ADITYA

COLLEGE OF ENGINEERING

Aditya Nagar, ADB Road, Surampalem - 533437

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LABORATORY RECORD

NAME	:	
ROLL NO.	:	
YEAR	:	
SEMESTER	:	
LAB	:	

ADITYA

COLLEGE OF ENGINEERING

Aditya Nagar, ADB Road, Surampalem - 533437

Department of

Name : **Roll No. :** | | | | | | | | | | |

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 year .....

No. of Practicals Conducted : ☐ No. of Practicals Attended : ☐

Signature - Faculty Incharge **Signature - Head of the Department**

Submitted for the Practical examinationheld on .....

EXAMINER - 1 **EXAMINER - 2**

VISION & MISSION OF THE INSTITUTE

VISION

To induce higher planes of learning by imparting technical education with

- International standards
- Applied research
- Creative Ability
- Value based instruction and to emerge as a premiere institute.

MISSION

Achieving academic excellence by providing globally acceptable technical education by forecasting technology through

- Innovative Research and development
- Industry Institute Interaction
- Empowered Manpower

VISION & MISSION OF THE DEPARTMENT

VISION

To be a recognized Computer Science and Engineering hub striving to meet the growing needs of the Industry and Society.

MISSION

M1: Imparting Quality Education through state-of-the-art infrastructure with industry Collaboration

M2: Enhance Teaching Learning Process to disseminate knowledge.

M3: Organize Skill based, Industrial and Societal Events for overall Development.

POINTER

S.No.	Date	Name of the Experiment	Page No.	Remarks

POINTER

S.No.	Date	Name of the Experiment	Page No.	Remarks

POINTER

S.No.	Date	Name of the Experiment	Page No.	Remarks

EXPERIMENT 1:

1 . Do the Requirement Analysis and Prepare SRS .

Requirement Analysis:

1. **Gather Information:**
    - Meet with stakeholders (clients, users, managers) to understand their needs and expectations.
    - Collect existing documentation, such as business plans, project proposals, and any relevant data.
2. **Identify Stakeholders:**
    - Identify all stakeholders involved in the project, including end-users, administrators, managers, and support staff.
3. **Define Project Scope:**
    - Clearly define the boundaries of the project. What is included, and what is not?
    - Determine the constraints, such as budget, timeline, and available resources.
4. **Document Requirements:**
    - Elicit requirements through interviews, surveys, workshops, or questionnaires.
    - Classify requirements as functional (what the system should do) and non-functional (performance, security, etc.).
    - Ensure that requirements are clear, concise, complete, and unambiguous.
5. **Analyze Requirements:**
    - Prioritize requirements based on their importance to stakeholders.
    - Resolve conflicting requirements and identify dependencies between them.
    - Validate requirements to ensure they align with the project objectives.
6. **Get Feedback:**
    - Review the gathered requirements with stakeholders to confirm their accuracy and understanding.
    - Address any feedback and make necessary revisions.

Software Requirements Specification (SRS) Preparation:

1. **Introduction:**
    - Provide an overview of the document.
    - Describe the purpose, scope, objectives, and stakeholders of the project.
2. **Functional Requirements:**
    - Detail each functional requirement identified during the analysis.
    - Specify input data, processes, and output results for each requirement.
    - Use diagrams, flowcharts, or pseudocode to illustrate complex processes if necessary.
3. **Non-Functional Requirements:**
    - Describe non-functional requirements such as performance, security, usability, and scalability.
    - Include constraints like regulatory compliance, hardware limitations, and compatibility.
4. **System Architecture:**
    - Provide an overview of the system architecture, including components and their interactions.
    - Describe the data flow, data storage, and external interfaces.
5. **Use Cases:**
    - Describe various use cases to provide a detailed understanding of how users will interact with the system.
6. **Data Model:**
    - Define the data model, including entities, relationships, and attributes.
    - Specify database requirements and data integrity constraints.
7. **Assumptions and Dependencies:**
    - Document any assumptions made during the analysis process.
    - Identify external dependencies that may affect the project.
8. **Traceability Matrix:**
    - Create a traceability matrix to link requirements back to their source (stakeholders, documents, discussions).
9. **Review and Validation:**
    - Have the SRS reviewed by stakeholders to ensure accuracy and completeness.
    - Revise the document based on feedback received

EXPERIMENT 1:

2. Draw E -R diagrams ,DFD,CFD and Structured Charts for the Project.

1.Course Registration System

2.Students Marks Analyzing System

3.Online Ticket Reservation Sytem

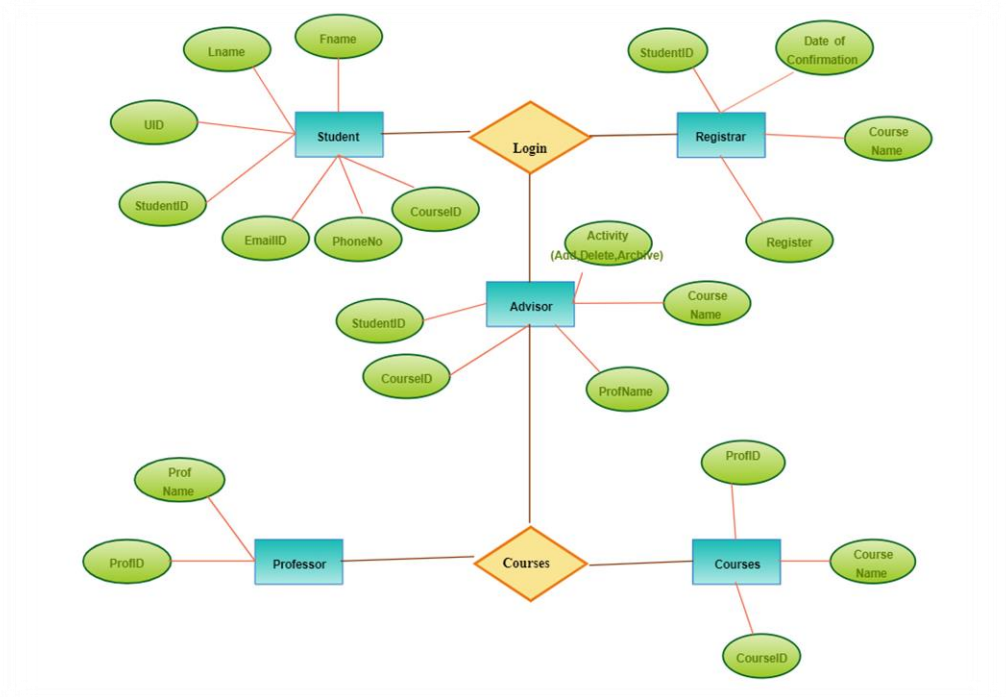
4.Stock Maintenance

- ## Entity-Relationship (E-R) Diagrams:

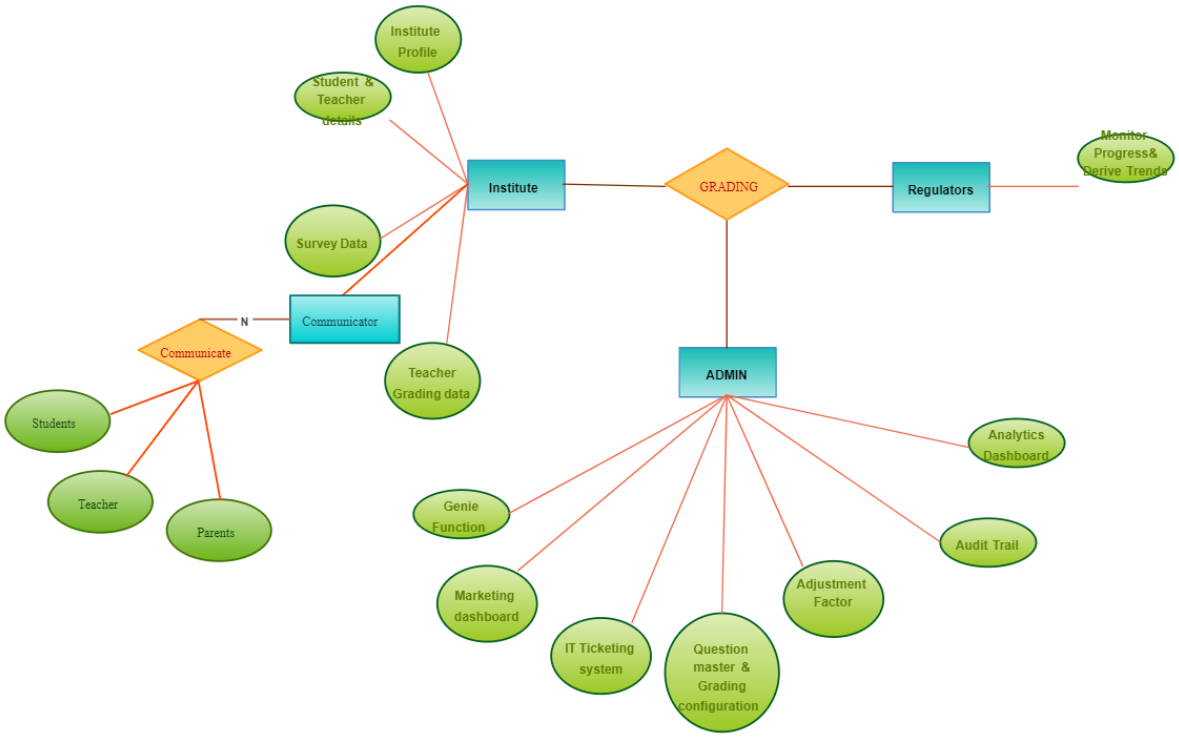
1. **Identify Entities:** Determine the main entities in your project (e.g., customer, product, order).
2. **Identify Relationships:** Define how these entities are related to each other (e.g., one customer can place many orders).
3. **Attributes:** Specify the attributes for each entity (e.g., customer name, order date).
4. **Draw the Diagram:** Use tools like Lucidchart, draw.io, or even pen and paper to draw the entities, relationships, and attributes.

E-R Diagrams :

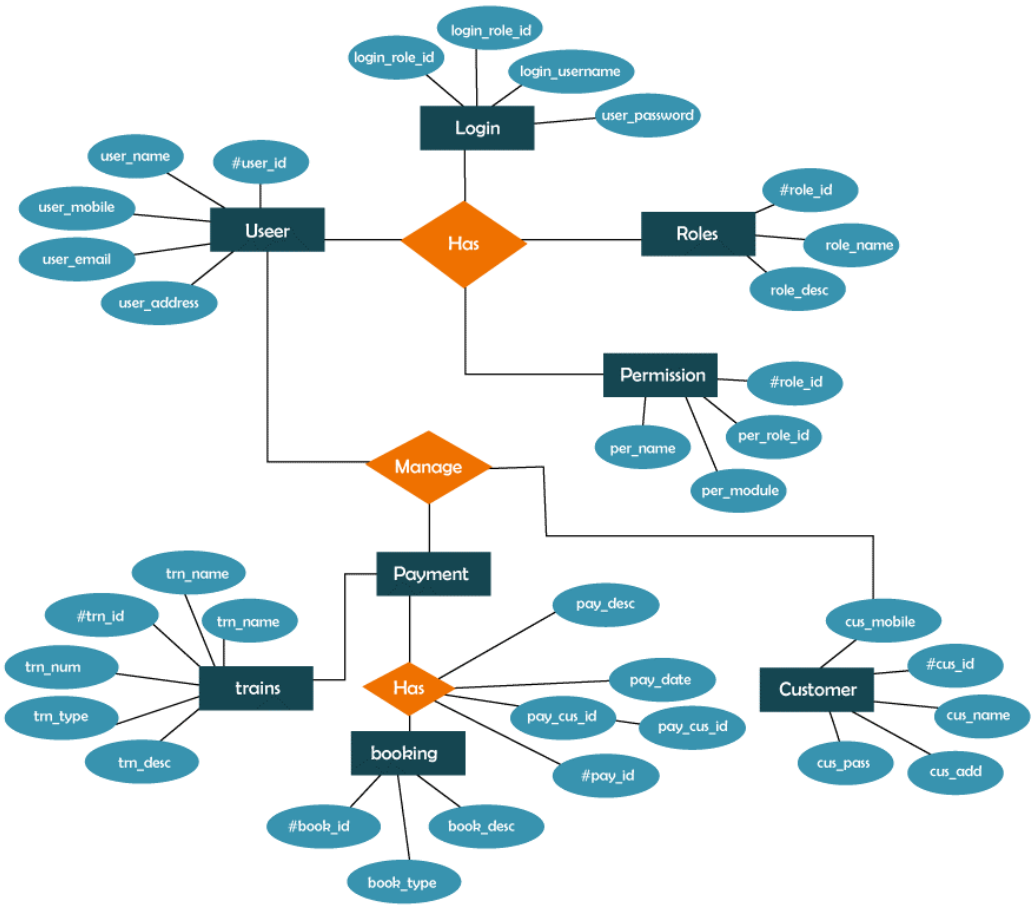
1. Course Registration System



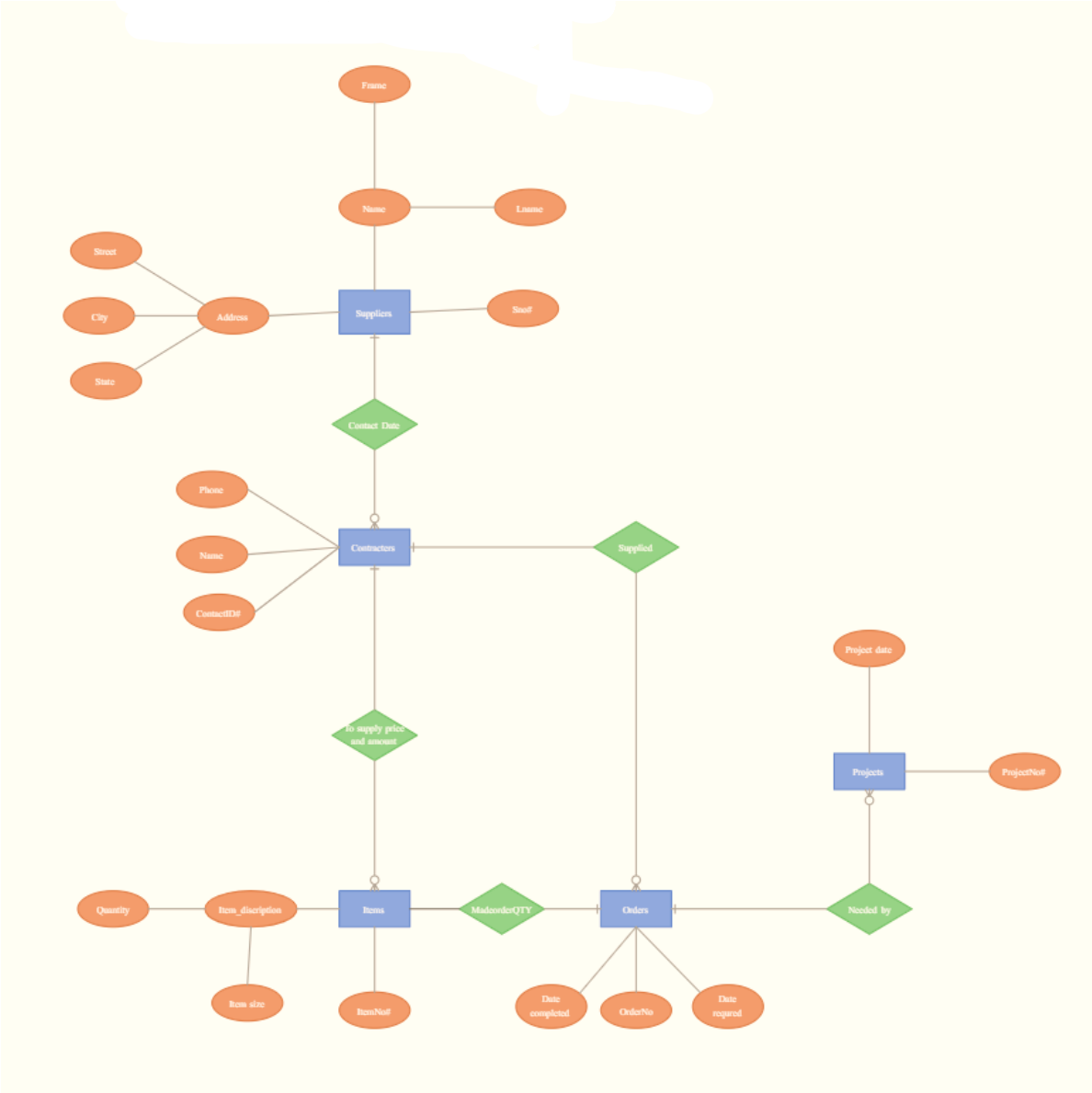
EXP-2:Student Marks Analyzing System



EXP-3:Online Ticket Reservation System



EXP-4 :Stock Maintenance



- **Data Flow Diagrams (DFD):**

1.Identify Processes: Determine the processes in your system (e.g., order processing, inventory management).

2.Identify Data Stores: Identify where data is stored (e.g., databases, file systems).

3.Identify Data Flows: Determine how data moves between processes and data stores.

4.Draw the DFD: Use DFD drawing tools or software like Microsoft Visio to create the diagram.

1.Course Registration System

Level 0 DFD: Course Registration System

At the highest level, you have a context-level diagram showing the course registration system as a whole. It interacts with external entities like students, administrators, and the course database.

External Entities:

1. **Student:** Provides registration details and checks course availability.
2. **Administrator:** Manages course offerings, updates the course database, and handles registration-related tasks.
3. **Course Database:** Stores information about available courses, their schedules, and capacities.

Processes:

1. **Course Registration:** Manages the registration process, including validating student information, checking course availability, and updating the database.
2. **Course Management:** Allows administrators to add new courses, modify existing ones, and update the course database.

Generate Reports: Generates various reports related to course registrations, available courses, and student enrollment.

Data Flows:

1. **Student Registration Data:** Information entered by students during registration.
2. **Course Information:** Details about available courses, including course codes, names, schedules, and capacities.
3. **Registration Confirmation:** Confirmation message sent to students after successful registration.
4. **Course Updates:** Information about changes in course offerings and schedules, sent from administrators to the database.
5. **Reports:** Generated reports about course registrations, enrollment statistics, and other related data.

Data Stores:

1. **Student Database:** Stores student information, including names, IDs, contact details, and registered courses.
2. **Course Database:** Contains data about courses, including course codes, names, schedules, capacities, and availability status.

Level 1 DFD: Detailed Processes

At this level, each process from the Level 0 DFD can be broken down into more detailed subprocesses.

Course Registration Process:

1. **Validate Student Information:** Verifies the accuracy and validity of the student's registration details.
2. **Check Course Availability:** Determines if the desired course has available slots.
3. **Update Student Database:** Adds the student's registration details to the student database.
4. **Update Course Availability:** Decreases the available slots for the registered course in the course database.
5. **Send Registration Confirmation:** Notifies the student about the successful registration.

Course Management Process:

1. **Add New Course:** Adds new courses to the course database with relevant information.
2. **Modify Course Details:** Allows administrators to update course names, schedules, and other details.
3. **Remove Course:** Removes courses from the database if they are no longer offered.
4. **Update Course Database:** Updates the course database with changes made by administrators.

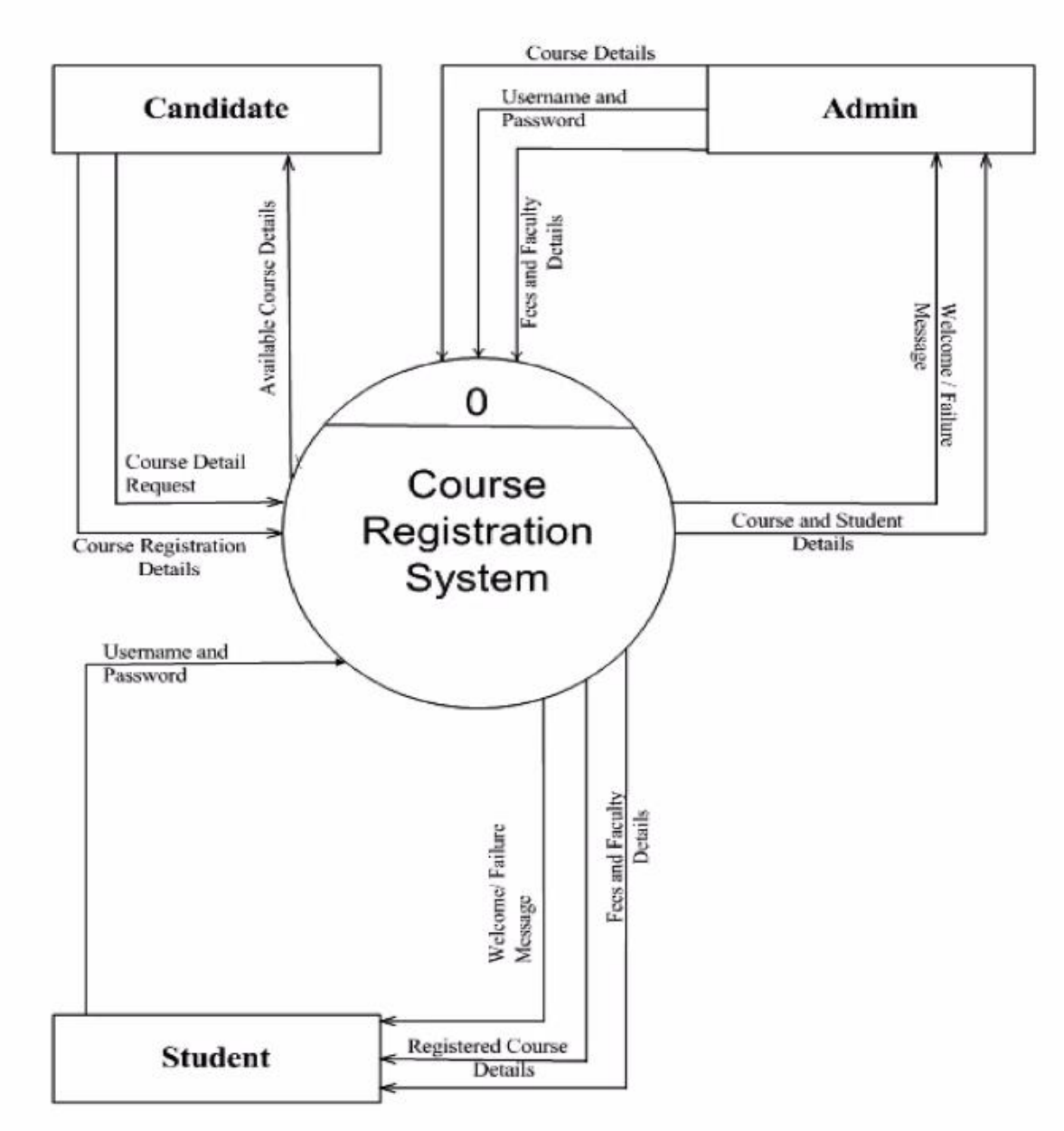
Generate Reports Process:

1. **Collect Data:** Gathers relevant information from the student and course databases.
2. **Generate Enrollment Reports:** Creates reports detailing student enrollments for various courses.
3. **Generate Course Availability Reports:** Provides information about available slots in different courses.
4. **Send Reports:** Distributes generated reports to administrators and other relevant parties.

