GLOBAL STANDARDS GIET
An Autonomous Institution
NBA Accredited & NAAC A+

GODAVARI INSTITUTE OF ENGINEERING AND TECHNOLOGY

Department of **COMPUTER SCIENCE AND ENGINEERING**

Name:_________________________________ Pin No: | | | | | | | | | | |

CERTIFICATE

Certified that this is the bonafide record of practical work done by

Mr./Ms.____

a student of_____________________________________with Pin No:_____________________

in the_____________________________Laboratory during the Academic year_____________

No. of Experiments Conducted: | | | No. of Experiments attended: | | |

Faculty In-charge   Head of the Department

Submitted for Practical Examination Conducted on___________

Examiner – 1   Examiner – 2

INDEX

Exp No	Date	Name of the Experiment	Page No.	Signature
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				

EXERCISE-1: DRAW E-R DIAGRAMS, DFD FOR THE PROJECT.

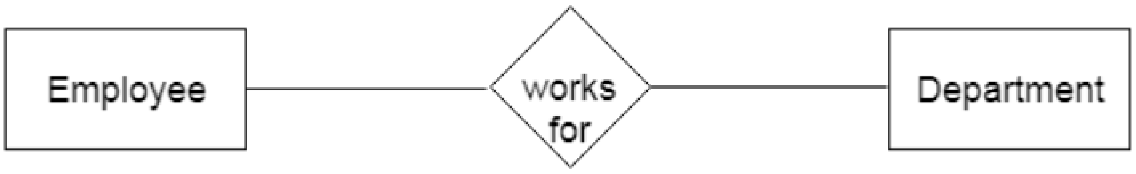
ER DIAGRAMS:

ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system. In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

ENTITY:

An **entity** may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

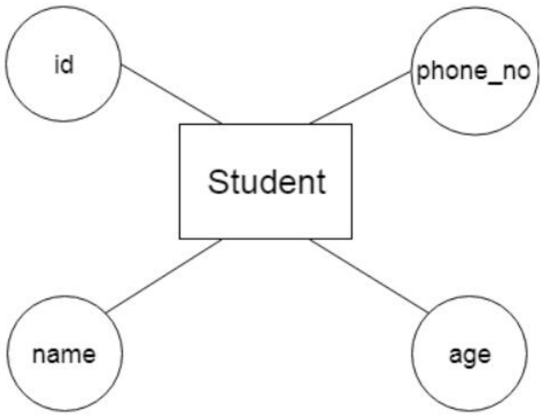
EX: Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



ATTRIBUTE:

The **attribute** is used to describe the property of an entity. Eclipse is used to represent an attribute.

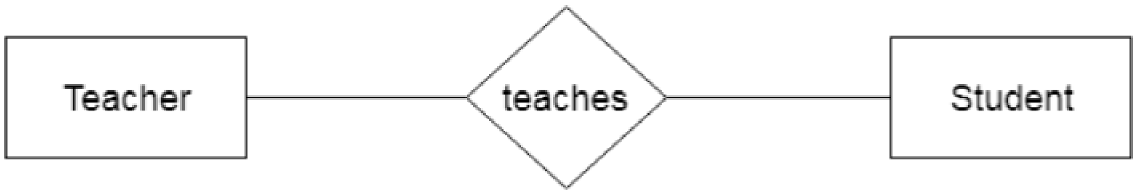
EX: id, age, contact number, name, etc. can be attributes of a student.



RELATIONSHIP:

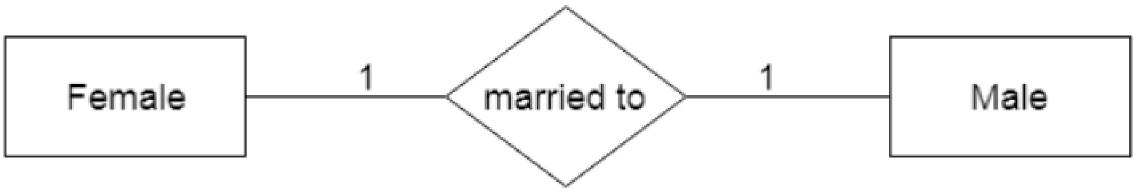
Relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

EX:

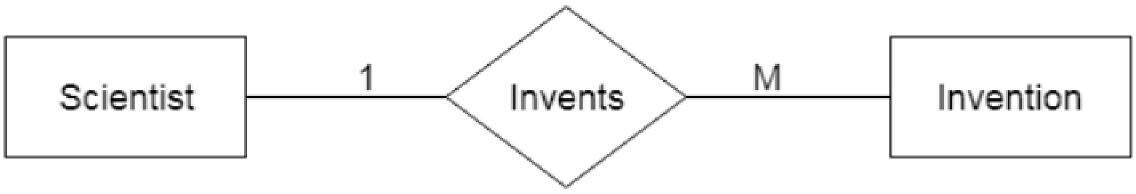


Types of relationship are as follows:

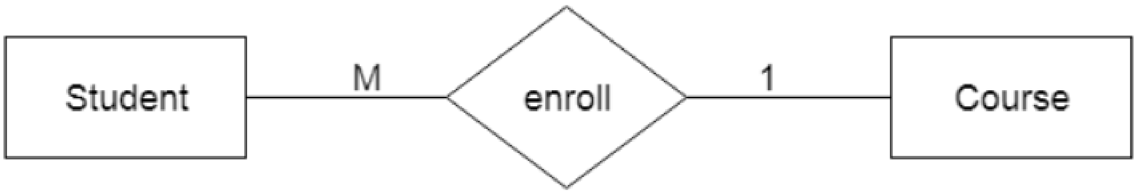
a. **One-to-One Relationship:** When only one instance of an entity is associated with the relationship, then it is known as one-to-one relationship.
EX: A female can marry to one male, and a male can marry to one female.



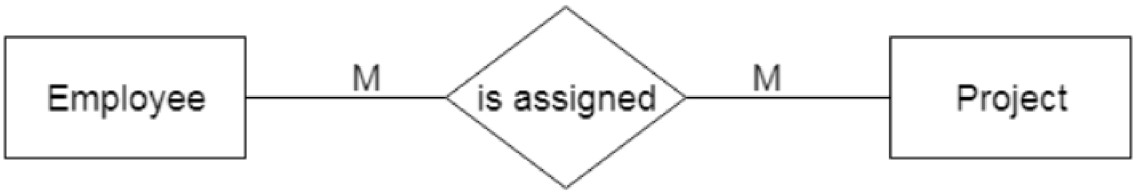
b. **One-to-many relationship:** When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.
EX: Scientist can invent many inventions, but the invention is done by the only specific scientist.



c. **Many-to-one relationship:** When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.
EX: Student enrols for only one course, but a course can have many students.



d. **Many-to-many relationship:** When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.
EX: Employee can assign by many projects and project can have many employees.



Course Registration System:

The course registration system helps the students to gather information about a particular course and then they can easily register themselves in a particular course.
This **ER** (Entity Relationship) Diagram represents the model of Course Registration System Entity. The entity-relationship diagram of Course Registration System shows all the visual instrument of database tables and the relations between Fees, Students, Course, Trainers etc.

Course Registration System **entities and their attributes**:

Course Entity: Attributes of Course are course_id, course_student_id, course_registration, course_name, course_type, course_year, course_description)

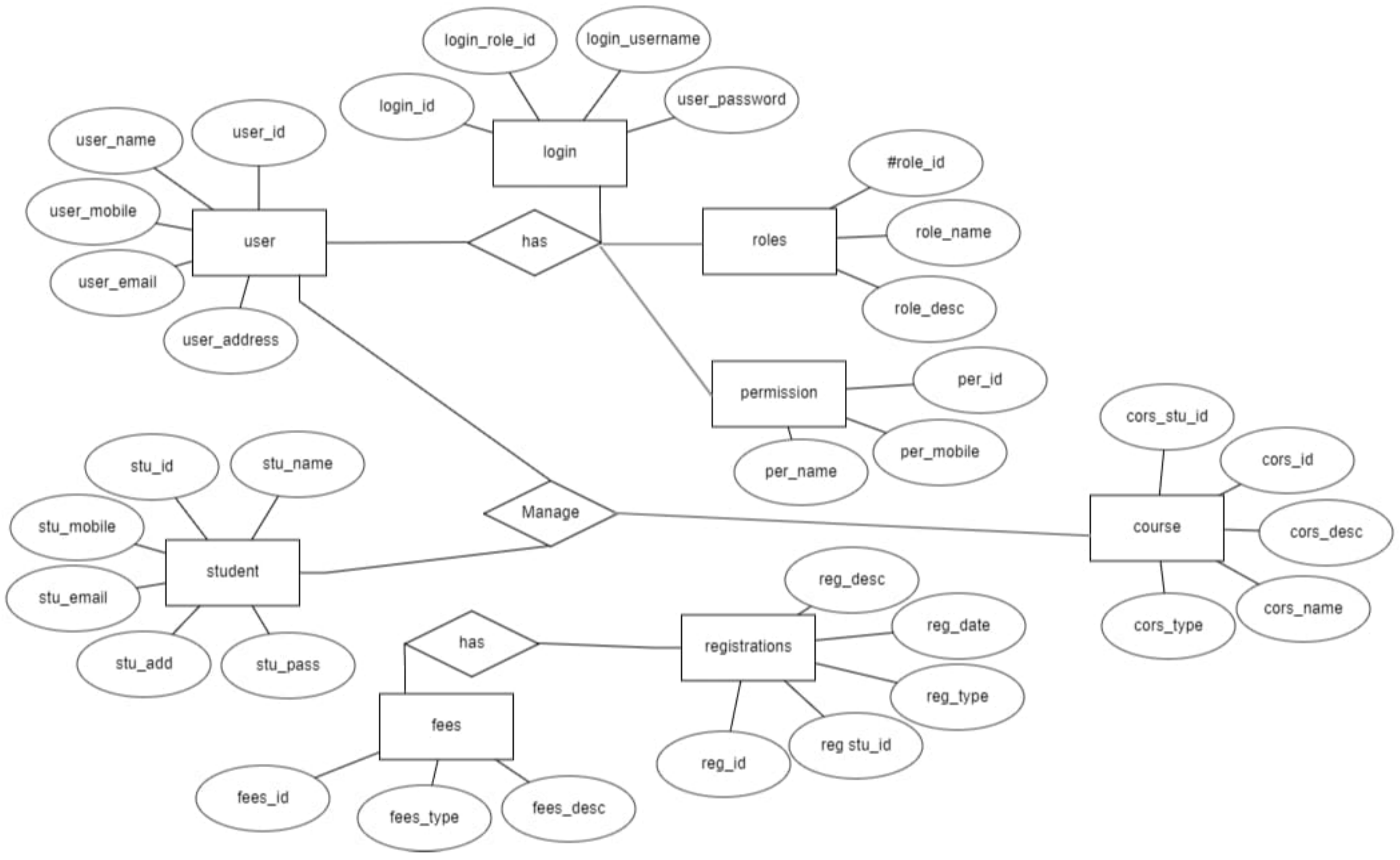
Fees Entity: Attributes of Fees are fees_id, fees amount, fees_type, fees_description

Syllabus Entity: Attributes of Syllabus are syllabus_id, syllabus_course_id, syllabus_name, syllabus_type, syllabus_description

Students Entity: Attributes of Students are student_id, student_college_id, student_name, student_mobile, student_email, student_username, student password, student_address

Registrations Entity: Attributes of Registrations are registration_id, registration_student_id, registration_name, registration_type, registration_number, registration_date, registration_description

Trainers Entity: Attributes of Trainers are trainer_id, trainer_course_id, trainer_name, trainer_mobile, trainer_email, trainer_usemame, trainer_password, trainer_address



Students Marks Analysing System:

Students marks analysing system has been designed to carry out the mark analysis process in an educational institution. The results of respective departments can be efficiently computed without much of manual involvement. This **ER** (Entity Relationship) Diagram represents the model of Students Marks Analysing System Entity The entity-relationship diagram of Students Marks Analysing System shows all the visual instrument of database tables and the relations between Class Exam, Student Teacher etc.

Result Management System **entities and their attributes**:

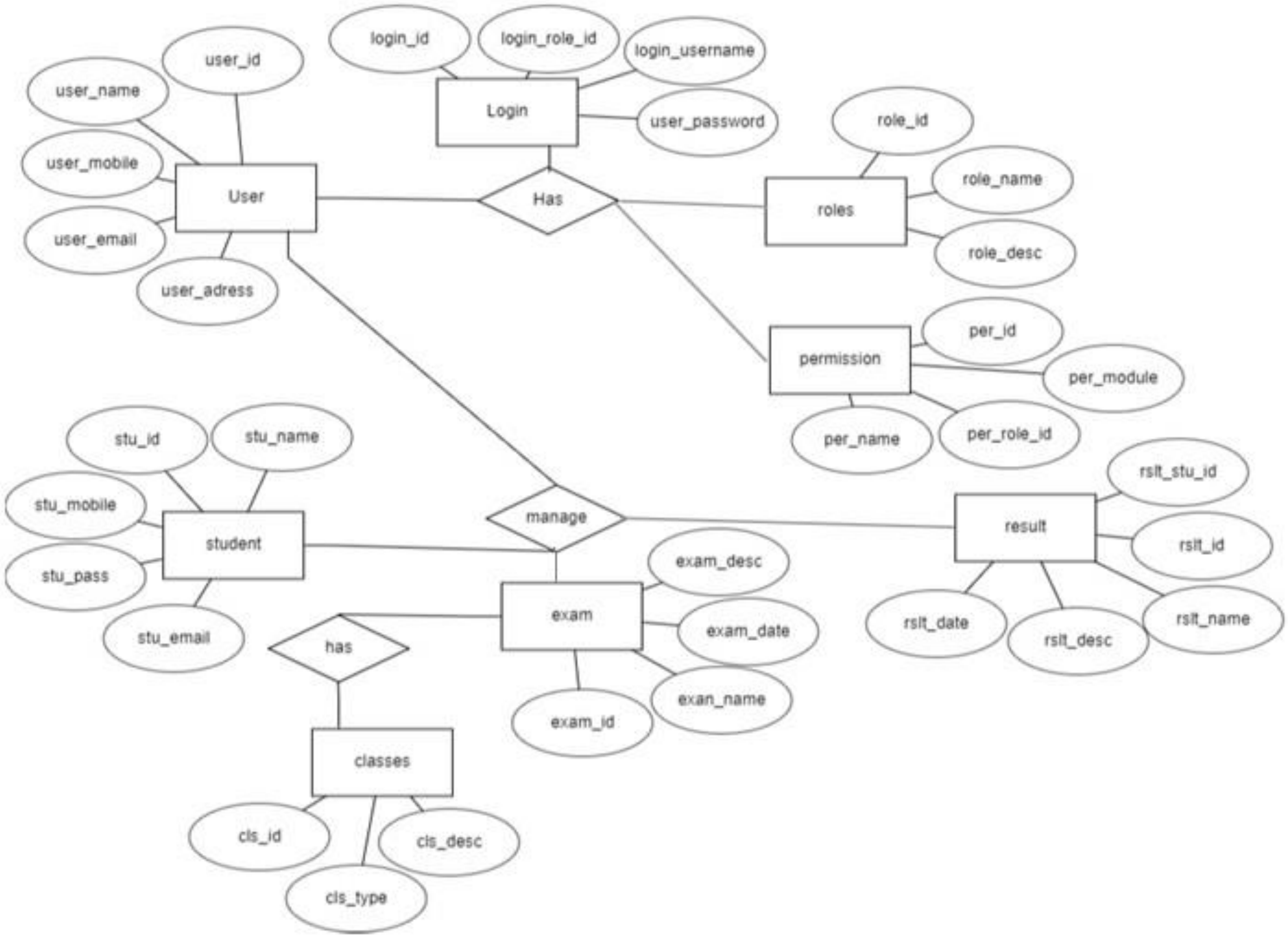
Student Entity: Attributes of Student are student_id, student_college_id, student_name, student mobile, student email, student username, student password, student address

Class Entity: Attributes of Class are class id, class student id, class name, class room, class type, class description

Subject Entity: Attributes of Subject are subject id, subject_course_id, subject student_id, subject_name, subject type, subject_description

Exam Entity: Attributes of Exam are exam id, exam student_id, exam roll number, exam_date exam_name, exam_type exam_description Result Entity Attributes of Result are result id, result student_id, result_name, result_description

Teacher Entity: Attributes of Teacher are teacher_id, teacher_college_id, teacher_name, teacher_mobile, teacher_email, teacher_username teacher password, teacher address



Online Ticket Reservation System:

An online reservation system is a software solution that allows customers to book their reservations or appointments online via a company's website or app instead of over the phone or in person. This **ER** (Entity Relationship) Diagram represents the model of Ticket Reservation System Entity. The entity-relationship diagram of Ticket Reservation System shows all the visual instrument of database tables and the relations between Seats Availability, Stations, Trains, Passengers etc.

