INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
SONEPAT

Software Engineering Lab

(CSC 507)

Submitted To:-

Ms. Sunita Sharma

Submitted By:-

Dipankar Yadav
12111070
Branch: CSE
Semester: V
Session: 2021-25

Index

Course Management System

Aim: The aim of the Course Management System is to provide an efficient and user-friendly platform for educational institutions to manage their courses, student enrollments, and resources.

User stories and Requirements:

I. Student: –

- As a student, I want to be able to browse and enroll in courses.
- As a student, I want to access course materials and receive updates from my instructors.
- As a student, I want to track my progress and view my grades.

II. Teacher: -

- As a teacher, I want to be able to manage course materials, assignments, and grades for my students.
- As a teacher, I want to communicate with my students and provide feedback on their assignments.
- As a teacher, I want to schedule and manage class sessions, including virtual meetings.

III. Administrator: -

- As an administrator, I want to be able to create, update, and delete courses.
- As an administrator, I want to generate reports on course enrollment and student progress.
- As an administrator, I want to manage user accounts, roles, and permissions.

Requirements:

Functional Requirements:

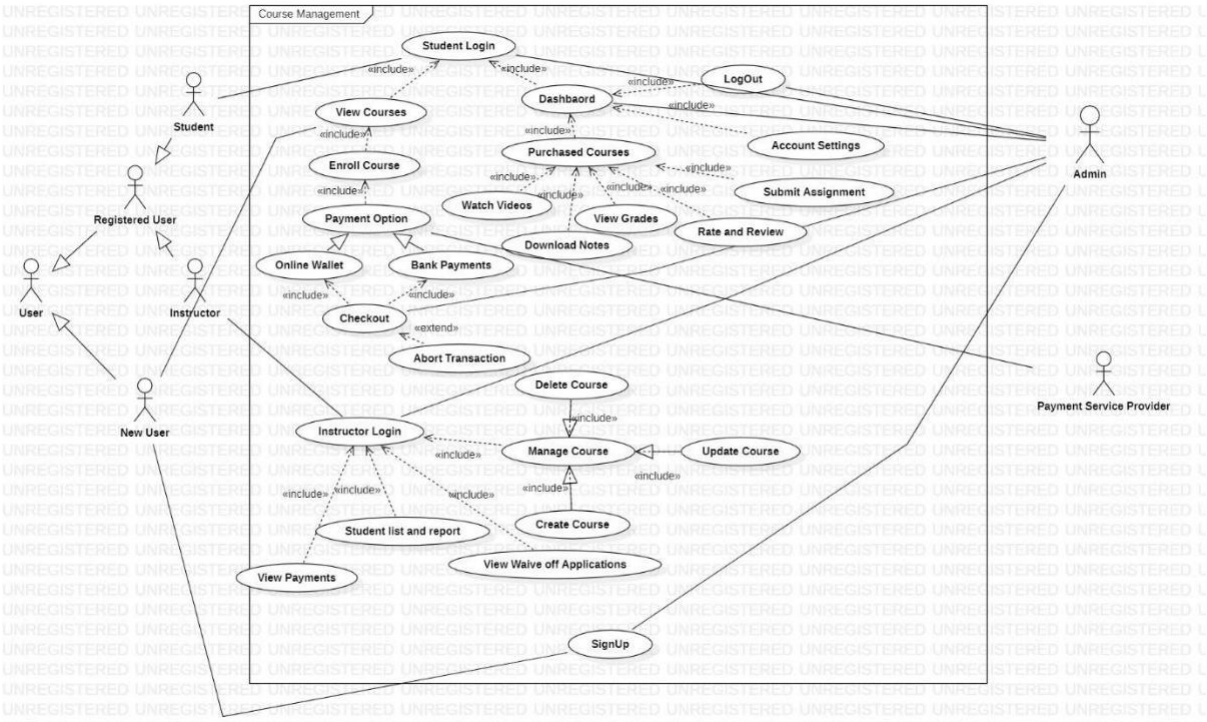
- **User Authentication and Authorization:** Users should be able to register, log in, and have different roles (admin, teacher, student) with appropriate permissions.
- **Course Management:** Create, update, and delete courses. Assign instructors to courses.
- **User Profile Management:** Users should be able to update their profiles, including contact information.
- **Course Materials:** Instructors should be able to upload and manage course materials, including documents, videos, and links.
- **Enrollments:** Students should be able to enroll in courses, and instructors should be able to view and manage course rosters.
- **Communication:** Implement messaging and notification features for communication between students and instructors.
- **Assignments and Grading:** Instructors should be able to create assignments, and students should submit assignments. Instructors can grade and provide feedback.
- **Scheduling:** Allow the scheduling of class sessions, including in-person and virtual meetings, with calendar integration.
- **Reporting:** Provide reporting capabilities for administrators to monitor course enrollment, student progress, and system usage.