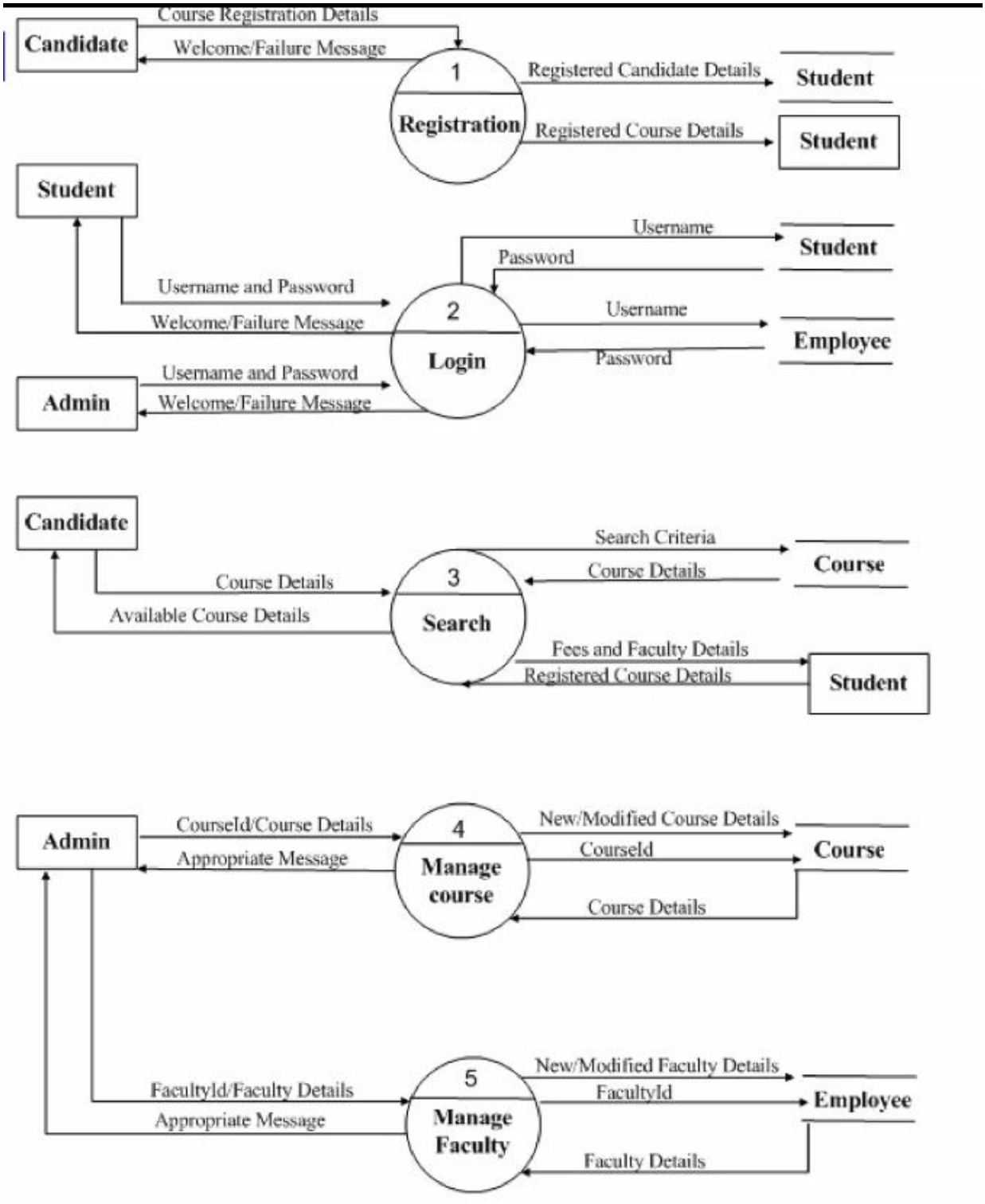
This detailed DFD provides a structured overview of the course registration system, illustrating how data flows between processes, data stores, and external entities at different levels of the system.

Course Registration System (DFD):

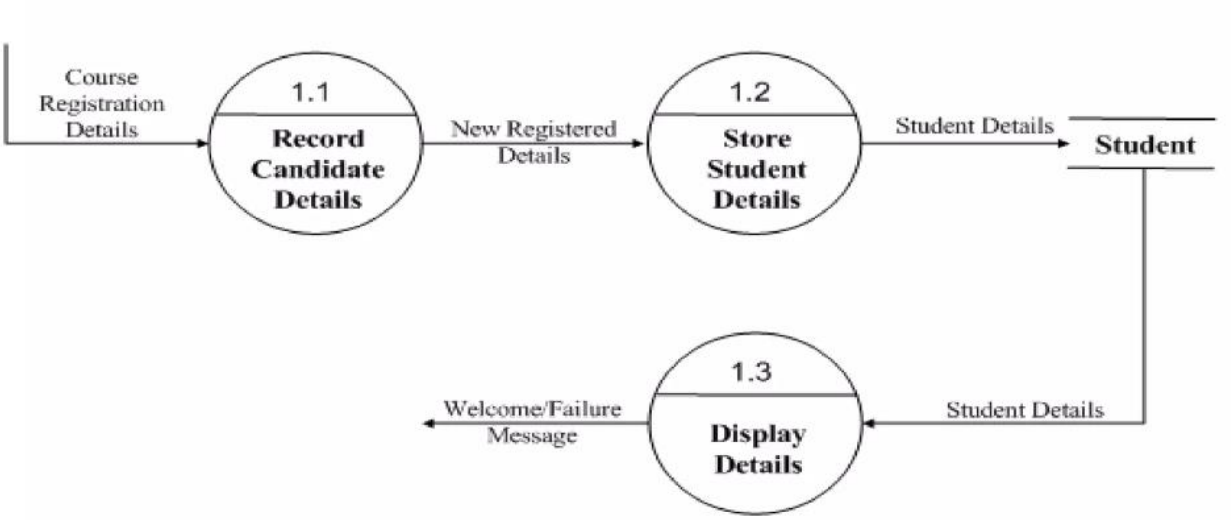
Level 0 DFD :



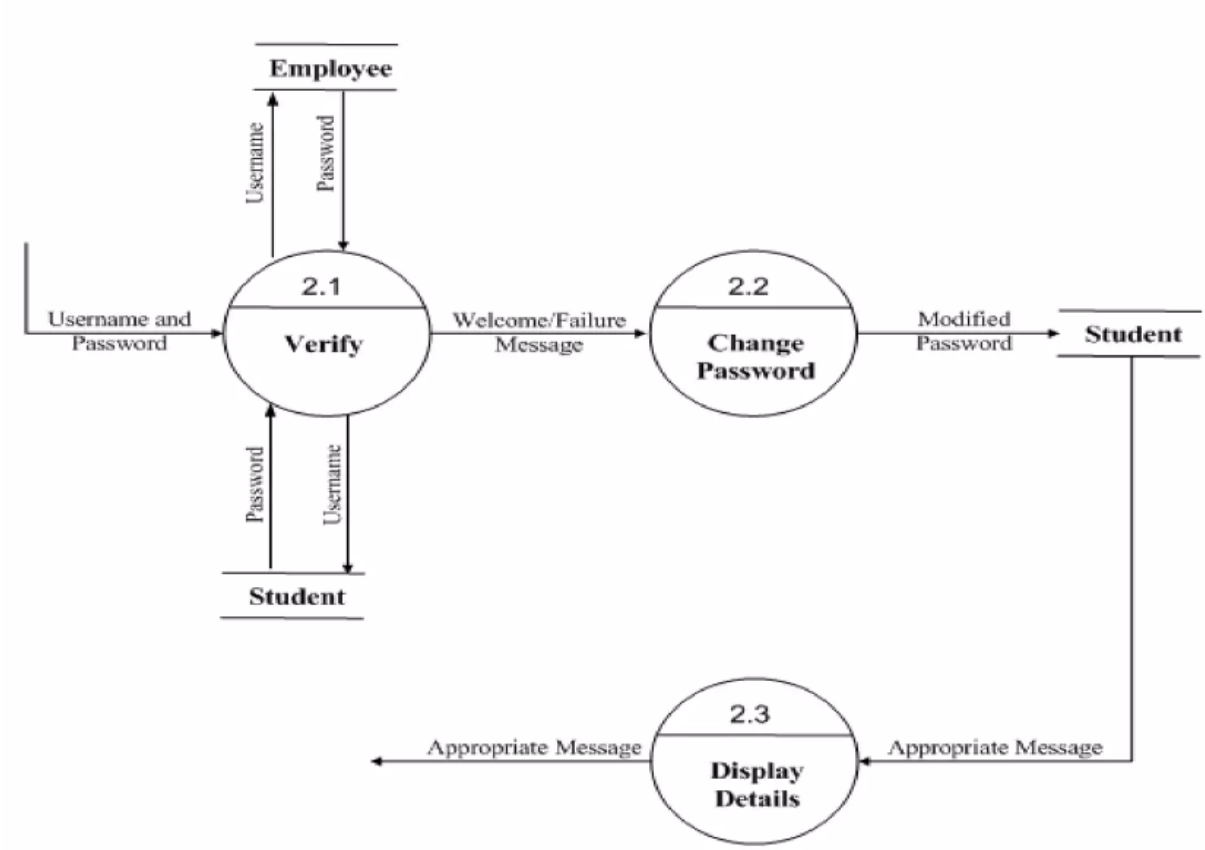
Level 1 DFD :


Candidate
Course Registration Details
Welcome/Failure Message
1
Registration
Registered Candidate Details
Student
Registered Course Details
Student
Student
Username
Password
Student
Username and Password
Welcome/Failure Message
2
Login
Username
Password
Employee
Admin
Username and Password
Welcome/Failure Message
Candidate
Search Criteria
Course Details
Course
Course Details
Available Course Details
3
Search
Fees and Faculty Details
Registered Course Details
Student
Admin
CourseId/Course Details
Appropriate Message
4
Manage course
New/Modified Course Details
CourseId
Course
Course Details
FacultyId/Faculty Details
Appropriate Message
5
Manage Faculty
New/Modified Faculty Details
FacultyId
Employee
Faculty Details

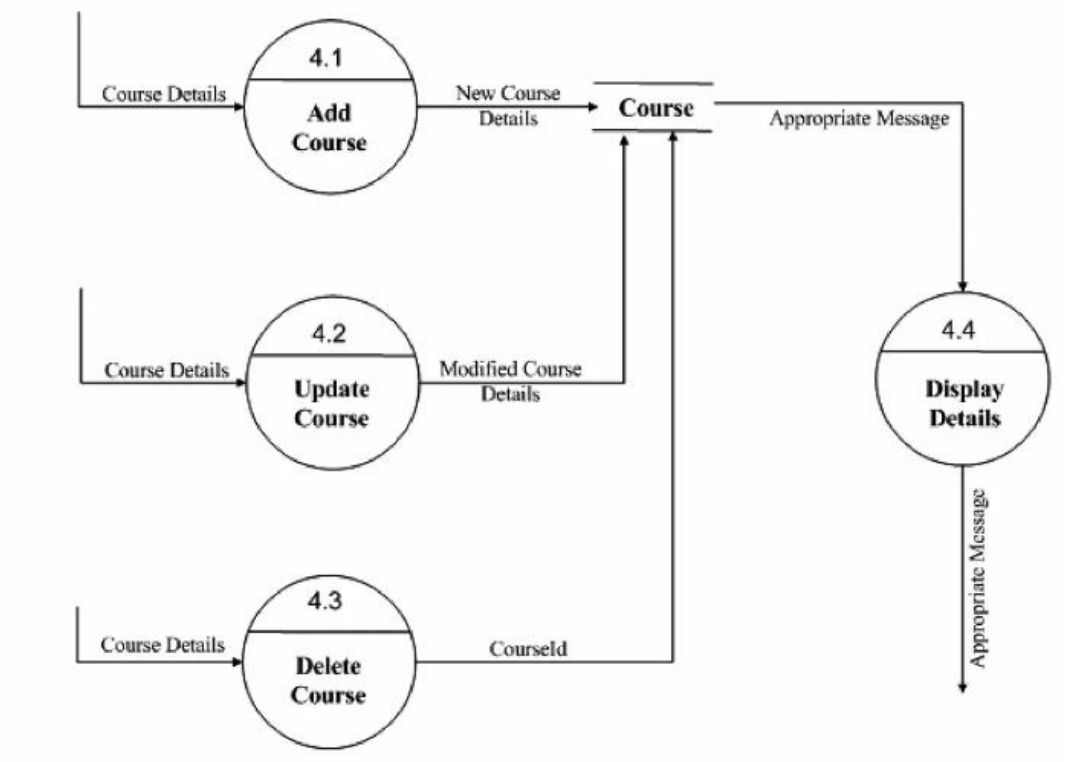
Level 2 DFD For Registration:



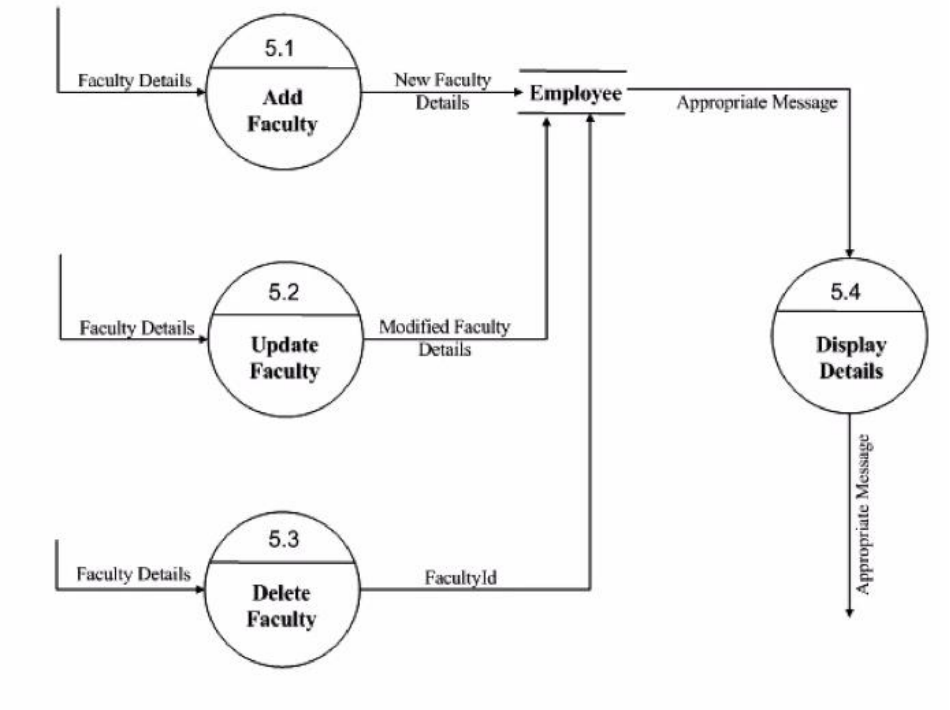
Level 2 DFD For Login:



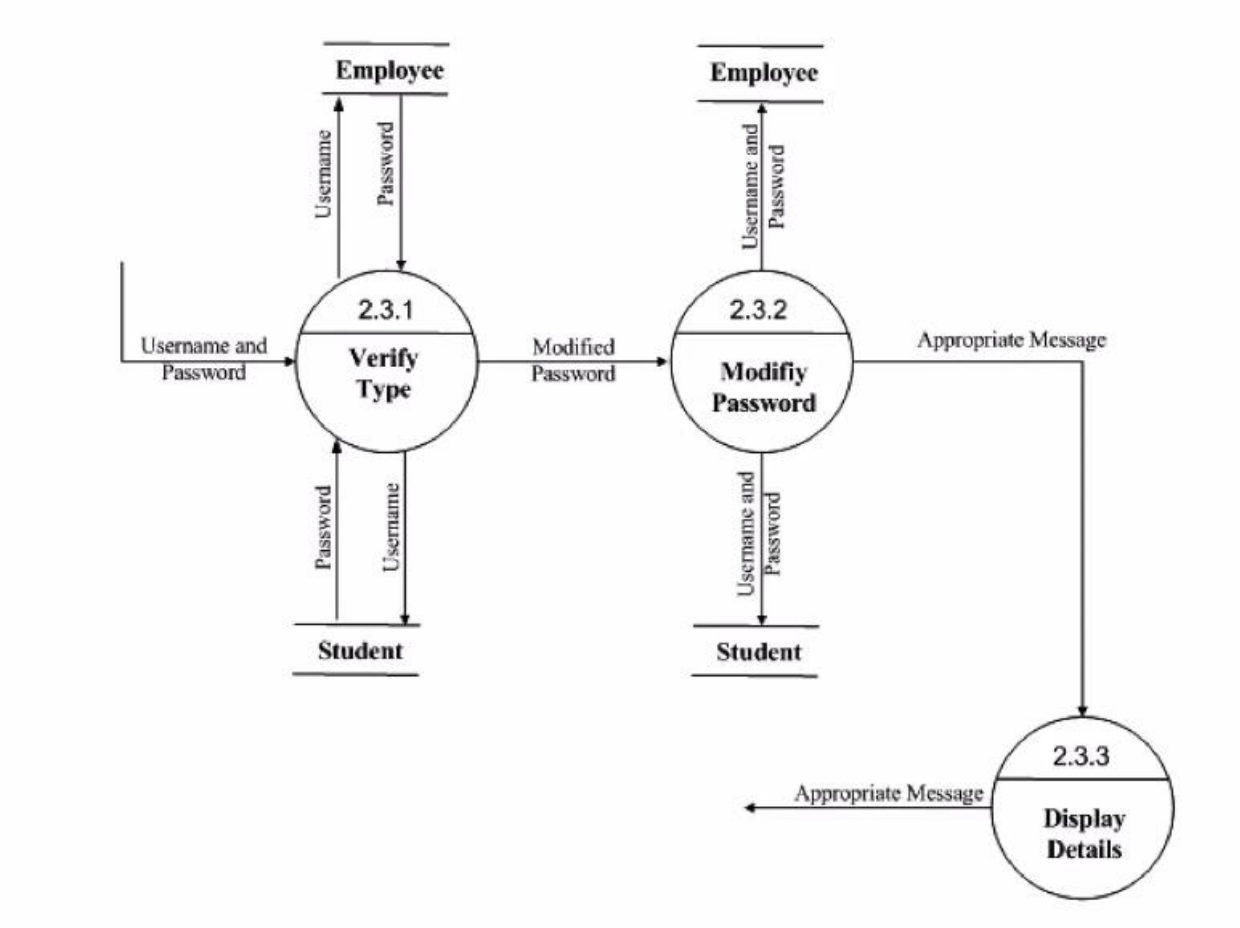
Level 2 DFD For Manage Course:



Level 2 DFD For Manage Faculty:



Level 3 DFD For Login:



2.Sudent Marks Analyzing System:

Level 0 DFD: Student Marks Analyzing System

At the highest level, you have a context-level diagram showing the student marks analyzing system as a whole. It interacts with external entities like students, teachers, administrators, and the database system.

External Entities:

1. **Student:** Provides their marks and other relevant information.
2. **Teacher:** Inputs marks and feedback for students.
3. **Administrator:** Manages user accounts, system settings, and overall system functionality.
4. **Database System:** Stores student information, marks, and other relevant data.

Processes:

1. **Input Student Marks:** Allows teachers to input marks and feedback for students.
2. **Analyze Student Performance:** Analyzes student marks and generates performance reports.
3. **Manage User Accounts:** Allows administrators to add, modify, or delete user accounts.
4. **System Settings:** Manages system configurations and settings.
5. **Generate Reports:** Creates various reports such as individual student reports, class-wise performance reports, and subject-wise analysis reports.

