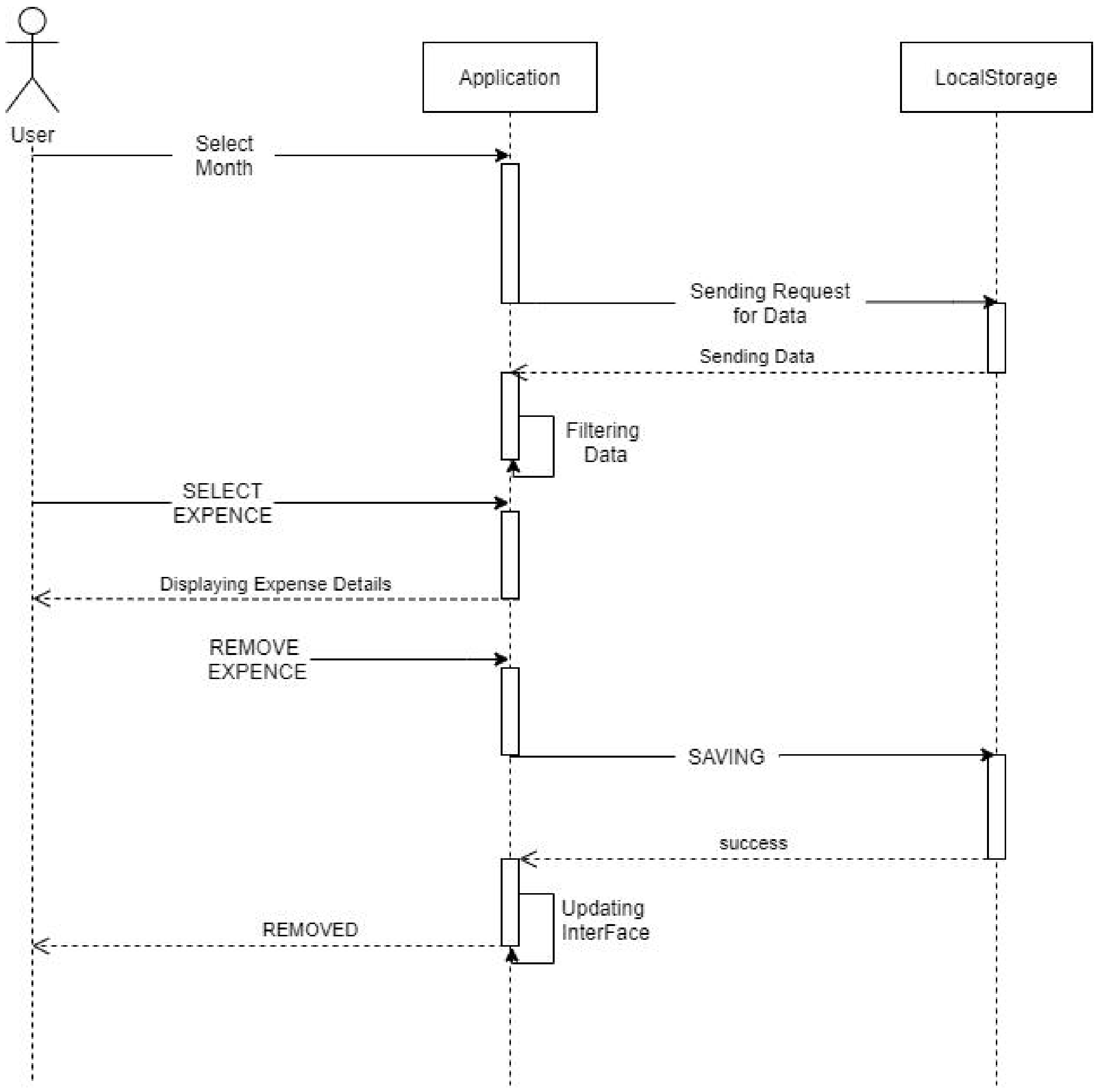
Name:-Atharva Gangrade
Enrollment No: EN19CS305014



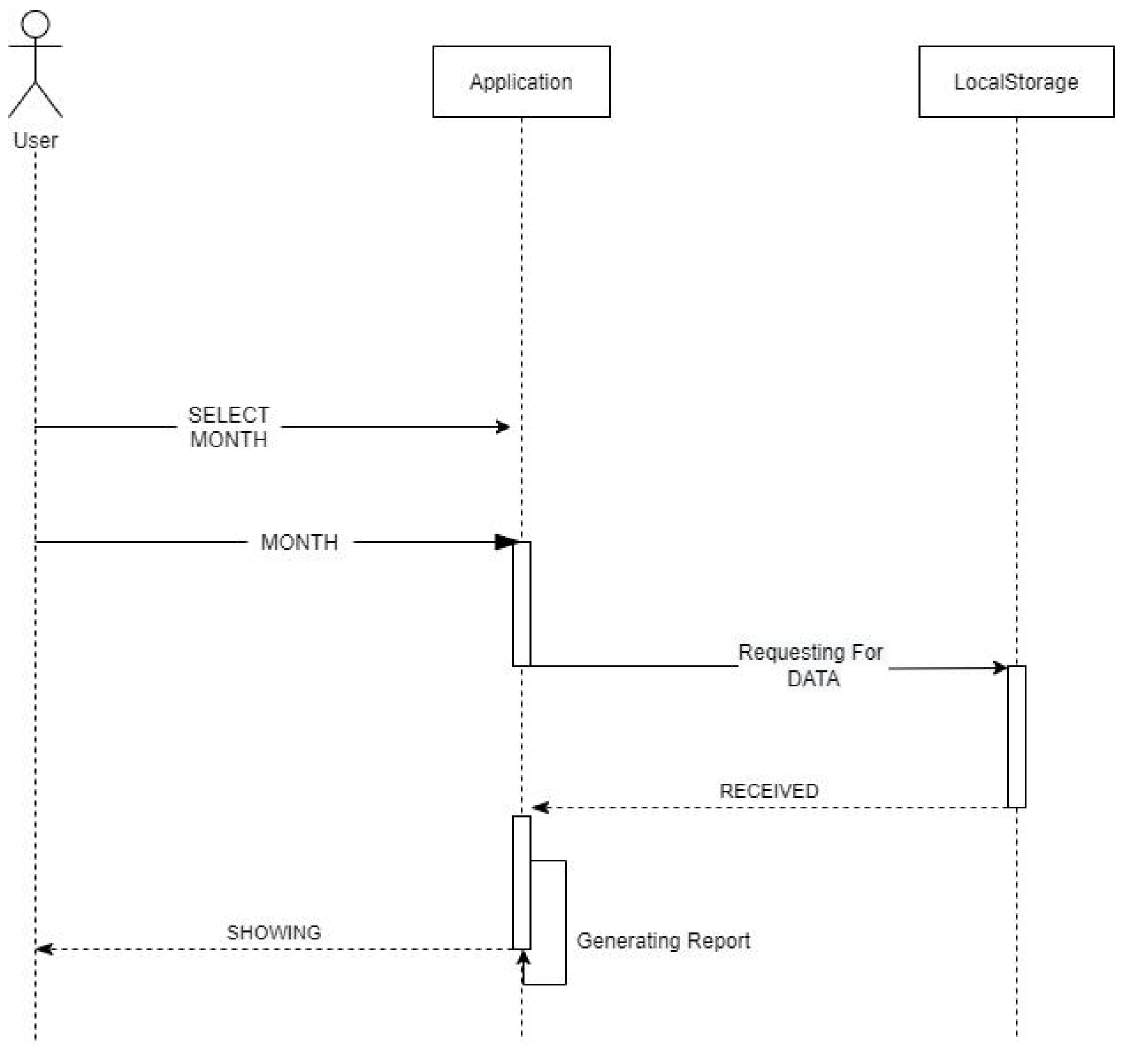
Name:-Atharva Gangrade
Enrollment No: EN19CS305014



Name:-Atharva Gangrade
Enrollment No: EN19CS305014



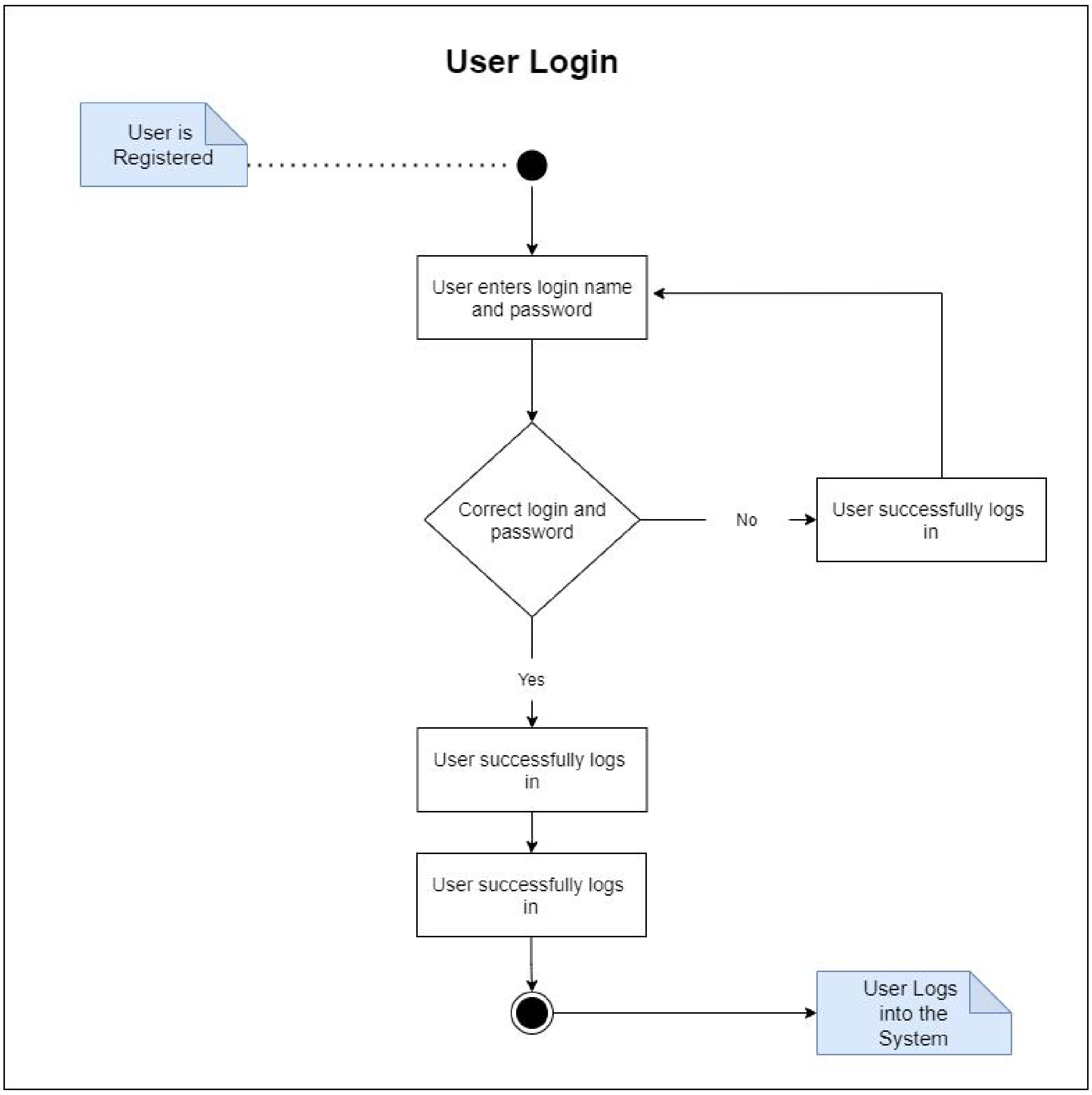
Name:-Atharva Gangrade
Enrollment No: EN19CS305014