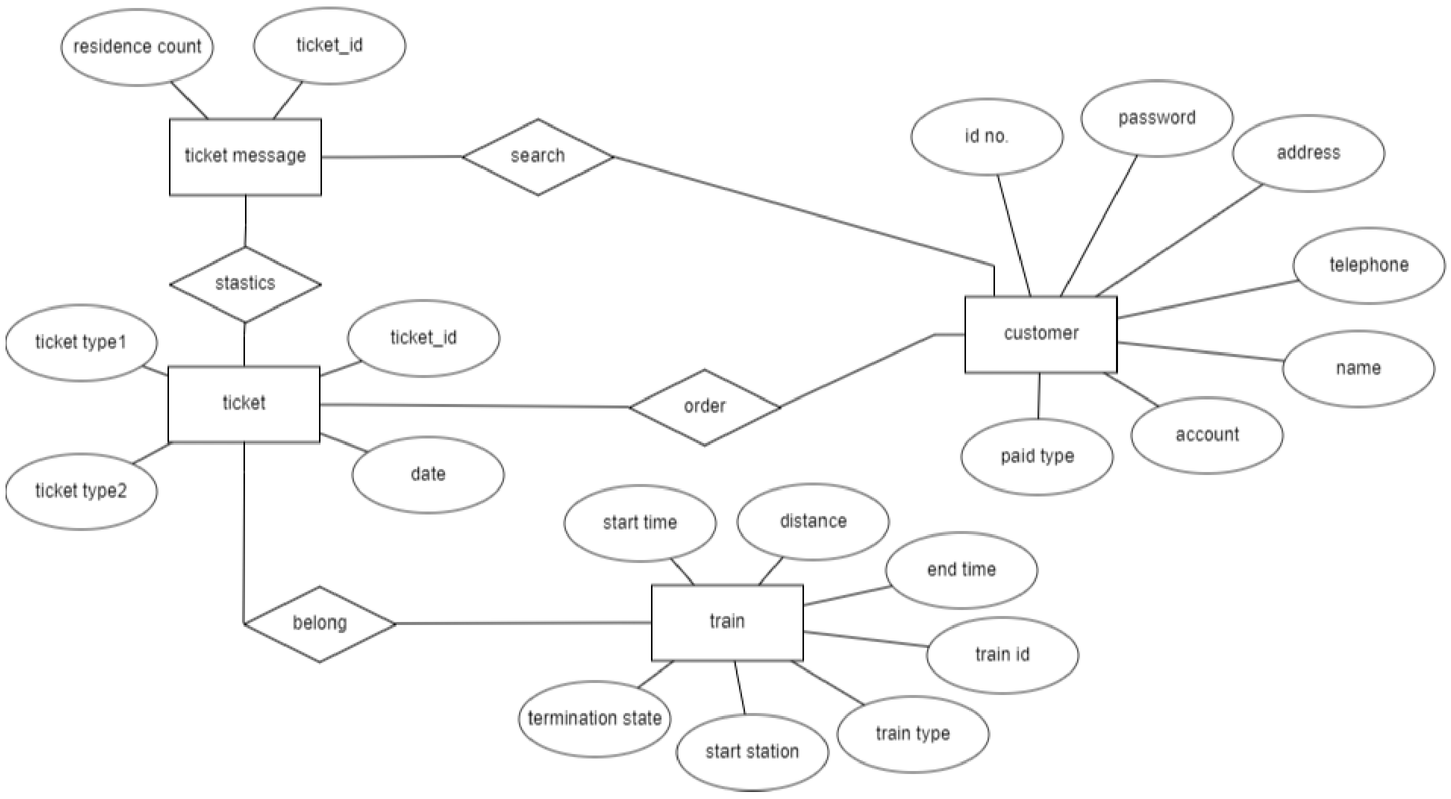
Ticket Reservation System **entities and their attributes**:

Trains Entity: Attributes of Trains are train_id, train_name, train_number, train seat number, train ticket, train_type, train_description • Seats Availability Entity: Attributes of Seats Availability are seat_id, seat_train_id, seat_customer_id, seat_number, seat_type, seat_description

Fare Entity: Attributes of Fare are fare_id, fare_ticket_id, fare_title, fare_type, fare_description

Stations Entity: Attributes of Stations are station_id, station_name, station_type, station description. Booking Entity: Attributes of Booking are booking_id, booking_ticket_id, booking_title, booking type, booking_date, booking_description

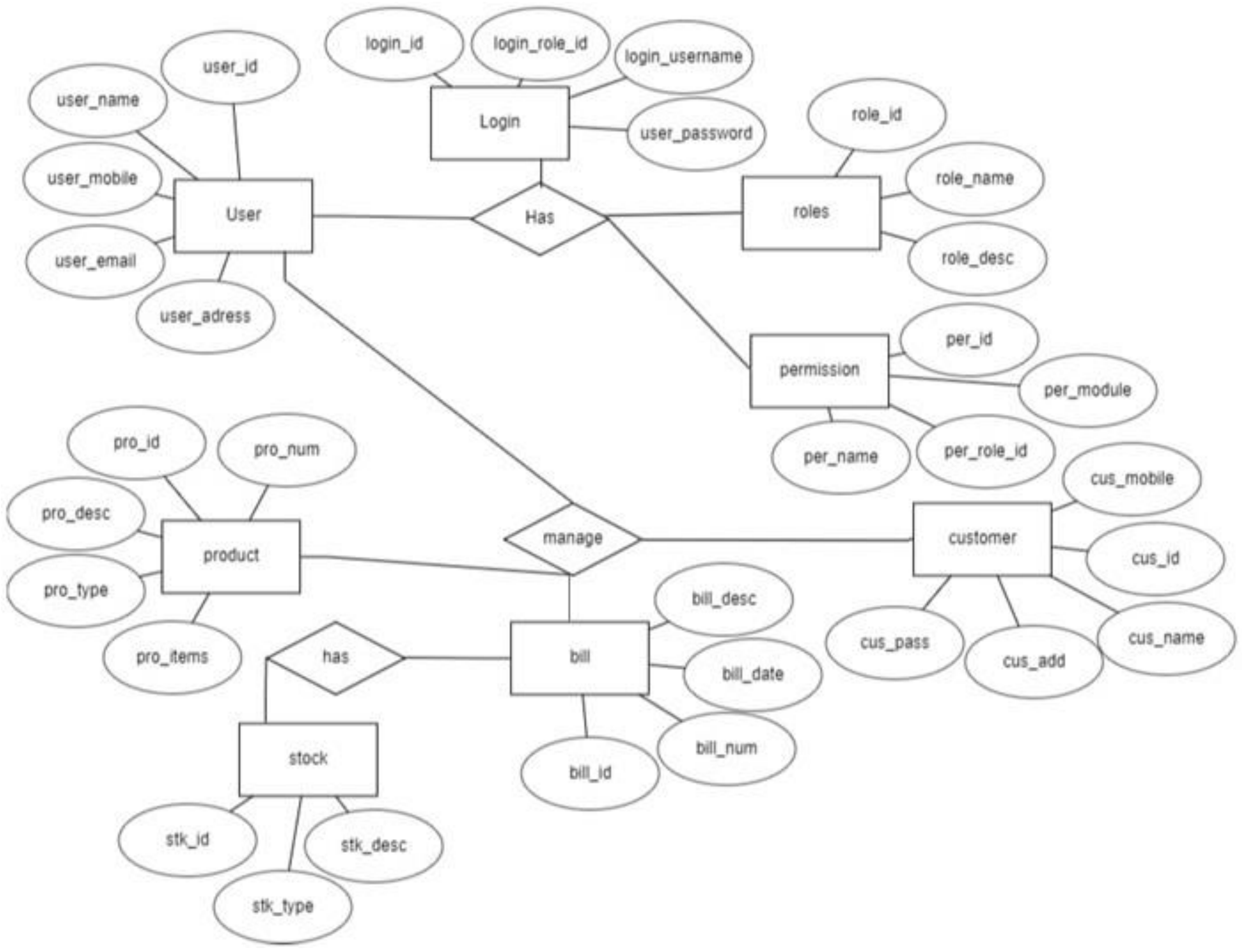
Passengers Entity: Attributes of Passengers are passenger_id, passenger_name, passenger_mobile, passenger_email, passenger_username. passenger_password, passenger_address



Stock Maintenance:

This **ER** (Entity Relationship) Diagram represents the model of Stock Management System Entity The entity relationship diagram of Stock Management System shows all the visual instrument of database tables and the relations between Product Bill, Stock, Store etc.

Stock Management System **entities and their attributes**:
Stock Entity: Attributes of Stock are stock_id, stock items, stock number, stock_type, stock_description
Product Entity: Attributes of Product are product_id, product_customer_id product items, product number, product type product description
Product Quality: Entity Attributes of Product Quality are product quality_id product_quality_name, product quality_type product quality description
Bill Entity: Attributes of Bill are bill_id, bill customer_id bill_number, bill type, bill receipt, bill_description
Customer Entity: Attributes of Customer are customer_id, customer_name, customer_mobile, customer email, customer_usemame customer password, customer_address
Store Entity: Attributes of Store are store_id, store_name, store type, store_description

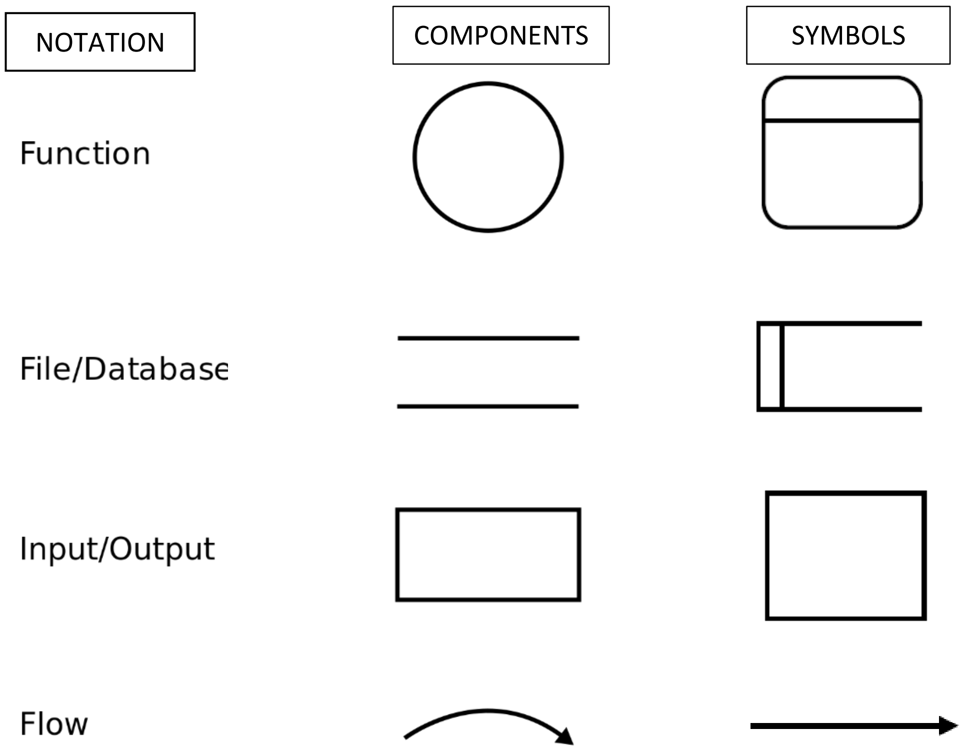


DATA FLOW DFD:

A **Data Flow Diagram** (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. They can be used to analyse an existing system or model a new one.
For each data flow, at least one of the endpoints (source and / or destination) must exist in a process. The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes. The data-flow diagram is a tool that is part of **structured analysis** and **data modelling**.

SYMBOLS AND NOTATIONS USED IN DFD'S:

Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams.



Function/Process: any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as "Submit payment."

Database/Data store: files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as "Orders."

(Input/Output)/External entity: an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

Flow/Data flow: the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labelled with a short data name, like "Billing details."

RULES FOR DATA FLOW DIAGRAM:

1) **Data cannot flow between two entities**.

Data flow must be from entity to a process or a process to an entity. There can be multiple data flows between one entity and a process.

2) **Data cannot flow between two data stores.**

Data flow must be from data store to a process or a process to a data store. Data flow can occur from one data store to many processes.

3) **Data cannot flow directly from an entity to data store.**

Data Flow from entity must be processed by a process before going to data store and vice versa.

4) **A process must have at least one input data flow and one output data flow.**

Every process must have input data flow to process the data and an output data flow for the processed data.

5) **A data store must have at least one input data flow and one output data flow.**

Every data store must have input data flow to store the data and an output data flow for the retrieved data.

6) **Two data flows cannot cross each other.**
7) **All the process in the system must be linked to minimum one data store or any other process.**

LEVELS IN DATA FLOW DIAGRAMS (DFD):

DFD (Data Flow Diagram) can be drawn to represent the system of different levels of abstraction. Higher-level DFDs are partitioned into low levels-hacking more information and functional elements. Levels in DFD are numbered 0, 1, 2. The main 3 levels in the data flow diagram, are: **0-level DFD**, **1-level DFD**, and **2-level DFD**.

0-level DFD:
It is also known as a context diagram. It’s designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

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

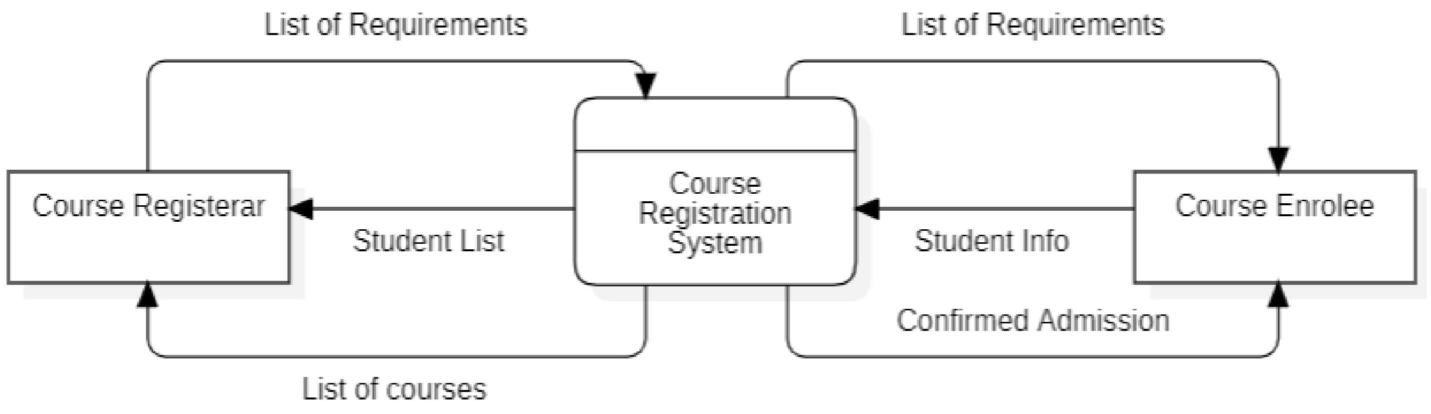
2-level DFD:
2-level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system’s functioning.

Progression to level 3, 4 and more is possible, but usually going beyond level 3 is rare.

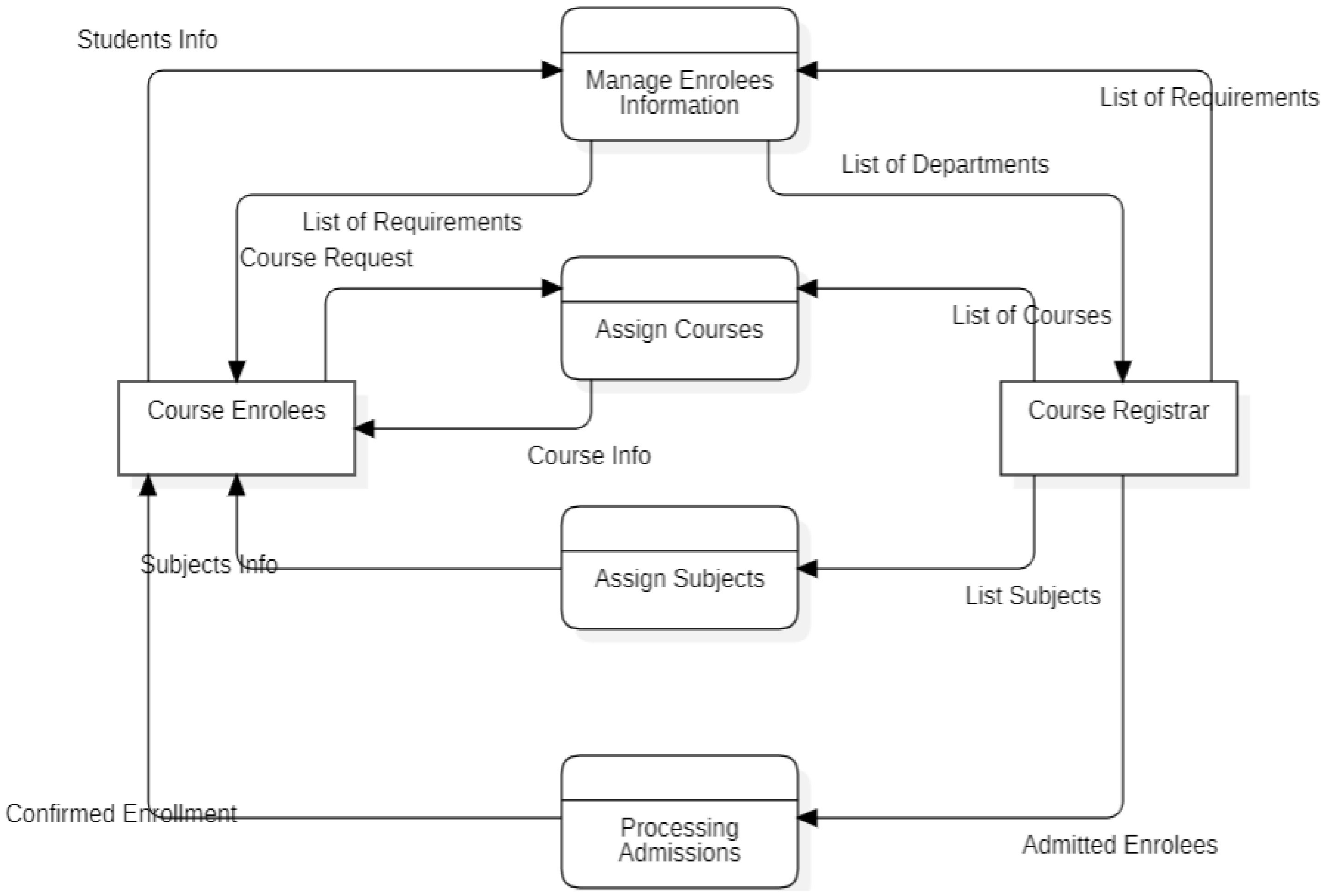
COURSE REGISTRATION SYSTEM:

The course registration system helps the students to gather information about a particular course and then they can easily register themselves in a particular course.