Data Flows:

1. **Student Marks Data:** Marks and feedback provided by teachers for individual students.
2. **Student Information:** Details about students, including names, IDs, and other relevant information.
3. **User Account Information:** Data related to user accounts, including usernames, passwords, and roles.
4. **System Configuration Data:** Information about system settings and configurations.
5. **Performance Reports:** Reports generated based on the analysis of student marks and other data.

Data Stores:

1. **Student Database:** Stores student information, including names, IDs, and other relevant details.
2. **Marks Database:** Contains data about student marks, feedback, and related information.
3. **User Account Database:** Stores information about user accounts, including usernames, passwords, and roles.
4. **System Configuration Database:** Contains data about system settings and configurations.

Level 1 DFD: Detailed Processes

At this level, each process from the Level 0 DFD can be broken down into more detailed subprocesses.

Input Student Marks Process:

1. **Verify Teacher:** Authenticates the teacher's credentials before allowing access to input marks.
2. **Enter Marks:** Allows teachers to enter marks and additional feedback for students.
3. **Store Marks:** Stores the entered marks and feedback in the marks database.

Analyze Student Performance Process:

1. **Retrieve Student Marks:** Retrieves marks and feedback data from the marks database.
2. **Perform Analysis:** Analyzes student marks to identify trends, strengths, and areas for improvement.
3. **Generate Performance Reports:** Creates performance reports based on the analysis, including graphical representations if required.
4. **Notify Teachers and Students:** Sends notifications to teachers and students about their performance and feedback.

Manage User Accounts Process:

1. **Add User Account:** Allows administrators to add new user accounts with appropriate roles and permissions.
2. **Modify User Account:** Enables administrators to modify existing user account information.
3. **Delete User Account:** Allows administrators to delete user accounts if necessary.

System Settings Process:

1. **Retrieve Configuration Data:** Retrieves system configuration data from the system configuration database.
2. **Modify Settings:** Allows administrators to modify system settings, such as report formats and analysis parameters.

Generate Reports Process:

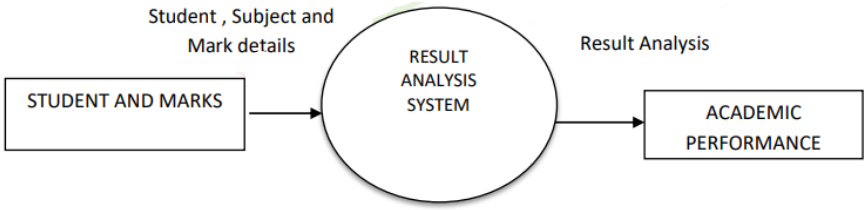
1. **Collect Data:** Gathers relevant information from the marks database and student/user databases.
2. **Generate Individual Student Reports:** Creates detailed reports for each student, including marks, feedback, and performance analysis.

3. **Generate Class-wise Reports:** Provides class-wise performance reports, comparing students' marks within the same class.
4. **Generate Subject-wise Reports:** Generates reports analyzing student performance in specific subjects.

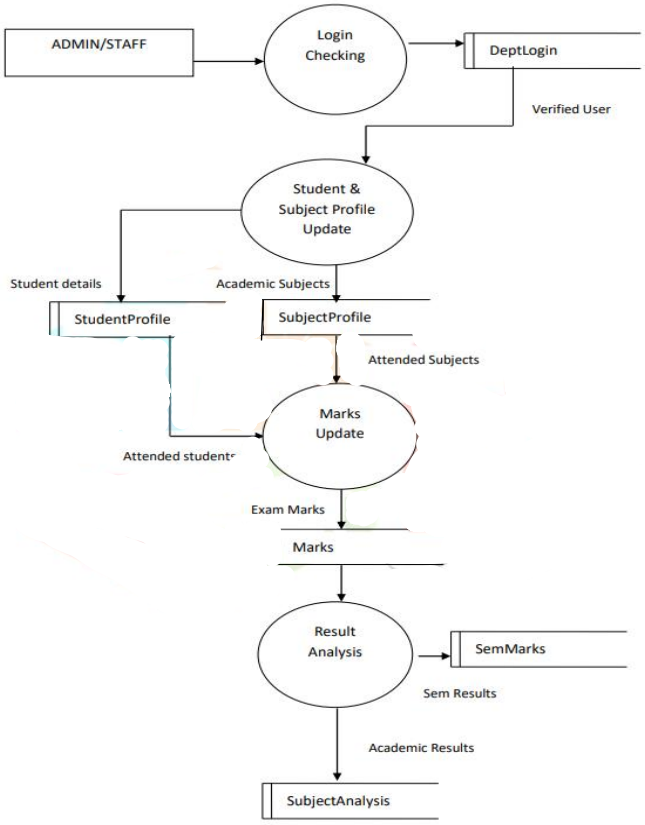
This detailed DFD provides a structured overview of the student marks analyzing system, illustrating how data flows between processes, data stores, and external entities at different levels of the system.

Student Marks Analyzing System (DFD):

Level 0 DFD:



Level 1 DFD:



3.Online Ticket Reservation System:

Level 0 DFD: Online Ticket Reservation System

At the highest level, you have a context-level diagram showing the online ticket reservation system as a whole. It interacts with external entities like customers, administrators, payment gateways, and the database system.

External Entities:

1. **Customer:** Initiates the ticket reservation process by providing travel details and making payments.
2. **Administrator:** Manages the system, including adding/modifying routes, schedules, and handling customer issues.
3. **Payment Gateway:** Processes payment transactions made by customers.
4. **Database System:** Stores information about routes, schedules, seat availability, customer details, and payment records.

Processes:

1. **Search for Routes:** Allows customers to search for available routes based on their travel preferences.
2. **Check Seat Availability:** Verifies the availability of seats on selected routes and schedules.
3. **Reserve Ticket:** Allows customers to reserve tickets after selecting a route and available seats.
4. **Process Payment:** Handles payment transactions made by customers using the payment gateway.
5. **Manage System:** Involves activities related to adding or modifying routes, schedules, and resolving customer issues.
6. **Generate Tickets:** Generates electronic tickets for confirmed reservations.
7. **Notify Customer:** Notifies customers about successful reservations, payment confirmations, and booking details.

Data Flows:

1. **Customer Travel Details:** Information provided by the customer, including departure city, destination city, travel date, and number of passengers.

2. **Route and Seat Availability Data:** Details about available routes, schedules, and the number of seats available on each route.
3. **Reservation Request:** Data indicating the customer's selected route, schedule, and number of passengers.
4. **Payment Information:** Customer payment details sent to the payment gateway for processing.
5. **Confirmation Details:** Information confirming the reservation, including seat numbers, booking reference, and payment status.
6. **System Updates:** Data regarding changes made to routes, schedules, and seat availability, updated by administrators.
7. **Notification Messages:** Messages sent to customers to confirm reservations, payment status, and booking details.

Data Stores:

1. **Route and Schedule Database:** Contains information about available routes, schedules, and seat availability.
2. **Customer Database:** Stores customer information, including names, contact details, and booking history.
3. **Payment Records:** Records of payment transactions made by customers.
4. **System Configuration Data:** Information about system settings and configurations.
5. **Ticket Database:** Stores electronic ticket details for confirmed reservations.

Level 1 DFD: Detailed Processes

At this level, each process from the Level 0 DFD can be broken down into more detailed subprocesses.

Search for Routes Process:

1. **Receive Search Criteria:** Accepts customer travel details as input.
2. **Retrieve Route Information:** Retrieves available routes and schedules based on the provided travel details.
3. **Display Search Results:** Displays the search results to the customer for selection.

Check Seat Availability Process:

1. **Receive Route and Date:** Accepts selected route and travel date from the customer.

2. **Check Seat Availability:** Verifies the availability of seats on the selected route and date.
3. **Display Seat Availability:** Shows the customer the number of available seats and seat configurations.

Reserve Ticket Process:

1. **Receive Reservation Request:** Accepts the reservation request from the customer, including selected seats and passenger details.
2. **Update Seat Availability:** Adjusts the available seat count based on the reserved seats.
3. **Generate Booking Reference:** Creates a unique booking reference for the reservation.
4. **Store Reservation Details:** Stores reservation details, including customer information, reserved seats, and booking reference.

Process Payment Process:

1. **Receive Payment Information:** Accepts payment details from the customer.
2. **Process Payment:** Initiates the payment transaction through the payment gateway.
3. **Receive Payment Confirmation:** Receives payment confirmation from the payment gateway.
4. **Update Payment Records:** Records the payment transaction in the payment records database.

Manage System Process:

1. **Receive System Modification Request:** Accepts requests from administrators to add, modify, or delete routes and schedules.
2. **Update Route and Schedule Database:** Modifies the route and schedule information based on the administrator's request.
3. **Handle Customer Issues:** Addresses customer issues, such as cancellations, refunds, and other inquiries.

Generate Tickets Process:

1. **Retrieve Reservation Details:** Retrieves reservation details using the booking reference.
2. **Generate Electronic Ticket:** Creates an electronic ticket with passenger names, seat numbers, and travel details.

3. **Store Electronic Ticket:** Stores the generated electronic ticket in the ticket database.
4. **Send Ticket to Customer:** Sends the electronic ticket to the customer via email or mobile app notification.

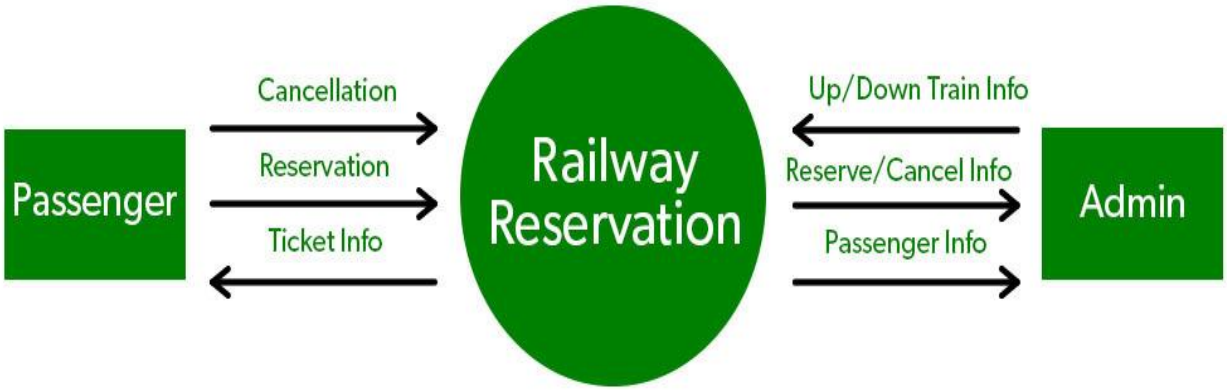
Notify Customer Process:

1. **Prepare Notification Messages:** Generates notification messages confirming reservations, payment status, and booking details.
2. **Send Notifications:** Sends the prepared notification messages to customers via email or SMS.

This detailed DFD provides a structured overview of the online ticket reservation system, illustrating how data flows between processes, data stores, and external entities at different levels of the system.

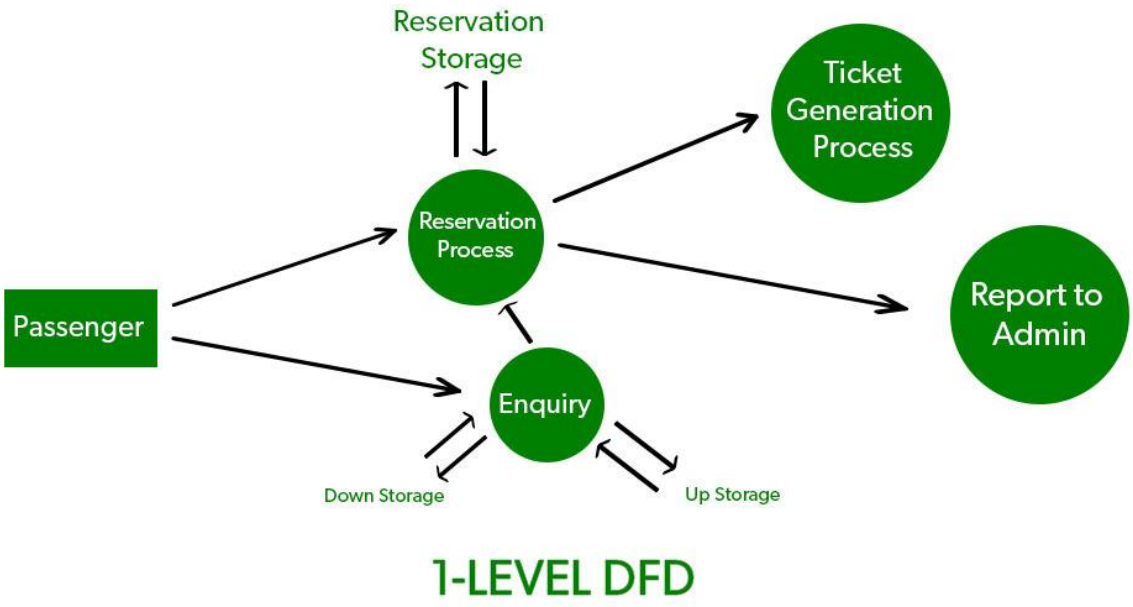
Online Ticket Resevation System (DFD):

Level 0 DFD:



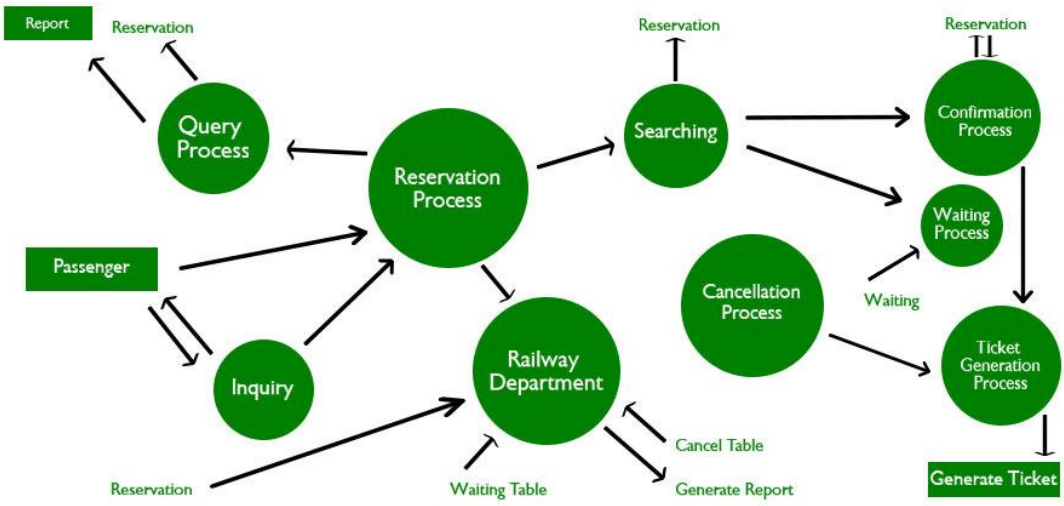
0-LEVEL DFD

Level 1 DFD:



1-LEVEL DFD

Level 2 DFD:



2-LEVEL DFD

4.Stock Maintenance:

Level 0 DFD: Stock Maintenance System

At the highest level, you have a context-level diagram showing the stock maintenance system as a whole. It interacts with external entities like suppliers, customers, and administrators.

External Entities:

1. **Supplier:** Provides information about new stock items, including quantity, description, and price.
2. **Customer:** Places orders for stock items.
3. **Administrator:** Manages the stock system, including updating stock levels, adding new items, and generating reports.

Processes:

1. **Update Stock Levels:** Adjusts the stock levels based on information received from suppliers and customer orders.
2. **Manage Stock Items:** Involves adding new items to the stock, updating item details, and removing items that are no longer available.
3. **Process Customer Orders:** Handles customer orders by checking stock availability, processing orders, and updating stock levels accordingly.
4. **Generate Reports:** Creates various reports such as stock levels, sales reports, and order histories.

Data Flows:

1. **Stock Item Information:** Details about stock items, including item codes, descriptions, quantities, and prices.
2. **Supplier Information:** Information about new stock items provided by suppliers.
3. **Customer Orders:** Orders placed by customers, including item codes, quantities, and delivery details.
4. **Stock Level Updates:** Information about changes in stock levels, including additions and deductions.
5. **Reports:** Generated reports about stock levels, sales, and order histories.

Data Stores:

1. **Stock Database:** Contains information about stock items, including item codes, descriptions, quantities, prices, and current stock levels.
2. **Supplier Database:** Stores information about suppliers, including names, contact details, and items supplied.
3. **Customer Database:** Contains information about customers, including names, contact details, and order histories.

4. **Order History Database:** Records information about customer orders, including order dates, items ordered, quantities, and delivery details.

Level 1 DFD: Detailed Processes

At this level, each process from the Level 0 DFD can be broken down into more detailed subprocesses.

Update Stock Levels Process:

1. **Receive Supplier Information:** Accepts new stock item details from suppliers.
2. **Update Stock Database:** Adds the new stock items to the stock database and adjusts quantities based on the received information.

Manage Stock Items Process:

1. **Receive Stock Item Updates:** Accepts requests from administrators to add new items, update existing items, or remove items from the stock.
2. **Modify Stock Database:** Updates the stock database based on the changes requested by administrators.

Process Customer Orders Process:

1. **Receive Customer Orders:** Accepts customer orders including item codes, quantities, and delivery details.
2. **Check Stock Availability:** Verifies if the ordered items are available in the stock database.
3. **Update Stock Levels:** Reduces the stock levels for the items included in the processed order.
4. **Record Order History:** Stores information about the processed order in the order history database.

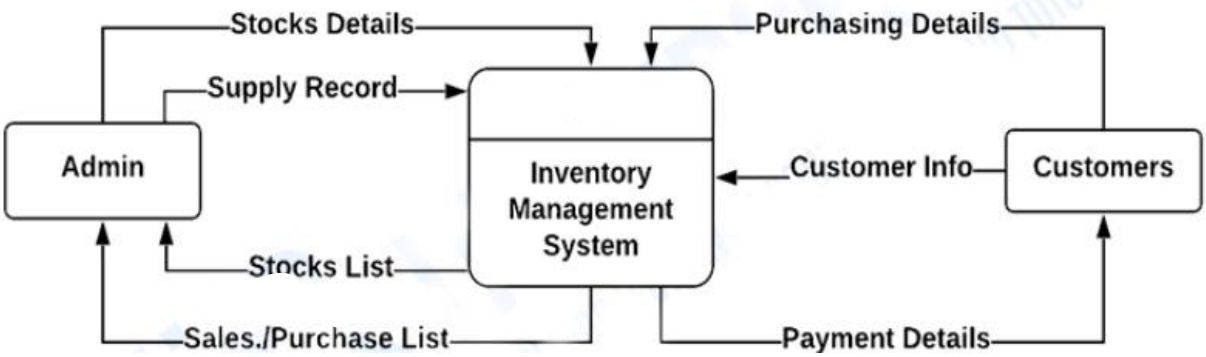
Generate Reports Process:

1. **Collect Data:** Gathers relevant information from the stock database and order history database.
2. **Generate Stock Level Reports:** Creates reports detailing current stock levels for various items.
3. **Generate Sales Reports:** Provides information about sales, including item-wise sales, revenue, and customer details.
4. **Generate Order History Reports:** Generates reports summarizing customer orders, including order dates, items ordered, and delivery details.

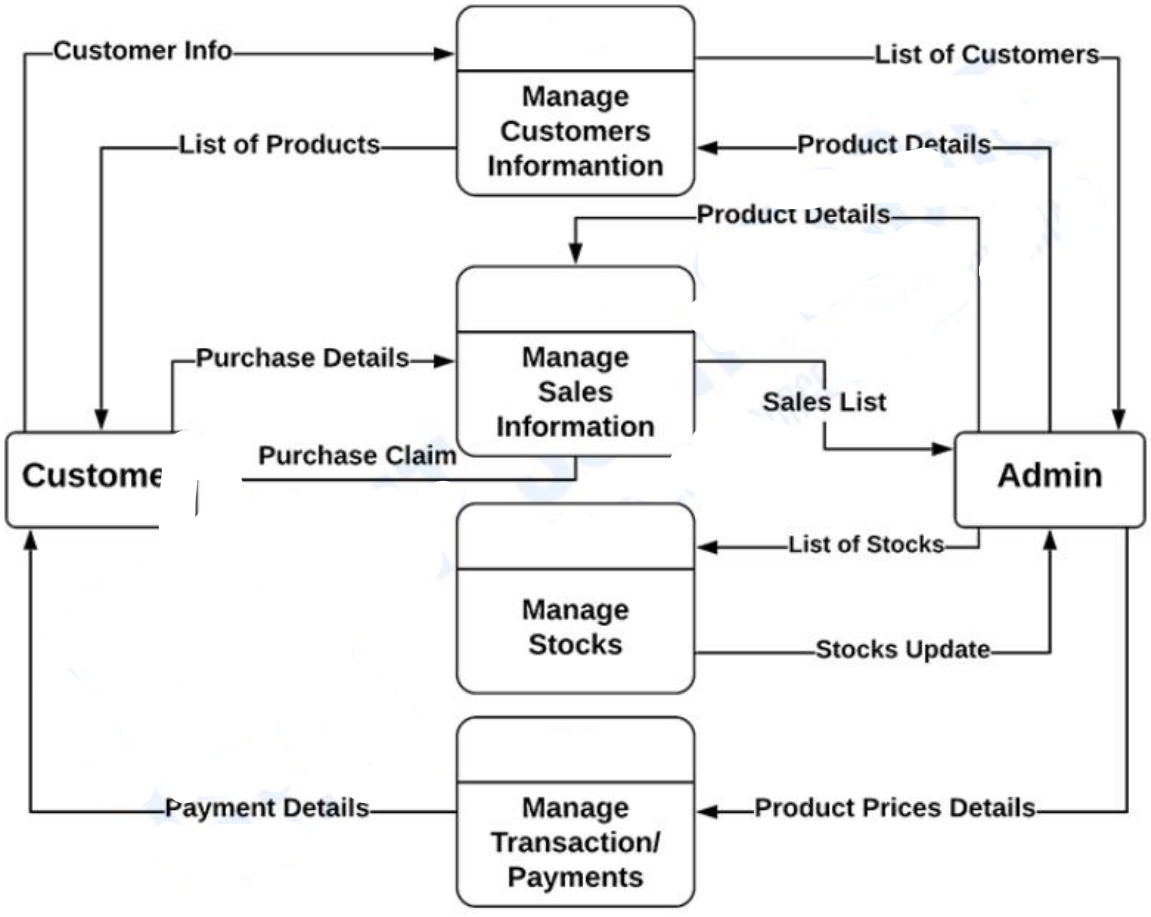
This detailed DFD provides a structured overview of the stock maintenance system, illustrating how data flows between processes, data stores, and external entities at different levels of the system.