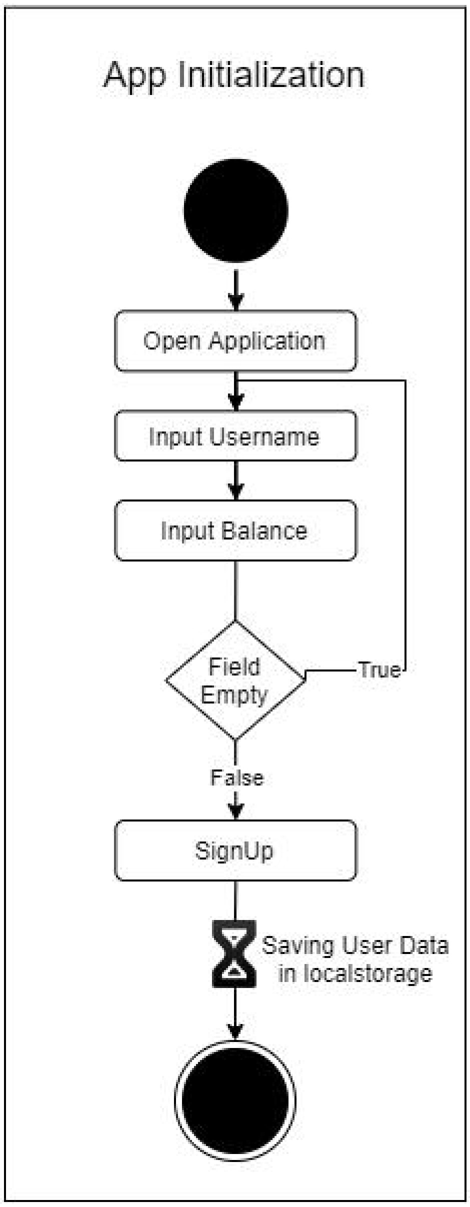
Name:-Atharva Gangrade
Enrollment No: EN19CS305014

Activity Diagram:

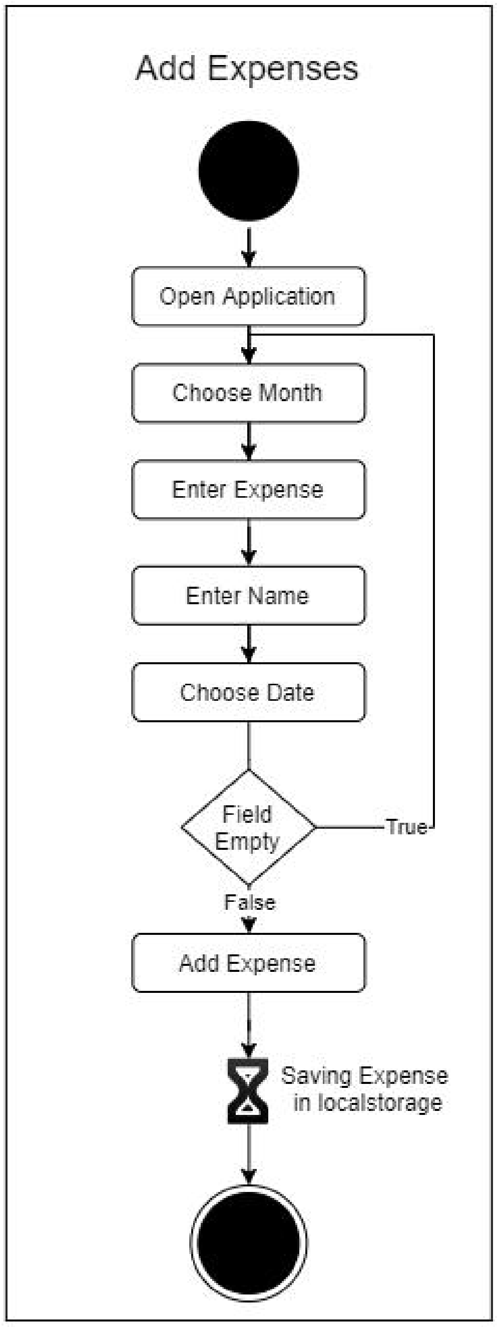
Activity Diagrams are used to depict the flow of control in a system. An activity diagram may also be used to describe the processes required in carrying out a use case. Using activity diagrams, we represent sequential and concurrent activities. As a result, we use an activity diagram to visually represent workflows. An activity diagram focuses on the state of flow and the order in which it occurs. Using an activity diagram, we describe or represent what causes a certain occurrence.

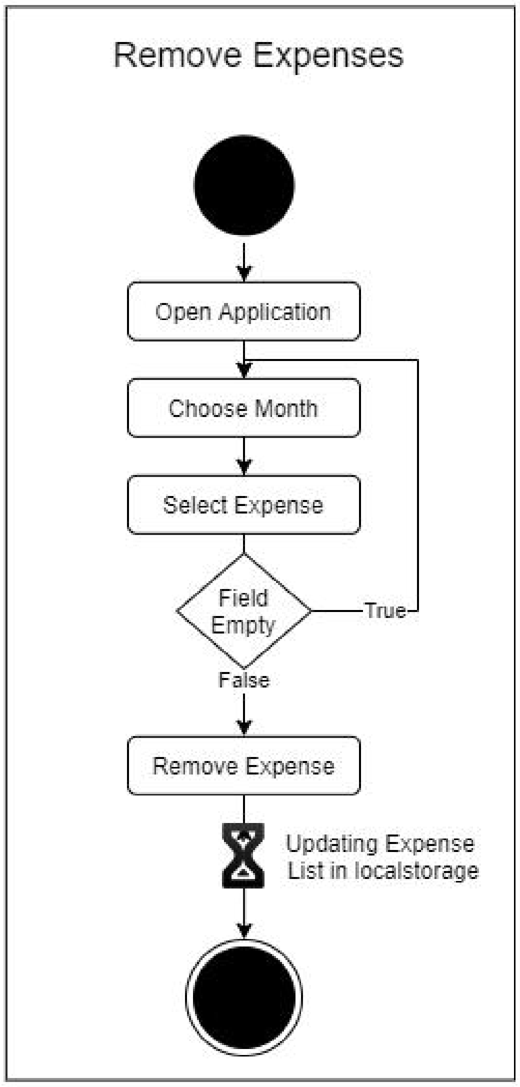

User Login
User is Registered
User enters login name and password
Correct login and password
No
User successfully logs in
Yes
User successfully logs in
User successfully logs in
User Logs into the System

Name:-Atharva Gangrade
Enrollment No: EN19CS305014

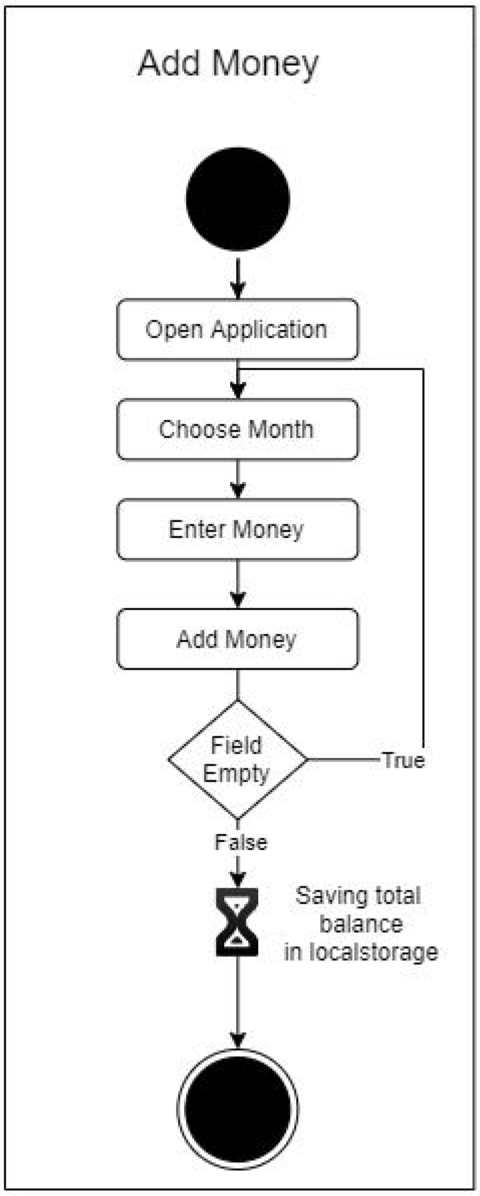


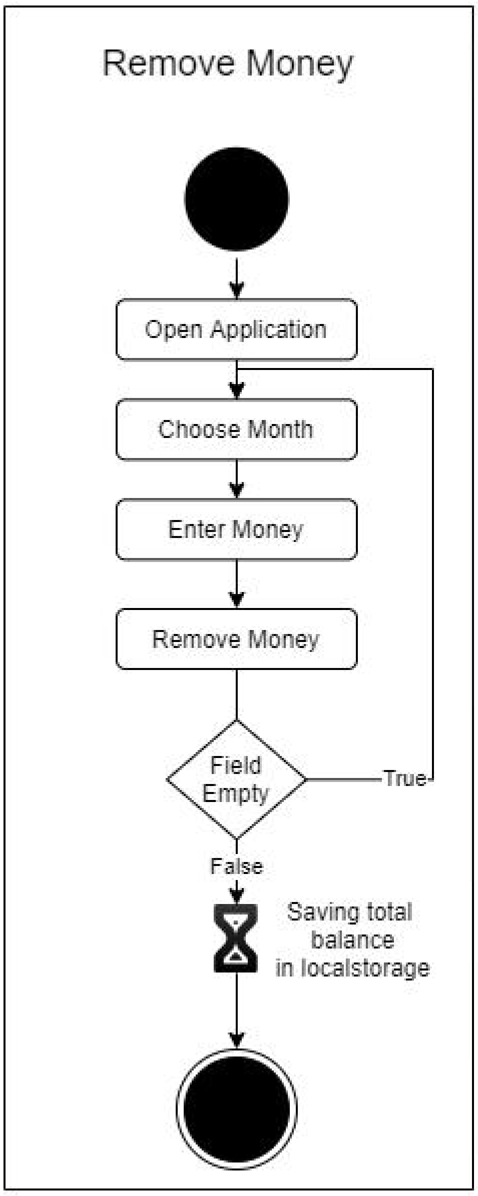
Name:-Atharva Gangrade
Enrollment No: EN19CS305014