Non-Functional Requirements:

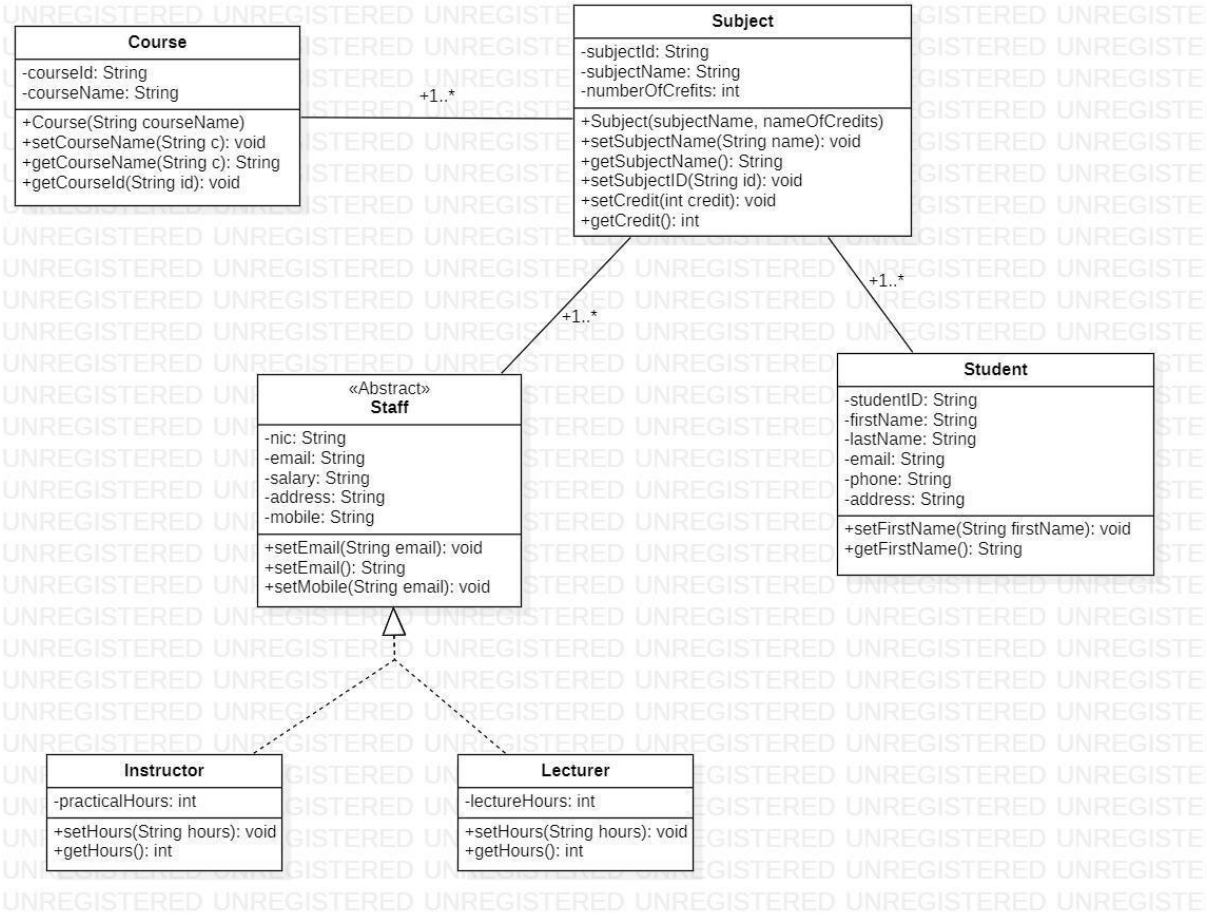
- **Usability:** The system should be user-friendly, with an intuitive interface and responsive design for various devices.
- **Performance:** The system should be able to handle a large number of users and course data efficiently, with minimal latency.
- **Security:** Implement data security measures to protect user data and ensure compliance with data privacy regulations.
- **Integration:** Support integration with external systems, such as a Learning Management System (LMS) or student information system.
- **Scalability:** The system should be scalable to accommodate growth in the number of users and courses.
- **Reliability:** Ensure system uptime and availability with minimal downtime for maintenance.

- **Accessibility:** The system should be accessible to users with disabilities, following accessibility standards.
- **Data Backup and Recovery:** Regularly back up data and have a plan for data recovery in case of system failures.
- **Compliance:** Comply with relevant regulations and standards, especially regarding data protection and privacy.