Stock Maintenance (DFD):

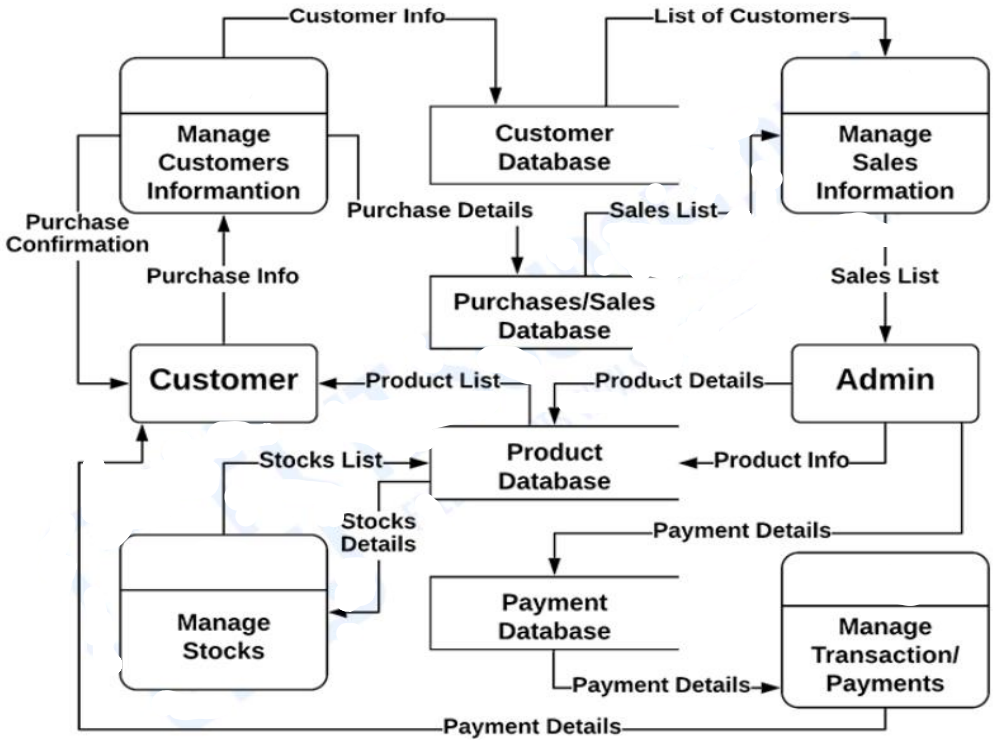
Level 0 DFD:



Level 1 DFD:



Level 2 DFD:



RESULT: Thus the DFD for an Stock Maintenence system has been developed.

EXPERIMENT 5:

5.Consider any application using COCOMO model estimate the effort.

COCOMO Model: Boehm proposed COCOMO (Constructive Cost Estimation Model) in 1981.COCOMO is one of the most generally used software estimation models in the world. COCOMO predicts the efforts and schedule of a software product based on the size of the software.

The necessary steps in this model are:

1. Get an initial estimate of the development effort from evaluation of thousands of delivered lines of source code (KDLOC).

2. Determine a set of 15 multiplying factors from various attributes of the project.

3. Calculate the effort estimate by multiplying the initial estimate with all the multiplying factors i.e., multiply the values in step1 and step 2.

The initial estimate (also called nominal estimate) is determined by an equation of the form used in the static single variable models, using KDLOC as the measure of the size. To determine the initial effort Ei in person-months the equation used is of the type is shown below.

$$Ei=a*(KDLOC)b$$

The value of the constant a and b are depends on the project type.

In COCOMO, projects are categorized into three types:

1. Organic

2. Semidetached

3. Embedded

1.Organic: A development project can be treated of the organic type, if the project deals with developing a well-understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar methods of projects.

Examples of this type of projects are simple business systems, simple inventory management systems, and data processing systems.

2. Semidetached: A development project can be treated with semidetached type if the development consists of a mixture of experienced and inexperienced staff. Team members may have finite experience.

Example of Semidetached system includes developing a new operating system (OS), a Database Management System (DBMS), and complex inventory management system.

3. Embedded: A development project is treated to be of an embedded type, if the software being developed is strongly coupled to complex hardware, or if the stringent regulations on the operational method exist. For Example: ATM, Air Traffic control.

For three product categories, Bohem provides a different set of expression to predict effort (in a unit of person month)and development time from the size of estimation in KLOC(Kilo Line of code) efforts estimation takes into account the productivity loss due to holidays, weekly off, coffee breaks, etc.

According to Boehm, software cost estimation should be done through three stages:

1. Basic Model

2. Intermediate Model

3. Detailed Model

1. Basic COCOMO Model: The basic COCOMO model provide an accurate size of the project parameters. The following expressions give the basic COCOMO estimation model.

Effort=a1*(KLOC) a2 PM

Tdev=b1*(efforts)b2 Months

Where,

KLOC is the estimated size of the software product indicate in Kilo Lines of Code,a1,a2,b1,b2 are constants for each group of software products,Tdev is the estimated time to develop the software, expressed in months.

Effort is the total effort required to develop the software product, expressed in person months **(PMs).**

Estimation of development effort

For the three classes of software products, the formulas for estimating the effort based on the code size are shown below:

Organic: Effort = 2.4(KLOC) 1.05 PM

Example1: Suppose a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three model i.e., organic, semi-detached & embedded.

Solution: The basic COCOMO equation takes the form:

Effort=a1*(KLOC) a2 PM

Tdev=b1*(efforts)b2 Months

Estimated Size of project= 400 KLOC

1.Organic Mode

E = 2.4 * (400)1.05 = 1295.31 PM

D = 2.5 * (1295.31)0.38=38.07 PM

2.Semidetached Mode

E = 3.0 * (400)1.12=2462.79 PM

D = 2.5 * (2462.79)0.35=38.45 PM

3.Embedded Mode

E = 3.6 * (400)1.20 = 4772.81 PM

D = 2.5 * (4772.8)0.32 = 38 PM

EXPERIMENT 6:

Consider any aplication,calculate effort using FP oriented estimation model.

FP (Function Point) ORIENTED ESTIMATION MODEL Calculating effort for Attendance Management System using FP oriented model. Objective Calculating effort for Attendance Management System using Function Point oriented estimation model. It is a method to break systems into smaller components, so they can be better understood and analyzed. It is used to express the amount of business functionality, an information system (as a product) provides to a user. Fps measure software size. They are widely accepted as an industry standard for functional sizing. Function points are used to compute a functional size measurement (FSM) of software. The cost (in dollars or hours) of a single unit is calculated from past projects.Function Point Analysis can provide a mechanism to track and monitor scope creep. Function Point Counts at the end of requirements, analysis, design, code, testing and implementation can be compared. The function point count at the end of requirements and/or designs can be compared to function points actually delivered. The amount of growth is an indication of how well requirements were gathered by and/or communicated to the project team. If the amount of growth of projects declines over time it is a natural assumption that communication with the user has improved. Overview Function-oriented software metrics use a measure of the functionality delivered by the application as a normalization value. Since 'functionality cannot be measured directly, it must be derived indirectly using other direct measures. Function-oriented metrics were first proposed by Albrecht, who suggested a measure called the function point. Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and assessments of software complexity. Function points are computed by completing the table as Laboratory Five information domain characteristics are determined and counts are provided in the appropriate table location. Information domain values are defined in the following manner:

			Weighting factor			
Measurement parameter	**Count**		**Simple**	**Average**	**Complex**	
Number of user inputs		×	3	4	6	=
Number of user outputs		×	4	5	7	=
Number of user inquiries		×	3	4	6	=
Number of files		×	7	10	15	=
Number of external interfaces		×	5	7	10	=
Count total						

Number of user inputs: Each user input that provides distinct application oriented data to the software is counted. Inputs should be distinguished from inquiries, which are counted separately.

Number of user outputs: Each user output that provides application oriented information to the user is counted. In this context output refers to reports, screens, error messages, etc. Individual data items within a report are not counted separately.

Number of user inquiries: An inquiry is defined as an on-line input that results in the generation of some immediate software response in the form of an on-line output. Each distinct inquiry is counted.

Number of files: Each logical master file (i.e., a logical grouping of data that may be one part of a large database or a separate file) is counted.

Number of external interfaces: All machine readable interfaces (e.g., data files on storage media) that are used to transmit information to another system are counted.Once these data have been collected.

FP POINTS COMPUTATION :

To compute function points (FP), the following relationship is used:

FP = count total [0.65 + 0.01 Σ (Fi)]

where count total is the sum of all FP entries .

The Fi (i = 1 to 14) are "complexity adjustment values" based on responses to the following questions :

1. Does the system require reliable backup and recovery?
2. Are data communications required?
3. Are there distributed processing functions?
4. Is performance critical?
5. Will the system run in an existing, heavily utilized operational environment?
6. Does the system require on-line data entry?
7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?
8. Are the master files updated on-line?
9. Are the inputs, outputs, files, or inquiries complex?
10. Is the internal processing complex?
11. Is the code designed to be reusable?
12. Are conversion and installation included in the design?

13. Is the system designed for multiple installations in different organizations?

14. Is the application designed to facilitate change and ease of use by the user?

Each of these questions is answered using a scale that ranges from 0 (not important or applicable) to 5 (absolutely essential). applied to information domain counts are determined empirically. Once function points have been calculated, they are used in a manner analogous to LOC as a way to normalize measures for,

software productivity, quality, and other attributes:

Errors per FP.

Productivity = FP/ Person-Month

Quality = No of faults / FP

Cost= $/FP

Documentation = Pages count / FP.

Effort = FP/ Person-Month

Count Total can be obtained using the following table

Domain characteristics	Count		Weighting factor			Count
			Simple	Average	Complex	
No of user input		*	3	4	6	
No of user output		*	4	5	7	
No of user queries		*	3	4	6	
No of files		*	7	10	15	
No of external interfaces		*	5	7	10	
Count total:						

Example:

Assume that....

Number of user input : 5

Number of user output : 5

Number of user enquires : 6

Number of files :5

Number of external interfaces : 5

Apply these assumptions on a simple project and calculate the Count Total

Domain characteristics	Count		Weighting factor			Count
			Simple	Average	Complex	
No of user input	5	*	3	4	6	15
No of user output	5	*	4	5	7	2
No of user queries	6	*	3	4	6	18
No of files	5	*	7	10	15	35
No of external interfaces	5	*	5	7	10	25
Count total:						113

Therefore Count Total =113

Now calculate the Functional Points using FP = count total *0.65 + 0.01 Σ (Fi)]

FP = count total *0.65 + 0.01 Σ (Fi)]

= 113 * (0.65 +0.01 * 25) where Σ (Fi)

=25

i.e the questions answered using a scale that ranges from 0 (not important or applicable) to 5 (absolutely essential) in total 14 questions

=113 *(0.65 + 0.25)

=113 * 0.9

= 101.7

FP = 101.7

Effort = FP / person-month

ADVANTAGES:

1. This method is independent of programming languages.
2. It is based on the data which can be obtained in early stage of project
3. Function Points are easily understood by the non technical user. This helps communicate sizing information to a user or customer.

DISADVANTAGES:

1. This method is more suitable for Business systems and can be developed for that domain
2. Many aspects of this method are not validated
3. The functional point has no significant ant meaning, it’s just a numerical value.

EXPERIMENT 7:

Draw the UML diagrams for the 1,2,3,4.

1.COURSE REGISTRATION SYSTEM (UML):

Aim: To create a UML model for course registration system.

1. ANALYSIS:

1.1 Identify the Actors

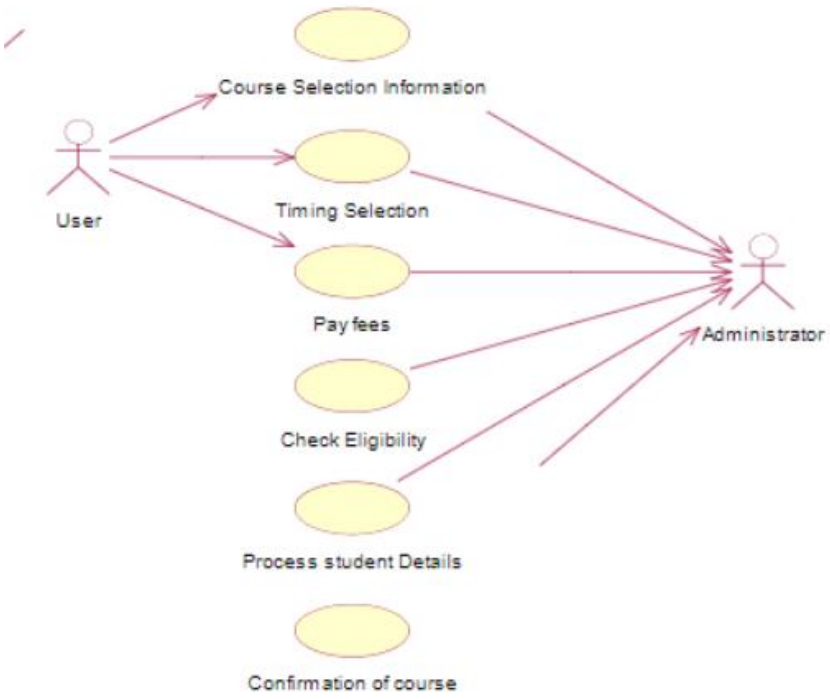
- User
- Administrator

1.2 Identify the Use Cases

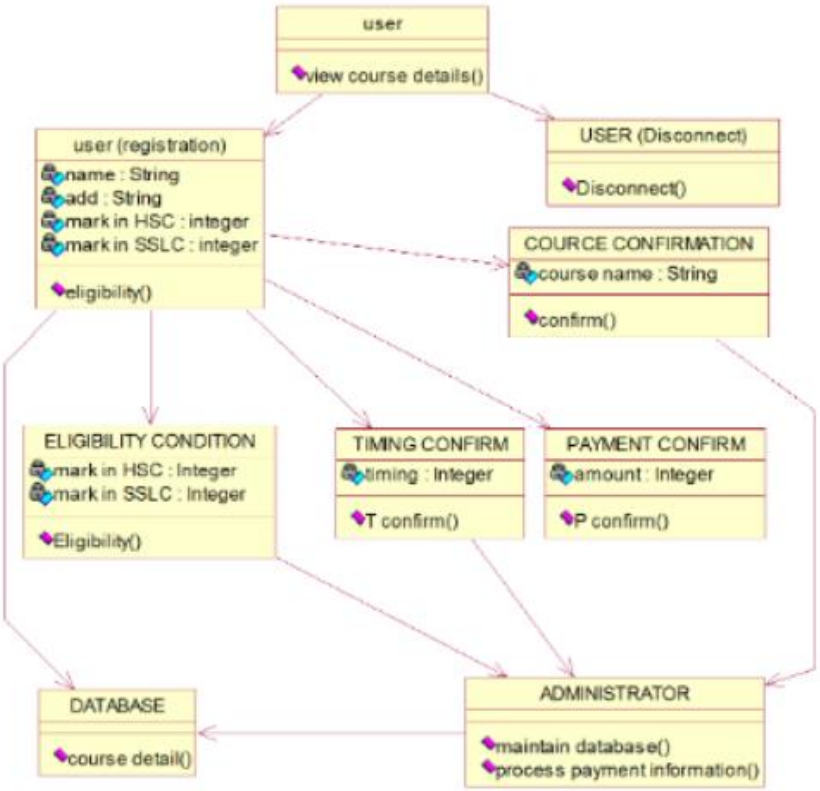
- Course Selection Information
- Timing Selection
- Pay fees