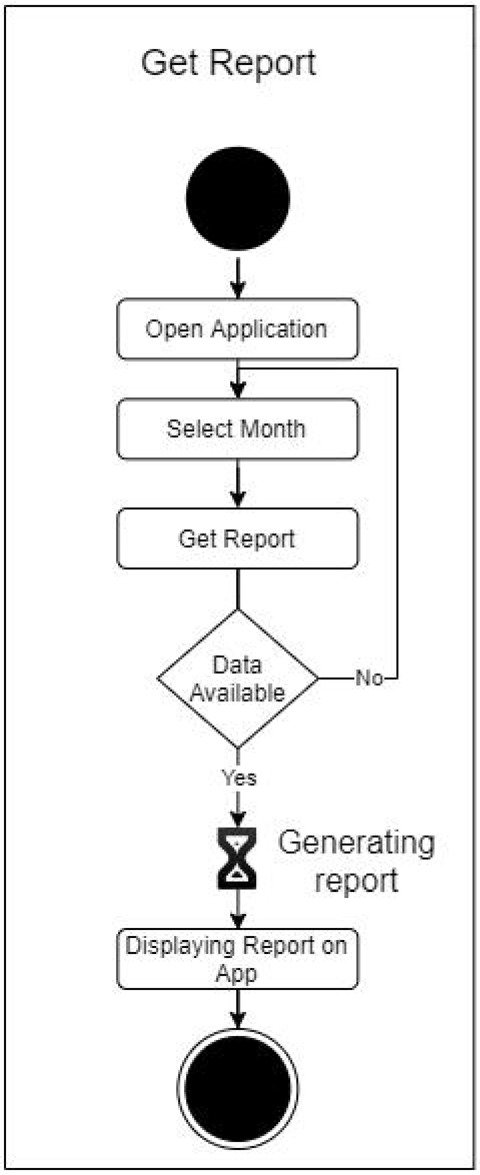


Name:-Atharva Gangrade
Enrollment No: EN19CS305014



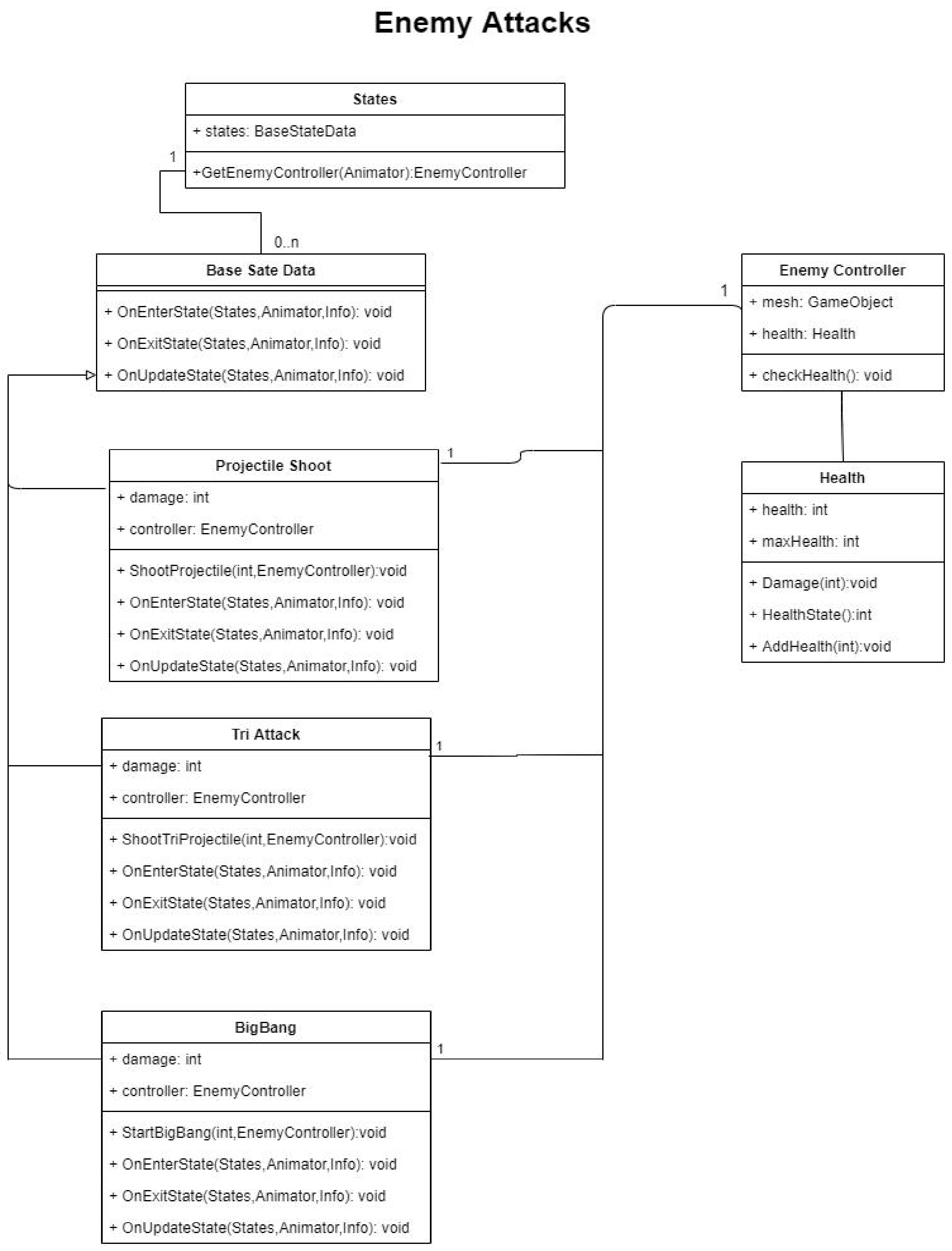


Name:-Atharva Gangrade
Enrollment No: EN19CS305014



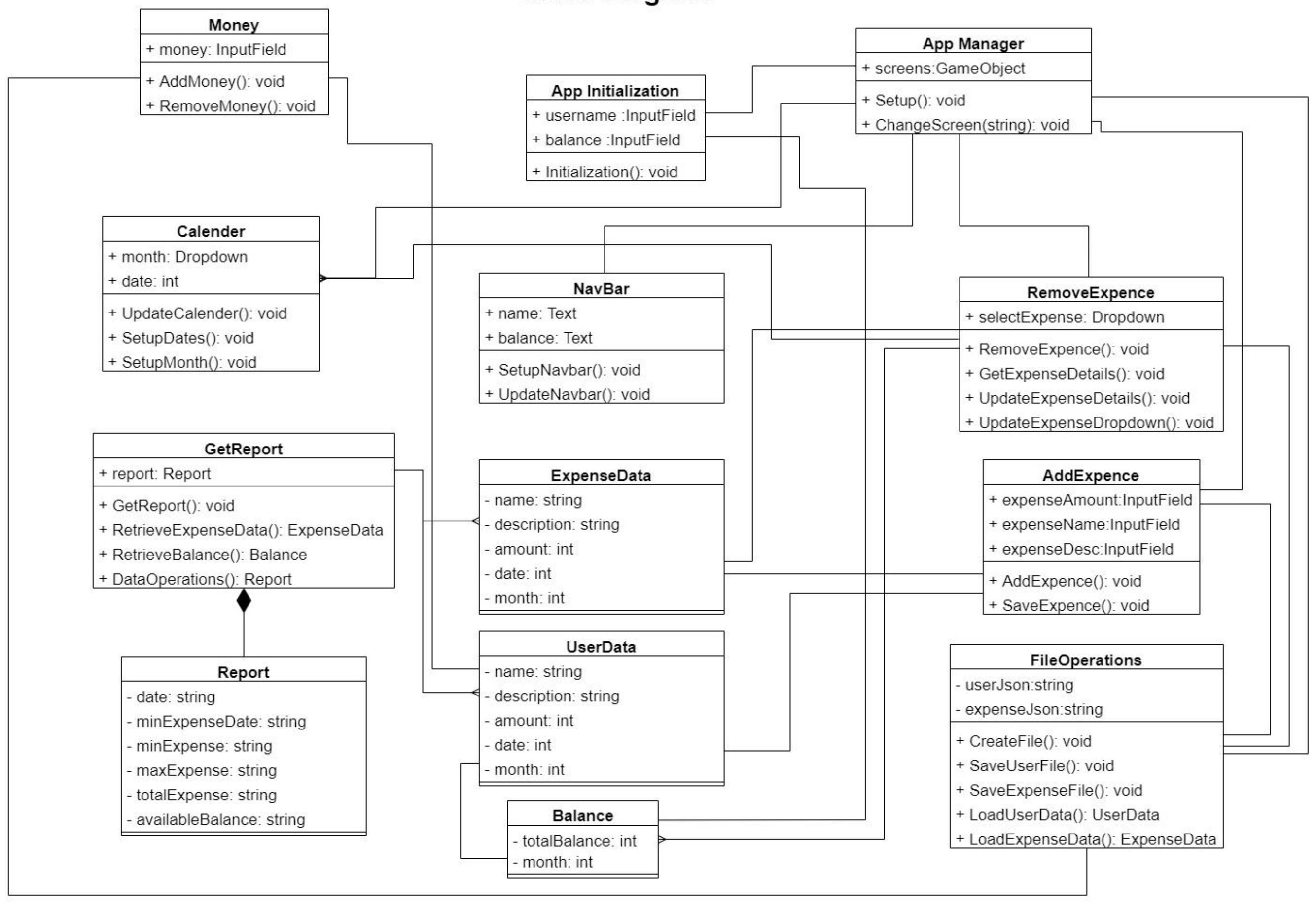
Name:-Atharva Gangrade
Enrollment No: EN19CS305014

Class Diagram:
The class diagram is the most commonly used UML diagram. It serves as the foundation for all object-oriented software systems. Class diagrams are used to describe the static structure of a system by displaying the system's classes, methods, and properties. Class diagrams also assist us in determining the relationships between various classes or objects.


Enemy Attacks
States
+ states: BaseStateData
+GetEnemyController(Animator):EnemyController
1
0..n
Base Sate Data
+ OnEnterState(States,Animator,Info): void
+ OnExitState(States,Animator,Info): void
+ OnUpdateState(States,Animator,Info): void
Enemy Controller
+ mesh: GameObject
+ health: Health
+ checkHealth(): void
1
Health
+ health: int
+ maxHealth: int
+ Damage(int):void
+ HealthState():int
+ AddHealth(int):void
Projectile Shoot
+ damage: int
+ controller: EnemyController
+ ShootProjectile(int,EnemyController):void
+ OnEnterState(States,Animator,Info): void
+ OnExitState(States,Animator,Info): void
+ OnUpdateState(States,Animator,Info): void
1
Tri Attack
+ damage: int
+ controller: EnemyController
+ ShootTriProjectile(int,EnemyController):void
+ OnEnterState(States,Animator,Info): void
+ OnExitState(States,Animator,Info): void
+ OnUpdateState(States,Animator,Info): void
1
BigBang
+ damage: int
+ controller: EnemyController
+ StartBigBang(int,EnemyController):void
+ OnEnterState(States,Animator,Info): void
+ OnExitState(States,Animator,Info): void
+ OnUpdateState(States,Animator,Info): void
1