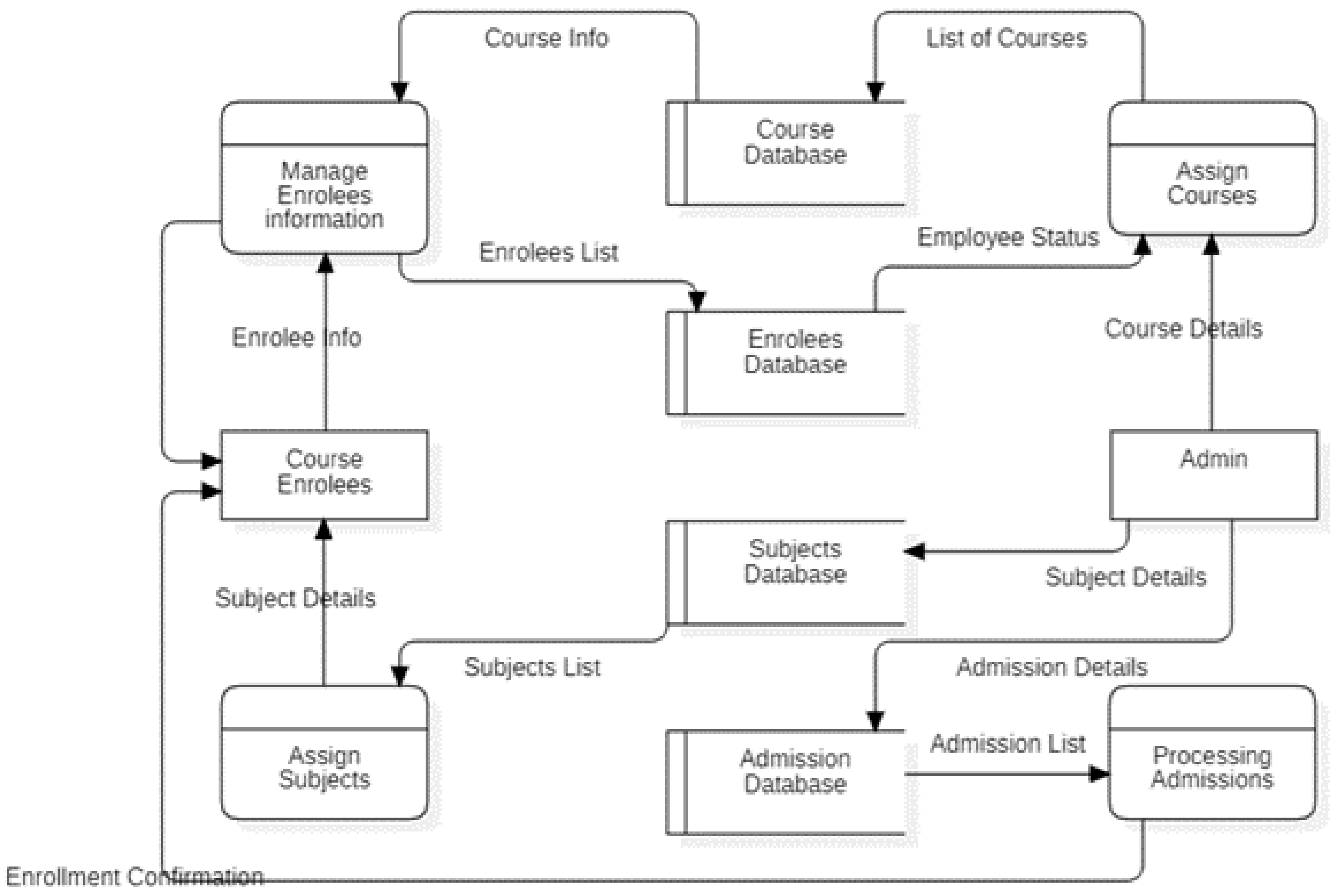
LEVEL-0:



LEVEL-1:



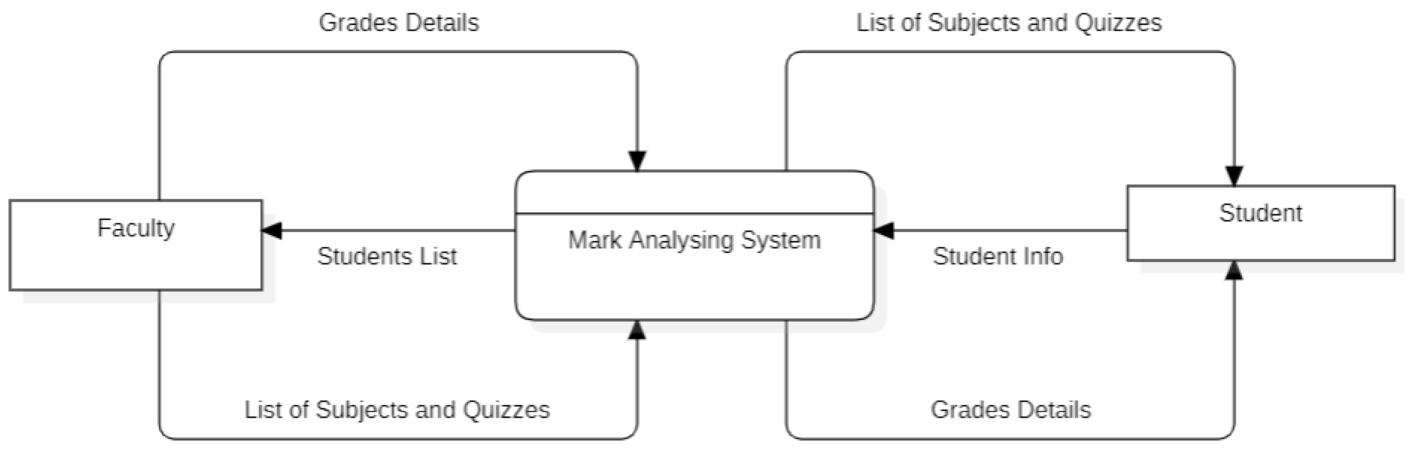
LEVEL-2:



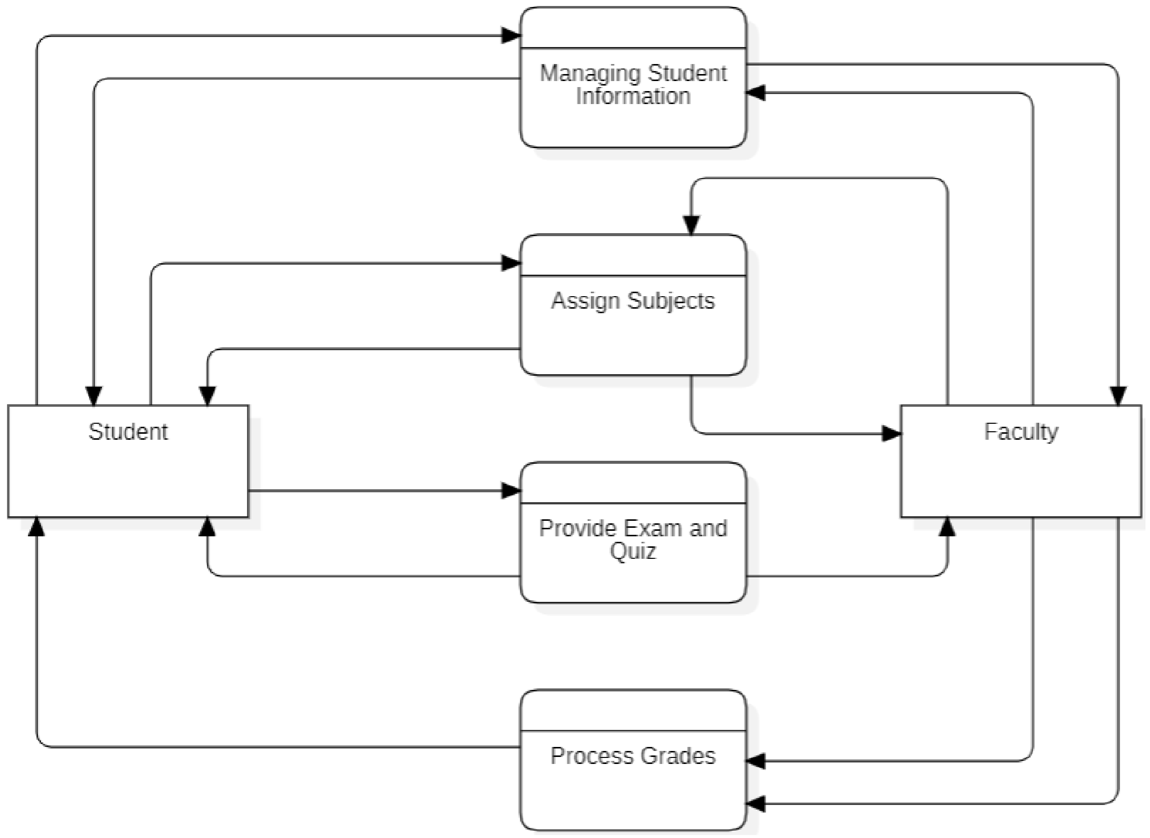
STUDENTS MARKS ANALYZING SYSTEM:

Student marks analysing system has been designed to carry out the mark analysis process in an educational institution. The results of respective departments can be efficiently computed without much of manual involvement.

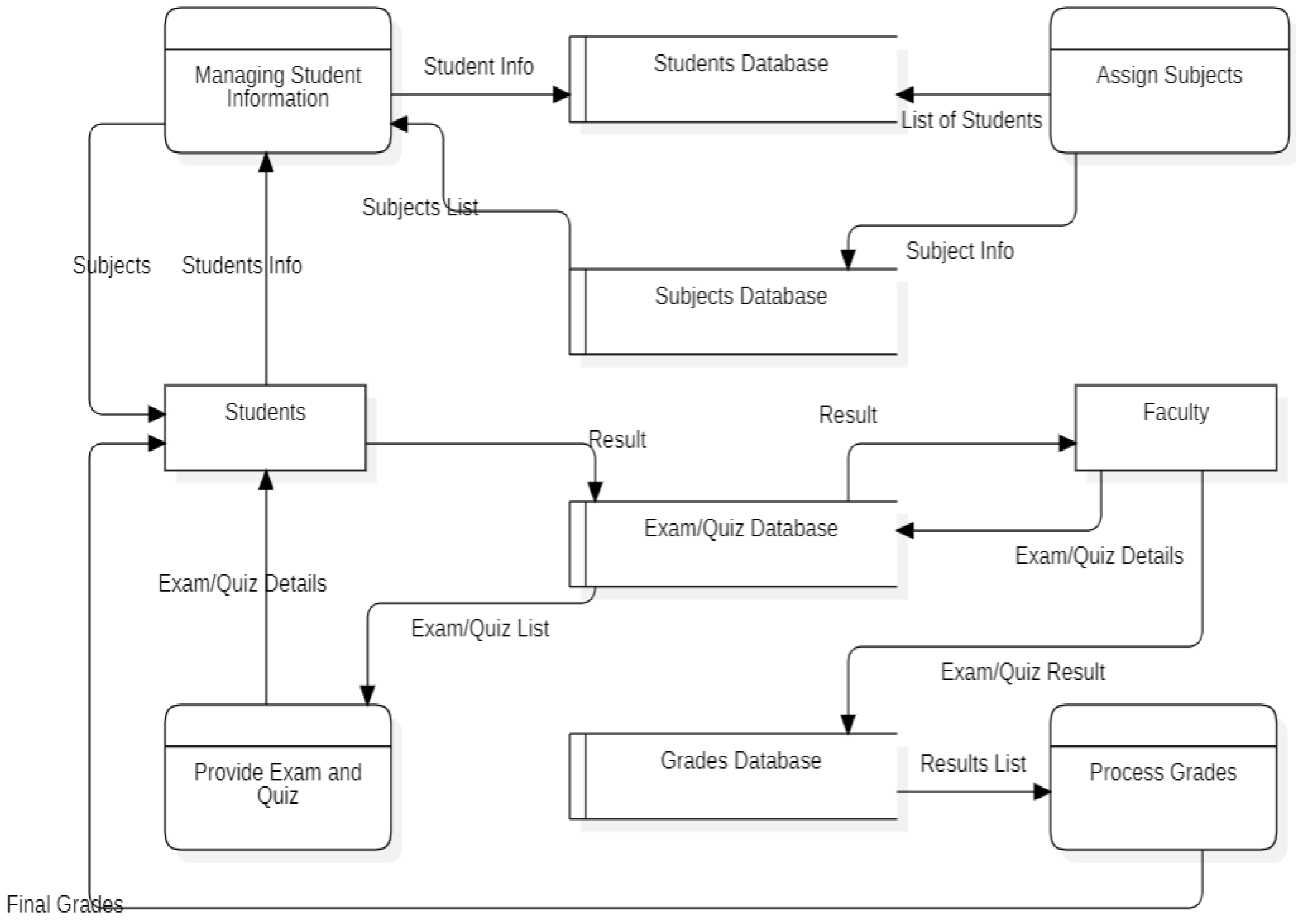
LEVEL-0:



LEVEL-1:



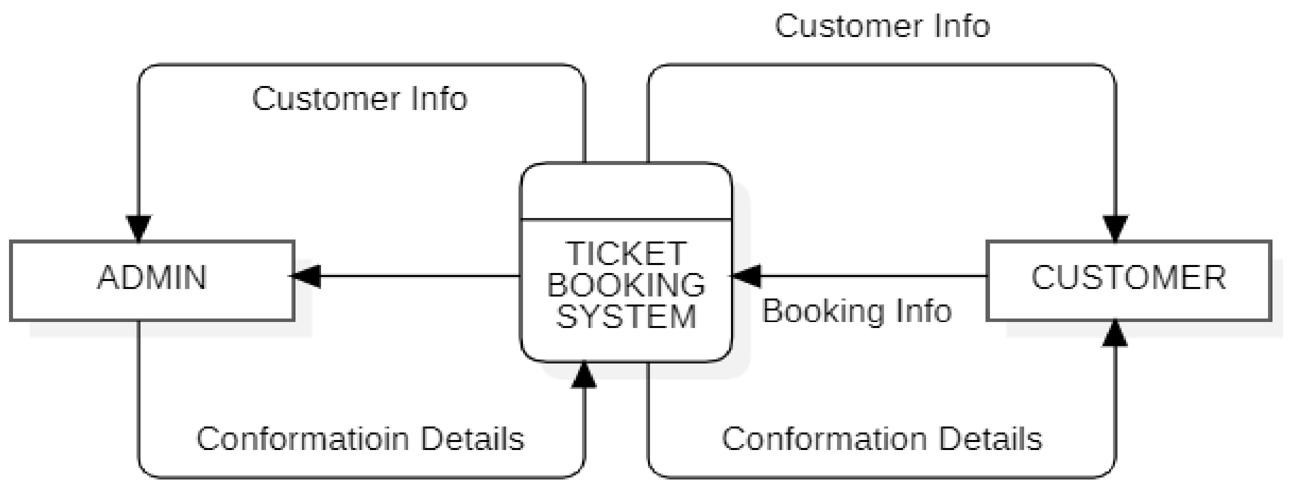
LEVEL-2:



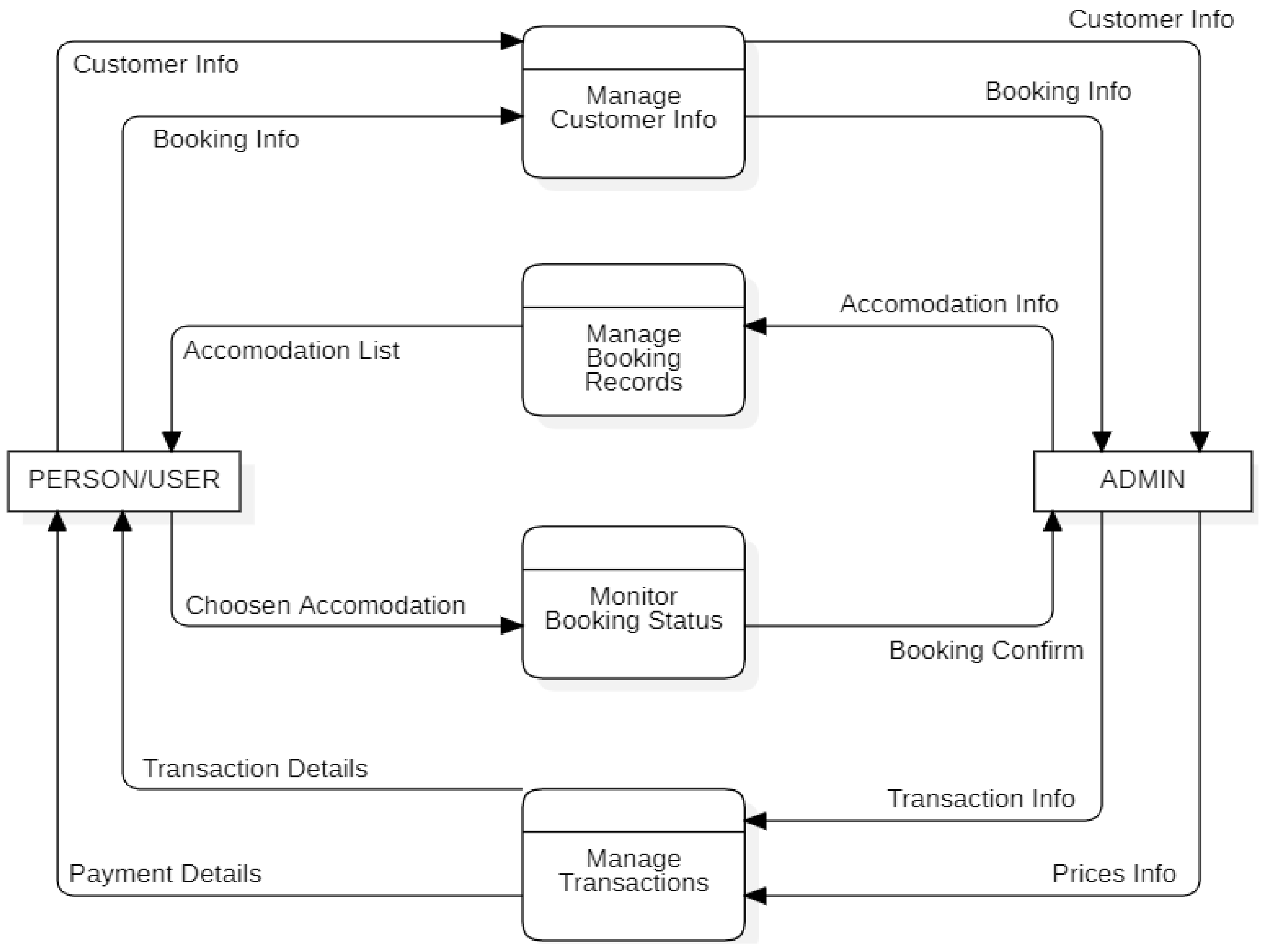
ONLINE TICKET RESERVATION SYSTEM:

An online reservation system is a software solution that allows customers to book their reservations or appointments online via a company's website or app instead of over the phone or in person.