2.DESIGN

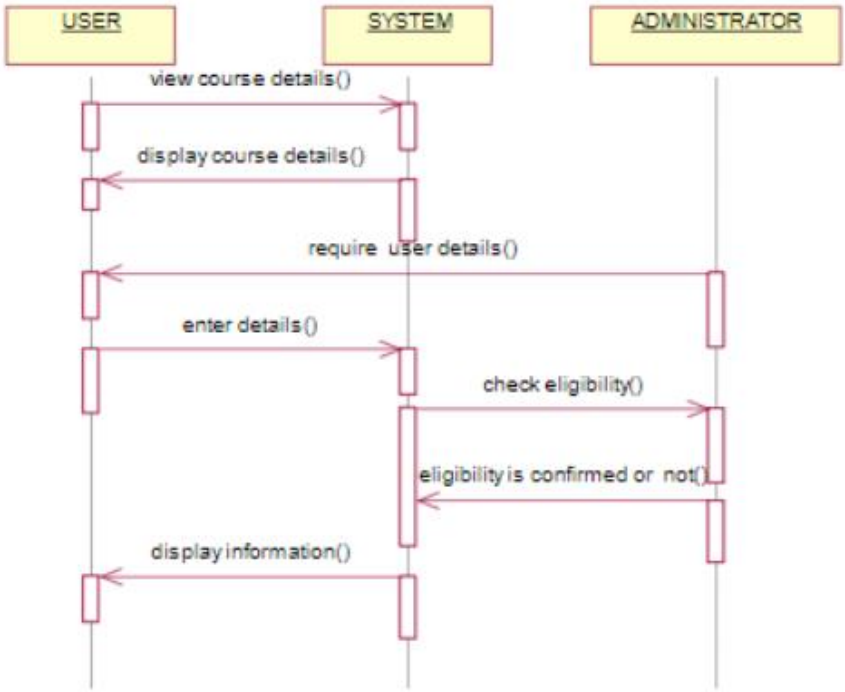
2.1 Use Case Diagram



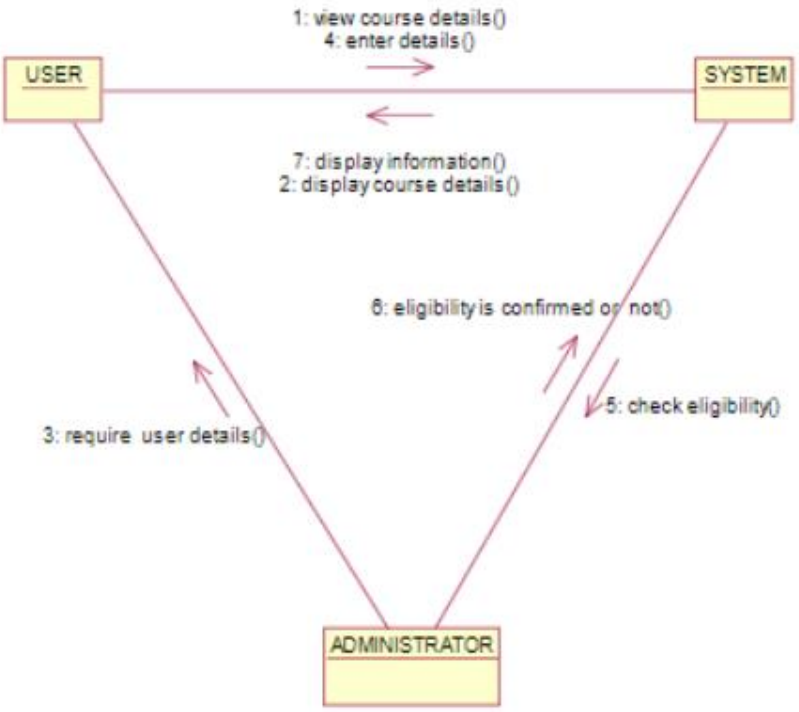
2.2 Class Diagram



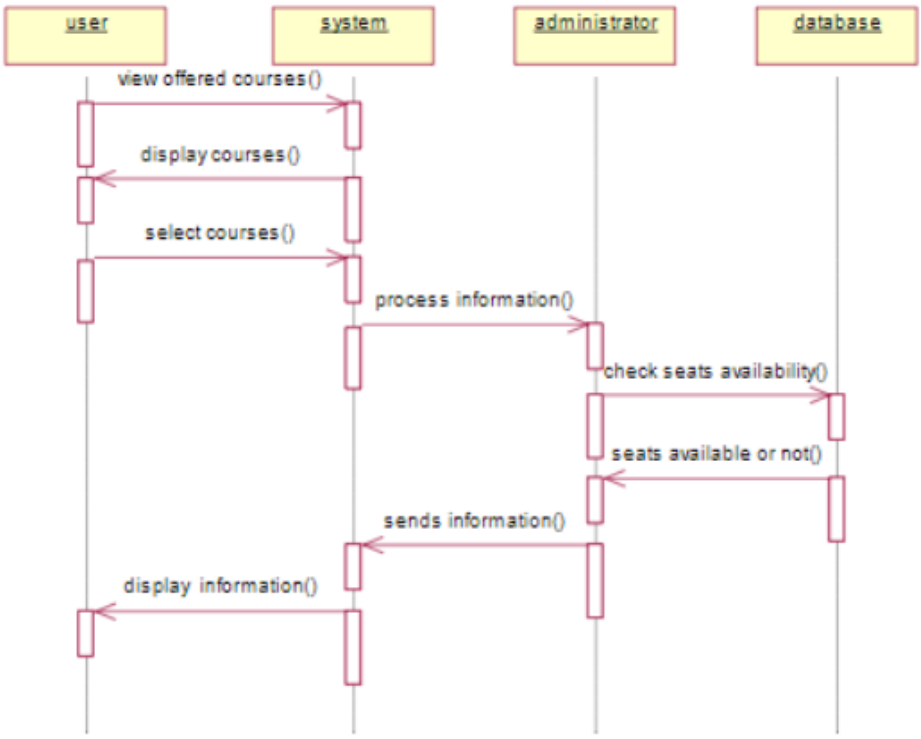
2.3 Sequence Diagram



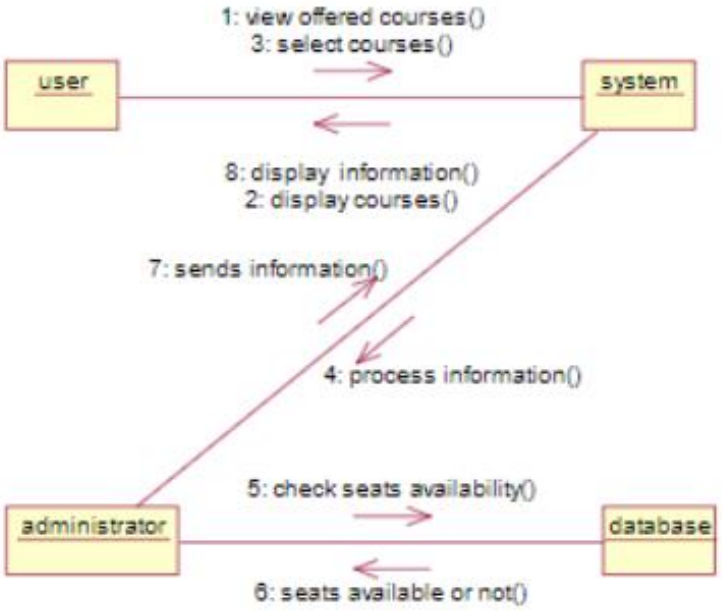
2.4 Collaboration Diagram



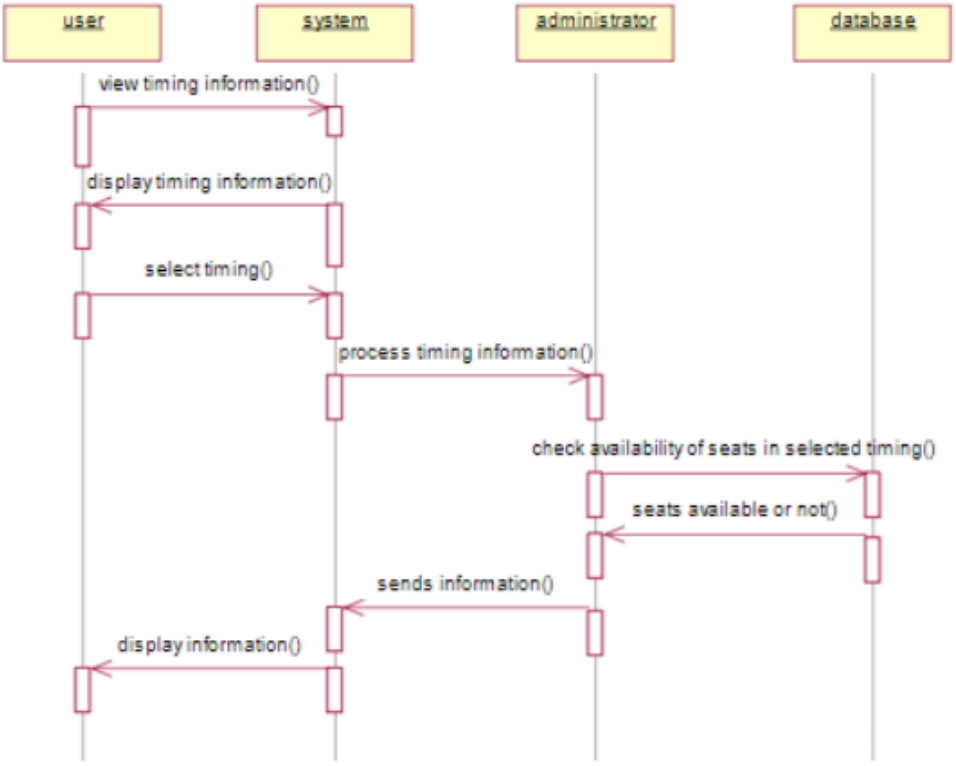
2.5 Sequence Diagram



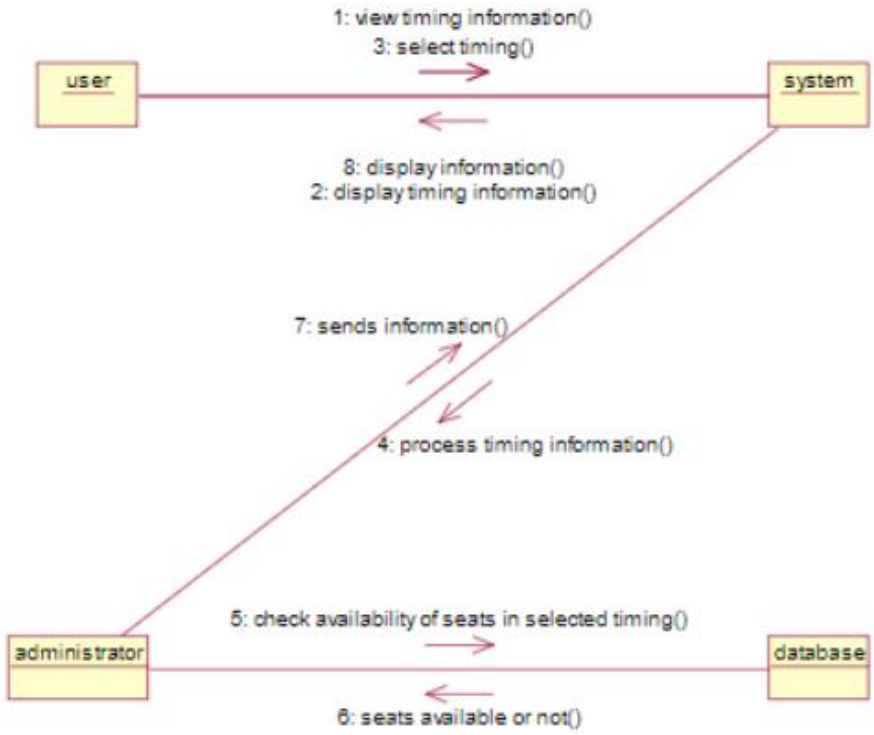
2.6 Collaboration Diagram



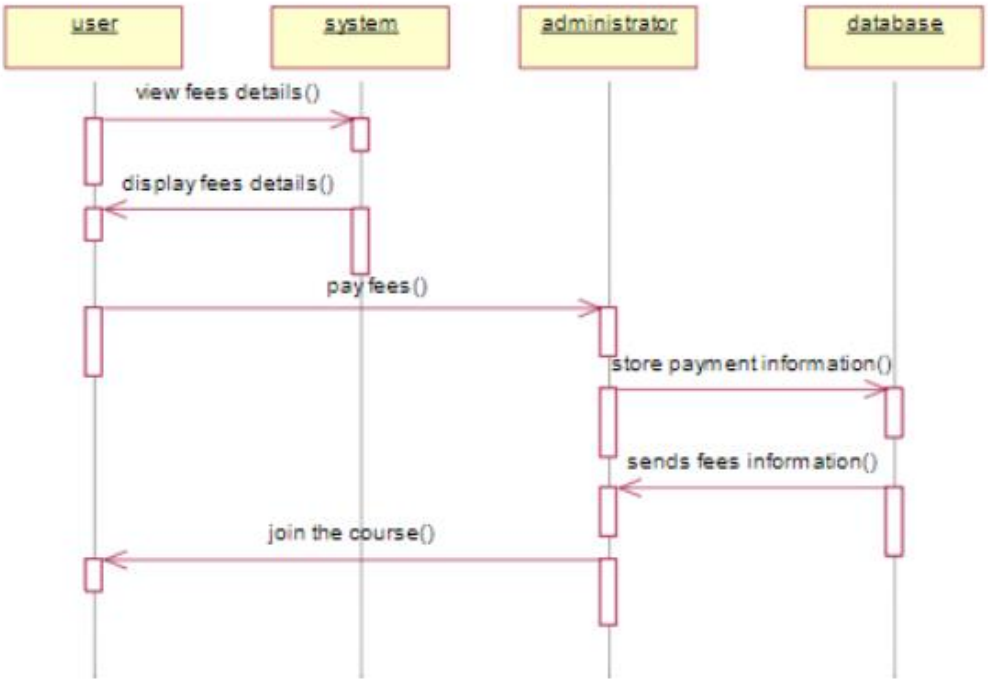
2.7 Sequence Diagram



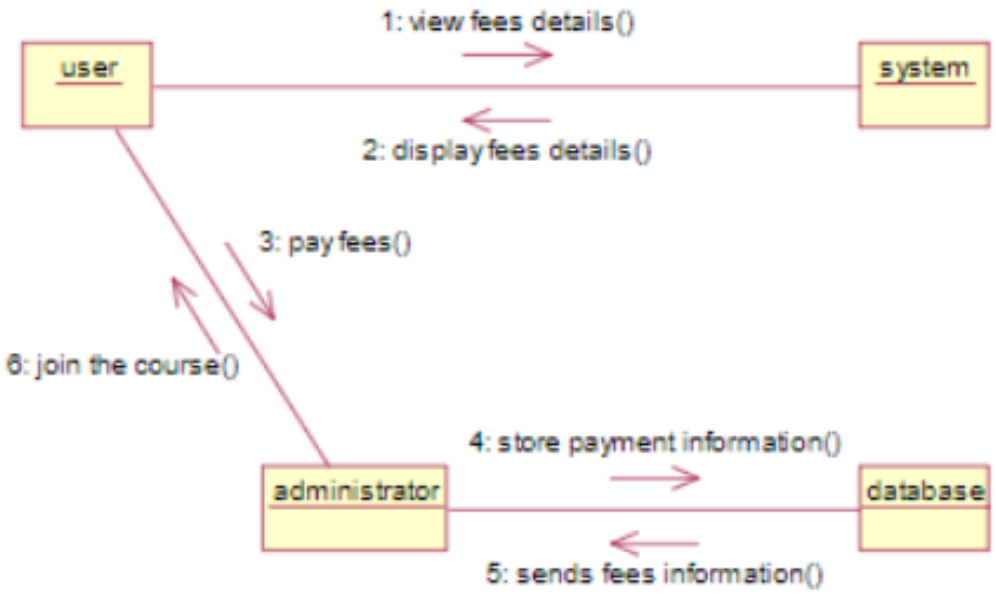
2.8 Collaboration Diagram



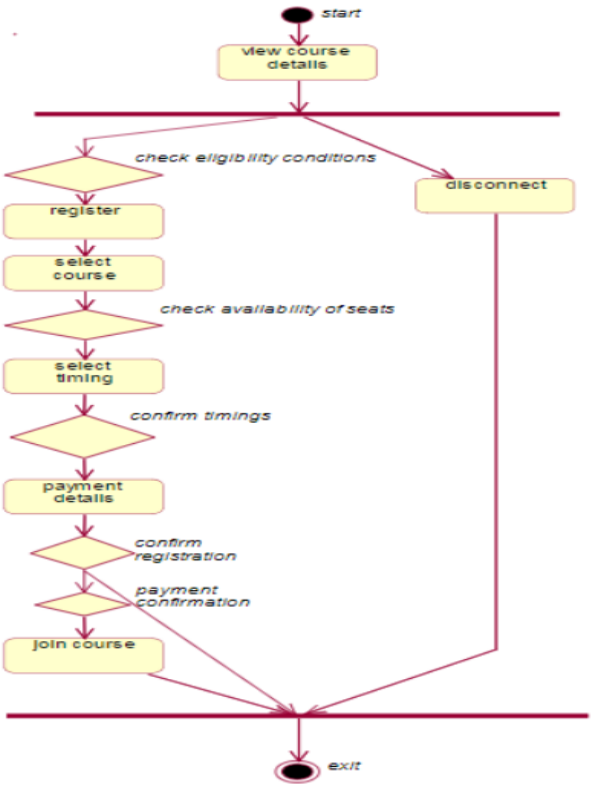
2.9 Sequence Diagram



2.10 Collaboration Diagram

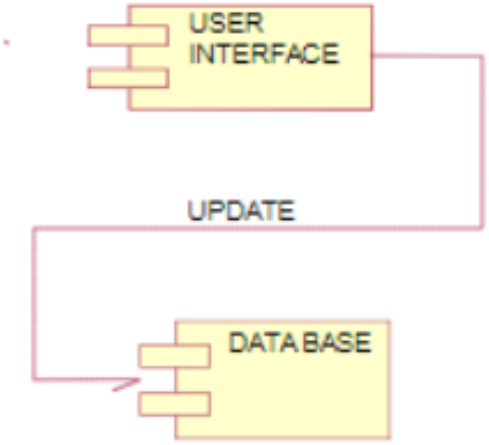


2.11 Activity Diagram

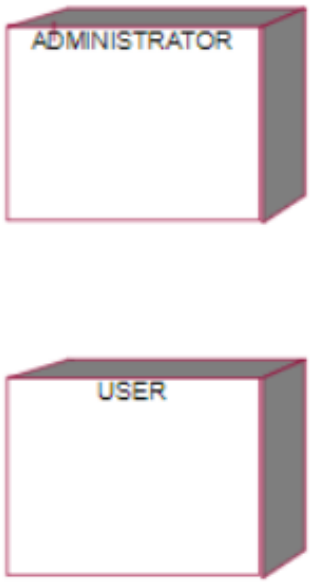


2.12 State chart Diagram

2.13 Component Diagram



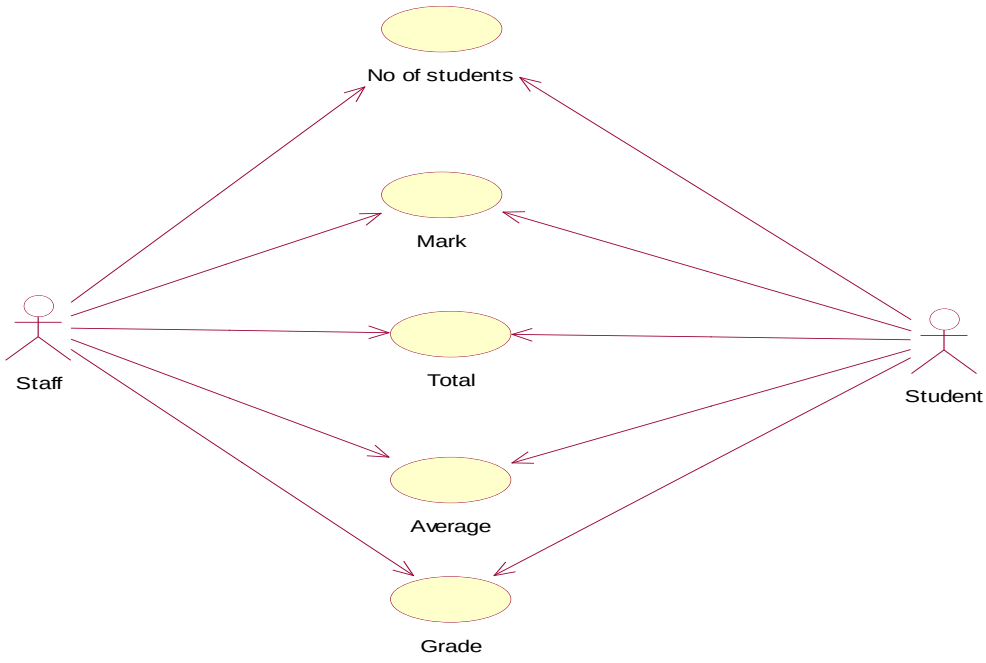
2.14 Deployment Diagram



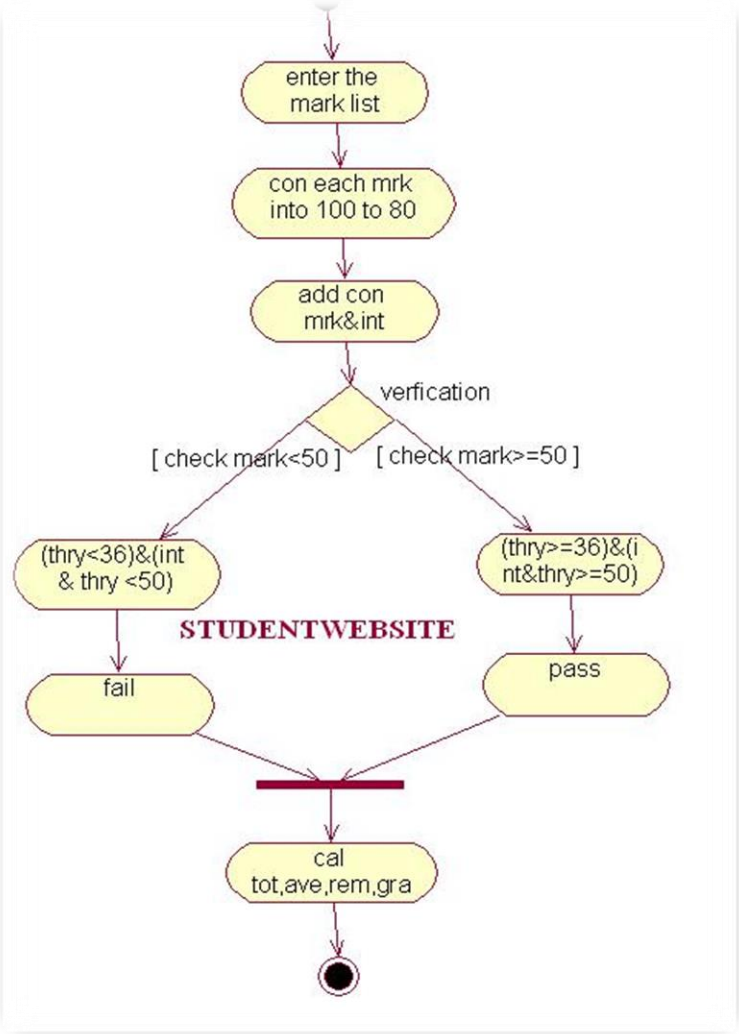
RESULT: Thus the UML model for an online banking systemhas beendeveloped.