Name:-Atharva Gangrade
Enrollment No: EN19CS305014

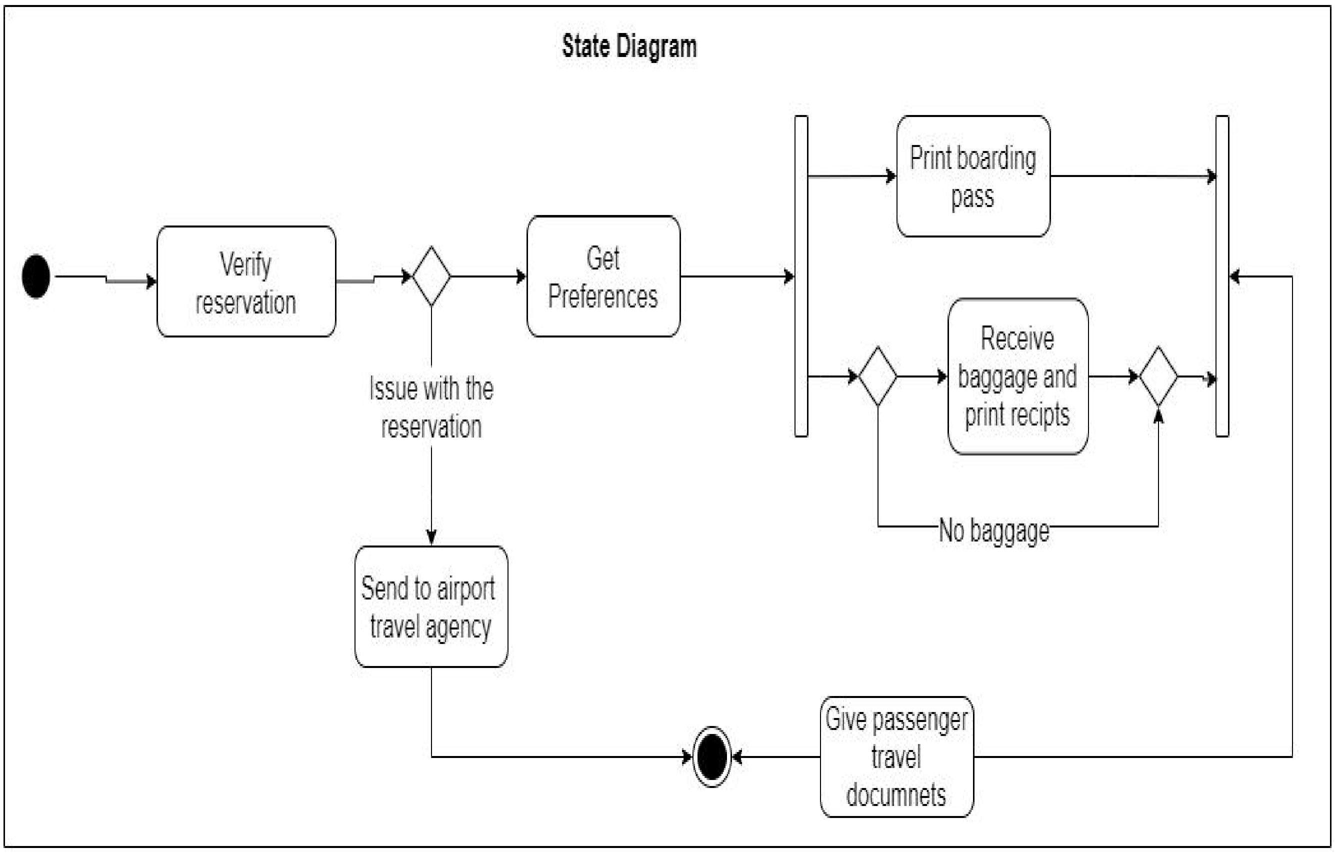
Class Diagram



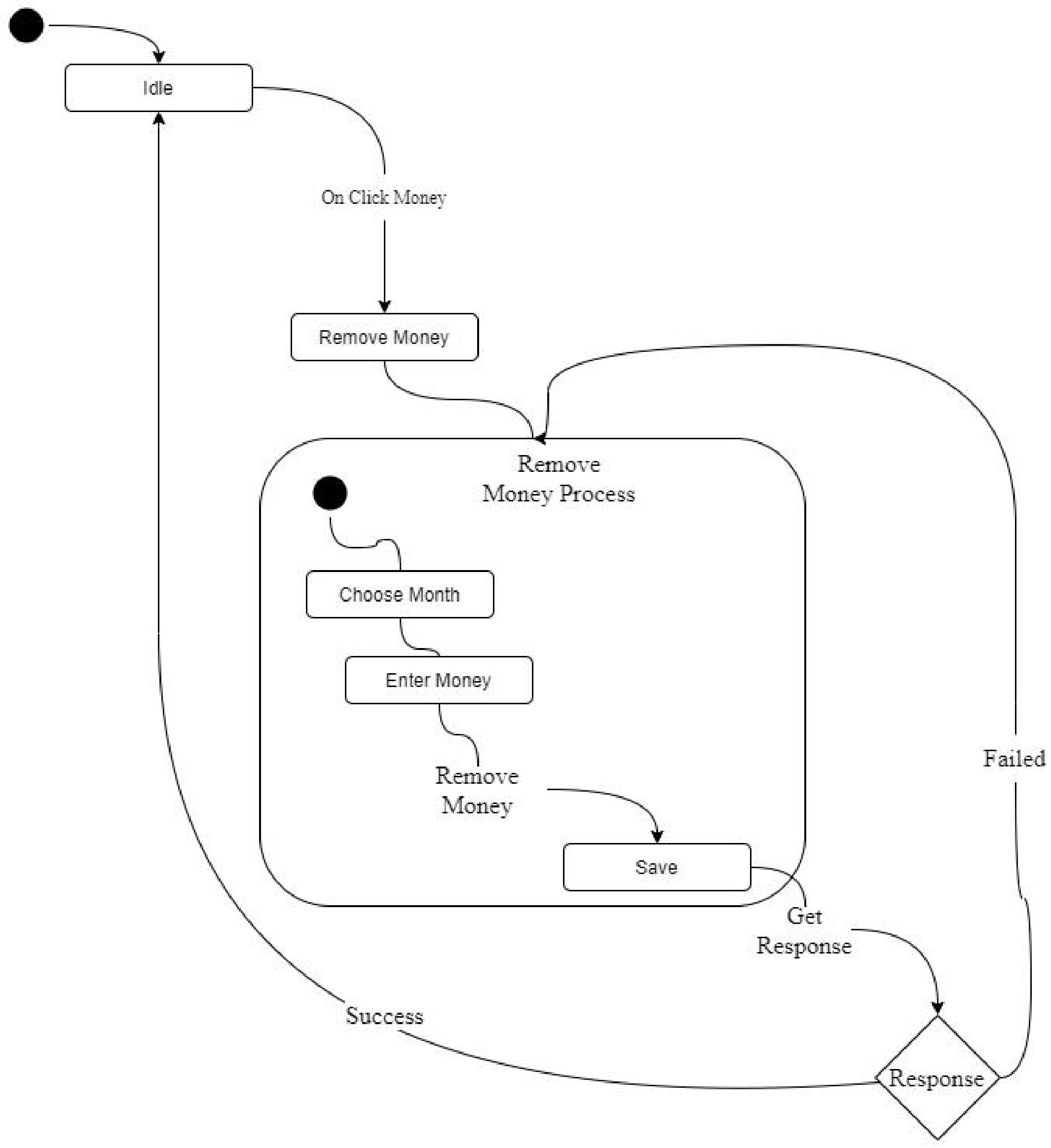
Name:-Atharva Gangrade
Enrollment No: EN19CS305014

State Diagram:

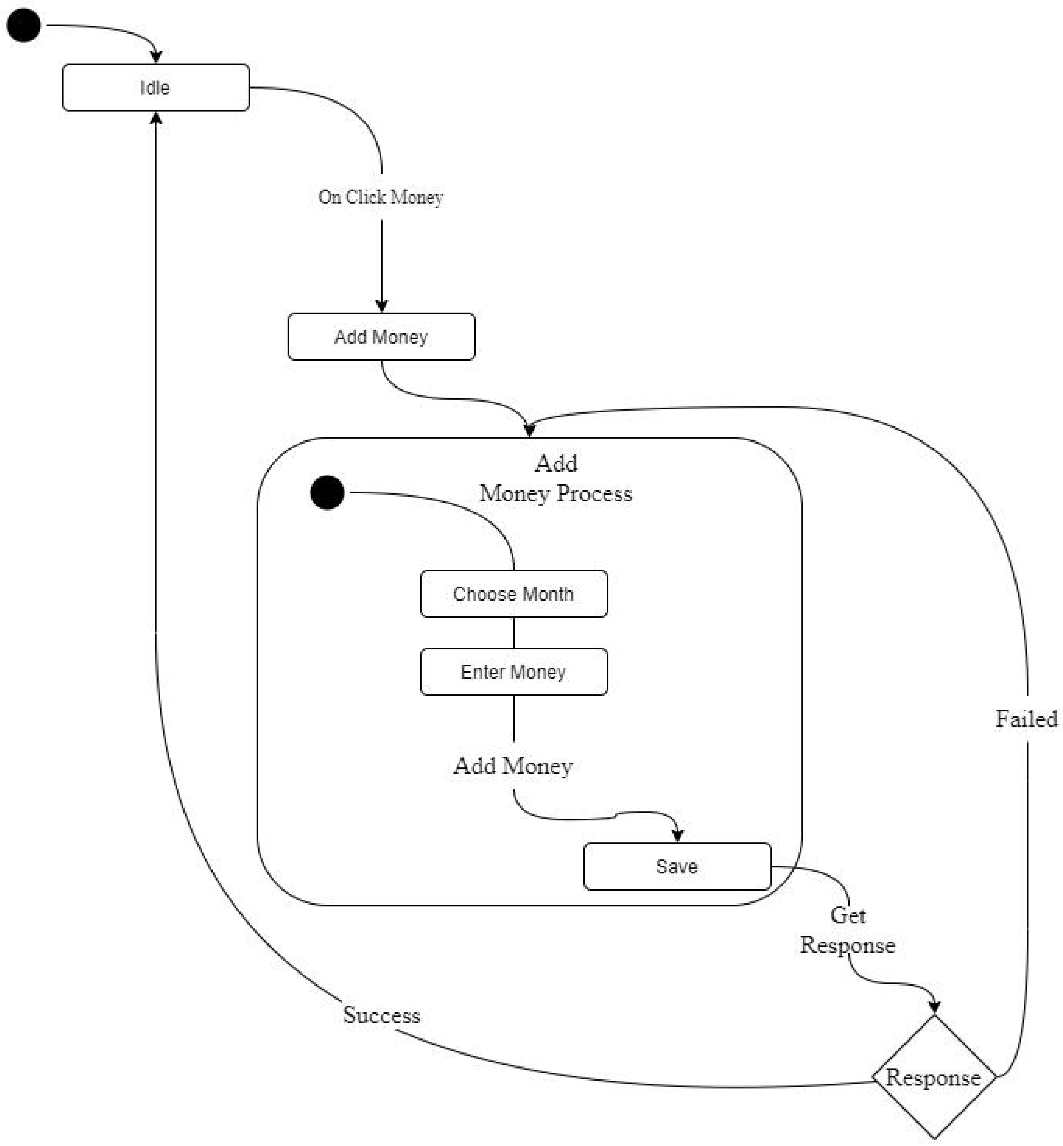
A state diagram is used to show the state of a system or component of a system at discrete points in time. It's a behavioural diagram that depicts behaviour with limited state changes. State diagrams can alternatively be called state machines or state-chart diagrams. These two names are frequently used interchangeably. Simply put, a state diagram is used to depict a class's dynamic behaviour in response to time and changing external stimuli**.**


State Diagram
Verify reservation
Get Preferences
Issue with the reservation
Send to airport travel agency
Print boarding pass
Receive baggage and print recipts
No baggage
Give passenger travel documnets