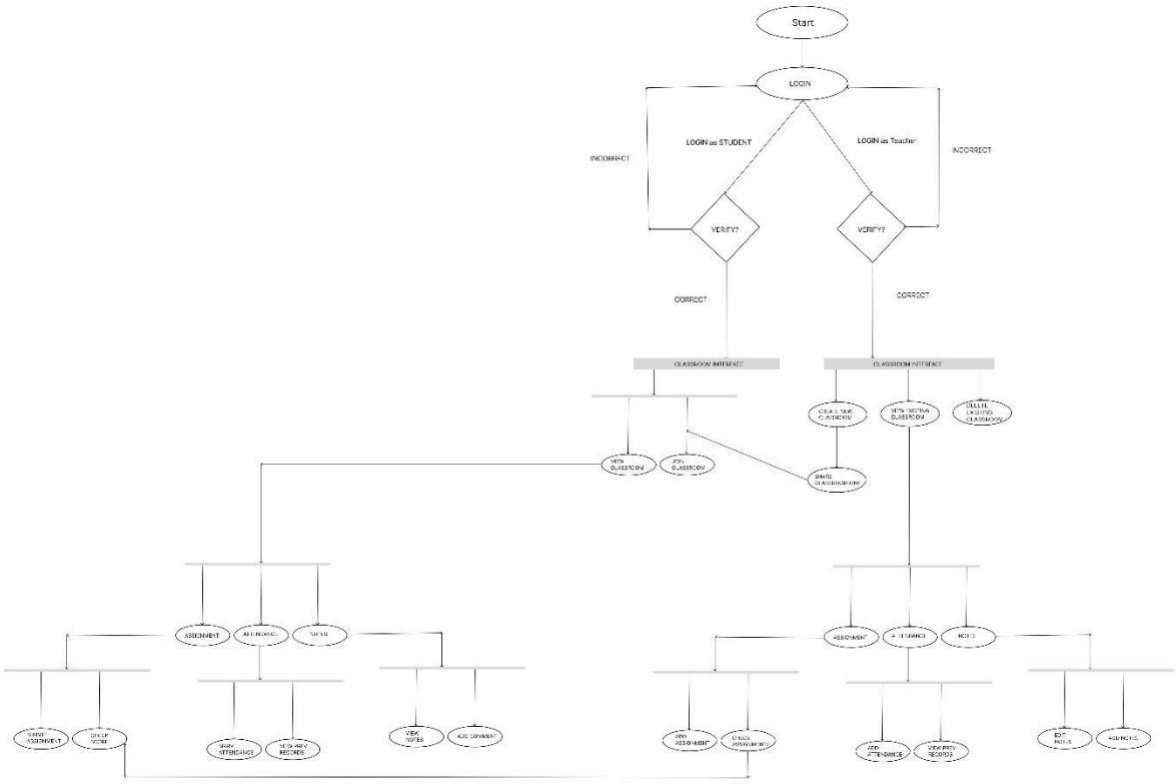
User Case Diagram: -



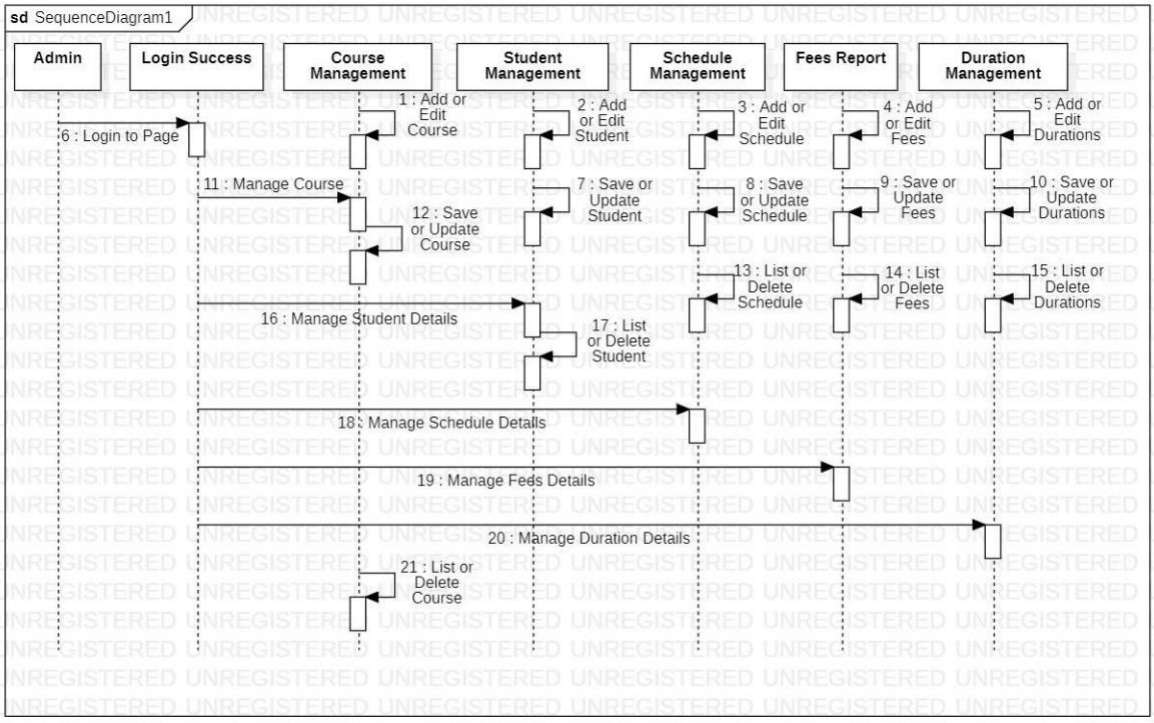
Class Diagram: -



Activity Diagram: -



Sequence Diagram: -



Easy Leave Application

Aim: This project is aimed at developing a web-based Leave Management Tool, which is of importance to either an organization or a college.

User stories and Requirements:

I. Employee :–

As an employee, I want to be able to submit a leave request by specifying the dates and the reason for taking leave so that I can efficiently manage my time off.

Requirements gathered from this user story include:

1. **Login**
    - Employees should be able to login securely to their accounts

2. **Submission of leave request**
    - Employees can submit leave request
    - Employees should be able to enter the dates on which they want to take leave.
    - Employees should be able to provide the reason regarding leave.

3. **Checking status of leave request**
    - Employees should be able to check status of their request
    - Notifications regarding the confirmation of leave request should be sent to employees showing the status of leave like approved, rejected, or pending.

4. **Leave Balances and History**
    - Employees should have access to a dashboard where they can view their leave balances
    - Employees should also have access to their previously taken leaves and their leave requests
    - Employees can also access their attendance record.

5. **User Profile Management**
    - Employees can edit their personal details on the application.