STUDENT MARKS ANALYZING SYSTEM:

UseCase Diagram:

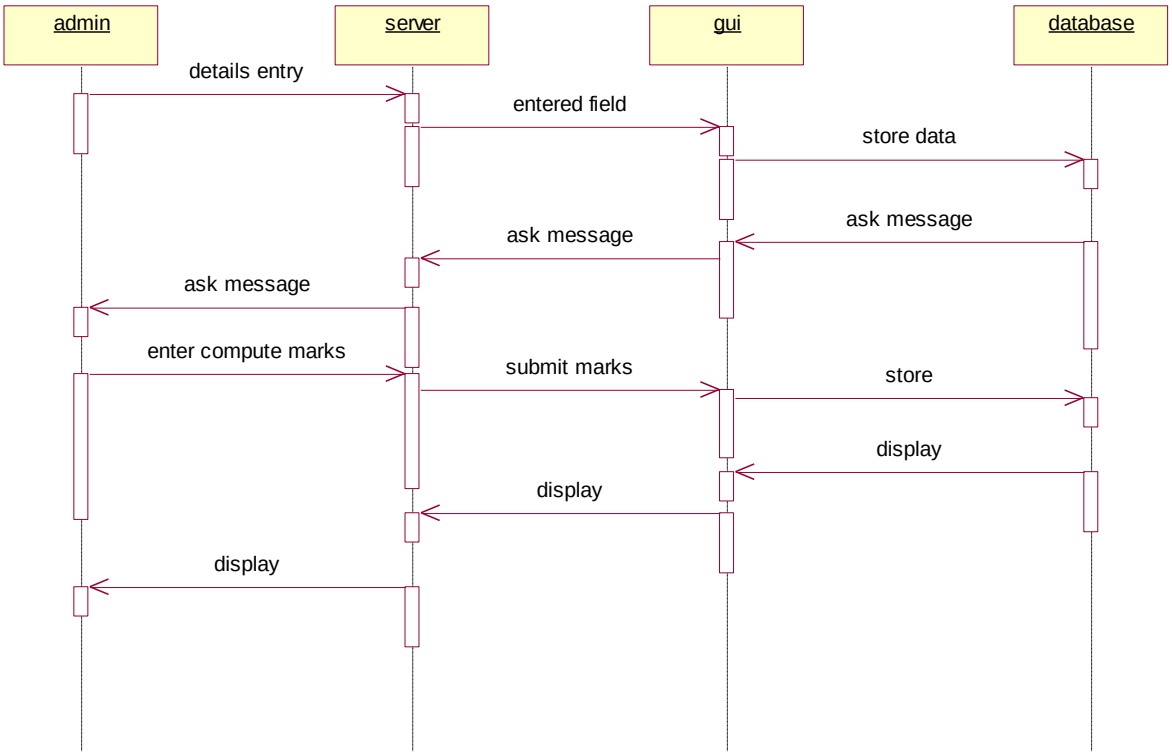


Activity Diagram:

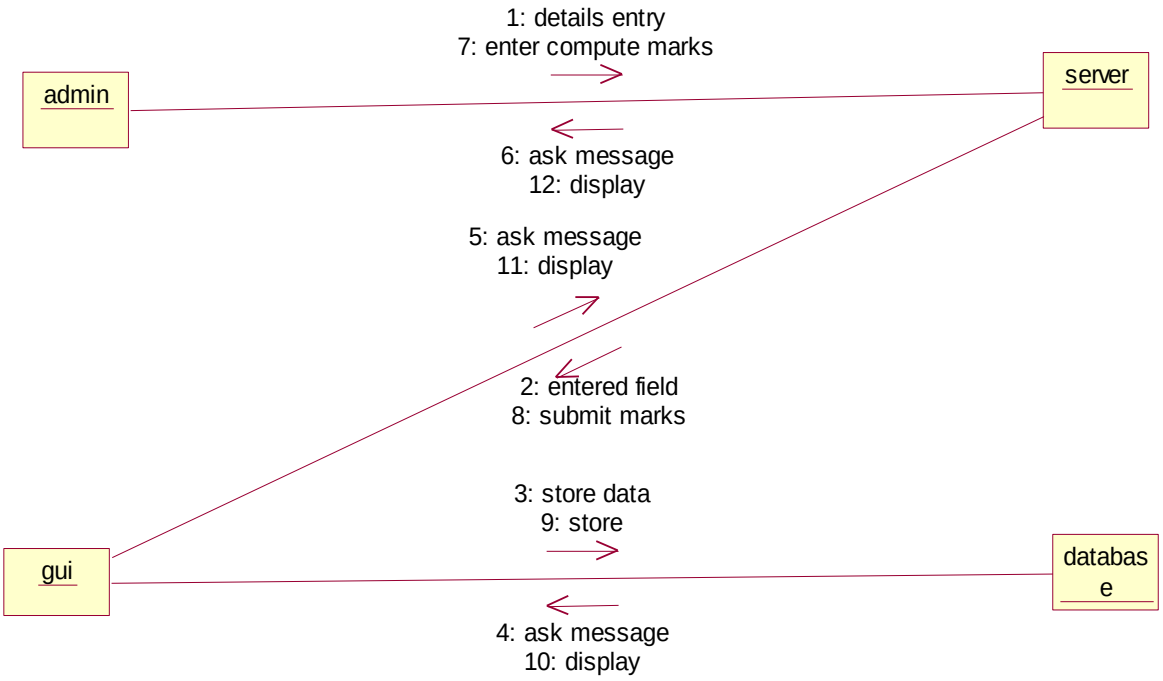


Class Diagram:

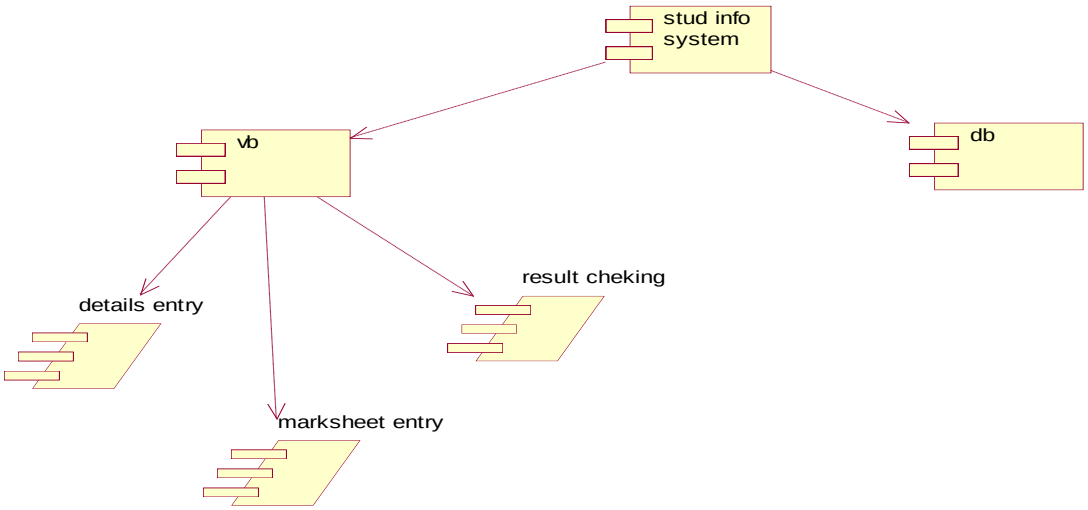
Sequential Diagram:



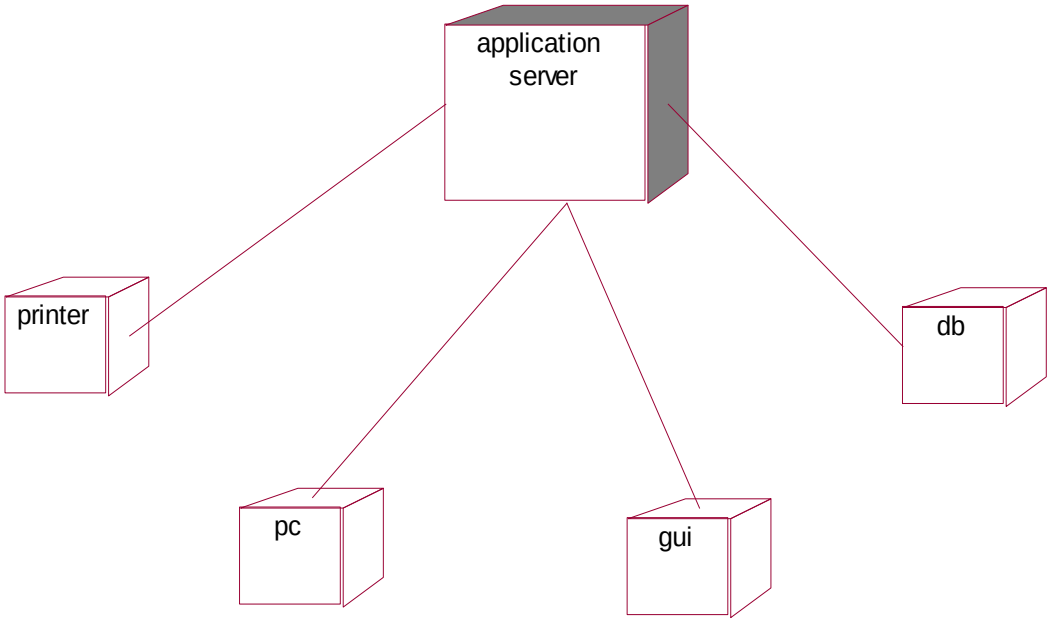
Collaboration diagram :



Component Diagram of student info System:



Deployment Diagram of Student info System:

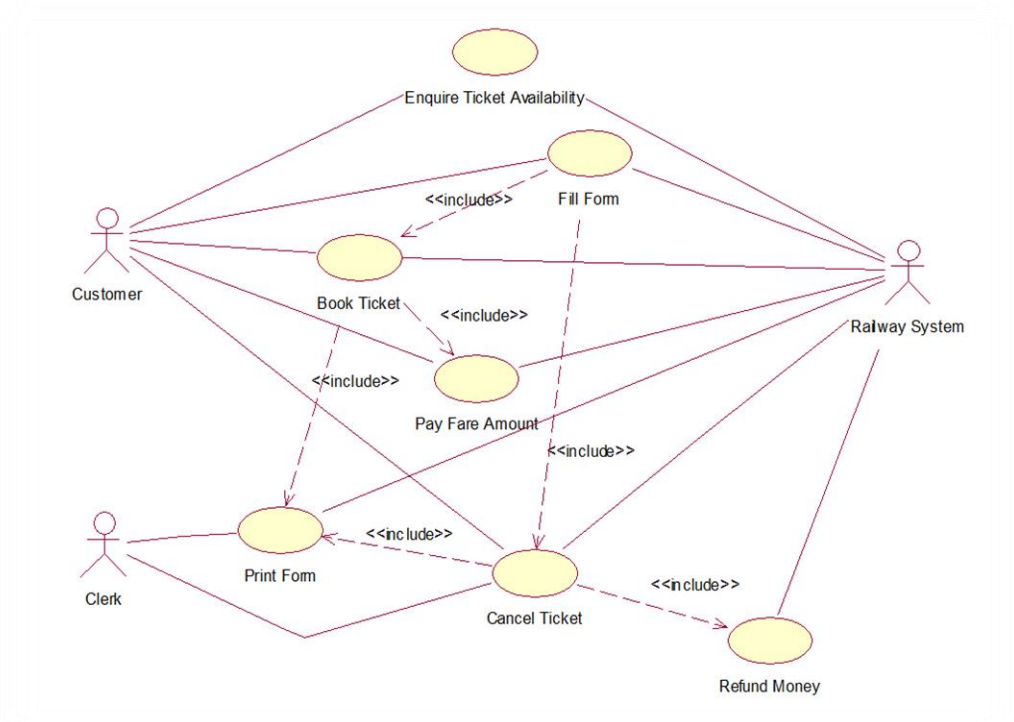


RESULT:

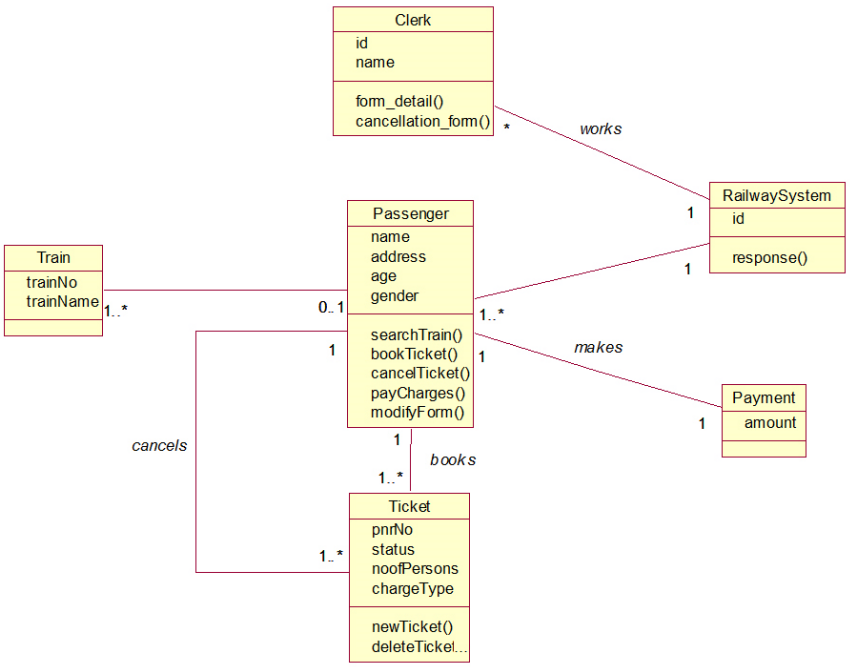
The Student marks analyzing system was designed and implemented successfully.

ONLINE TICKET RESERVATION SYSTEM:

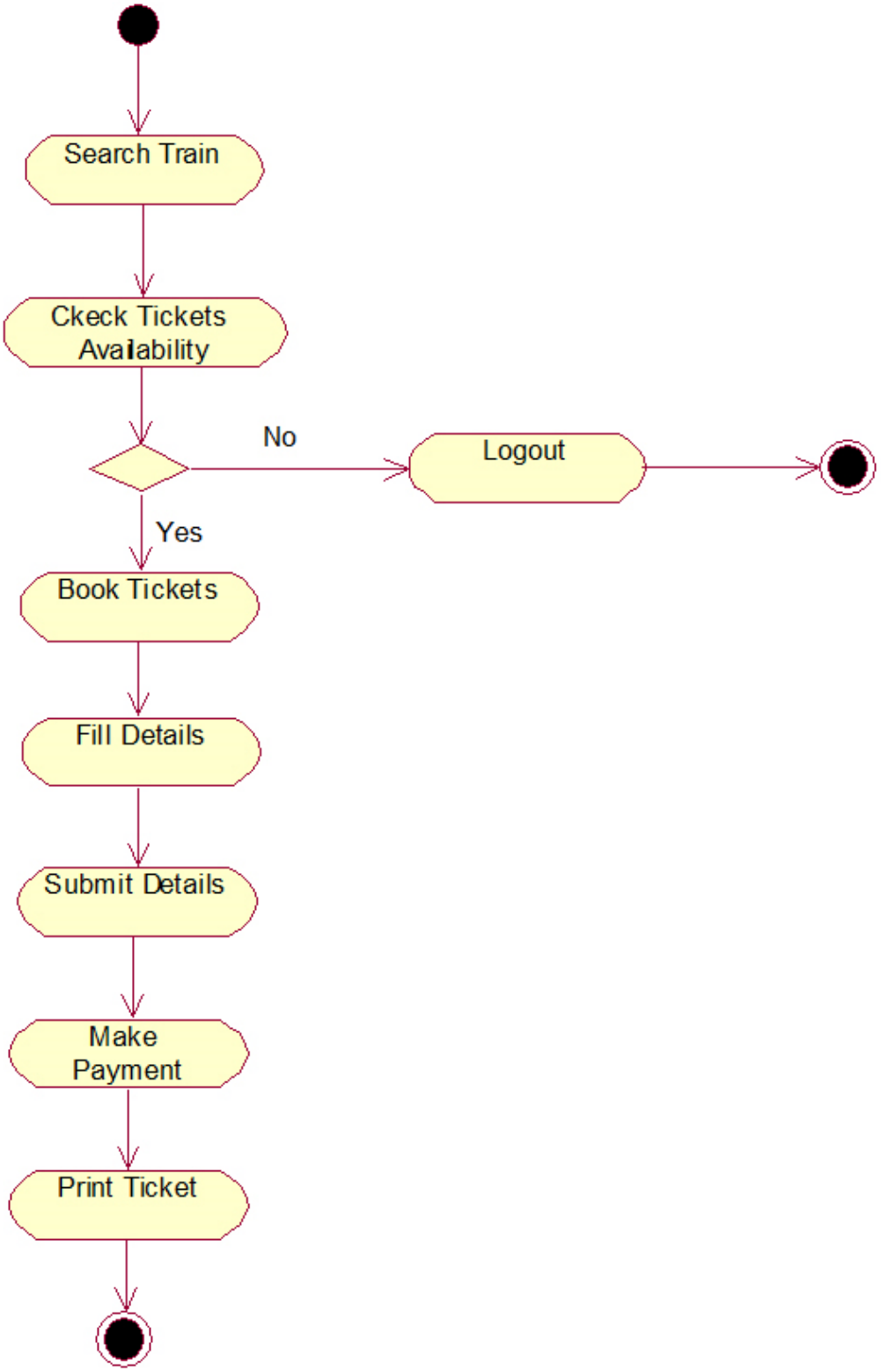
Usecase Diagram:



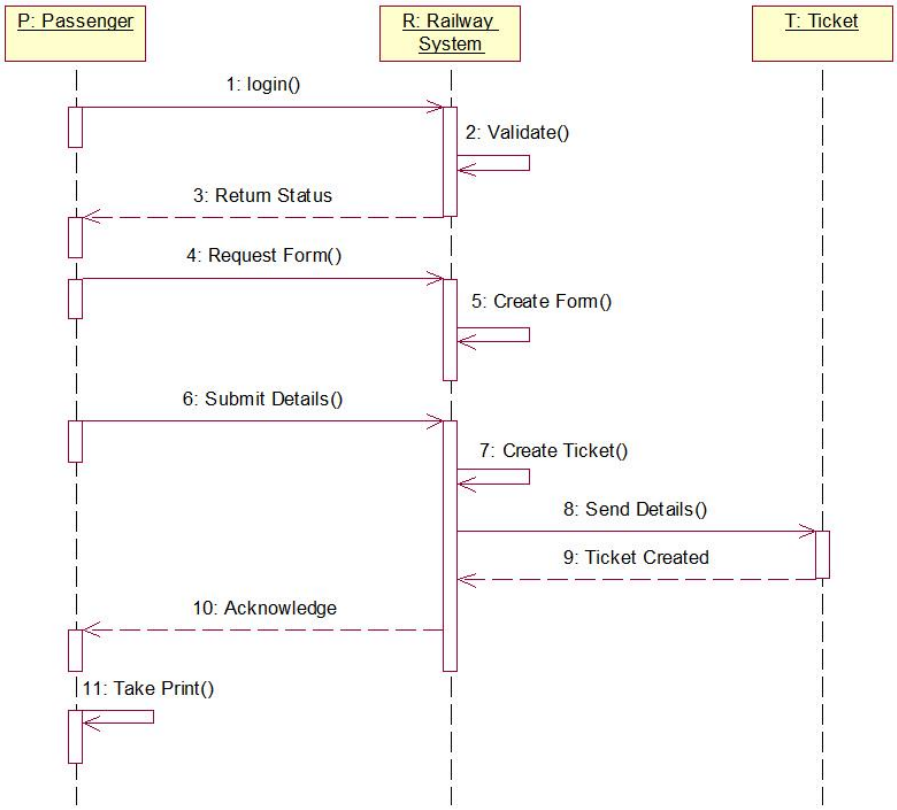
Class Diagram:



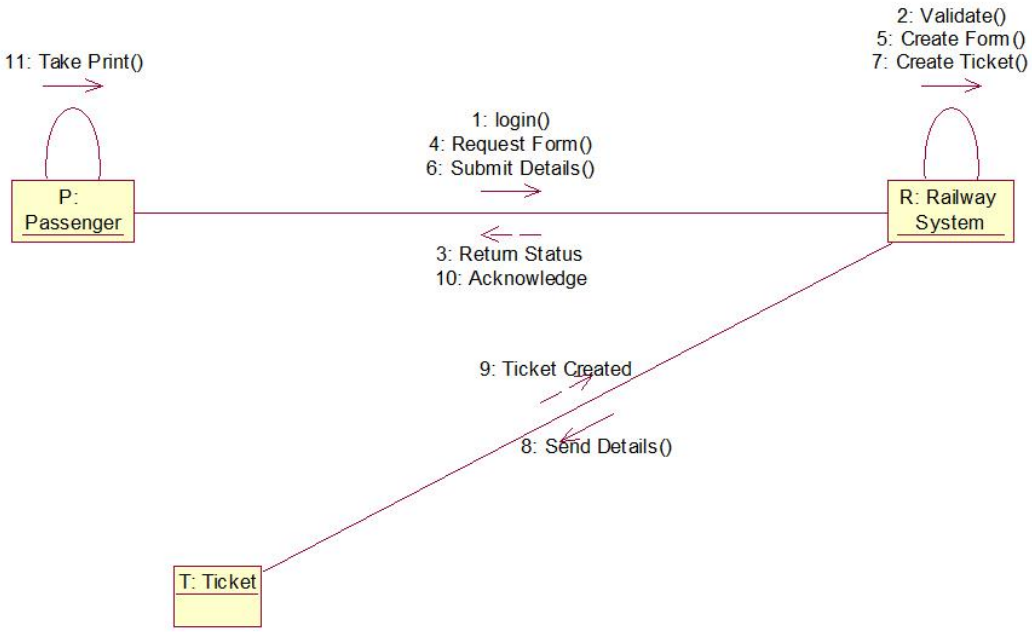
Activity Diagram:



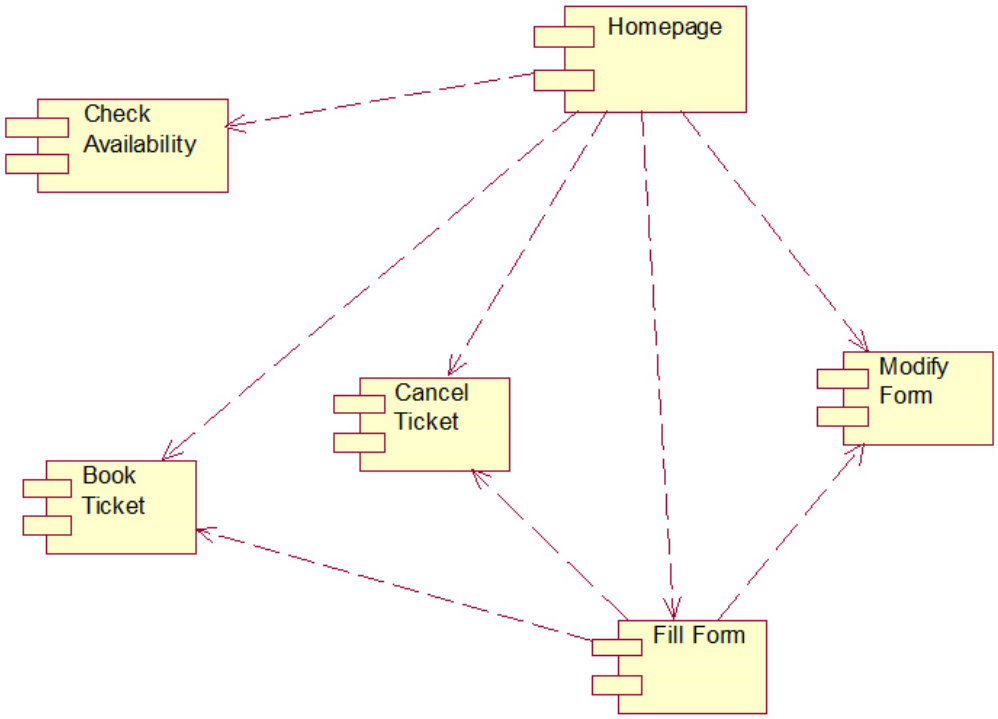
Sequential Diagram:



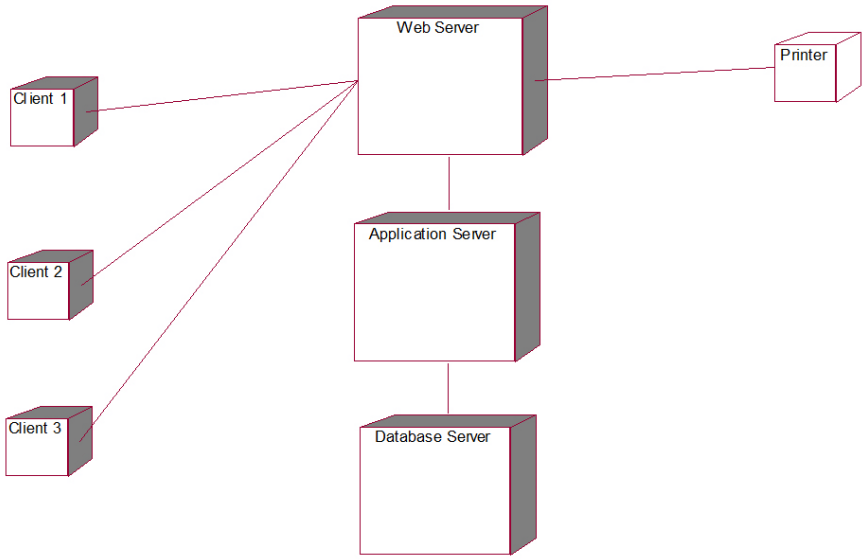
Colloboration Diagram:



Component Diagram:



Deployment Diagram:

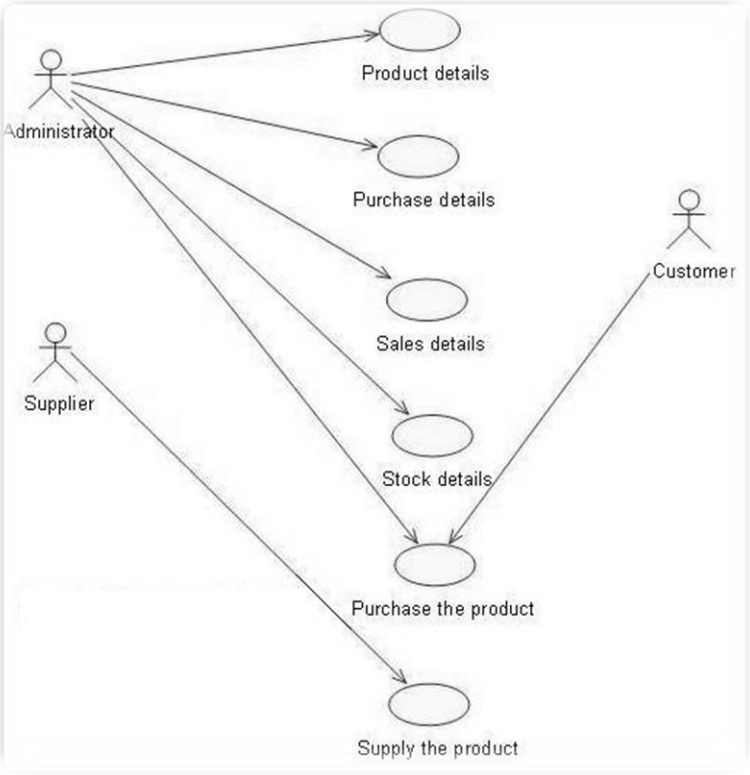


RESULT:

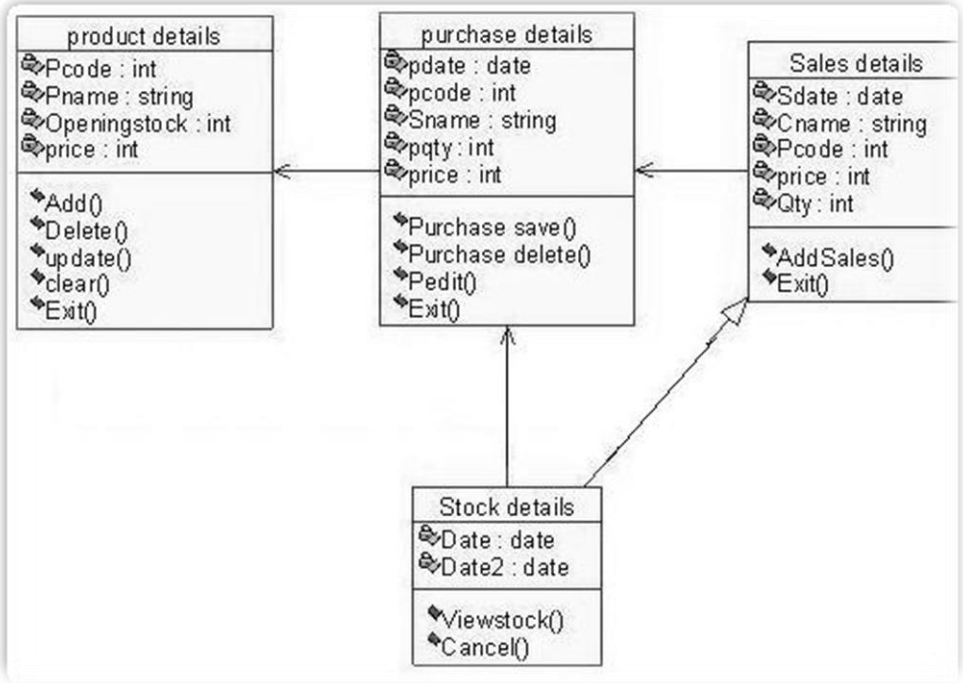
The Student marks analyzing system was designed and implemented successfully.

STOCK MANAGEMENT SYSTEM:

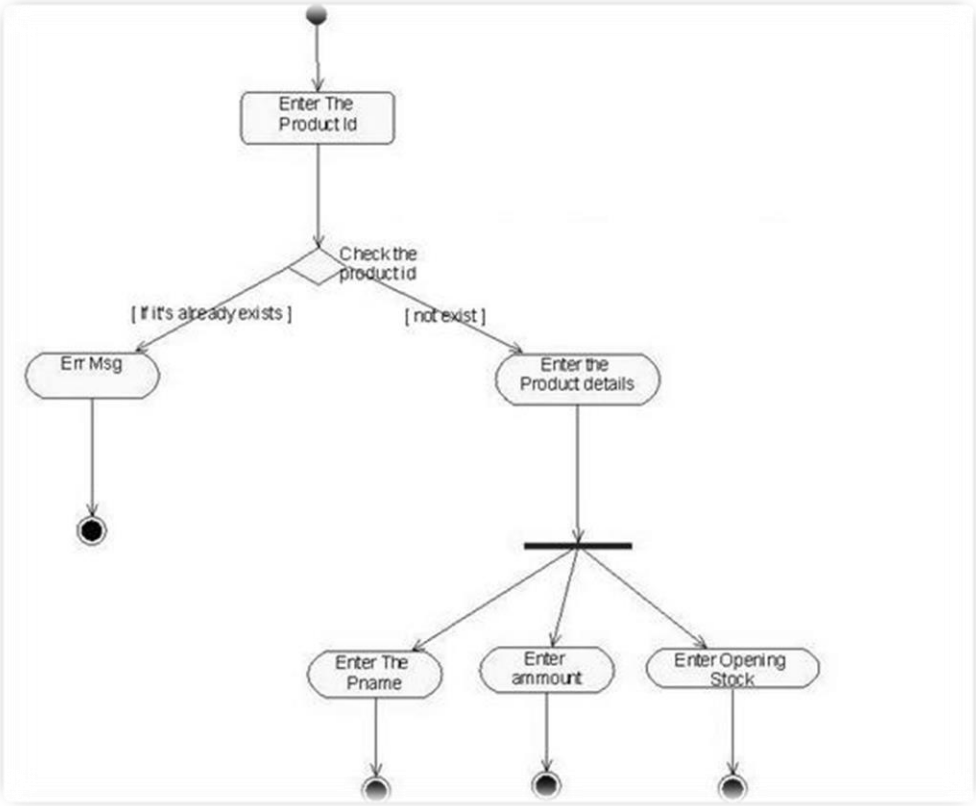
Usecase diagram:



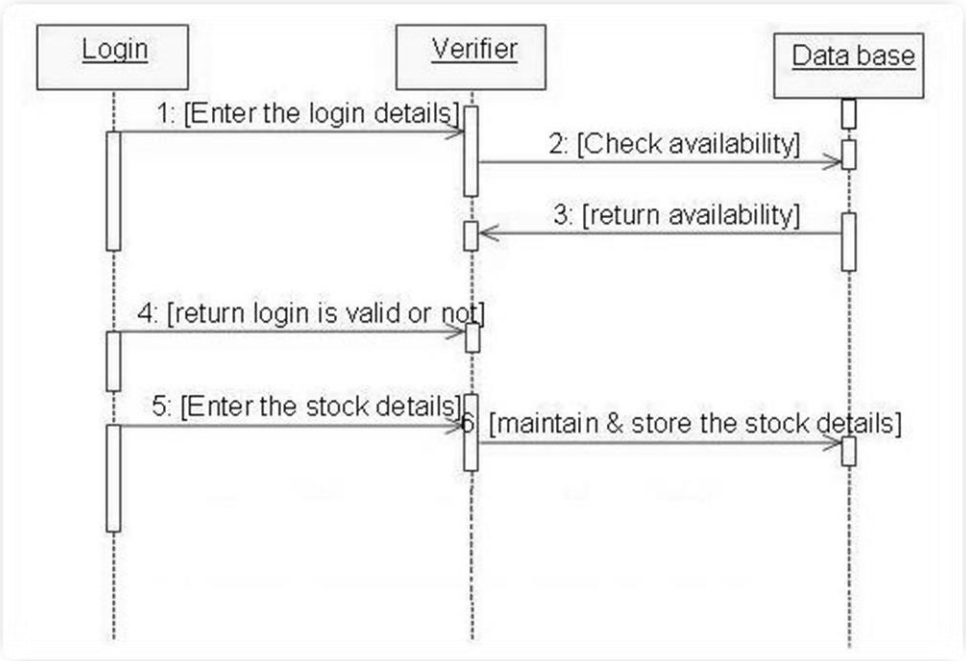
Class diagram:



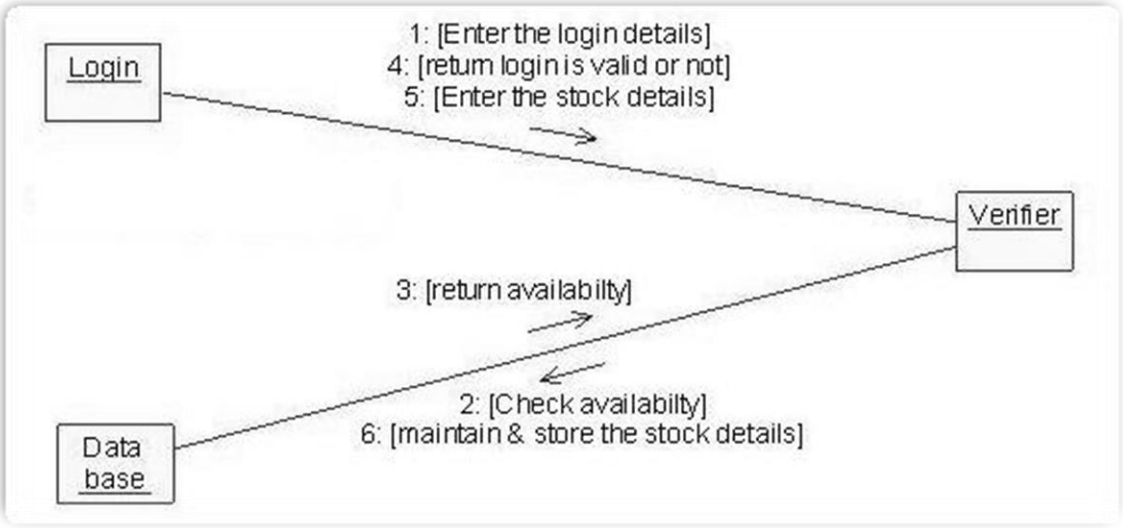
Activity Diagram:



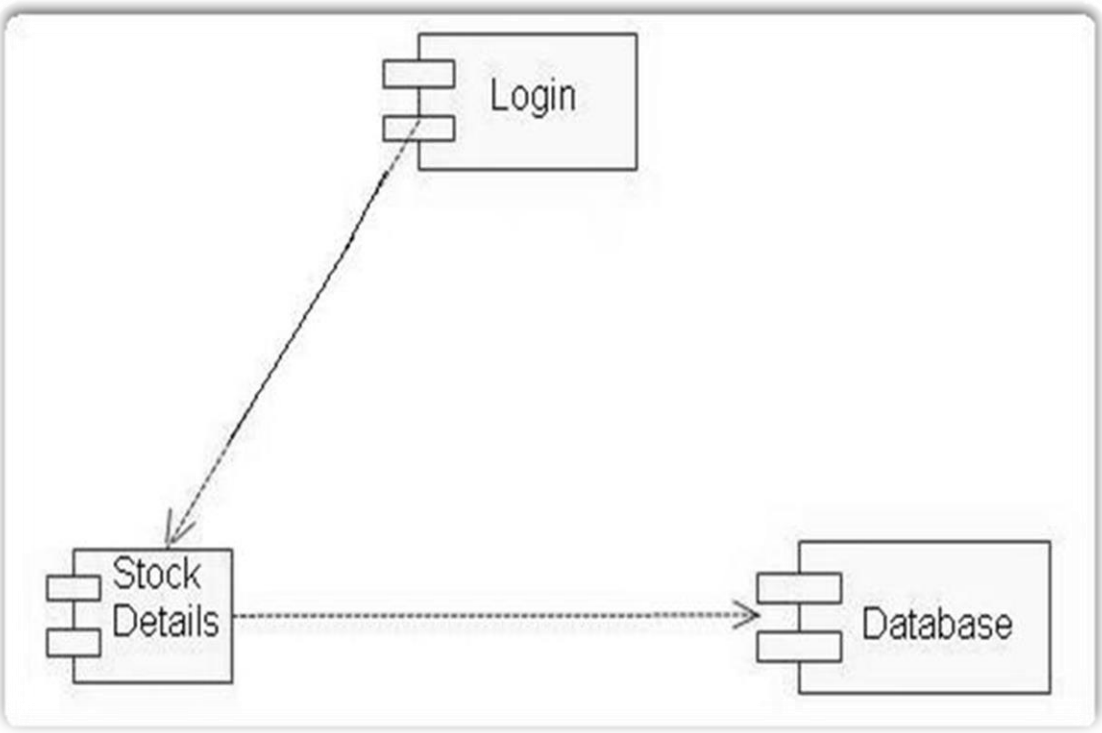
Sequence diagram:



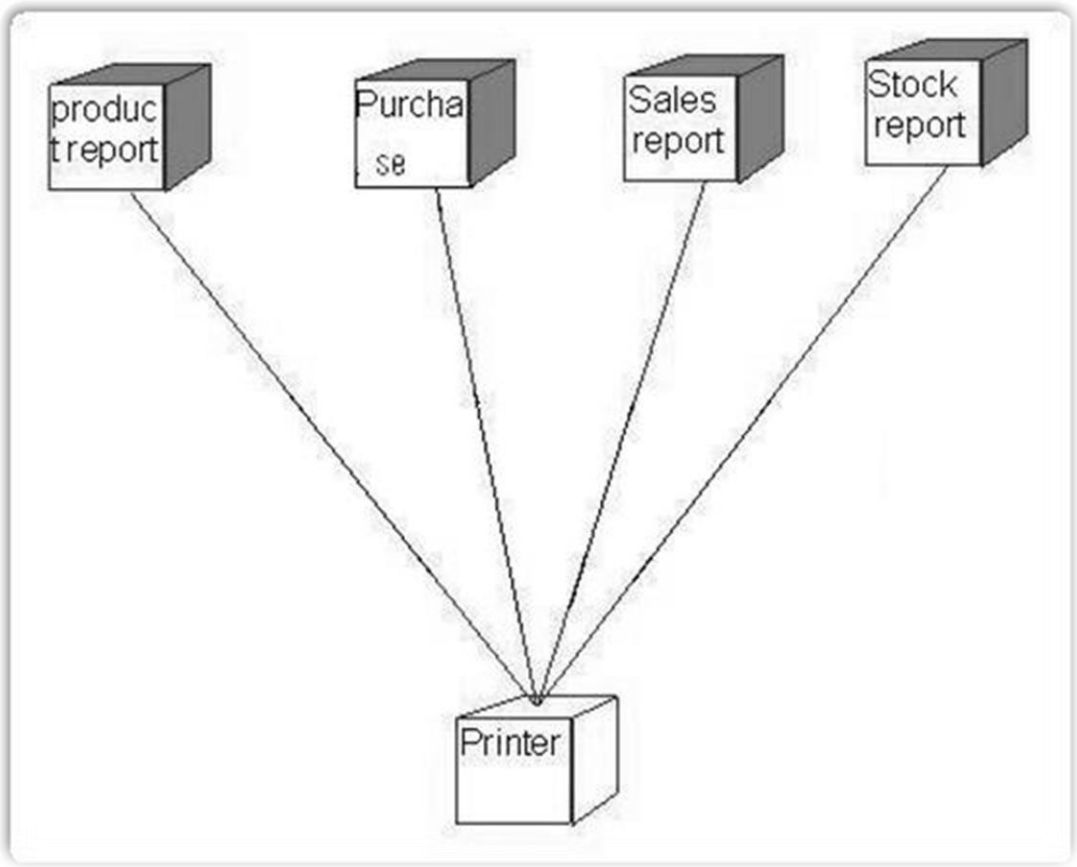
Collaboration diagram:



Component diagram:



Deployment Diagram:



RESULT:

The Stock management system was designed and implemented successfully.

EXPERIMENT 8:

- **Design the test cases for e-commerce application(Flipkart,Amazon)**

Designing test cases for an e-commerce application involves covering various aspects to ensure the application's functionality, usability, security, and performance. Below, I'll provide a set of test cases that you can use as a starting point for testing e-commerce applications like Flipkart or Amazon. Keep in mind that these are general examples, and you may need to customize them based on the specific features and functionalities of the application you are testing.

User Registration and Login:

Test Case 1: Verify that users can register with a valid email address and password.

Test Case 2: Verify that users cannot register with an invalid email format.

Test Case 3: Verify that users can log in with valid credentials.

Test Case 4: Verify that users cannot log in with invalid credentials**.**

Test Case 5: Verify that the "Forgot Password" functionality resets the user's password.

Product Search and Browsing:

Test Case 6: Verify that users can search for products using the search bar.

Test Case 7: Verify that users can filter search results based on various criteria (price, brand, etc.).

Test Case 8: Verify that users can view detailed product information by clicking on a product.

Test Case 9: Verify that users can add products to the shopping cart.

Shopping Cart:

Test Case 10: Verify that the shopping cart reflects the correct quantity and price of added items.

Test Case 11: Verify that users can update the quantity of items in the shopping cart.

Test Case 12: Verify that users can remove items from the shopping cart.

Test Case 13: Verify that the total price is updated correctly after modifying the cart.

Checkout Process:

Test Case 14: Verify that users can proceed to checkout from the shopping cart.

Test Case 15: Verify that users can enter shipping information.

Test Case 16: Verify that users can select a payment method.

Test Case 17: Verify that users can review and confirm their order before finalizing the purchase.

Order Management:

Test Case 18: Verify that users can view their order history.

Test Case 19: Verify that order details are accurate and include the correct products, prices, and shipping information.

Test Case 20: Verify that users receive a confirmation email after completing a purchase.

Security:

Test Case 21: Verify that the application protects against SQL injection attacks.

Test Case 22: Verify that sensitive information (e.g., passwords) is stored securely.

Test Case 23: Verify that the application uses encryption for sensitive data during transmission.

Performance:

Test Case 24: Verify that the application loads within an acceptable time frame.

Test Case 25: Verify that the application can handle a large number of simultaneous users.

Mobile Responsiveness:

Test Case 26: Verify that the application is responsive on various devices (mobile phones, tablets).

User Feedback:

Test Case 27: Verify that users receive appropriate error messages for invalid inputs.

Test Case 28: Verify that users receive confirmation messages for successful actions (e.g., successful order placement).

Cross-Browser Compatibility:

Test Case 29: Verify that the application functions correctly on different web browsers (Chrome, Firefox, Safari, Edge).

These test cases cover a range of functionalities, but the actual test suite will depend on the specific features of the e-commerce application you are testing. Additionally, it's important to consider both positive and negative test scenarios to ensure thorough testing.

EXPERIMENT 9:

- **Design the test cases for a mobile application (consider any example from Appstore)**

Designing test cases for a mobile application involves covering various aspects to ensure the application's functionality, usability, and performance. Below, I'll provide a set of test cases that you can use as a starting point for testing a mobile application. Keep in mind that these are general examples, and you may need to customize them based on the specific features and functionalities of the application you are testing.