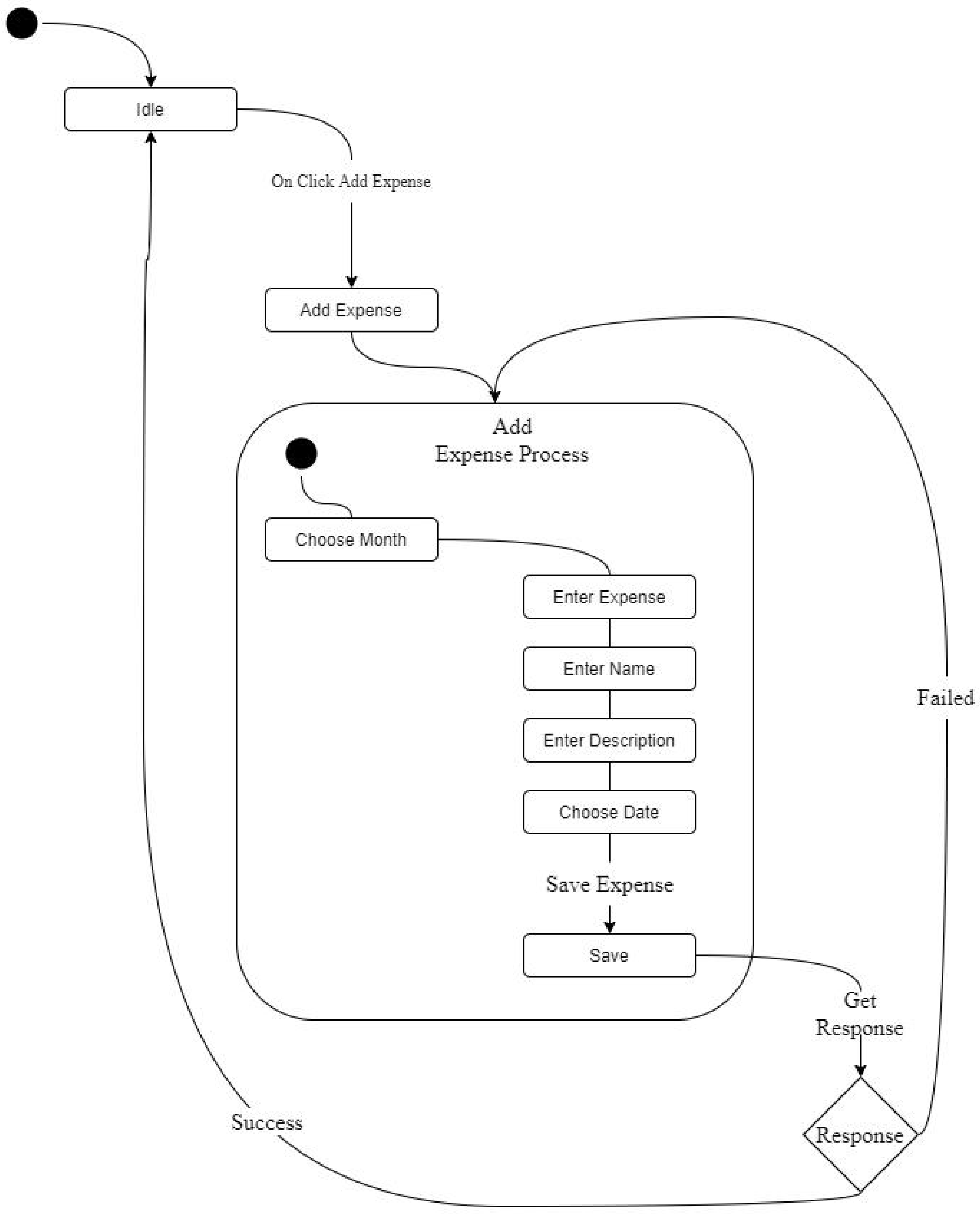
Name:-Atharva Gangrade
Enrollment No: EN19CS305014



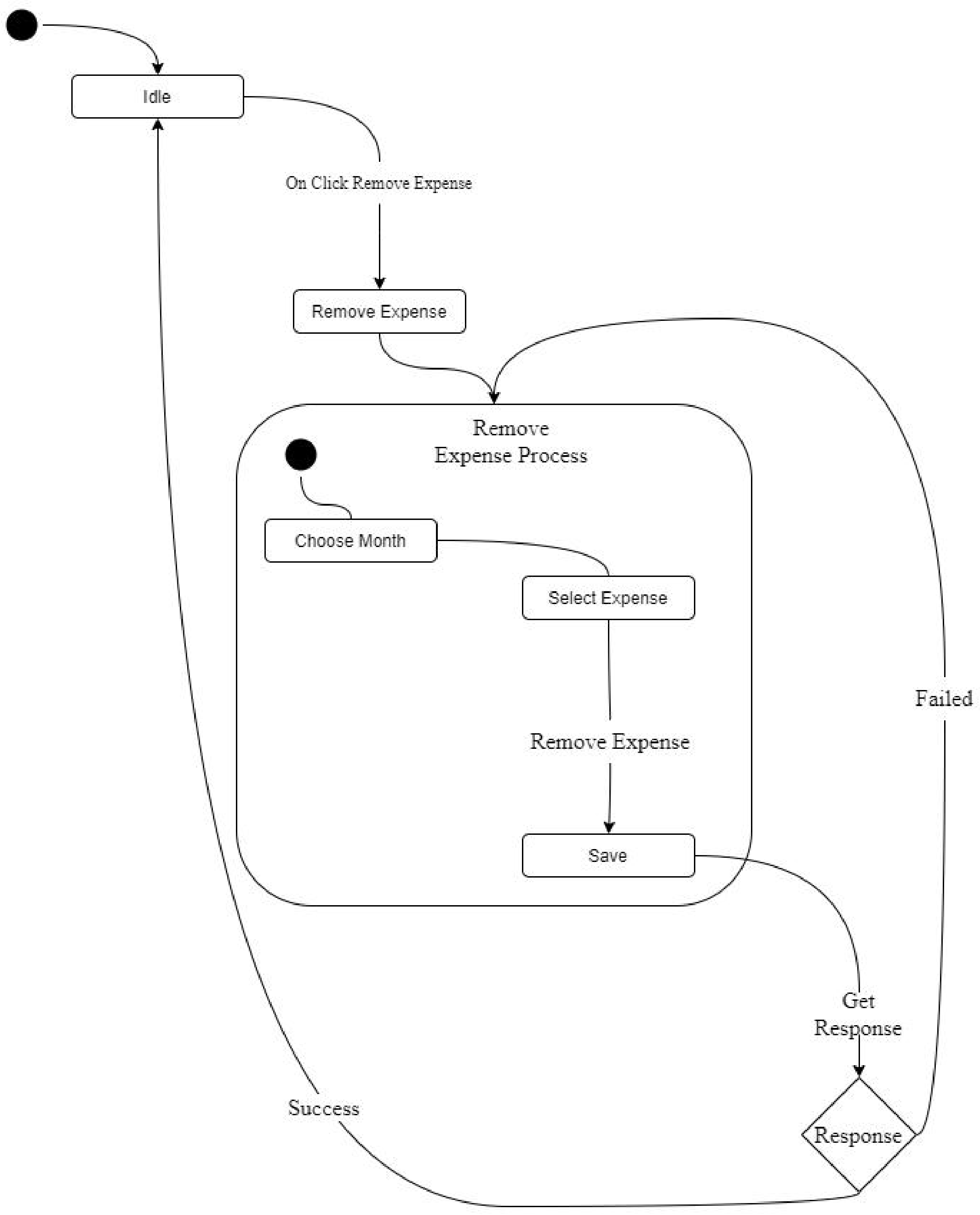
Name:-Atharva Gangrade
Enrollment No: EN19CS305014



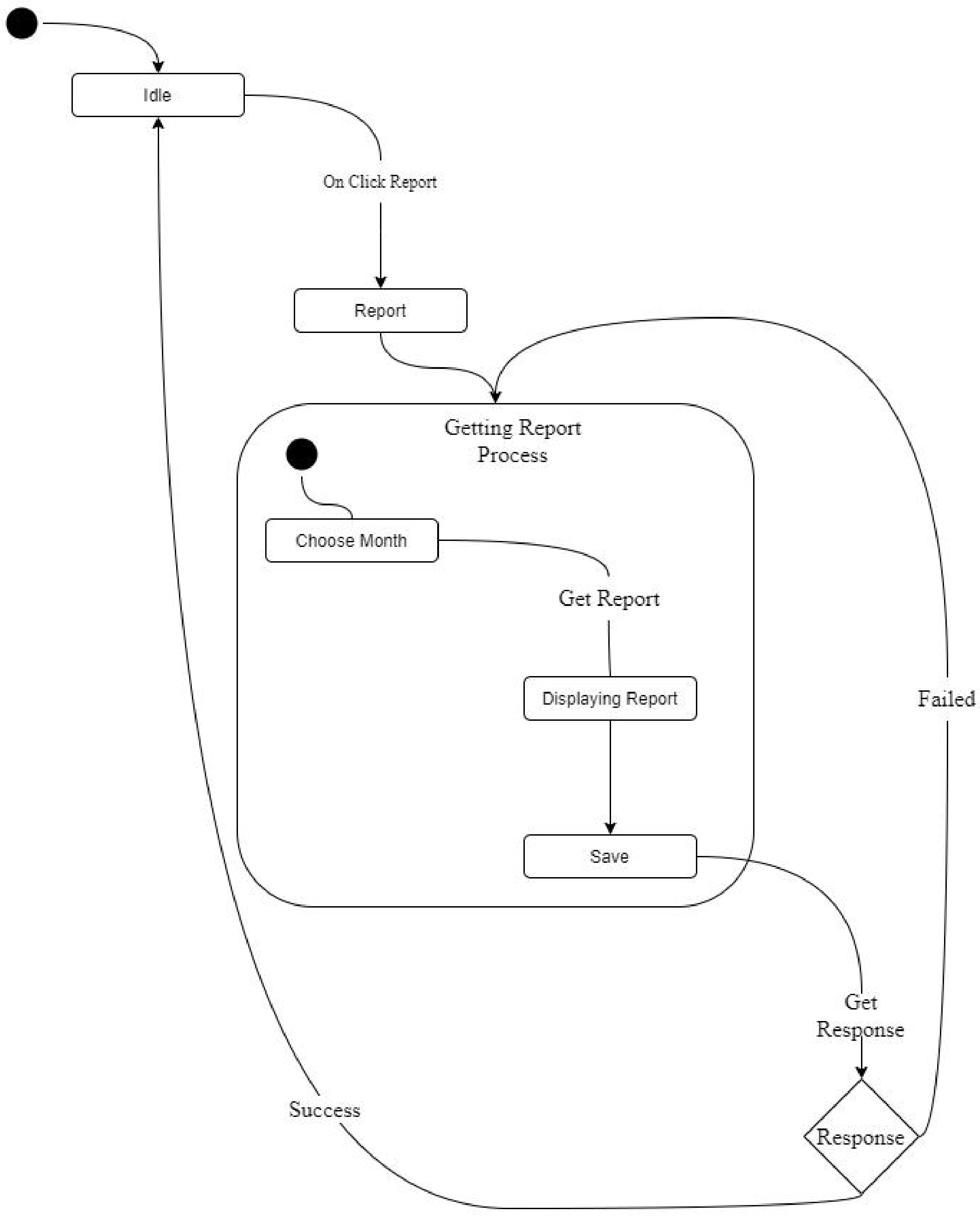
Name:-Atharva Gangrade
Enrollment No: EN19CS305014