II. Manager or Admin :–

As a manager, I want to efficiently review and manage leave requests of my team members to ensure proper workflow within the organization.

Requirements gathered from this user story include:

1. **Login**
    - Managers should be able to securely login to their accounts.

2. **Leave Request Review**
    - Managers should be able to view detailed information about each pending leave request, including the employee's name, reason for leave, and requested dates.
    - Managers should have the ability to approve or reject each leave request along with an optional comment or remark.

3. **Leave Request History**
    - Managers should be able to access a history log of all leave requests for their managed employees.
    - The leave request history should include details such as leave reason, requested dates, and approval status.

4. **Notifications**
    - Managers should receive notifications when an employee submits a leave request.

5. **Uploading Attendance Records**
    - Managers can additionally upload attendance records for the employees on the portal.

6. **User Profile Management**
    - Managers should have the option to update their personal information, including contact details, within their user profiles.

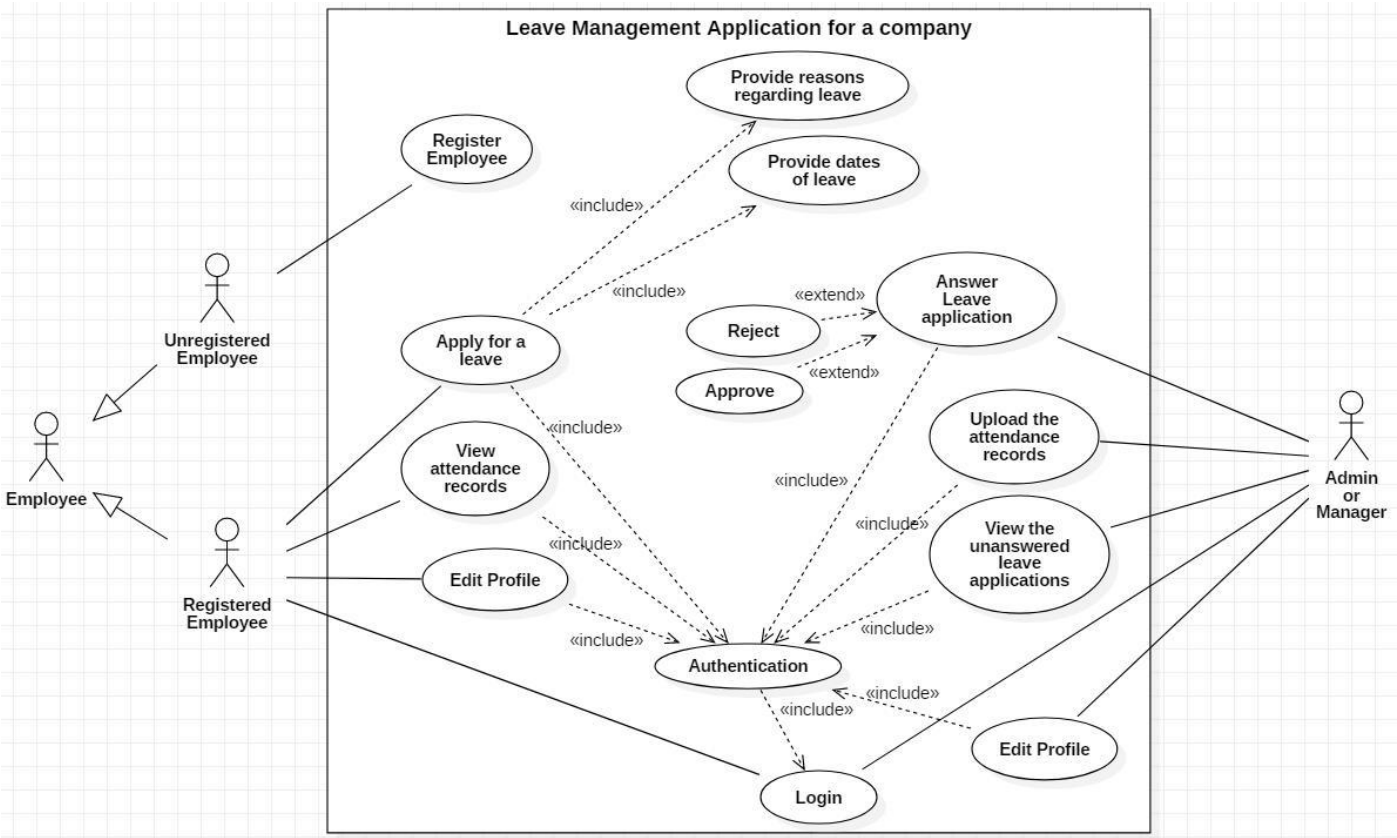
Functional Requirements:

- Registered employees can log in securely to access the system.
- Employees can submit leave requests by specifying the reason for leave, dates, and a brief reason
- Employees can view their leave balances to ensure accurate leave planning.
- Employees receive notifications regarding the status of their leave requests (approved, pending, or rejected).
- Employees can update their personal information, such as contact details and emergency contacts.
- Managers can log in securely to access the system.
- Managers can view the pending or unanswered leave requests on their dashboard.
- Managers can approve or reject leave requests from their team members.
- Managers can optionally add comments or remarks when approving or rejecting the leave requests.