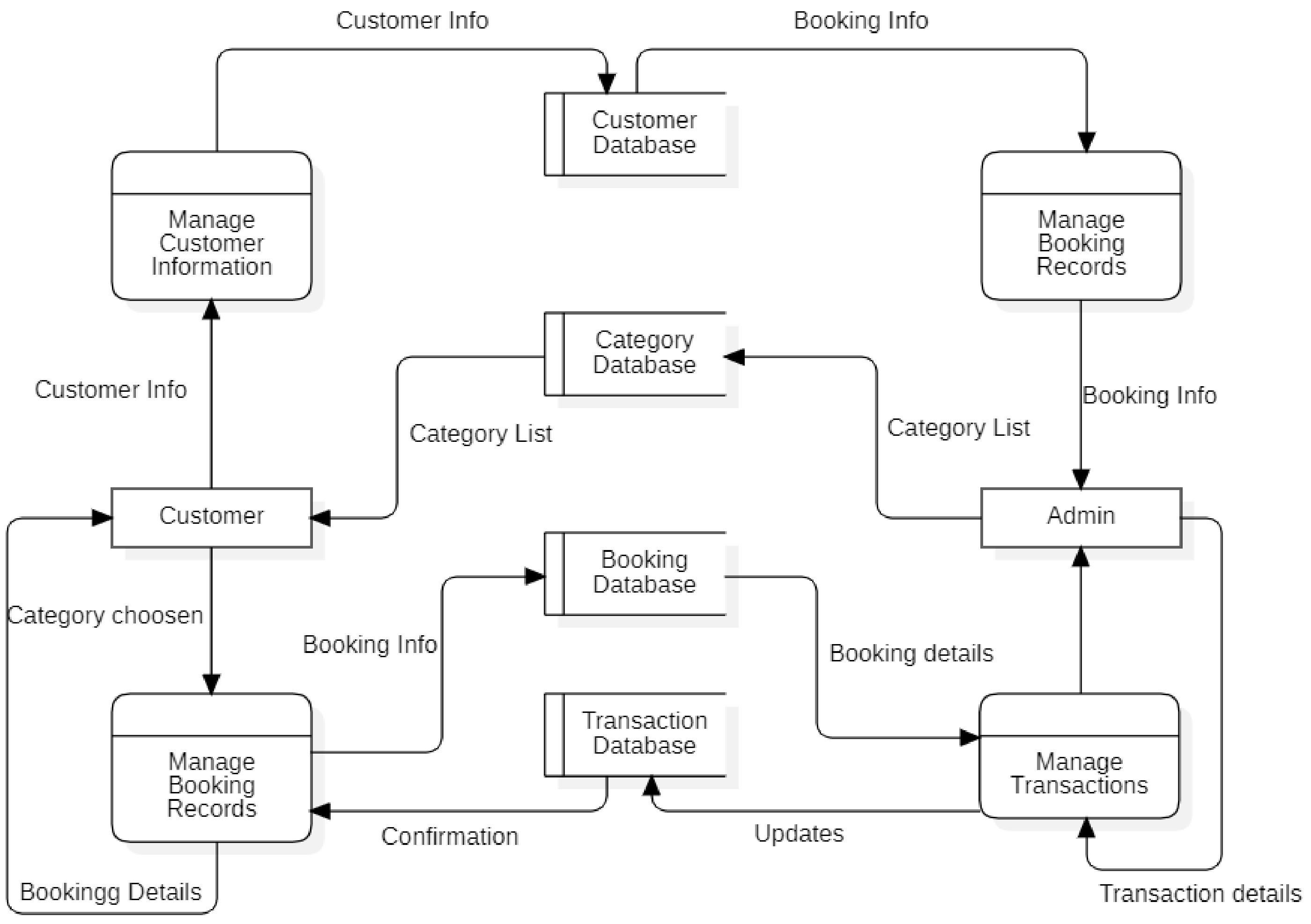
LEVEL-0:



LEVEL-1:



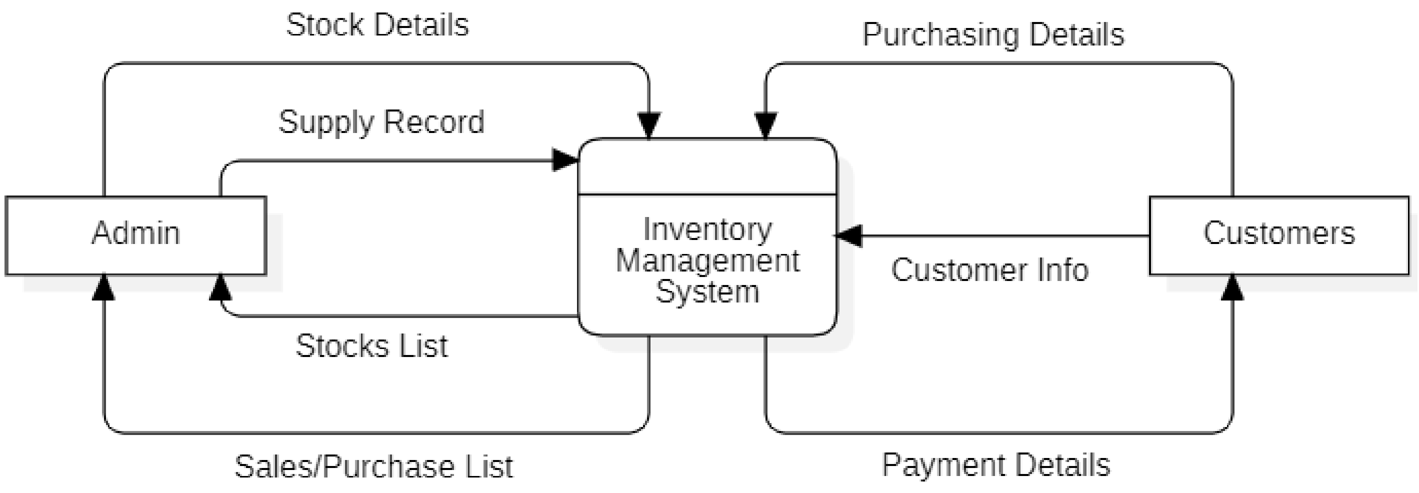
LEVEL-2:



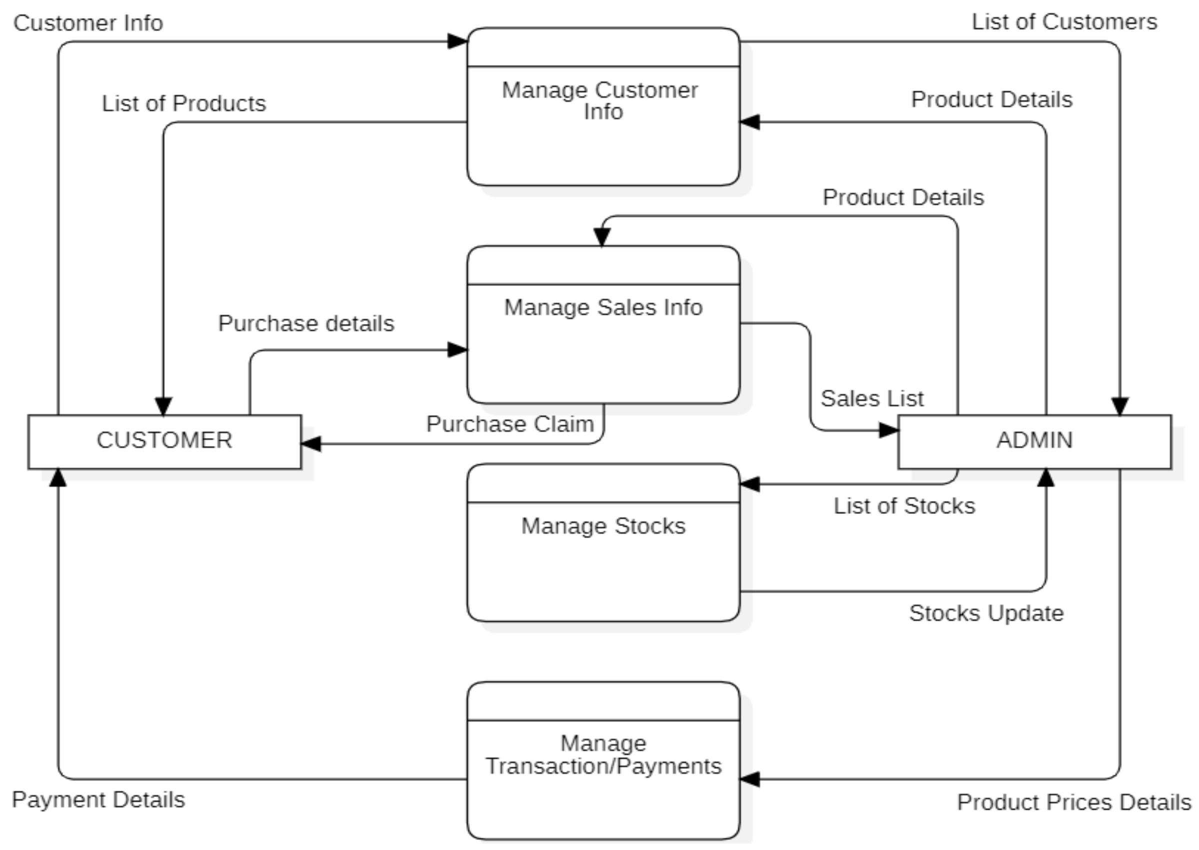
<u>STOCK MAINTENANCE</u>:

Stock management is the practice of ordering, storing, tracking, and controlling inventory.

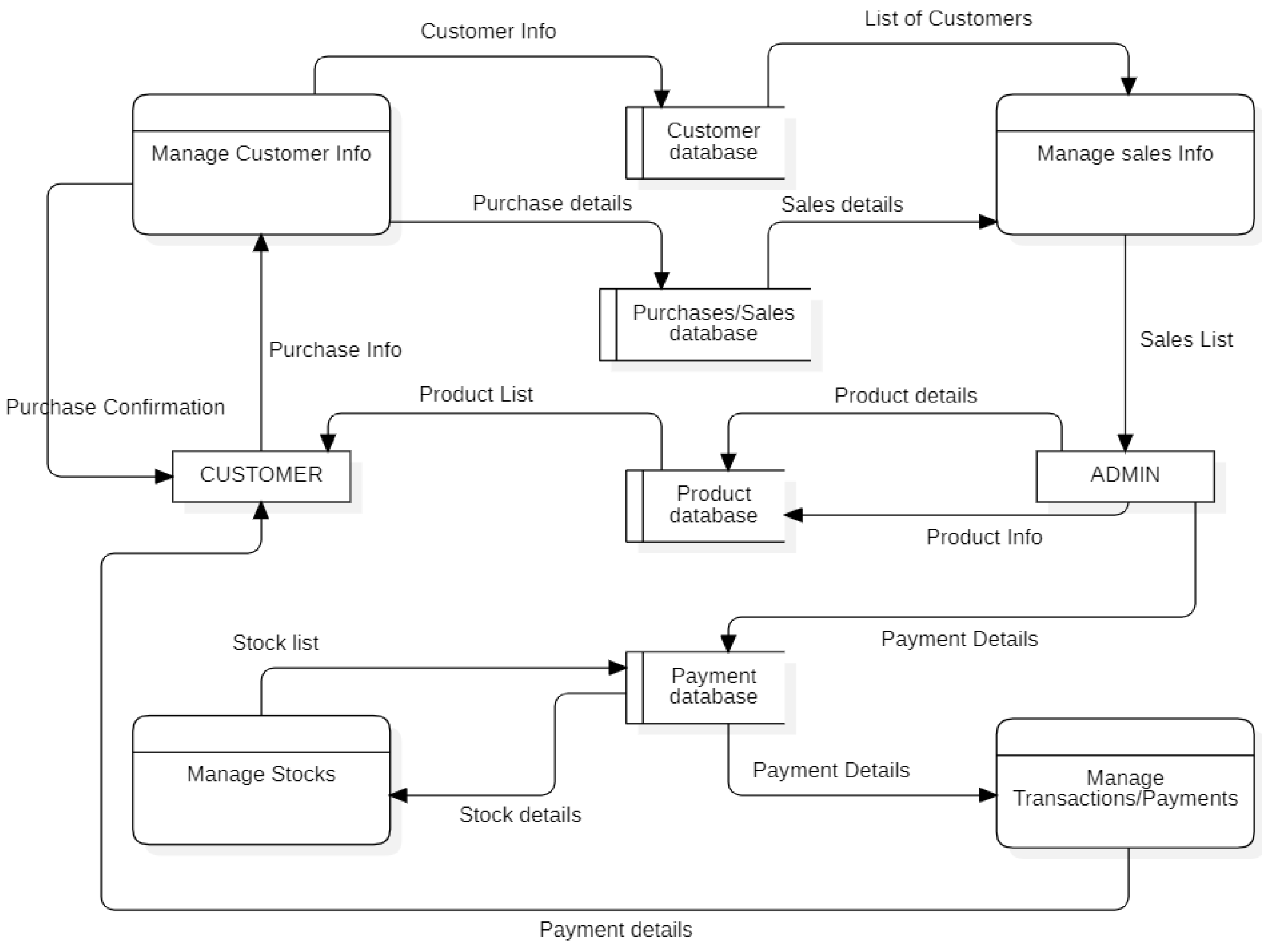
LEVEL-0:



LEVEL-1:



LEVEL-2:



EXERCISE-2: DRAW UML DIAGRAMS FOR THE PROJECT.

UML:

The **Unified Modelling Language** (UML) is a graphical language for OOAD that gives a standard way to write a software system's blueprint. It helps to visualize, specify, construct, and document the artifacts of an object-oriented system. It is used to depict the structures and the relationships in a complex system.

Brief History

It was developed in 1990s as an amalgamation of several techniques, prominently OOAD technique by Grady Booch, OMT (Object Modelling Technique) by James Rumbaugh, and OOSE (Object Oriented Software Engineering) by Ivar Jacobson. UML attempted to standardize semantic models, syntactic notations, and diagrams of OOAD.

Basic Building Blocks (The three building blocks of UML):

- Things
- Relationships
- Diagrams

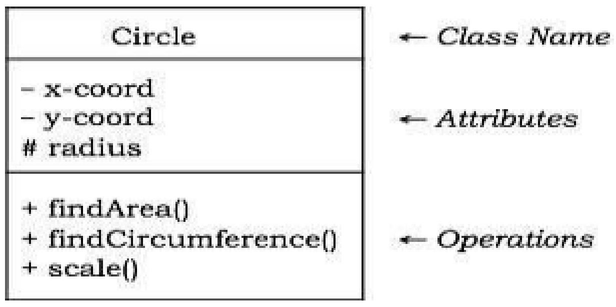
1) **THINGS:** There are four kinds of things in UML, namely:
    - **Structural Things:** These are the nouns of the UML models representing the static element that may be either physical or conceptual. The structural things are class, interface. Etc
    - **Behavioural Things:** These are the verbs of the UML models representing the dynamic behaviour over time and space
    - **Grouping Things:** They comprise the organizational parts of the UML models. There is only one kind of grouping thing, i.e., package.
    - **Annotational Things:** These are the explanations in the UML models representing the comments applied to describe elements.

2) **RELATIONSHIPS:** Relationships are the connection between things. The four types of relationships that can be represented in UML are:
    - **Dependency:** This is a semantic relationship between two things such that a change in one thing brings a change in the other
    - **Association:** This is a structural relationship that represents a group of links having common structure and common behaviour.
    - **Generalization:** This represents a generalization/specialization relationship in which subclasses inherit structure and behaviour from super-classes.
    - **Realization:** This is a semantic relationship between two or more classifiers such that one classifier lays down a contract that the other classifiers ensure to abide by.

3) **DIAGRAMS:** A diagram is a graphical representation of a system. It comprises of a group of elements generally in the form of a graph. UML includes nine diagrams in all, namely:

i. **Class Diagram**
ii. **Use Case Diagram**
iii. **Sequence Diagram**
iv. **State Chart Diagram**
v. **Activity Diagram**
vi. **Object Diagram**
vii. **Component Diagram**
viii. **Deployment Diagram**

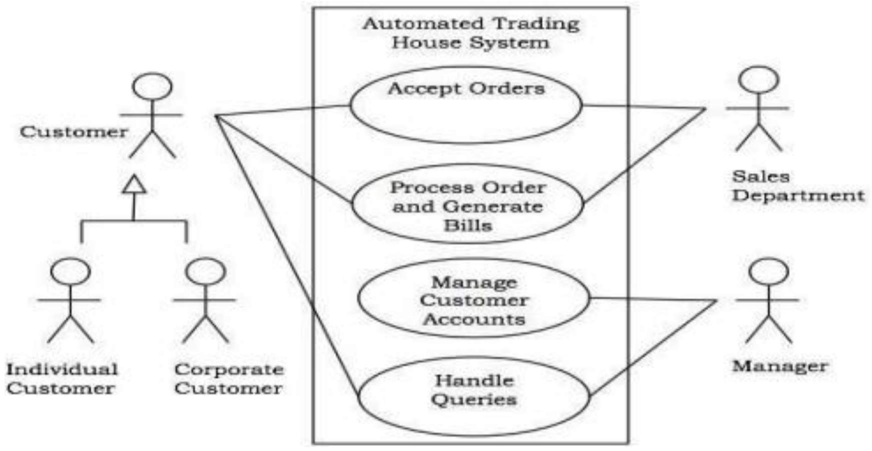
➢ **Class Diagram:** A class diagram models the static view of a system. It comprises of the classes, interfaces, and collaborations of a system; and the relationships between them.



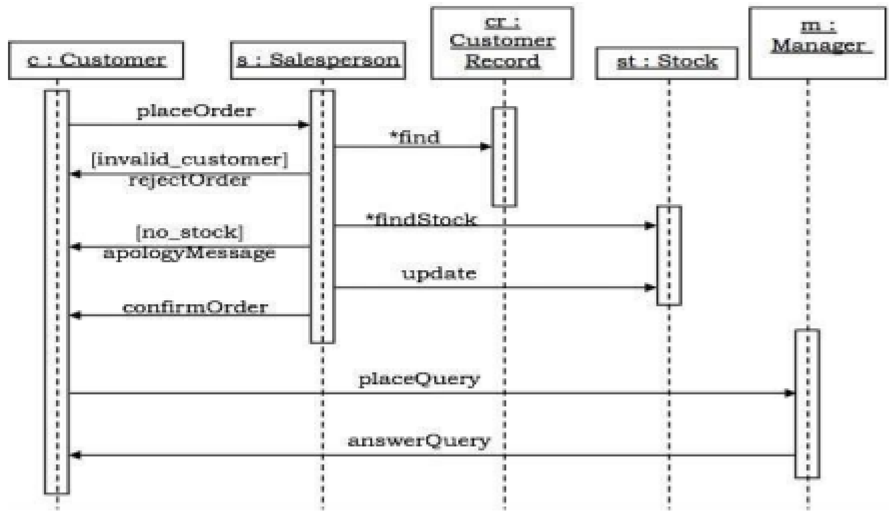
➢ **Use Case Diagram:** Use case diagrams comprise of:

- Use cases
- Actors
- Relationships like dependency, generalization, and association

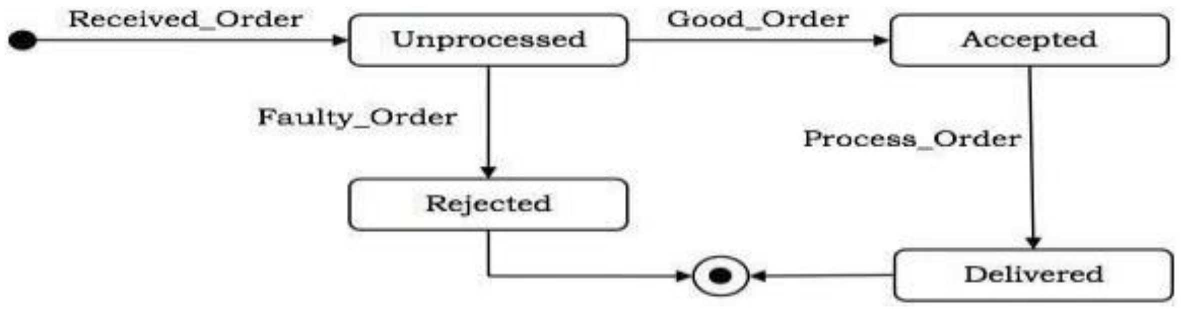
Use case: A use case describes the sequence of actions a system performs yielding visible results. It shows the interaction of things outside the system with the system itself. Use cases may be applied to the whole system as well as a part of the system.
Actor: An actor represents the roles that the users of the use cases play. An actor may be a person (e.g., student, customer), a device (e.g., workstation)



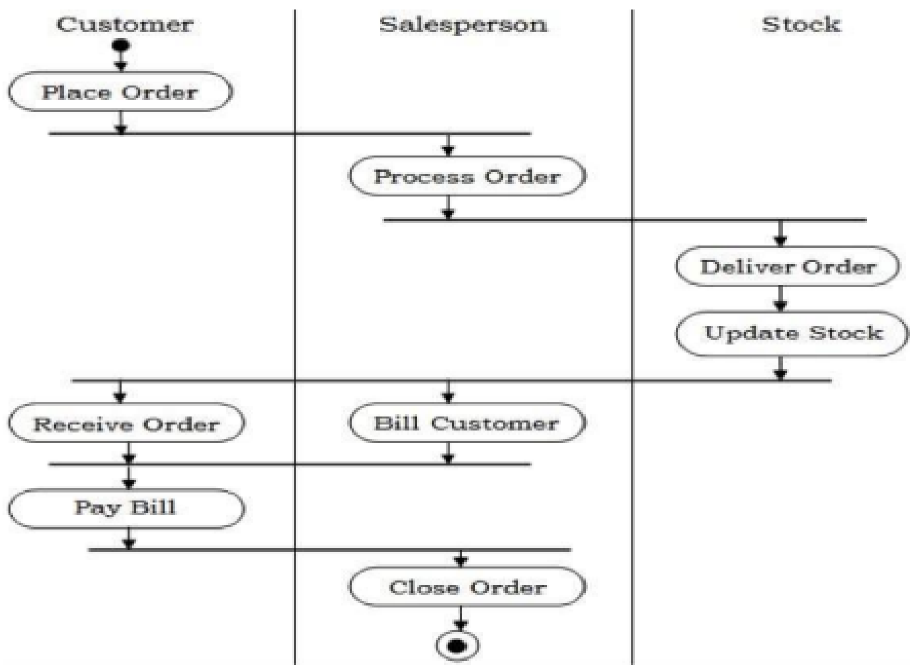
- **Sequence Diagram:** Sequence diagrams are interaction diagrams that illustrate the ordering of messages according to time.