Name:-Atharva Gangrade
Enrollment No: EN19CS305014



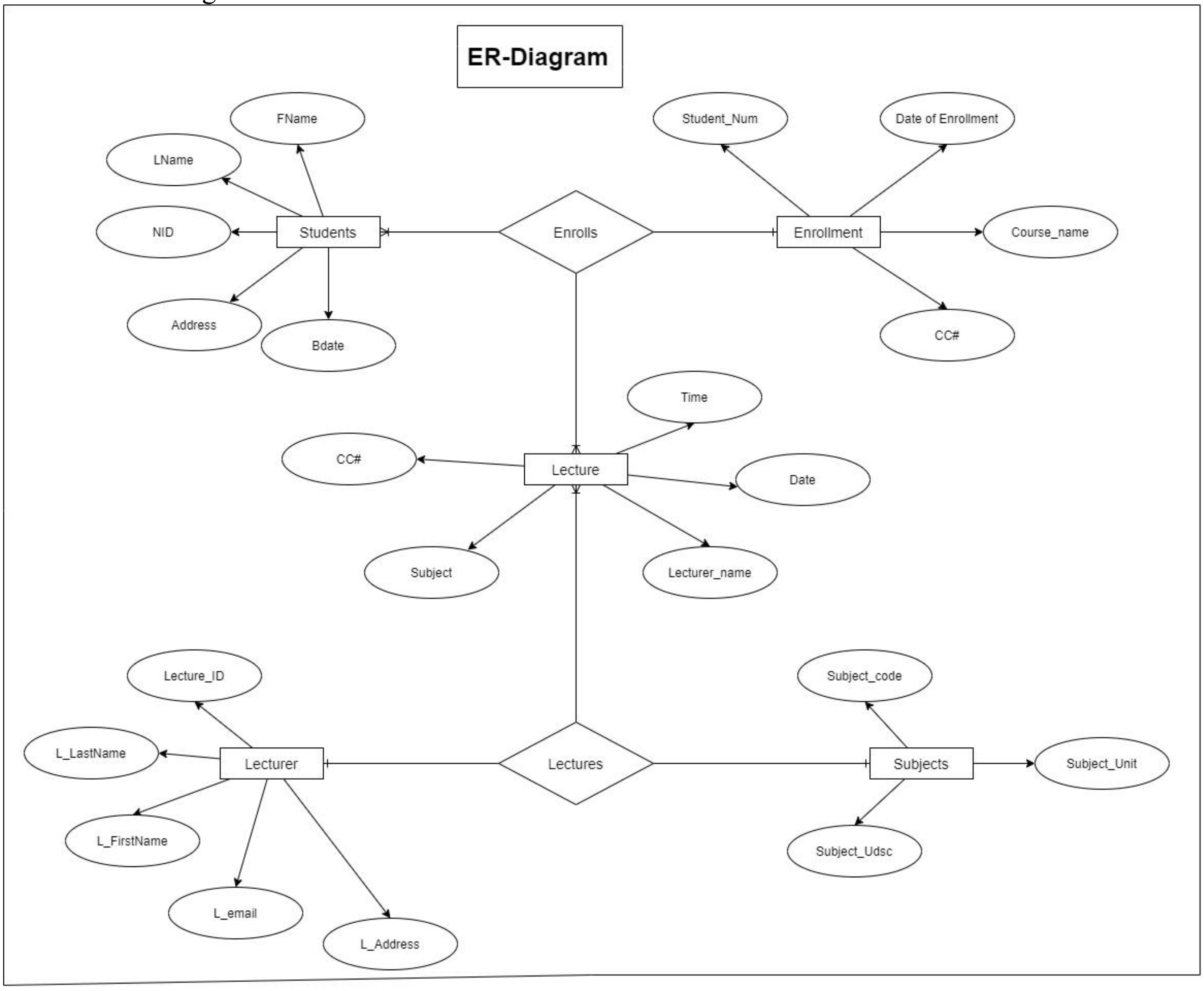
Name:-Atharva Gangrade
Enrollment No: EN19CS305014



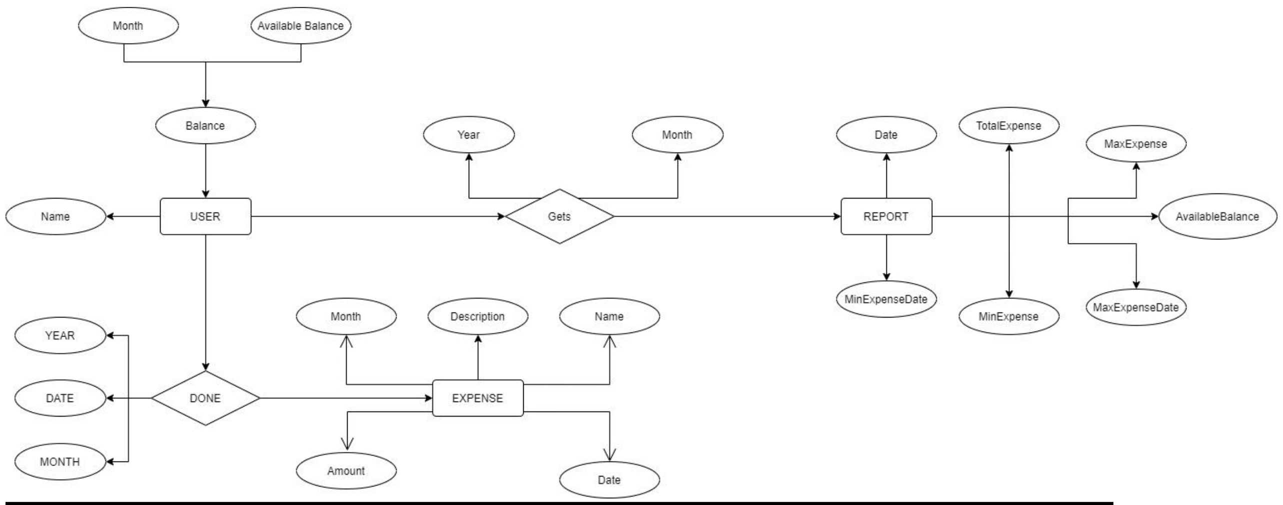
Name:-Atharva Gangrade
Enrollment No: EN19CS305014

ER Diagram:

An Entity Relationship (ER) Diagram is a sort of flowchart that shows how "entities" in a system, such as people, things, or concepts, interact with one another. ER Diagrams are most commonly used in the disciplines of software engineering, corporate information systems, education, and research to build or troubleshoot relational databases. They are sometimes referred to as ERDs or ER Models because they employ a predefined collection of symbols like as rectangles, diamonds, ovals, and connecting lines to illustrate the interconnectivity of entities, connections, and their properties. They reflect linguistic structure, with items acting as nouns and connections acting as verbs.



Name:-Atharva Gangrade
Enrollment No: EN19CS305014



Name:-Atharva Gangrade
Enrollment No: EN19CS305014

Data-Flow Diagram:

A Data Flow Diagram (DFD) is a classic visual depiction of a system's information flows. A tidy and clear DFD may graphically display the appropriate quantity of system need. It can be manual, automatic, or a hybrid of the two.

It demonstrates how data enters and exits the system, what alters the data, and where it is kept.

A DFD's goal is to represent the breadth and bounds of a system as a whole. It may be used as a method for communication between a system analyst and everyone who is involved in the order that serves as a beginning point for rebuilding a system. The data flow diagram (DFD) is also known as a data flow graph or bubble chart.

Levels in Data Flow Diagrams (DFD)

The DFD may be used to perform a system or software at any level of abstraction. Infact, DFDs may be partitioned into levels that represent increasing information flow and functional detail. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see primarily three levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

0-level DFDM

It is also known as fundamental system model, or context diagram represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows. Then the system is decomposed and described as a DFD with multiple bubbles. Parts of the system represented by each of these bubbles are then decomposed and documented as more and more detailed DFDs. This process may be repeated at as many levels as necessary until the program at hand is well understood. It is essential to preserve the number of inputs and outputs between levels, this concept is called leveling by DeMacro.

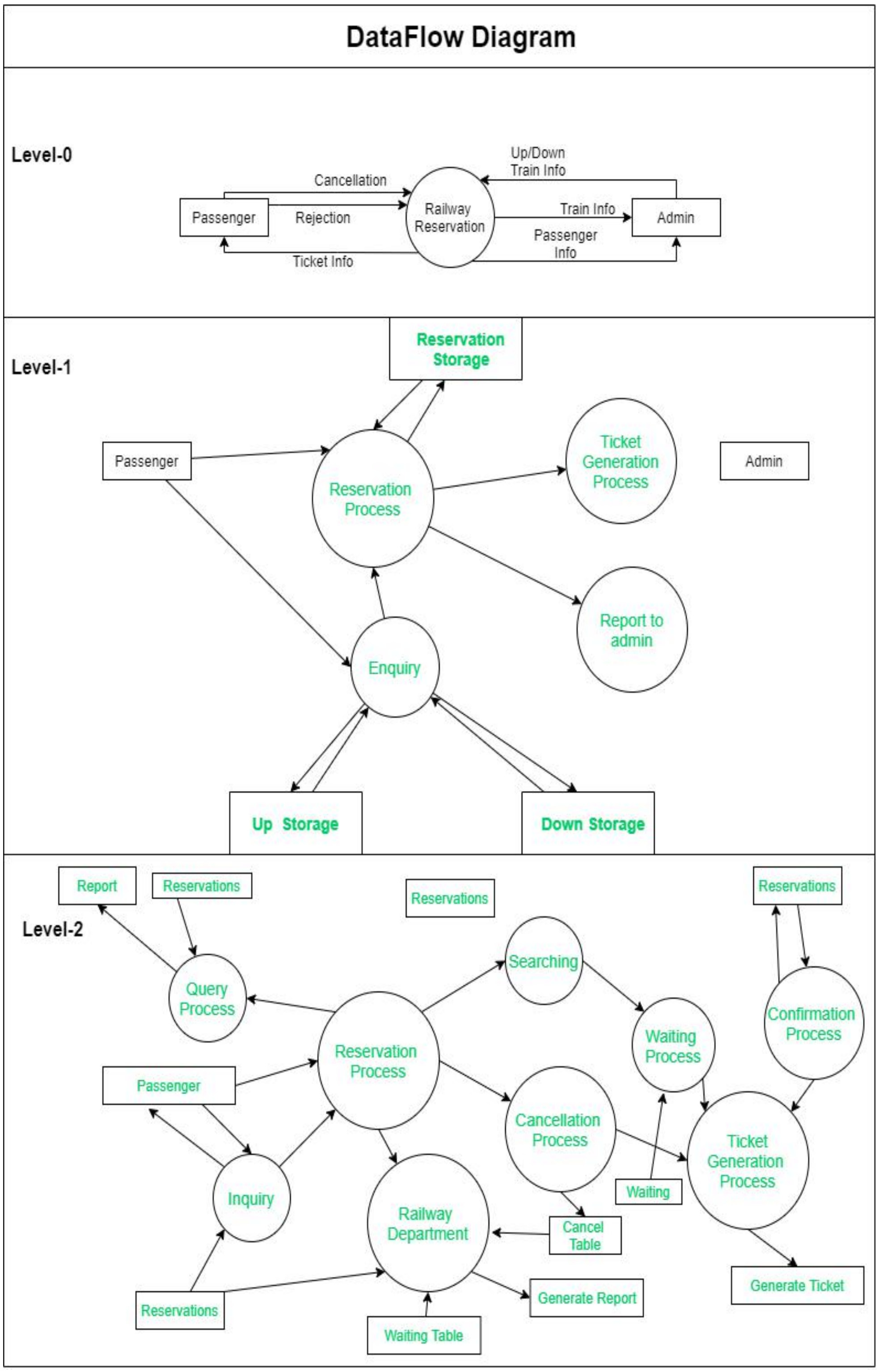
1-level DFD

In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main objectives of the system and breakdown the high-level process of 0-level DFD into sub-processes.