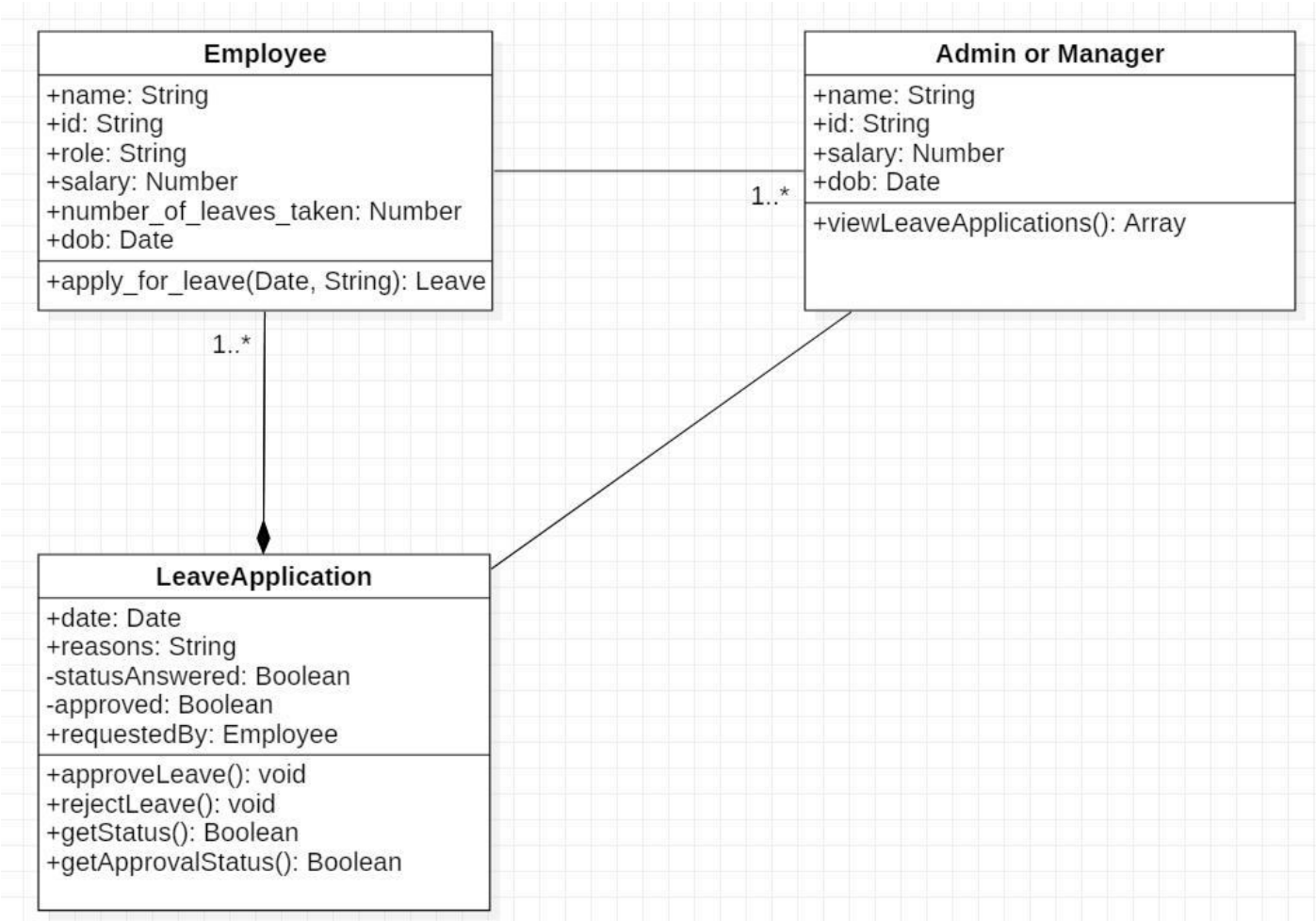
Non-functional Requirements:

- The system should be scalable to accommodate the organization's growth.
- The system should be available 24/7 with minimal downtime for maintenance.
- The system should comply with relevant governmental as well as organizational laws and regulations.
- The application should work properly on popular web browsers (e.g., Chrome, Firefox, Safari).
- The system should provide clear error messages and gracefully handle exceptions.
- The system should be compatible on various devices and also responsive to all the device screen sizes.