- **State–Chart Diagrams:** A state–chart diagram shows a state machine that depicts the control flow of an object from one state to another. State-chart diagrams are used for modelling objects which are reactive in nature.



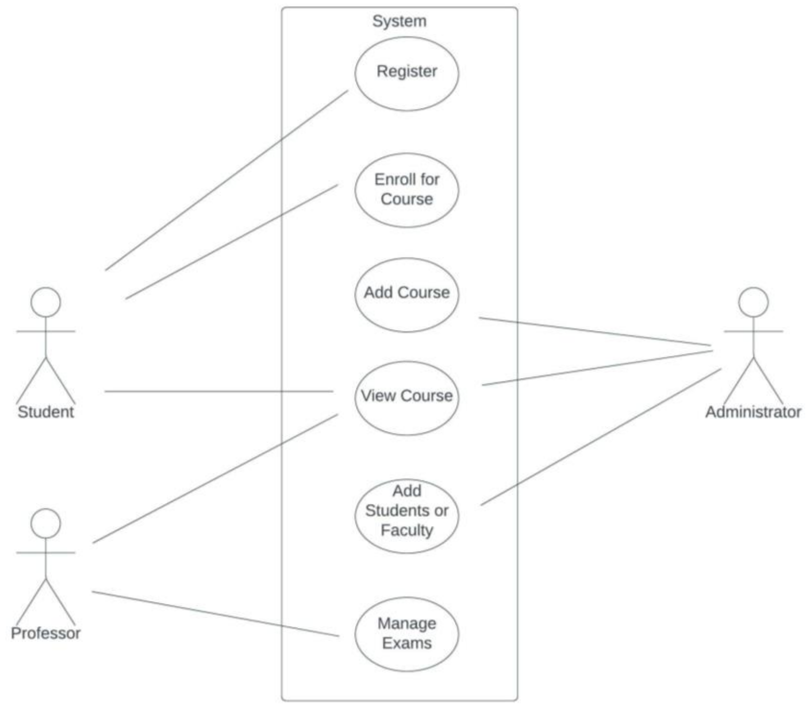
- **Activity Diagrams:** An activity diagram depicts the flow of activities which are ongoing non-atomic operations in a state machine. Activities result in actions which are atomic operations.



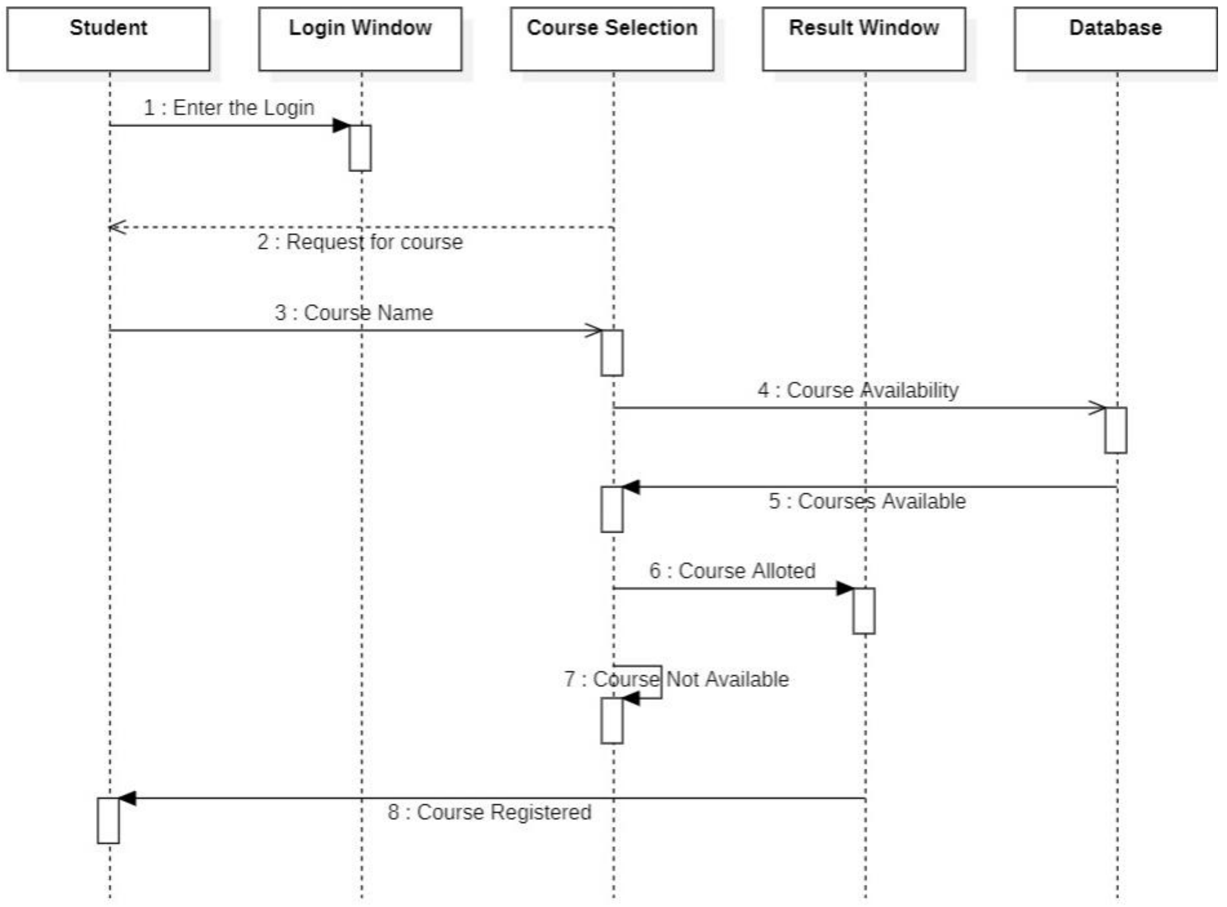
COURSE REGISTRATION SYSTEM :

The course registration system helps the students to gather information about a particular course and then they can easily register themselves in a particular course.

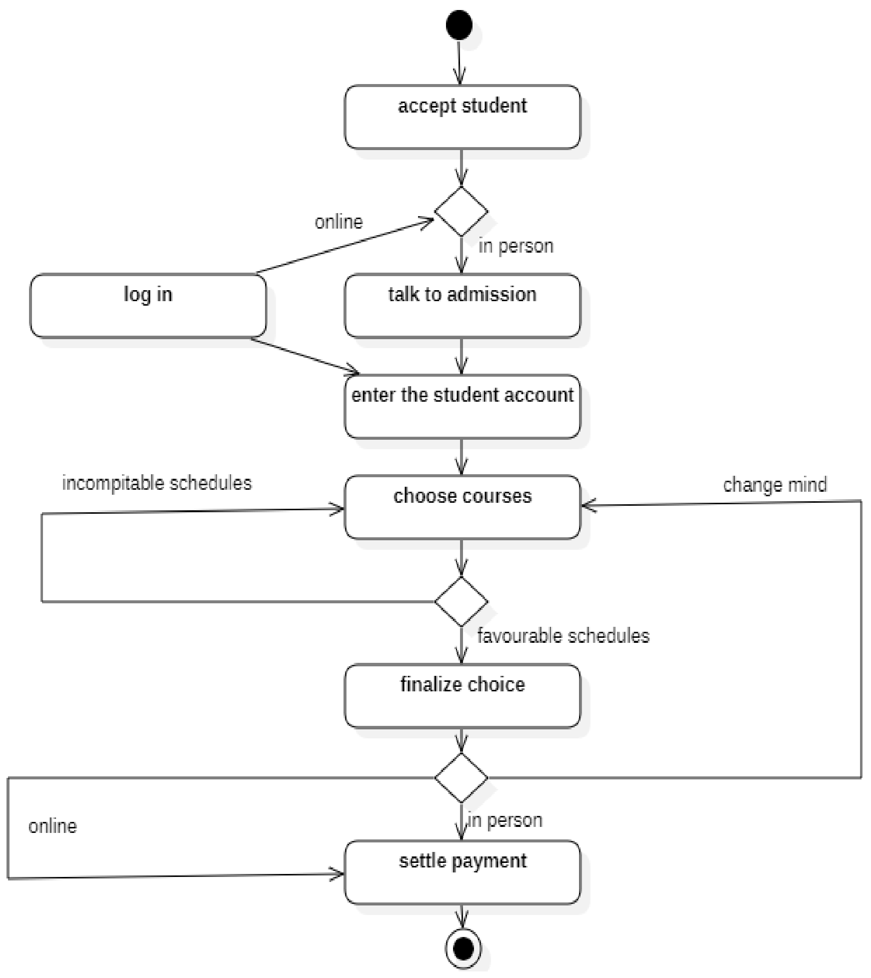
USECASE:



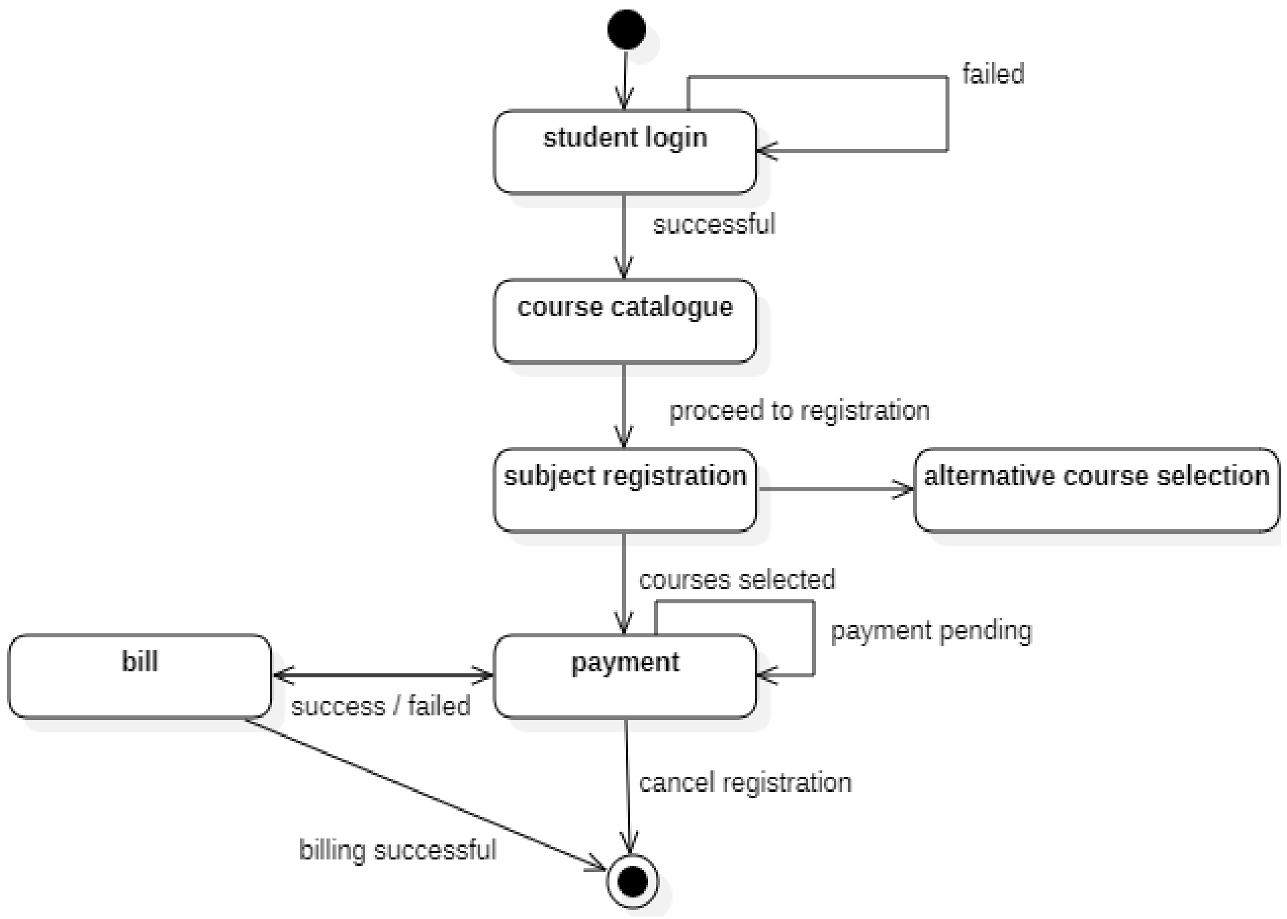
SEQUENCE DIAGRAM:



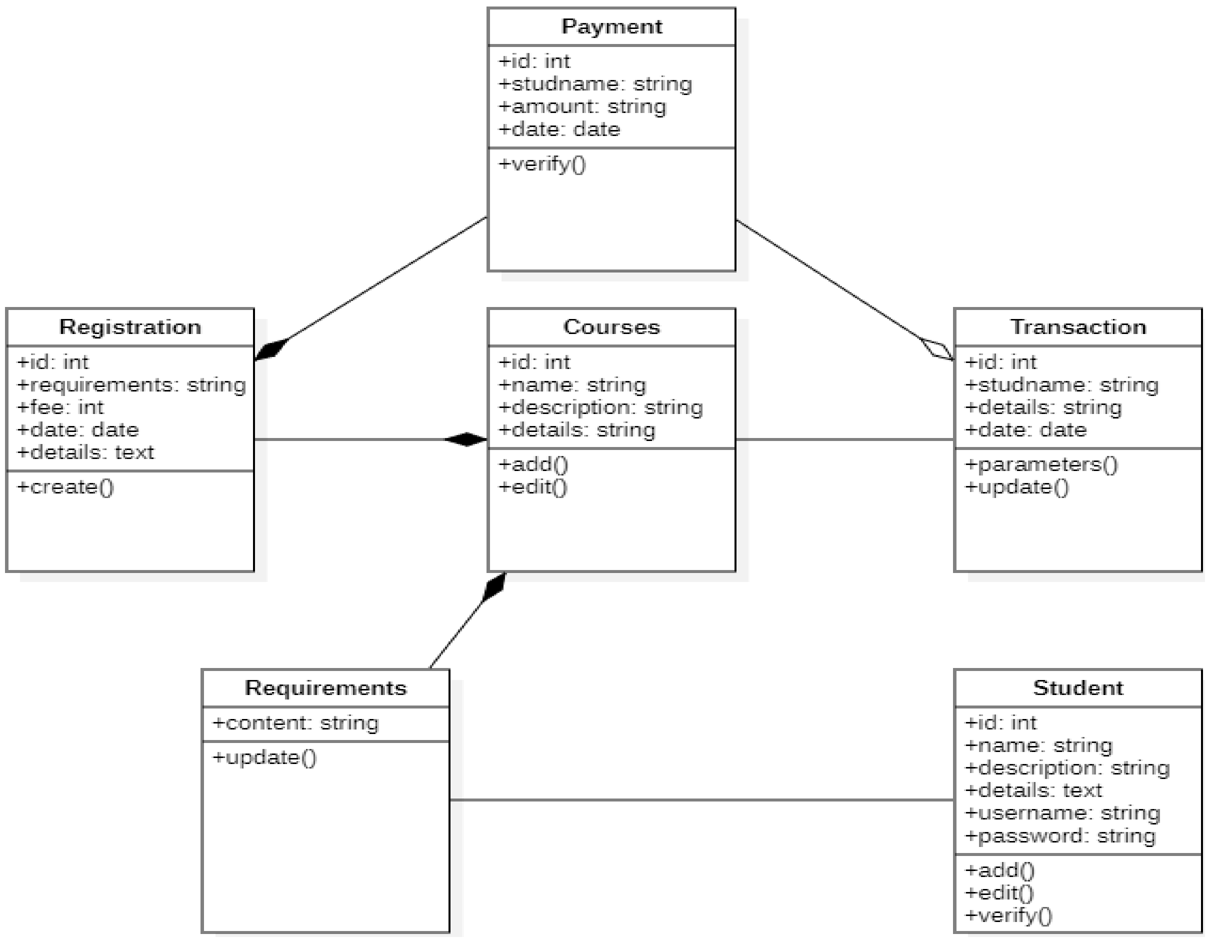
STATECHART:



ACTIVITY DIAGRAM:

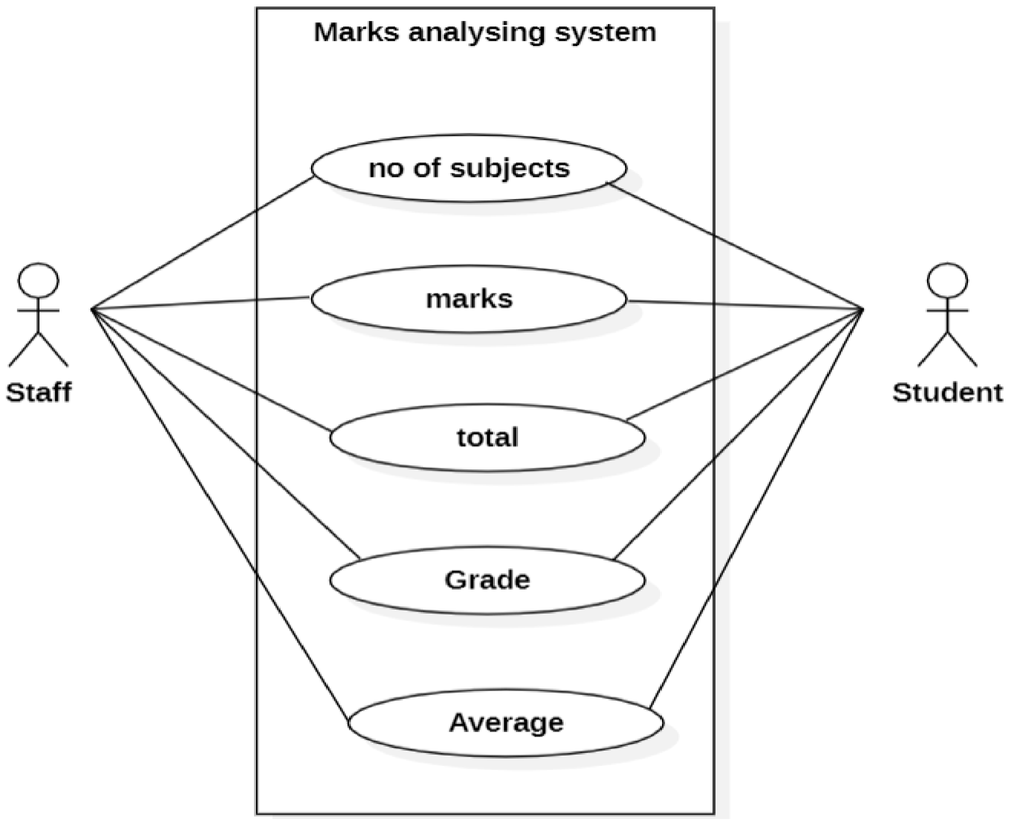


CLASS DIAGRAM:

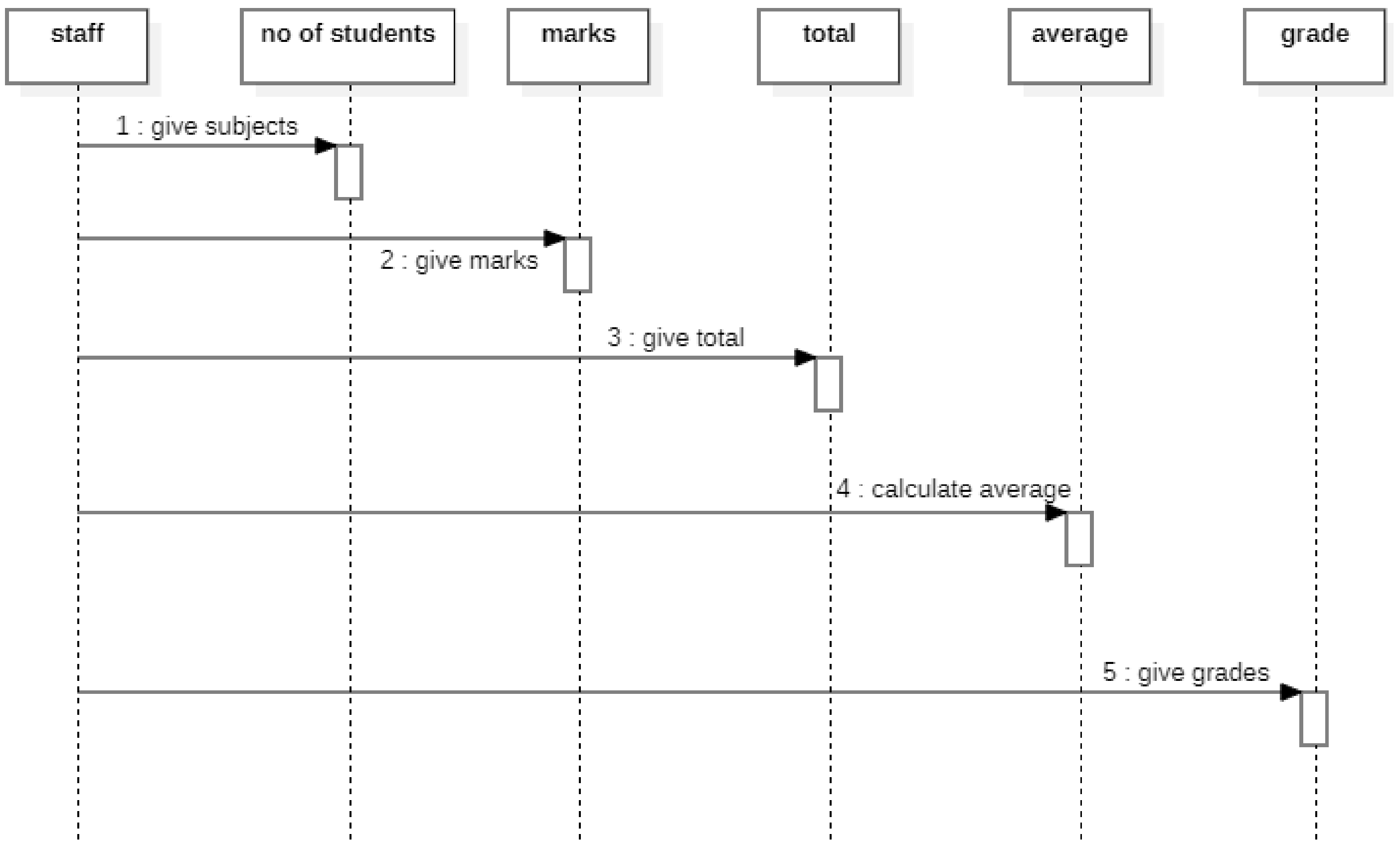


MARKS ANALYSING SYSTEM:

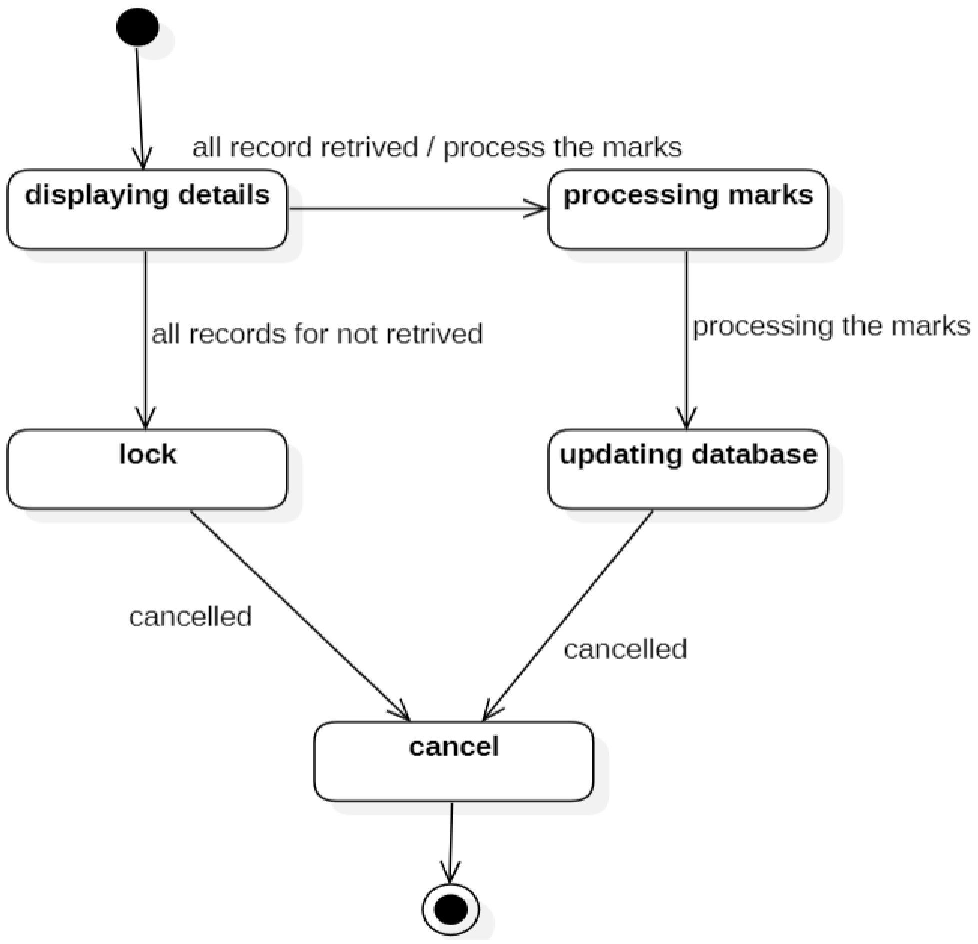
USECASE:



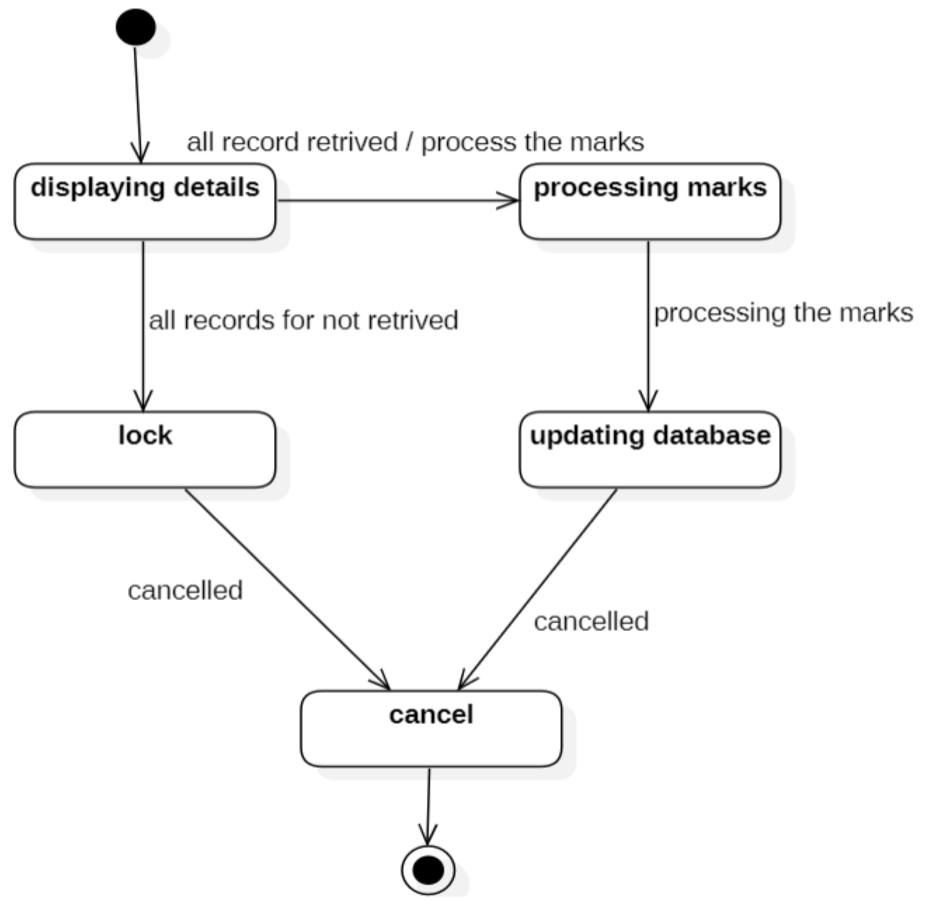
SEQUENCE:



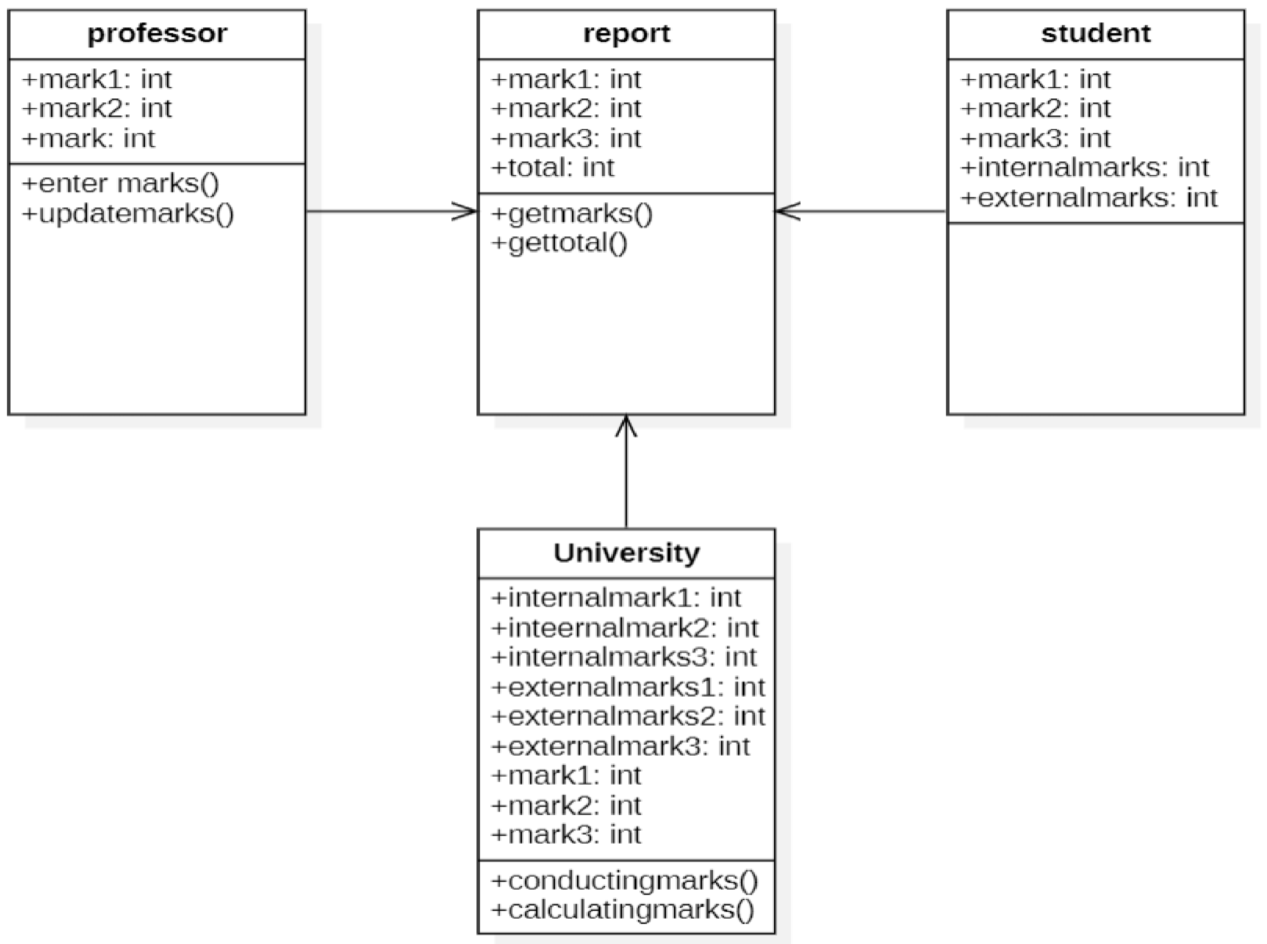
STATE CHART:



ACTIVITY DIAGRAM:

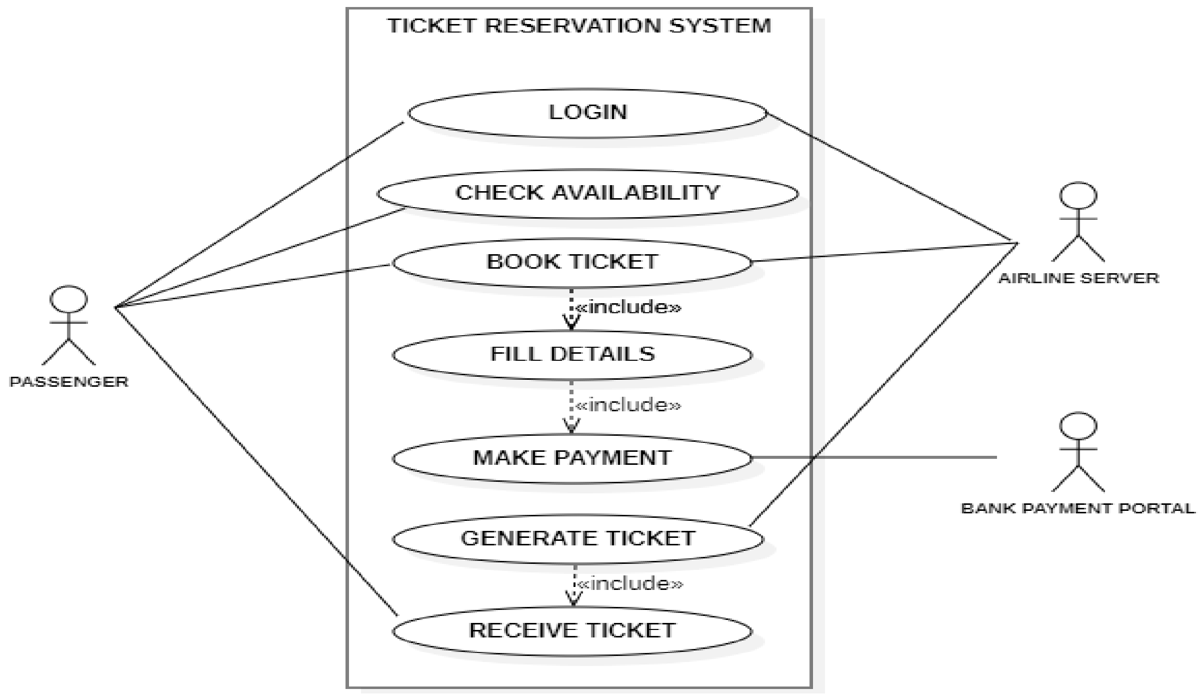


CLASS DIAGRAM:

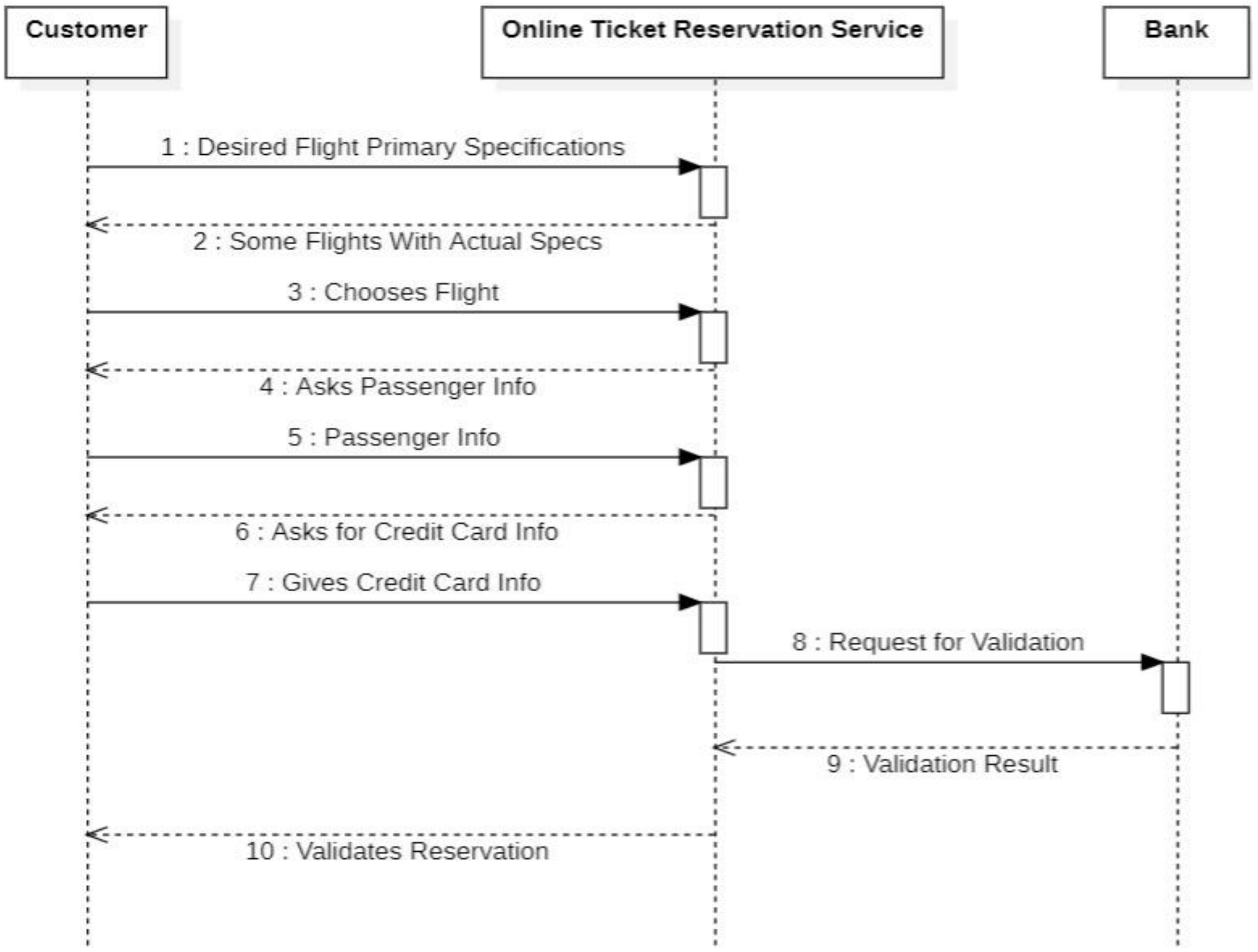


ONLINE TICKET RESERVATION SYSTEM:

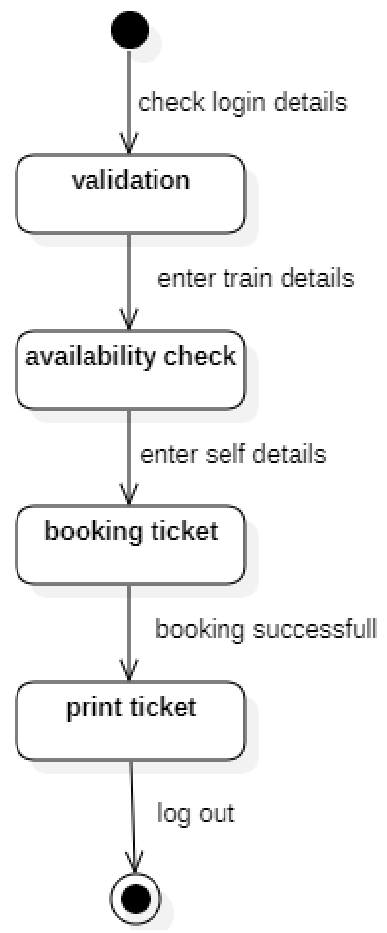
USECASE:



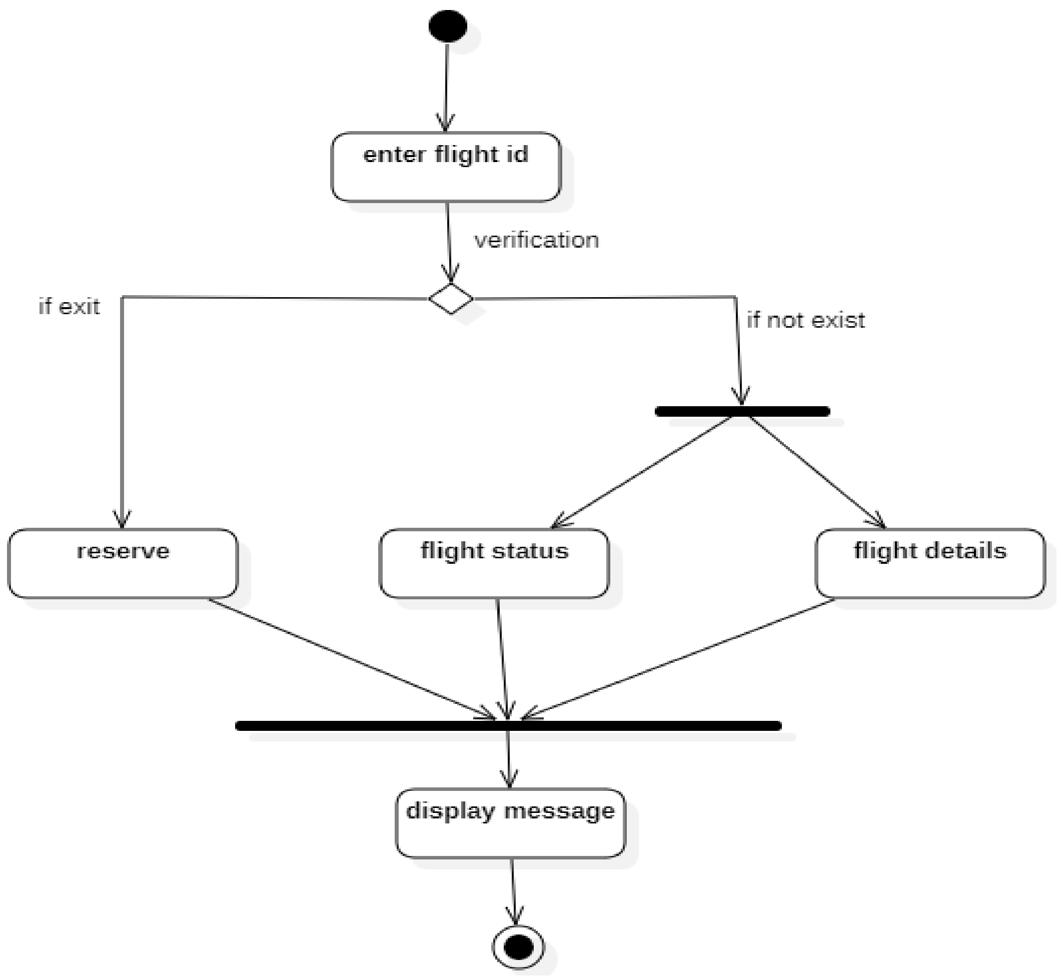
SEQUENCE:


Customer
Online Ticket Reservation Service
Bank
1 : Desired Flight Primary Specifications
2 : Some Flights With Actual Specs
3 : Chooses Flight
4 : Asks Passenger Info
5 : Passenger Info
6 : Asks for Credit Card Info
7 : Gives Credit Card Info
8 : Request for Validation
9 : Validation Result
10 : Validates Reservation

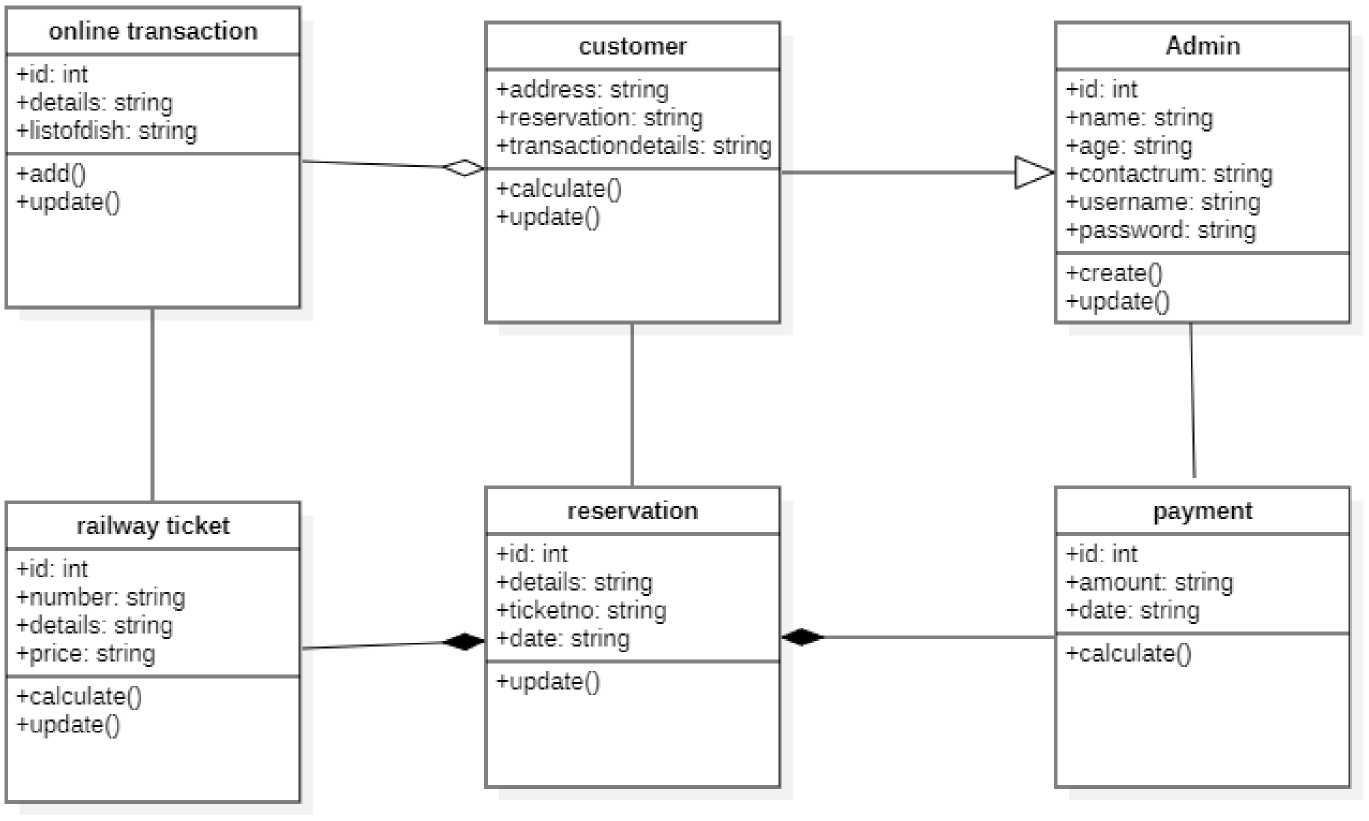
STATE CHART:



ACTIVITY DIAGRAM:

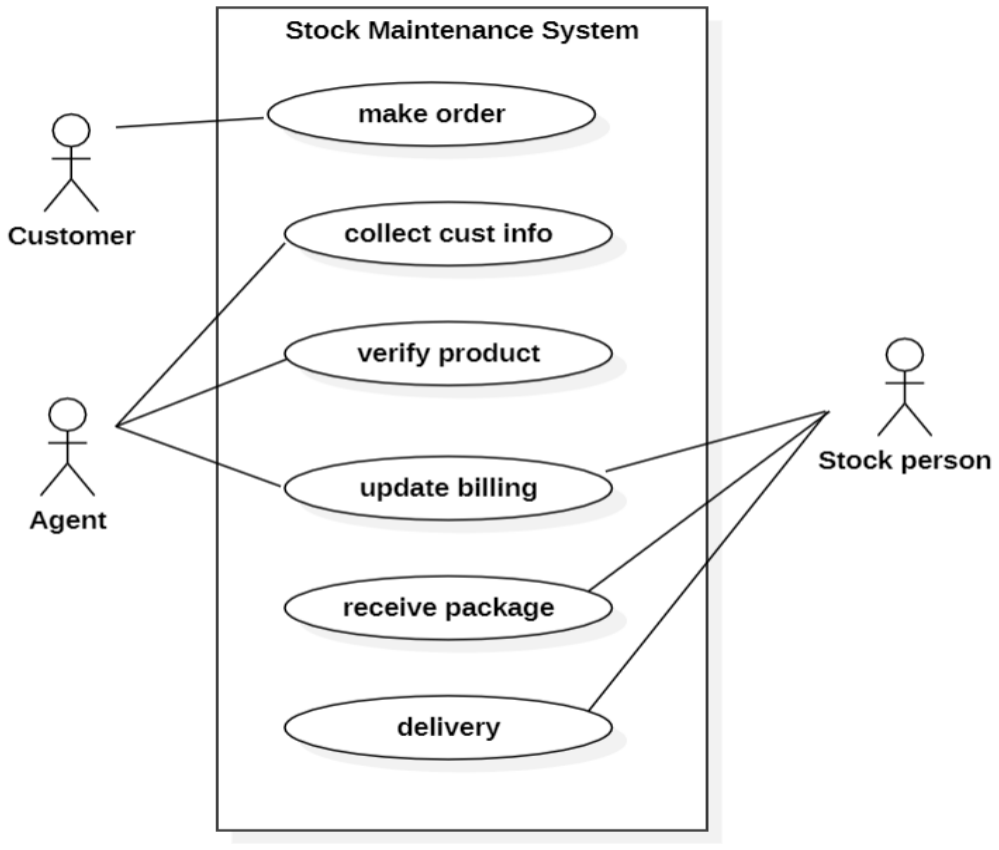


CLASS DIAGRAM:

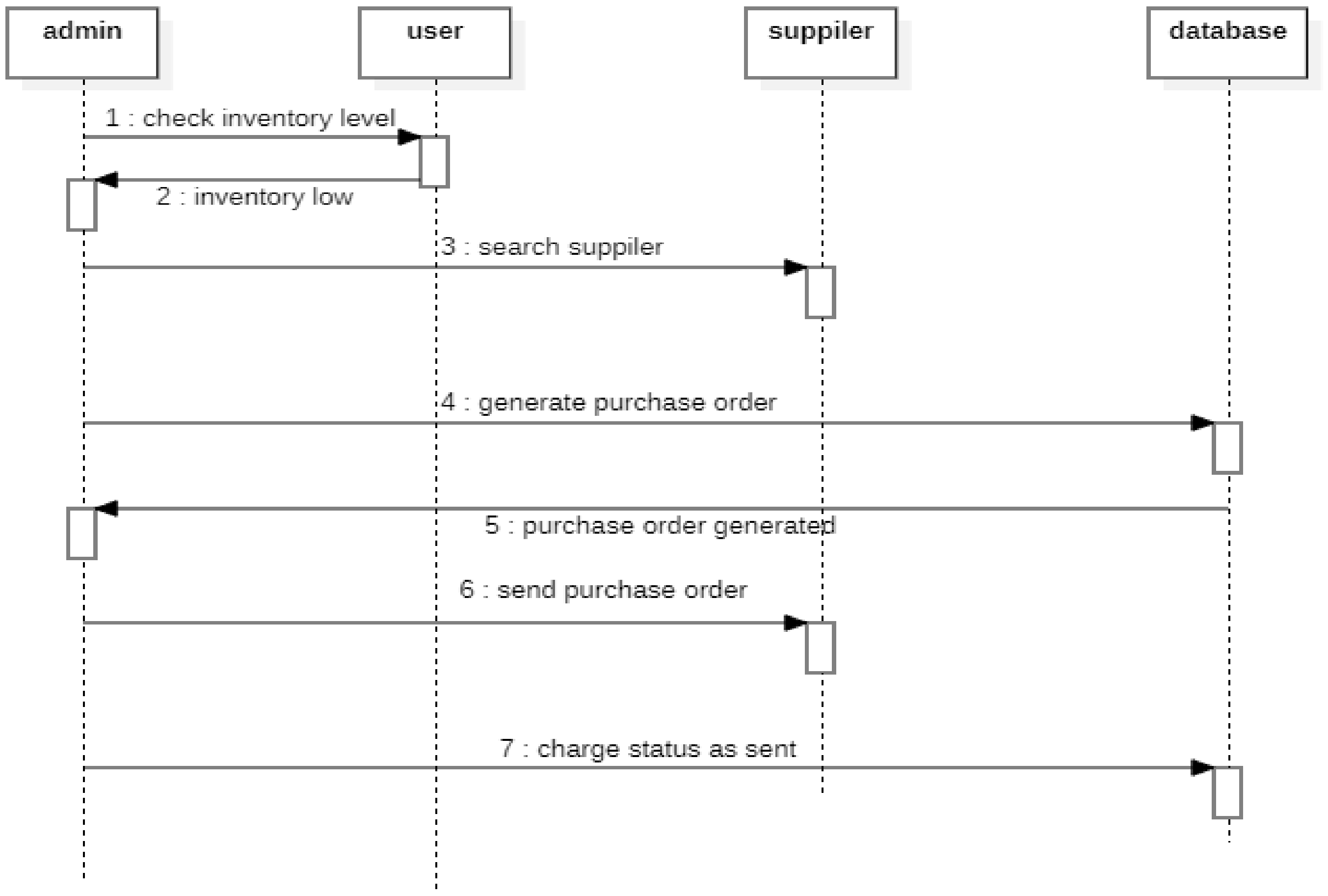


STOCK MAINTENANCE SYSTEM:

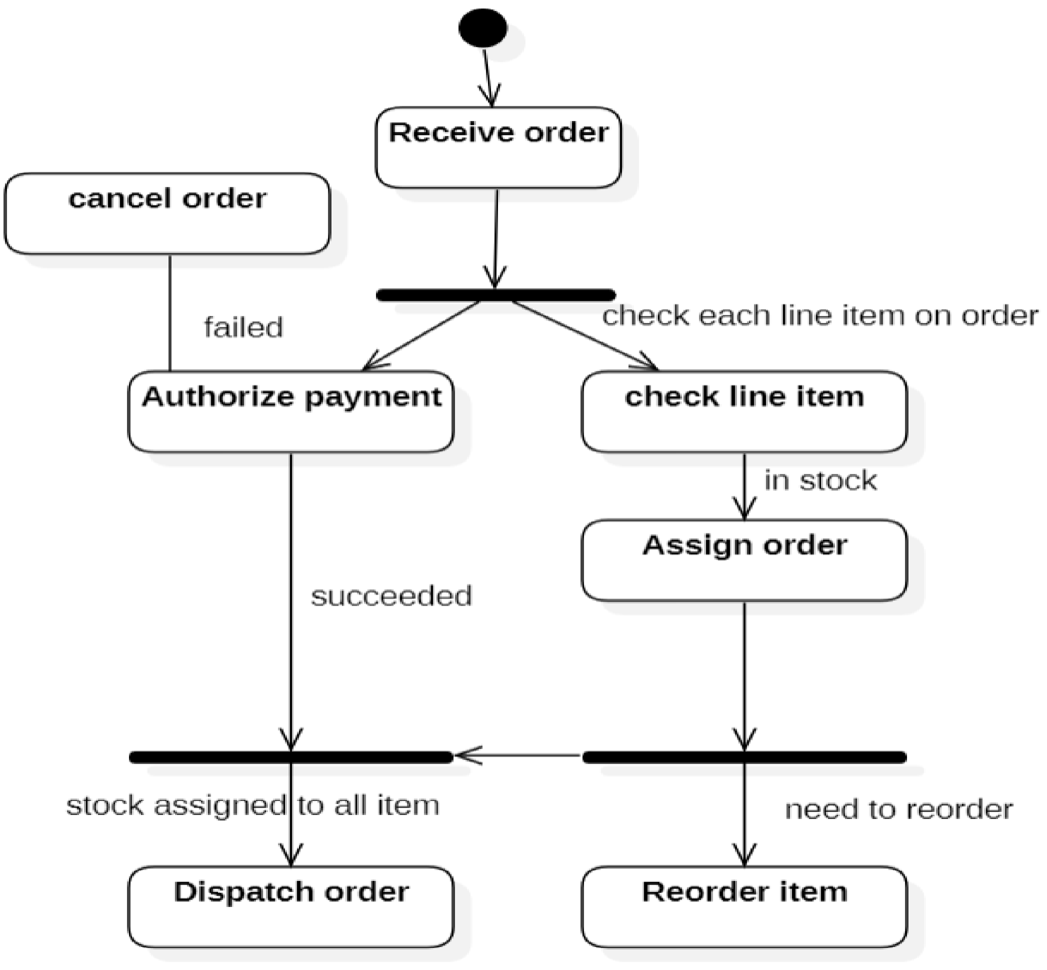
USECASE:



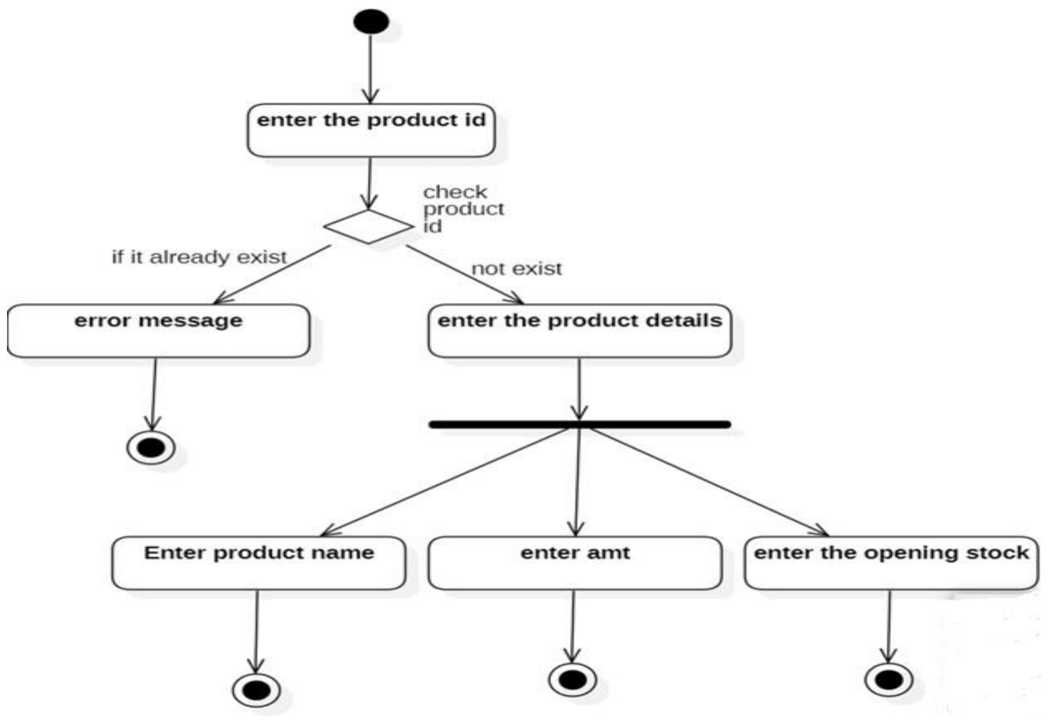
SEQUENCE DIAGRAM:


admin
user
suppiler
database
1 : check inventory level
2 : inventory low
3 : search suppiler
4 : generate purchase order
5 : purchase order generated
6 : send purchase order
7 : charge status as sent

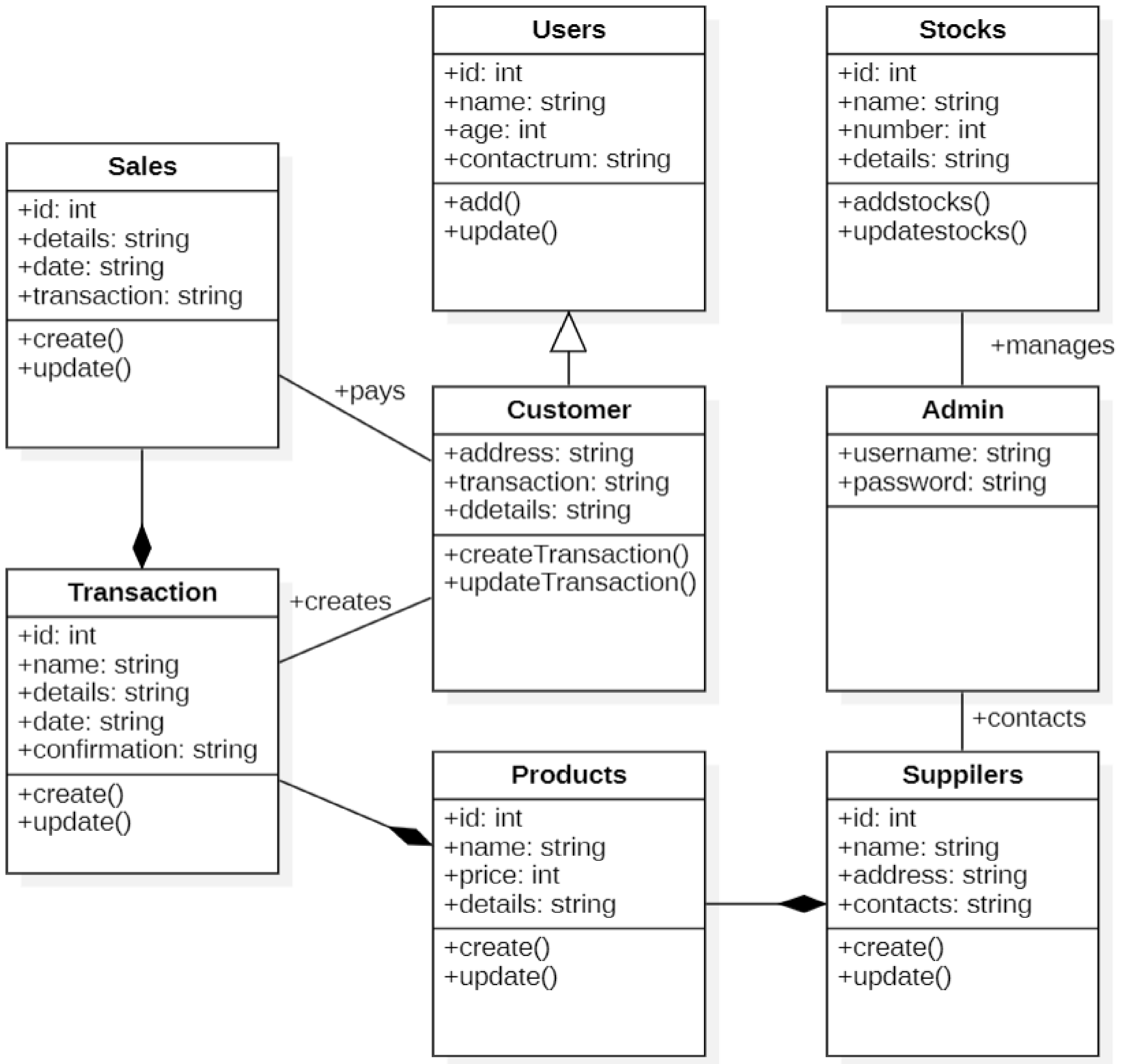
STATE CHART DIAGRAM:



ACTIVITY DIAGRAM:



CLASS DIAGRAM:



EXERCISE-3: DESIGN THE TEST CASES FOR E-COMMERCE APPLICATION (FLIPCART, AMAZON) AND MOBILE APPLICATION (CONSIDER ANY EXAMPLE FROM APP STORE)

TEST CASES:

A Test Case is a set of actions executed to verify a particular feature or functionality of your software application, it contains test steps, test data, precondition, postcondition developed for specific test scenario to verify any requirement.

The test case includes specific **variables or conditions**, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

How to Write Test Cases in Manual Testing

- **Test Case ID:**

Test cases should all bear unique IDs to represent them. In most cases, following a convention for this naming ID helps with organization, clarity, and understanding.

- **Test Description:**

This description should detail what unit, feature, or function is being tested or what is being verified.

- **Steps to be Executed:**

These should be easily repeatable steps as executed from the end user's perspective.
For **example**, a test case for logging into an email server might include these steps:

1. Open email server web page.
2. Enter username.
3. Enter password.
4. Click "Enter" or "Login" button.

- **Expected Result:**

This indicates the result expected after the test case step execution. Upon entering the right login information, the expected result would be a successful login.

- **Actual Result and Post-Conditions**

As compared to the expected result, we can determine the status of the test case. In the case of the email login, the user would either be successfully logged in or not. The post-condition is what happens as a result of the step execution such as being redirected to the email inbox.

- **Pass/Fail**

Determining the pass/fail status depends on how the expected result and the actual result compare to each other.

Same result = Pass
Different results = Fail

- **Pre-Conditions:**

This entails any conditions to be met before test case execution. One example would be requiring a valid Outlook account for a login.

- **Test Data:**

This relates to the variables and their values in the test case. In the example of an email login, it would be the username and password for the account.

ECOMMERCE APPLICATION (AMAZON & FLIPKART) TEST CASES:

The e-commerce business is increasing at a rapid pace especially since Covid-19. So, it is very important to test e-commerce websites and applications such as Amazon and Flipkart. **Regressive testing** of e-commerce applications is vital to make your app more robust and secure.
Effective **manual test** strategies should be designed to test eCommerce applications.

ECOMMERCE APPLICATION ARCHITECTURE

An eCommerce application has 4 important elements –

- ❖ Main Pages: It includes homepage, privacy page, about us page, careers, etc.
- ❖ Product Pages: It includes different options for the product such as size, color, and other attributes.
- ❖ Product Description Pages: It includes title, description, images, add to cart feature and additional info, etc.
- ❖ Shopping Cart: The shopping cart page should include payment options and removing a product from the cart.

an eCommerce website or an application includes different user roles such as customers, partners, staff, and service agents. Backend infrastructures include rules, analytics, security, logging, and content management.

Core Functionality includes Content, Marketing, Inventory Management, Orders, and Catalog.

Back-office functionality includes fulfillment, inventory, payment, and Catalog staging.

Core Functionality includes Content, Marketing, Inventory Management, Orders, and Catalog.

Back-office functionality includes fulfillment, inventory, payment, and Catalog staging.

LOGIN FUNTIONALITY IN ECOMMERCE APPLICATION:

TC-ID	TC-NAME	STEPS	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	STATUS
TC-01	Valid Username	**1)**Enter valid username	**Username:** shariff@gmail. Com	It should accept username	Accepting	PASS
TC-02	Invalid Username	**1)**Enter Invalid username	**Username:** shariff@gmail. Com	It should not accept username	Accepting	FAIL
TC-03	Valid password	**1)**Enter valid password **2)**password should contain combination od no. of char & special symbol	**Password:** CSM@231	It should accept password	Accepting	PASS
TC-04	Invalid password	**1)**Enter Invalid password **2)**password should contain combination od no. of char & special symbol	**Password:** CSM@231	It should accept password	Accepting	FAIL
TC-05	Submit	**1)**click on Submit Button	NO INPUT	It should go to next page	**Going to** next page	PASS
TC-06	Submit	**1)**click on Submit Button	NO INPUT	It should not go to next page	**Not Going** to next page	FAIL
TC-07	Cancel	**1)**click on cancel Button	NO INPUT	It should clear username & password and remain on same page	**Clearing** and on same page	PASS
TC-08	Cancel	**1)**click on cancel Button	NO INPUT	It should not clear username & password and not remain on same page	**Not clearing** and not on the same page	FAIL

MOBILE APPLICATION TEST CASES:

FUNCTIONAL TESTING:

The **Functional Testing** of Mobile Application is a process of testing functionalities of mobile applications like user interactions as well as testing the transactions that users might perform. The main purpose of mobile application functional testing is to ensure the quality, meeting the specified expectations, reducing the risk or errors and customer satisfaction.

The various **factors** which are relevant in functional testing are:

- **Type of application** based upon the business functionality usages (banking, gaming, social or business)
- **Target audience type** (consumer, enterprise, education)

Distribution channel which is used to spread the application (e.g. Apple App Store, Google play, direct distribution)

The general test scenarios for Functional Testing in a Mobile application are:

1. To validate whether all the required mandatory fields are working as required.
2. To validate that the mandatory fields are displayed in the screen in a distinctive way than the non-mandatory fields.
3. To validate whether the application works as per as requirement whenever the application starts/stops.
4. To validate whether the application goes into minimized mode whenever there is an incoming phone call. In order to validate the same we need to use a second phone, to call the device.

PERFORMANCE TESTING:

This type of testing's fundamental objective is to ensure that the application performs acceptably under certain performance requirements such as access by a huge number of users or the removal of a key infrastructure part like a database server.

The general test scenarios for Performance Testing in a Mobile application are:

1. To determine whether the application performs as per the requirement under different load conditions.
2. To determine whether the current network coverage is able to support the application at peak, average and minimum user levels.
3. To validate whether the response time of the application is as per as the requirements.

4. To validate that the battery consumption, memory leaks, resources like GPS, Camera performance is well within required guidelines.
5. To validate the application longevity whenever the user load is rigorous.
6. To validate the network performance while moving around with the device.

SECURITY TESTING TEST CASES:

The fundamental objective of security testing is to ensure that the application's data and networking security requirements are met as per guidelines.

The following are the most crucial areas for checking the security of Mobile applications.

The general test scenarios for security testing in a Mobile application are:

1. To validate that the application is able to withstand any brute force attack which is an automated process of trial and error used to guess a person's username, password or credit-card number.
2. To validate whether an application is not permitting an attacker to access sensitive content or functionality without proper authentication.
3. To validate that the application has a strong password protection system and it does not permit an attacker to obtain, change or recover another user's password.
4. To validate that the application does not suffer from insufficient session expiration.
5. To identify the dynamic dependencies and take measures to prevent any attacker for accessing these vulnerabilities.
6. To protect the application and the network from the denial-of-service attacks.
7. To analyse the data storage and data validation requirements.
8. To protect against malicious client-side injections.
9. To protect against malicious runtime injections.

USABILITY TESTING TEST CASES:

The usability testing process of the Mobile application is performed to have a quick and easy step application with less functionality than a slow and difficult application with many features. The main objective is to ensure that we end up having an easy-to-use, intuitive and similar to industry-accepted interfaces which are widely used.

The general test scenarios for usability testing in a Mobile application are:

1. To ensure that the buttons should have the required size and be suitable to big fingers.

2. To ensure that the buttons are placed in the same section of the screen to avoid confusion to the end users.
3. To ensure that the icons are natural and consistent with the application.
4. To ensure that the buttons, which have the same function should also have the same color.
5. To ensure that the validation for the tapping zoom-in and zoom-out facilities should be enabled.
6. To ensure that the text is kept simple and clear to be visible to the users.
7. To ensure that the short sentences and paragraphs are readable to the end users.
8. To ensure that the font size is big enough to be readable and not too big or too small.
9. To validate the application prompts the user whenever the user starts downloading a large amount of data which may be not conducive for the application performance.
10. To validate that the closing of the application is performed from different states and verify if it re-opens in the same state.

COMPATIBILITY TESTING TEST CASES:
Compatibility testing on mobile devices is performed to ensure that since mobile devices have different size, resolution, screen, version and hardware so the application should be tested across all the devices to ensure that the application works as desired.

The general test scenarios for compatibility testing in a Mobile application are:

1. To validate that the user Interface of the application is as per the screen size of the device, no text/control is partially invisible or inaccessible.
2. To ensure that the text is readable for all users for the application.
3. To ensure that the call/alarm functionality is enabled whenever the application is running. The application is minimized or suspended on the event of a call and then whenever the call stops the application is resumed.

RECOVERABILITY TESTING TEST CASES:

The general test scenarios for recoverability testing in a Mobile application are:

1. Crash recovery and transaction interruptions

2. Validation of the effective application recovery situation post unexpected interruption/crash scenarios.
3. Verification of how the application handles a transaction during a power failure (i.e., Battery dies or a sudden manual shutdown of the device)
4. The validation of the process where the connection is suspended, the system needs to re-establish for recovering the data directly affected by the suspended connection.

Important Checklist

1. **Installation testing** (whether the application can be installed in a reasonable amount of time and with required criterion)
2. **Uninstallation testing** (whether the application can be uninstalled in a reasonable amount of time and with required criterion)
3. **Network test cases** (validation of whether the network is performing under required load or not, whether the network is able to support all the necessary applications during the testing procedures)
4. Check Unmapped keys
5. Check application splash screen
6. Continued keypad entry during interrupts and other times like network issues
7. Methods which deal with exiting the application
8. Charger effect while an application is running in the background
9. Low battery and high performance demand
10. Removal of battery while an application is being performed
11. Consumption of battery by application
12. Check Application side effects

TEST CASES ON MOBILE APPLICATION:

TC-ID	TYPE	STATUS	TEST CONDITION	STEPS	EXPECTED RESULT	ACTUAL RESULT	STATUS
TC-01	SANITY	DESIGN	Verify mobile app registration screen	**1)**download and install mobile app **2)**open mobile app first time	Mobile app should open with in registration links.	NOT RUN	**N/A**
TC-02	SANITY	DESIGN	Verify registration screen	**1)**download and install mobile app **2)**open mobile app first time **3)**click on registration link	Registration link should display name, mobile no, email ID, password, continue button	NOT RUN	**N/A**
TC-03	MANUAL	DESIGN	Verify name field	**1)**download and install mobile app **2)**open mobile app first time **3)**click on registration link **4)**verify name field	Name field should be **TYEP:** Test box **ACCEPT:** Min 3 and max 16 alphabets **VALIDATION:** should provide validation message if field is empty while click on continue	NOT RUN	**N/A**
TC-04	MANUAL	DESIGN	Verify Email id field	**1)**mobile app download & install **2)**open mobile app first time **3)**click on registration link **4)**enter name **5)**verify email	Email id field should be **TYPE:** test box Accepts: email ID format	NOT RUN	**N/A**

TC-ID	TYPE	STATUS	TEST CONDITION	STEPS	EXPECTED RESULT	ACTUAL RESULT	STATUS
TC-05	MANUAL	DESIGN	Verify mobile number field	**1)**Download & Install mobile app **2)**open mobile app first time **3)**Click on registration link **4)**Enter name **5)**Enter Email no. **6)**Verify mobile number	**1)**mobile no field should be **TYPE:** text box **ACCEPTS:** only numeric **MAX:** 10 digits **VALIDATION:** should provide validation message if field is empty while click on continue button	NOT RUN	N/A
TC-06	MANUAL	DESIGN	Verify password field	**1)**Download & install mobile app **2)**Open mobile app first time **3)**Click registration link **4)**Enter name **5)**Enter email ID **6)**Enter mobile number **7)**Enter the password field	**1)**PASSWORD field should be **ACCEPTS:** password encrypted all characters **VALIDATION:** if empty should provide validation message while click on continue button	NOT RUN	N/A
TC-07	MANULA	DESIGN	Verify Facebook link	**1)**Download & Install mobile app **2)**Open mobile app for first time **3)**Click on registration link **4)**Click on Facebook link	**1)**mobile app should redirect page in order to login with Facebook existing account **2)**should display accept	NOT RUN	N/A
TC-08	MANUAL	DESIGN	Verify page link	Click on Google link	**1)**mobile app should redirect google page in order to login with google existing account	NOT RUN	N/A