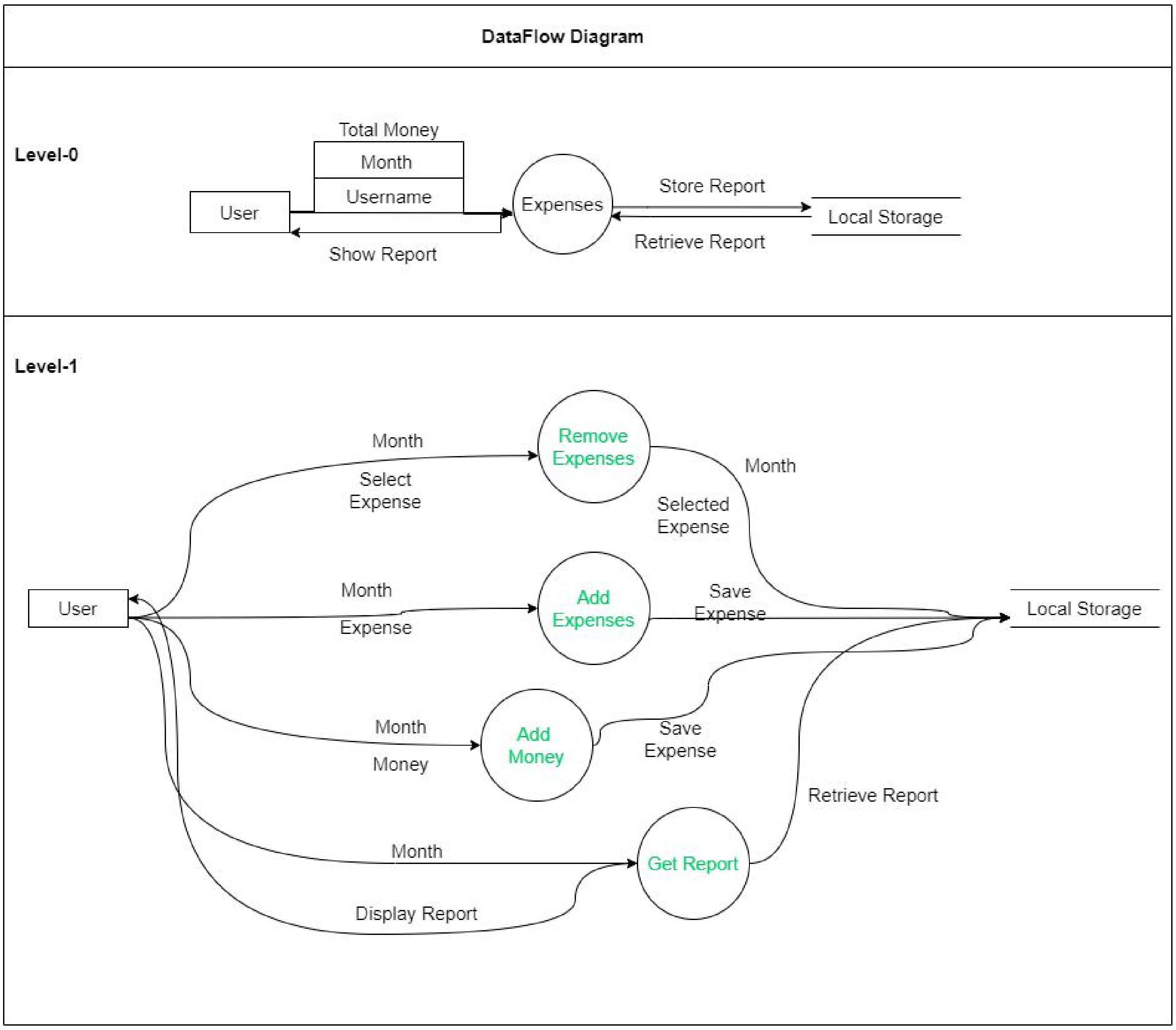
2-Level DFD

2-level DFD goes one process deeper into parts of 1-level DFD. It can be used to project or record the specific/necessary detail about the system's functioning.

Name:-Atharva Gangrade
Enrollment No: EN19CS305014

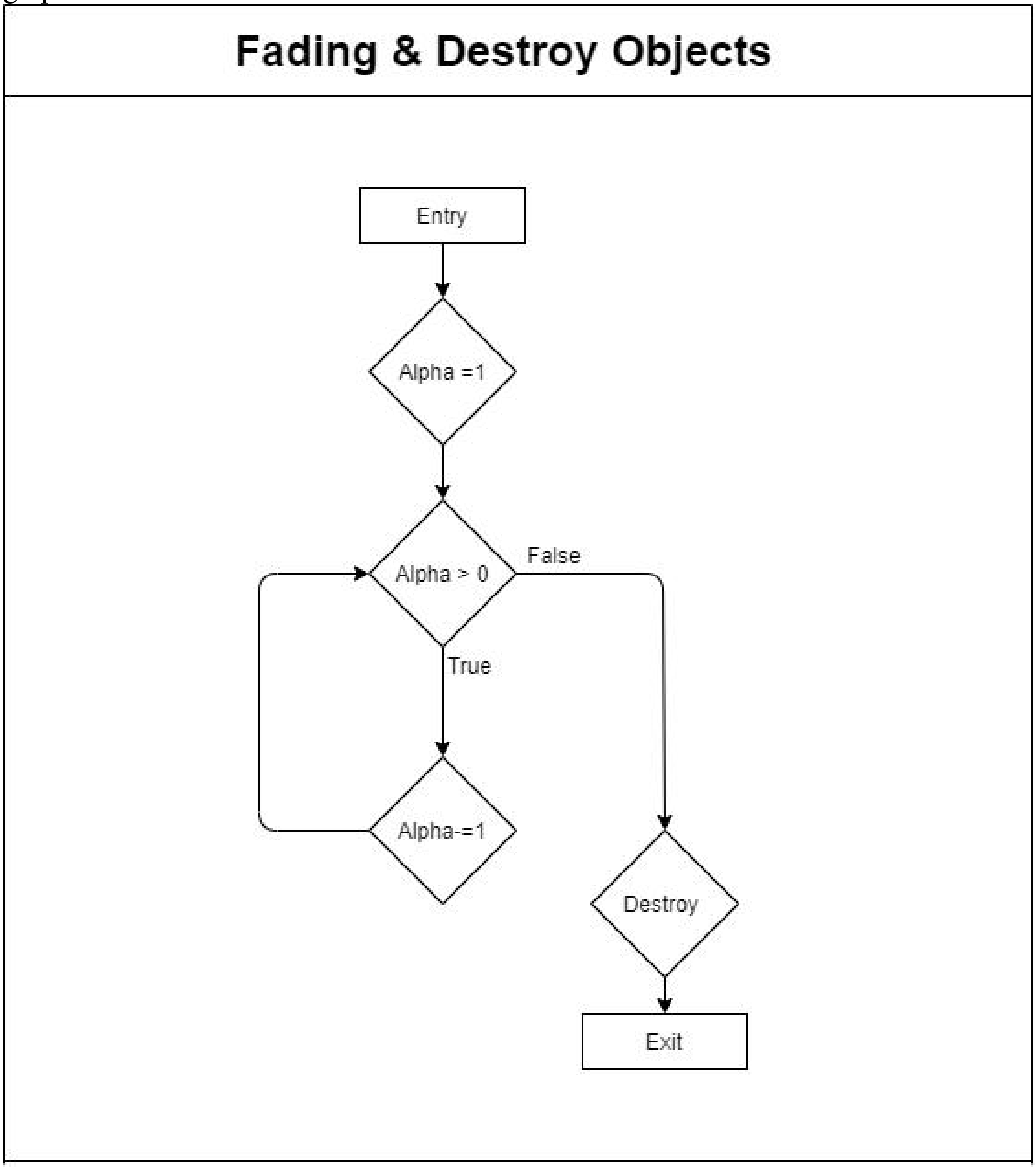


Name:-Atharva Gangrade
Enrollment No: EN19CS305014

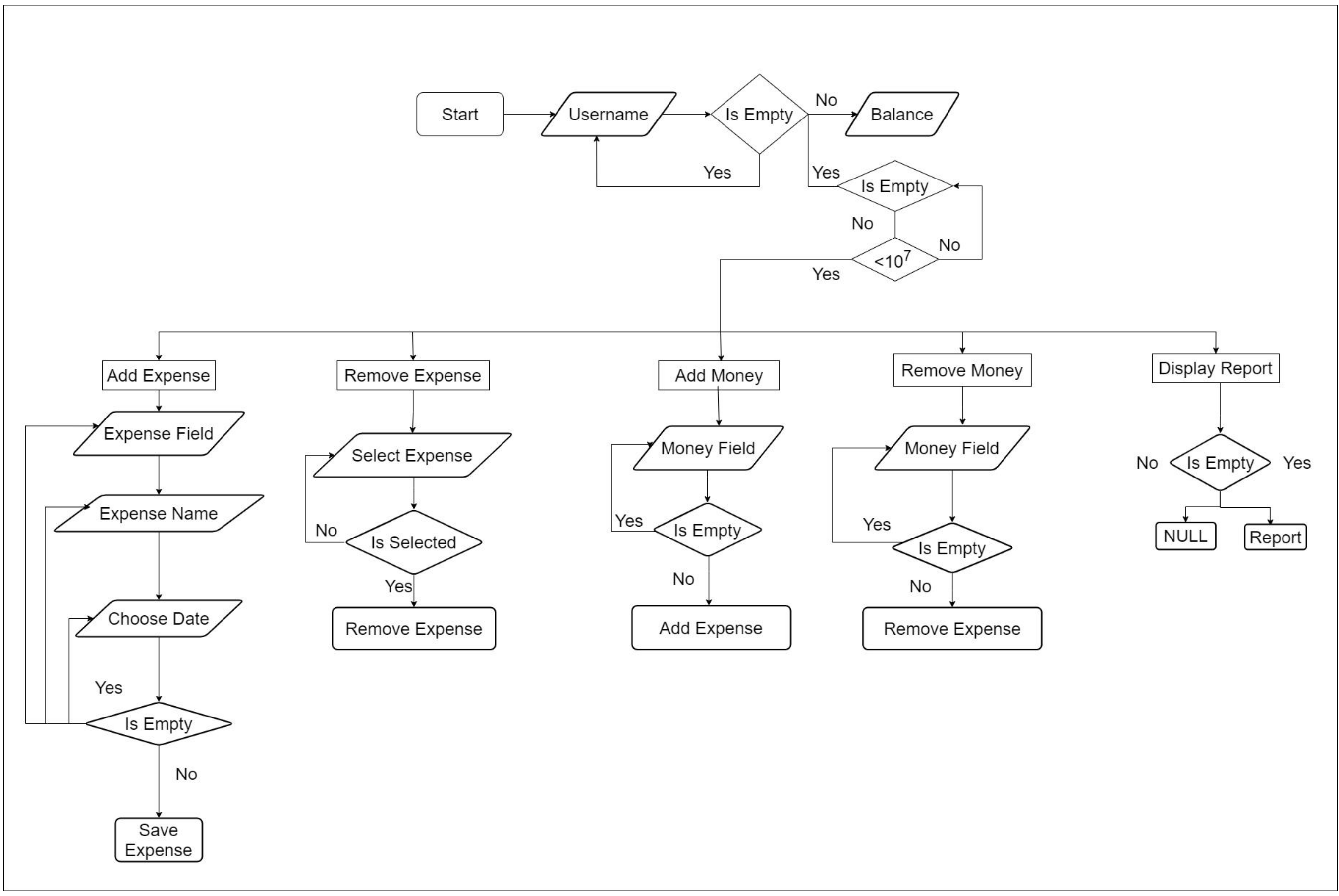


Name:-Atharva Gangrade
Enrollment No: EN19CS305014

Control-Flow Diagram:
A Control Flow Graph (CFG) is a graphical depiction of control flow or computation during programme or application execution. Control flow graphs are commonly utilised in static analysis and compiler applications because they properly reflect the flow inside a software unit. Frances E. Allen was the first to create a control flow graph.