Installation and Launch:

Test Case 1: Verify that the application installs without errors from the App Store.

Test Case 2: Verify that the application icon is displayed correctly on the device.

Test Case 3: Verify that the application launches without crashing**.**

User Registration and Login:

Test Case 4: Verify that users can register with a valid email address and password.

Test Case 5: Verify that users cannot register with an invalid email format.

Test Case 6: Verify that users can log in with valid credentials.

Test Case 7: Verify that users cannot log in with invalid credentials.

Test Case 8: Verify that the "Forgot Password" functionality resets the user's password.

Navigation:

Test Case 9: Verify that users can navigate between different sections of the app using the navigation menu.

Test Case 10: Verify that all navigation links and buttons work as expected.

Test Case 11: Verify that the back button functions correctly.

Functionality:

Test Case 12: Verify that all core functionalities of the app work as expected (e.g., if it's a social media app, test posting, liking, and commenting).

Test Case 13: Verify that any interactive elements (buttons, sliders, etc.) respond appropriately to user input.

Test Case 14: Verify that any forms or input fields validate user input correctly.

Offline Functionality:

Test Case 15: Verify that the app provides appropriate feedback when the device is offline.

Test Case 16: Verify that users can still access certain features or content when offline (if applicable).

Push Notifications:

Test Case 17: Verify that users receive push notifications when relevant.

Test Case 18: Verify that users can customize their notification preferences in the app settings.

Device Compatibility:

Test Case 19: Verify that the app is compatible with different devices (phones and tablets) and screen sizes.

Test Case 20: Verify that the app supports both landscape and portrait orientations.

Performance:

Test Case 21: Verify that the app responds quickly to user interactions.

Test Case 22: Verify that the app does not drain the device's battery excessively.

Security:

Test Case 23: Verify that the application protects against data breaches and unauthorized access.

Test Case 24: Verify that sensitive information (e.g., user data) is stored securely.

Updates and App Store Compatibility:

Test Case 25: Verify that the app functions correctly after updating to the latest version.

Test Case 26: Verify that the app complies with App Store guidelines and policies.

User Interface (UI) and User Experience (UX):

Test Case 27: Verify that the UI is intuitive and user-friendly.

Test Case 28: Verify that the app's design adheres to platform-specific guidelines (iOS Human Interface Guidelines for iOS apps, Material Design for Android apps).

Accessibility:

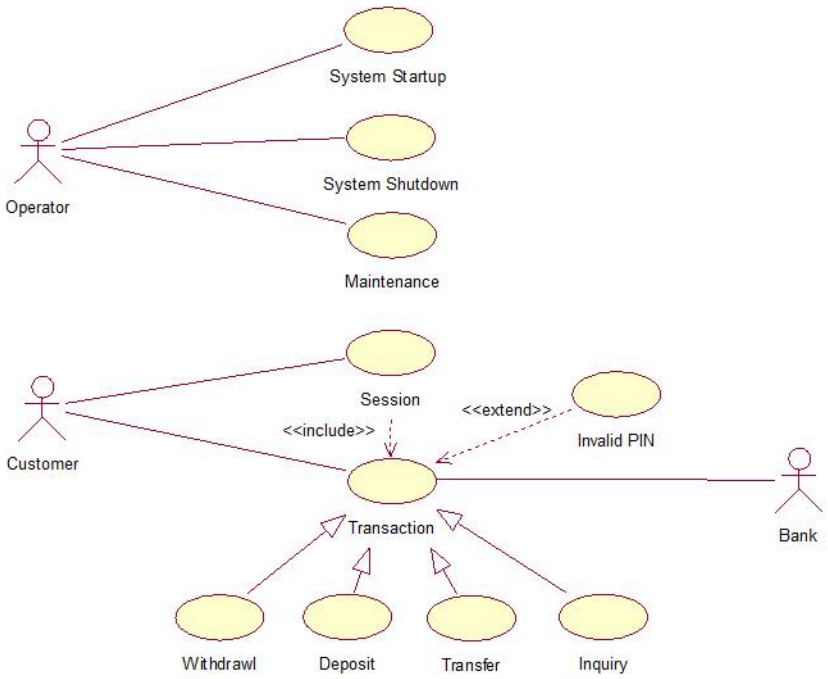
Test Case 29: Verify that the app is accessible to users with disabilities, including support for screen readers and other accessibility features.

These test cases cover a variety of aspects of mobile application testing. As with any testing effort, it's essential to consider the specific features and functionalities of the mobile application you are testing and tailor the test cases accordingly.

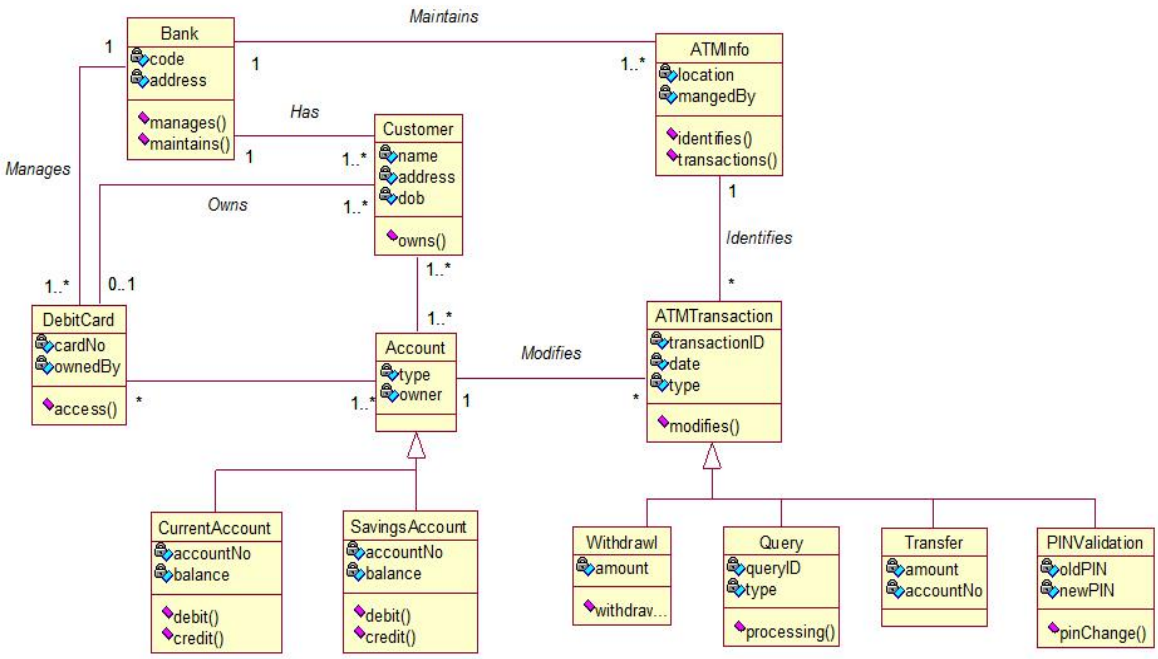
EXPERIMENT 10:

- **Design and Implement ATM system through UML Diagrams.**

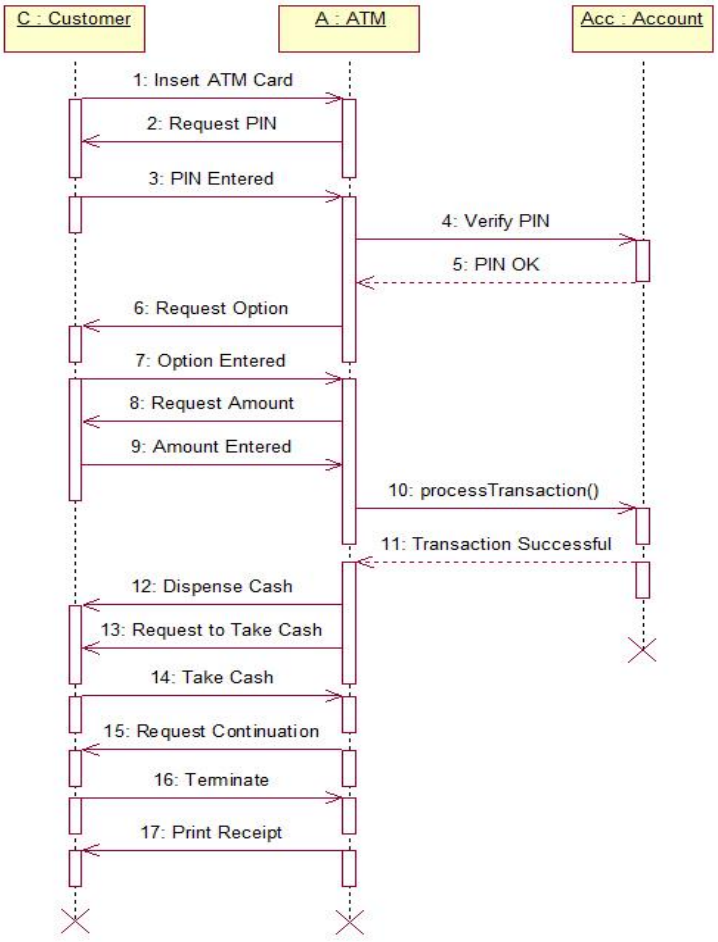
USE CASE DIAGRAM:



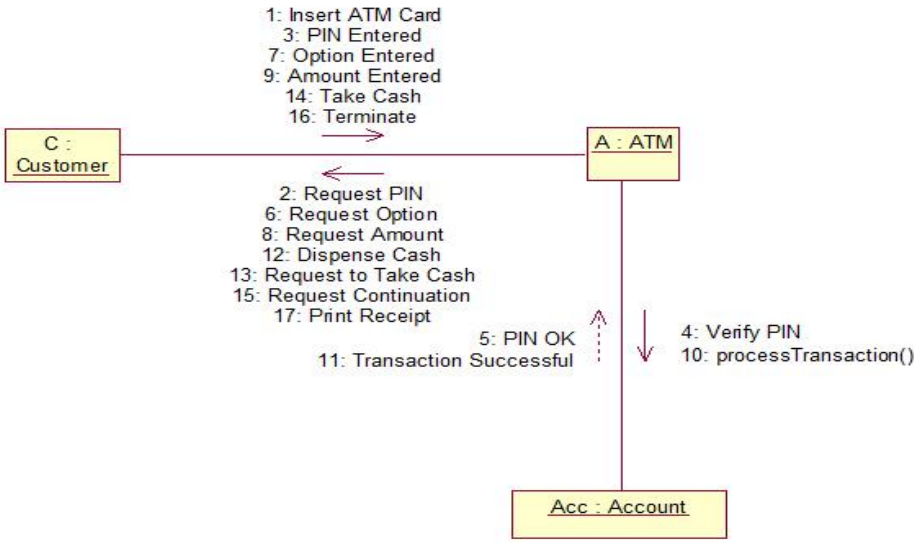
CLASS DIAGRAM:



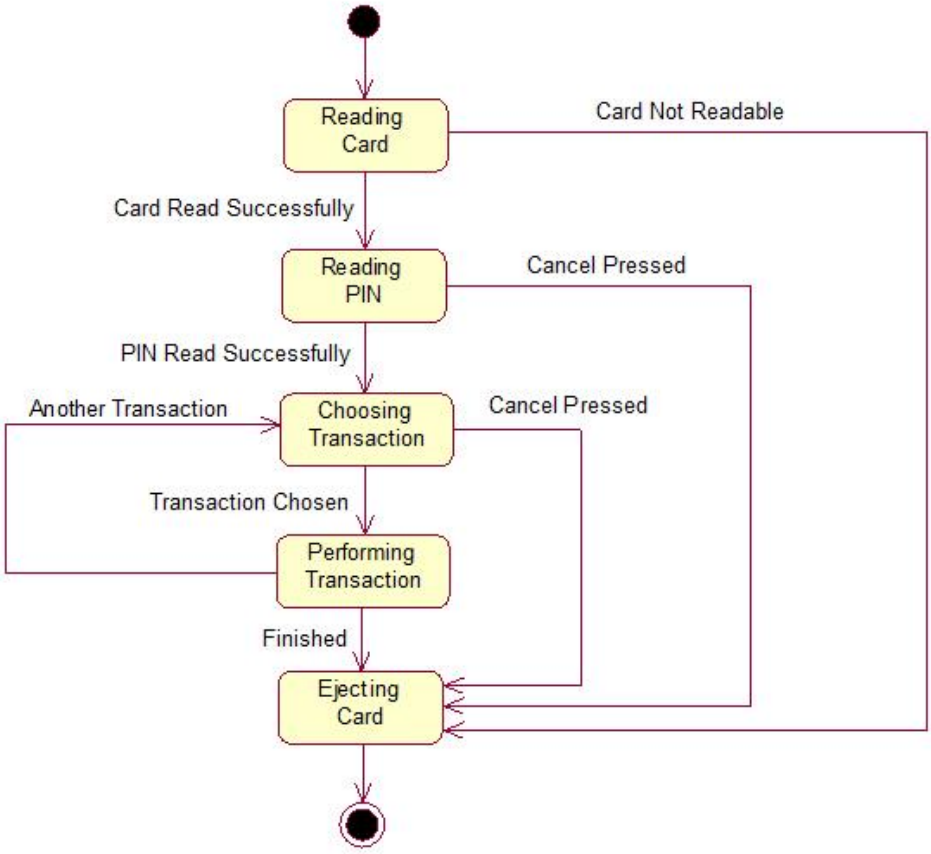
SEQUENCE DIAGRAM:



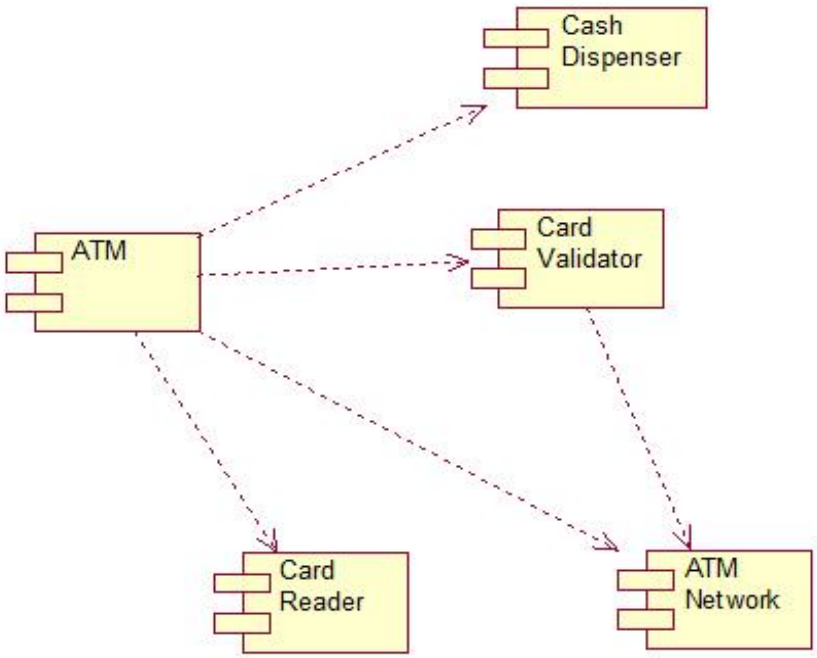
COLLABORATION DAGRAM:



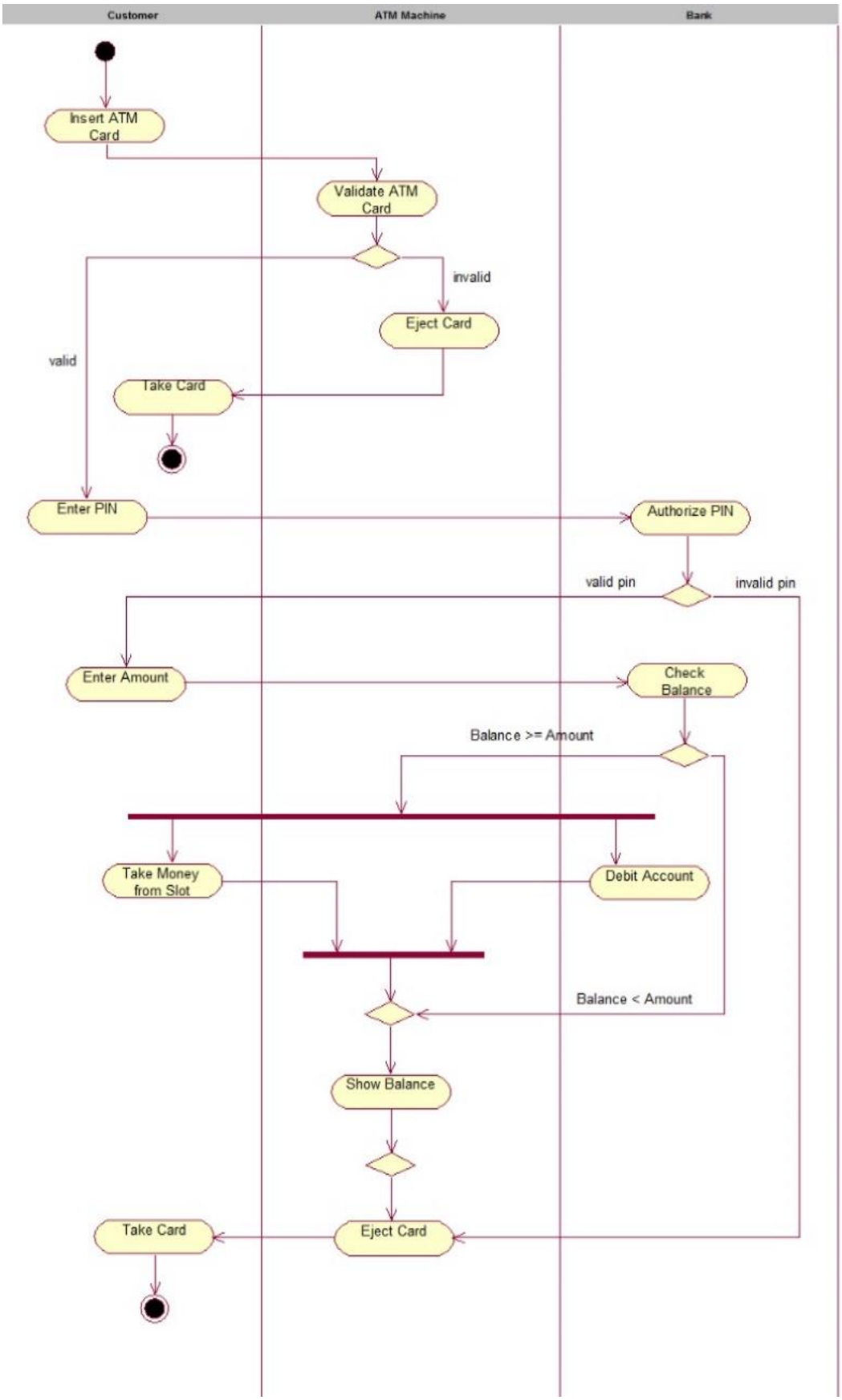
STATECHART DIAGRAM:



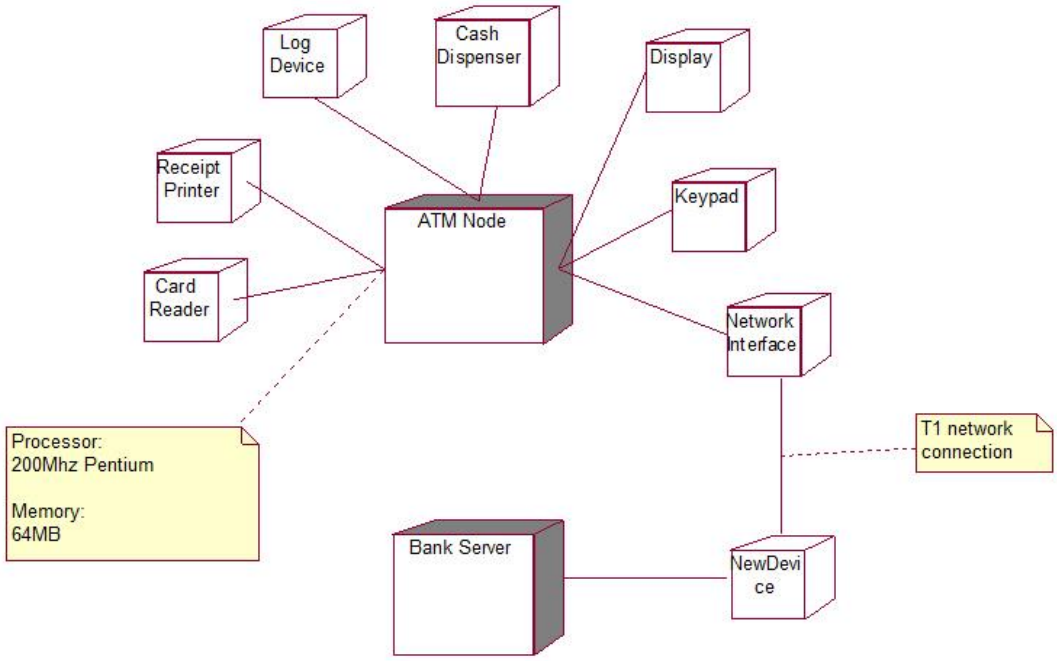
COMPONENT DIAGRAM:



ACTIVITY DIAGRAM:



DEPLOYMENT DIAGRAM:



RESULT:

The Stock management system was designed and implemented successfully.