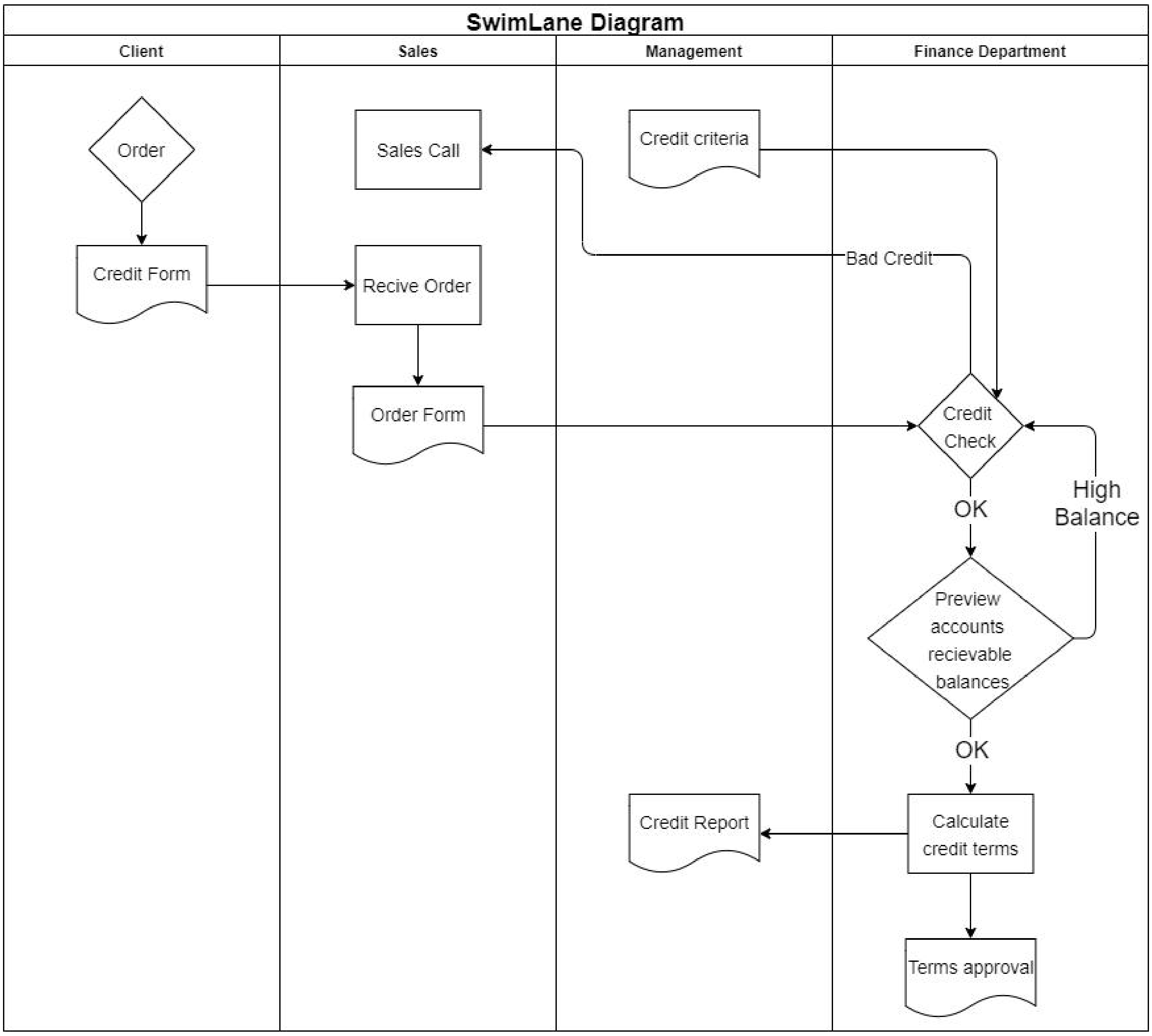
Name:-Atharva Gangrade
Enrollment No: EN19CS305014



Name:-Atharva Gangrade
Enrollment No: EN19CS305014

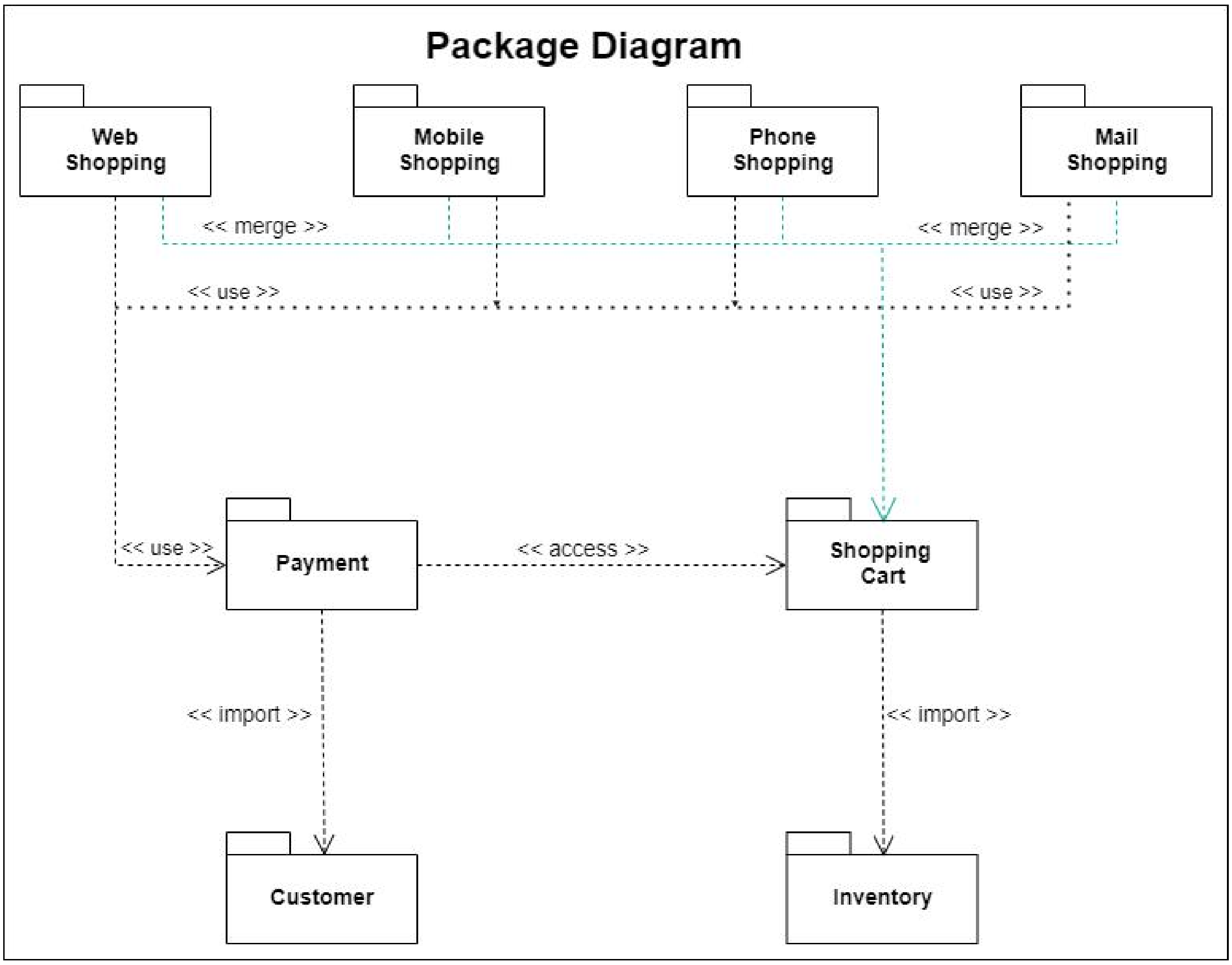
Swim-Lane Diagram:

A swimlane diagram is a form of flowchart that shows who is responsible for what in a process. A swimlane diagram, which uses the metaphor of pool lanes, promotes clarity and accountability by arranging process stages within the horizontal or vertical "swimlanes" of a specific person, work group, or department. It depicts the connections, communication, and handoffs that occur between various lanes, and it may be used to reveal waste, redundancy, and inefficiency in a process.



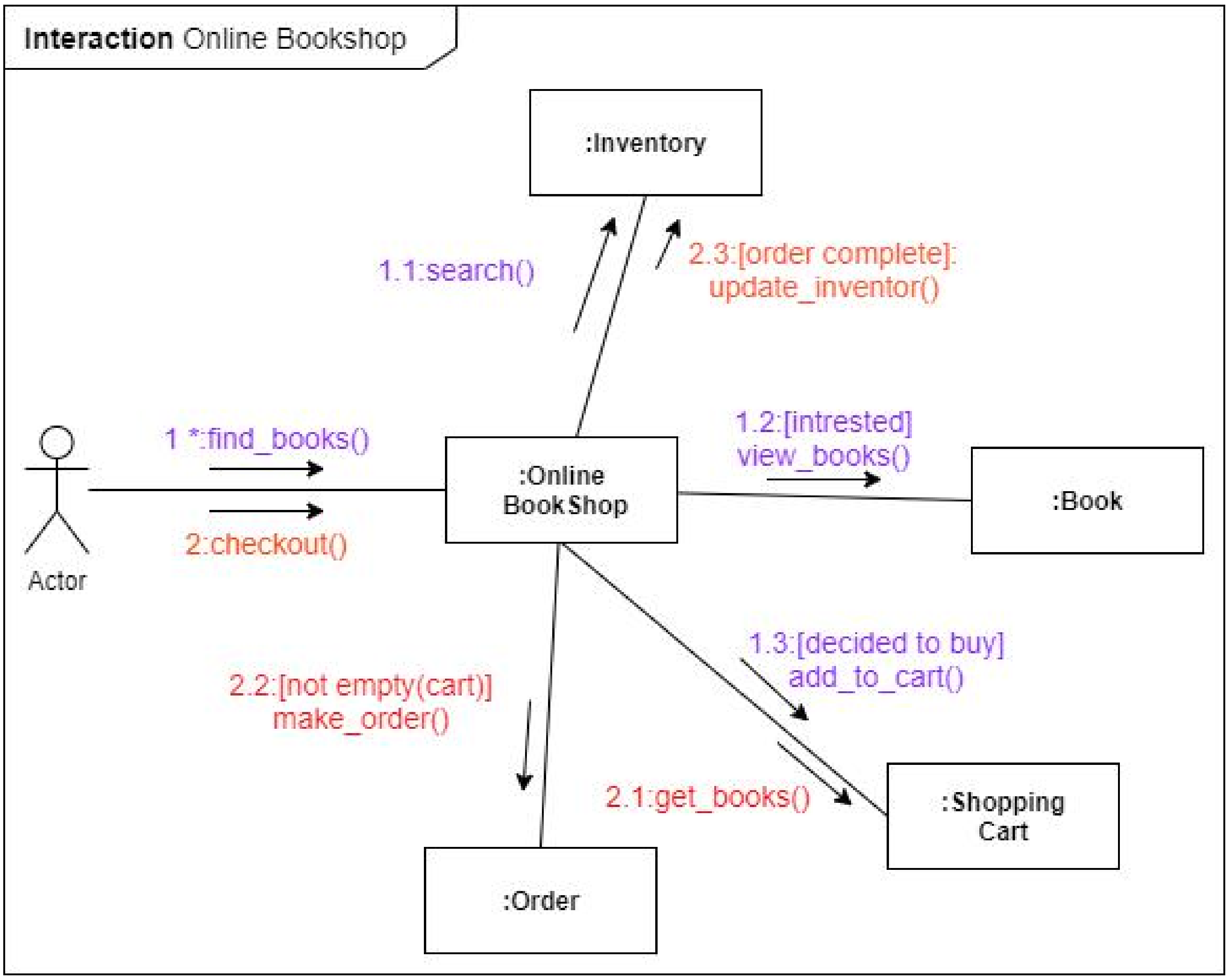
Name:-Atharva Gangrade
Enrollment No: EN19CS305014

Package Diagram:
Package Diagrams are used to show how packages and their elements are arranged. A package diagram simply displays the relationships between distinct packages as well as the internal structure of packages. Packages assist us in organising UML diagrams into meaningful groupings and making the diagram more understandable. They are typically employed in the organisation of class and use case diagrams.



Name:-Atharva Gangrade
Enrollment No: EN19CS305014

Communication Diagram:
They were known as collaboration diagrams in UML 1. Communication diagrams are similar to sequence diagrams in that they focus on messages that are passed between objects. A sequence diagram and other objects can be used to express the same information.


Interaction Online Bookshop
:Inventory
1.1:search()
2.3:[order complete]: update_inventor()
1 *:find_books()
2:checkout()
Actor
:Online Book Shop
1.2:[intrested] view_books()
:Book
1.3:[decided to buy] add_to_cart()
2.2:[not empty(cart)] make_order()
2.1:get_books()
:Shopping Cart
:Order