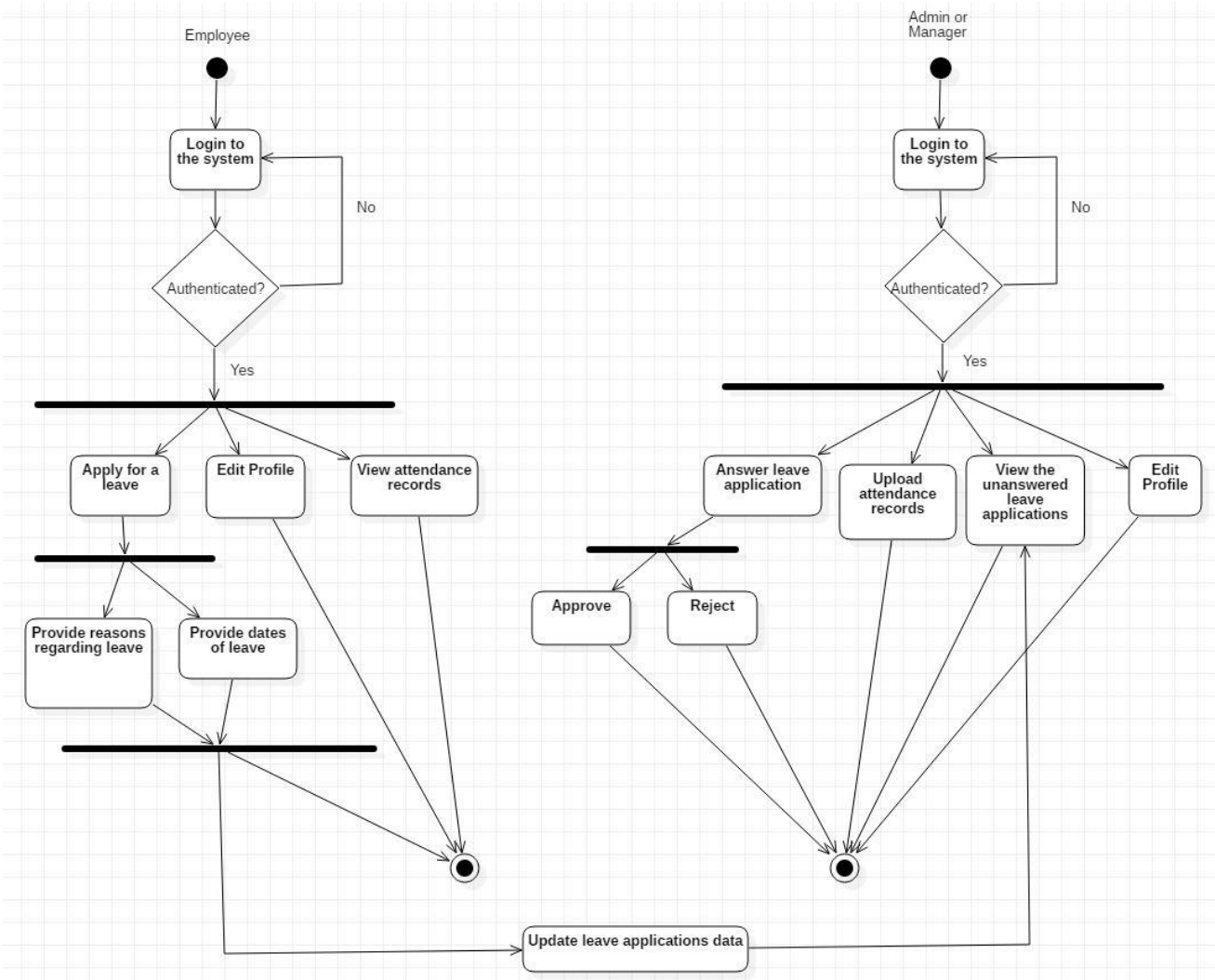
Use Case Diagram:



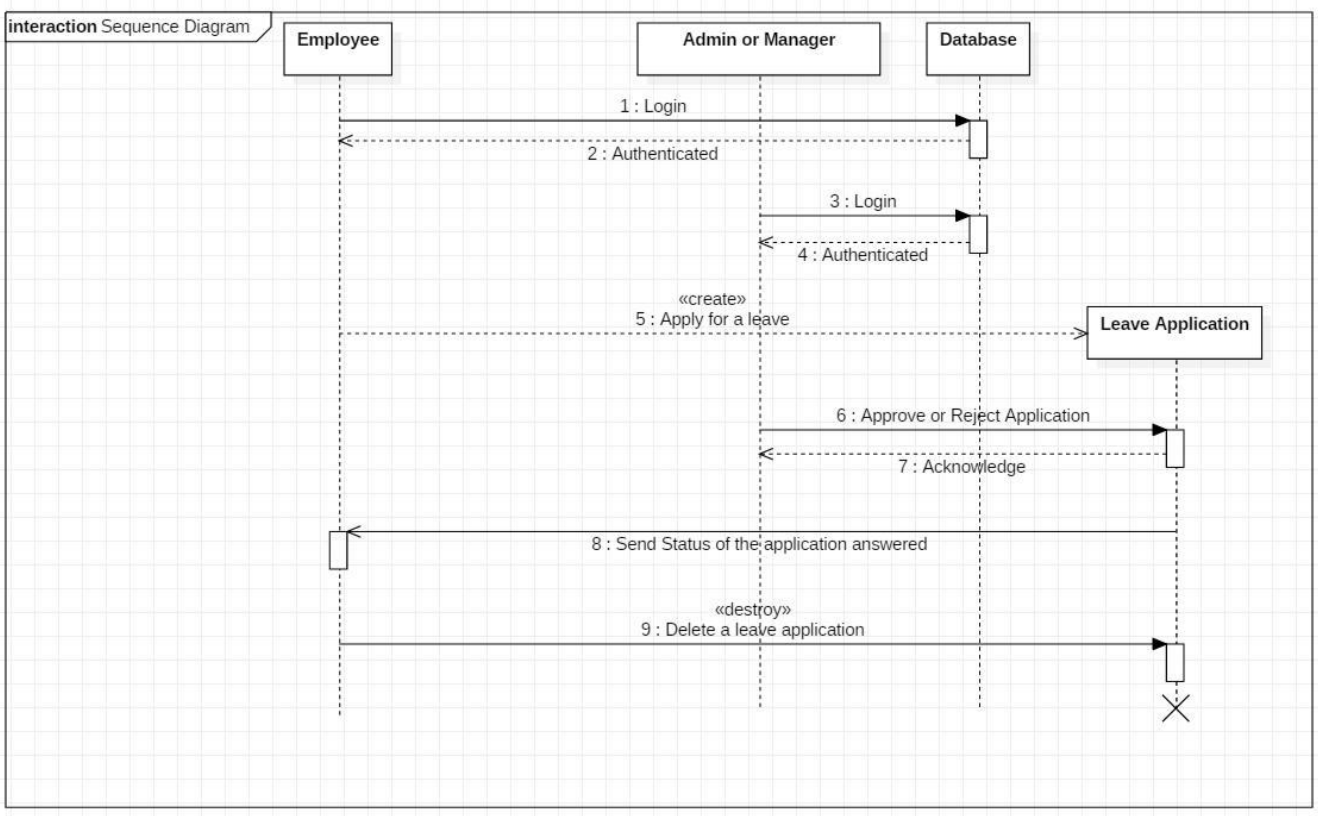
Class Diagram:



Activity Diagram:



Sequence Diagram:



E-bidding System

Aim:

The aim of the e-bidding system is to provide a platform for buyers and sellers to conduct online auctions efficiently. This system facilitates the buying and selling of various products through a competitive bidding process. The e-bidding system aims to offer a user-friendly interface, ensure secure transactions, and create a fair and transparent marketplace.

User Stories and Requirements:

I. Seller:-

As a seller, I want to list my products for auction, set reserve prices, receive bids, and manage my listings efficiently.

The requirements for these are as follows -

1. **Product Listing:**
   - Sellers should be able to create product listings, product descriptions, and initial bid prices.
   - Sellers can specify a reserve price to ensure that the product is not sold below a certain threshold.
2. **Auction Management:**
   - Sellers should be able to start and stop auctions for their listed products.
   - Sellers should have the option to extend the auction duration if desired.

II. Buyer:

As a buyer, I want to browse available products, place bids, and track the status of my bids.

The requirements for these are as follows -

1. **Product Browsing:**
   - Buyers should be able to browse and search for products listed in various categories.
   - Product listings should include all the details of the product.
2. **Bidding on Products:**
   - Buyers should be able to place bids on products they are interested in.
   - The system should update the current highest bid in real-time for each product.
3. **Bid Tracking:**
   - Buyers should have access to a dashboard where they can view all products, they have placed bids on.
   - The dashboard should display bid status, such as winning or losing.

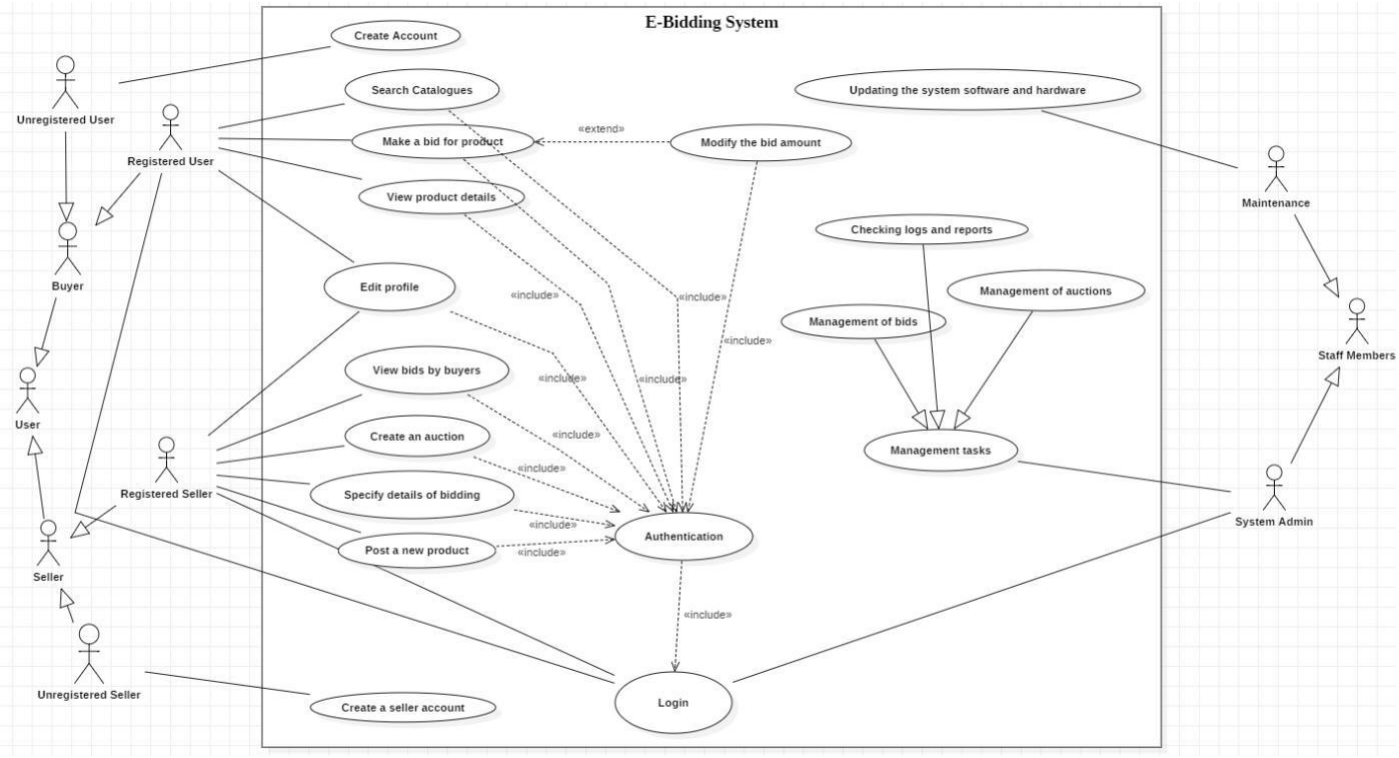
Functional Requirements:

- The system should provide user registration, login, and password recovery functionalities.
- User roles (seller and buyer) should be asked at sign up time.
- Sellers should be able to create, update, and delete product listings.
- Buyers should have the option to watch products without bidding.
- The system should facilitate the competitive bidding process.
- Bids should be timestamped, and the highest bid should be updated in real-time.
- The system should automatically close auctions when the duration expires.
- Products with bids above the reserve price should be marked as sold.
- The system should send notifications to both sellers and buyers for bid actions, auction closures, and winning bids.

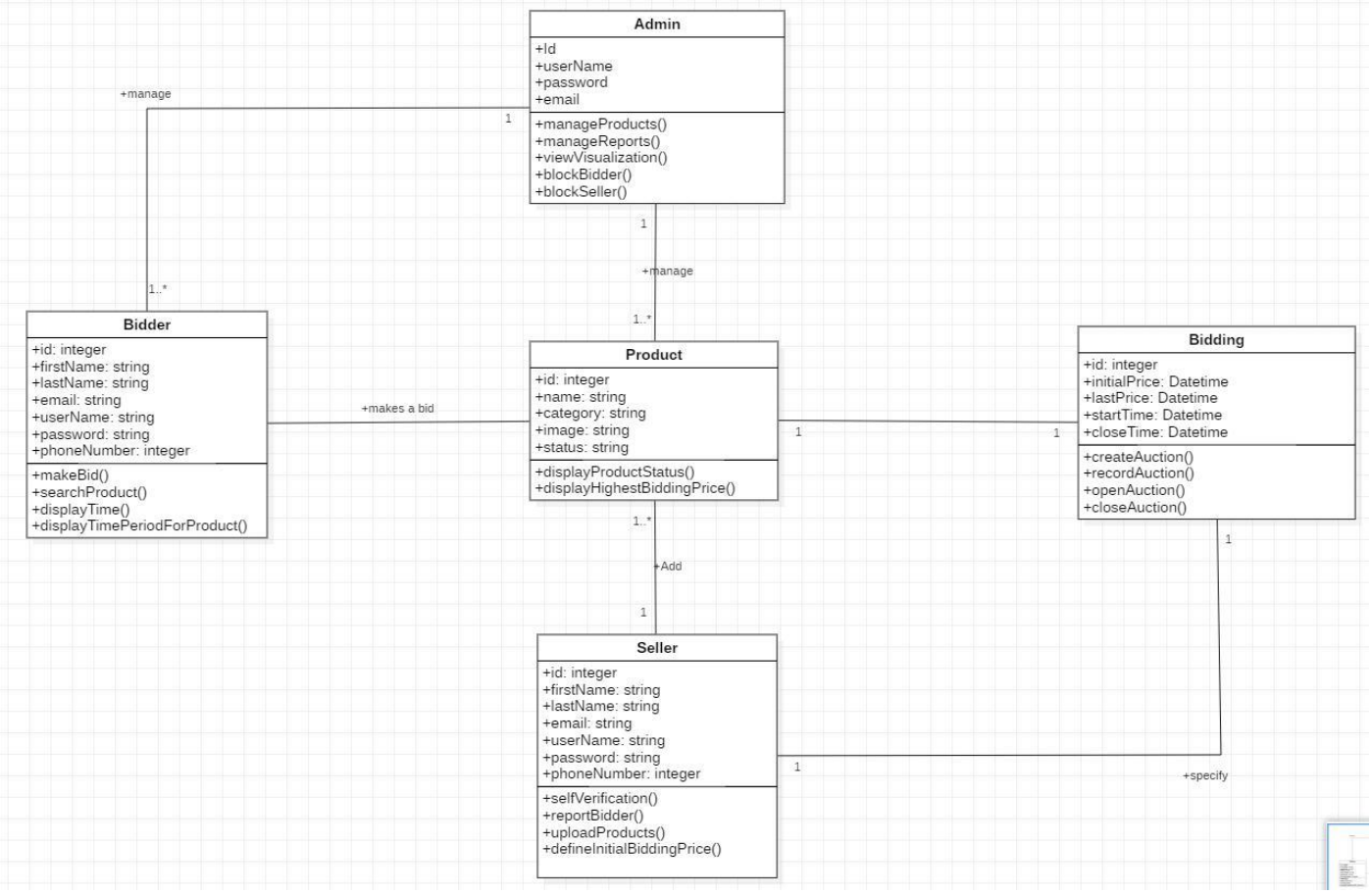
Non-Functional Requirements:

- **Security**: User data and transaction information should be securely stored and transmitted. Payment information should be protected.
- **Performance**: The system should be responsive and able to handle concurrent users during peak times. Load balancing and caching mechanisms should be in place to optimize performance.
- **Scalability**: The system should be scalable to accommodate a growing number of users and listings.
- **Usability**: The user interface should be intuitive and user-friendly for both sellers and buyers.
- **Payment Processing**: The system should integrate with a secure payment gateway to process payments for sold products.

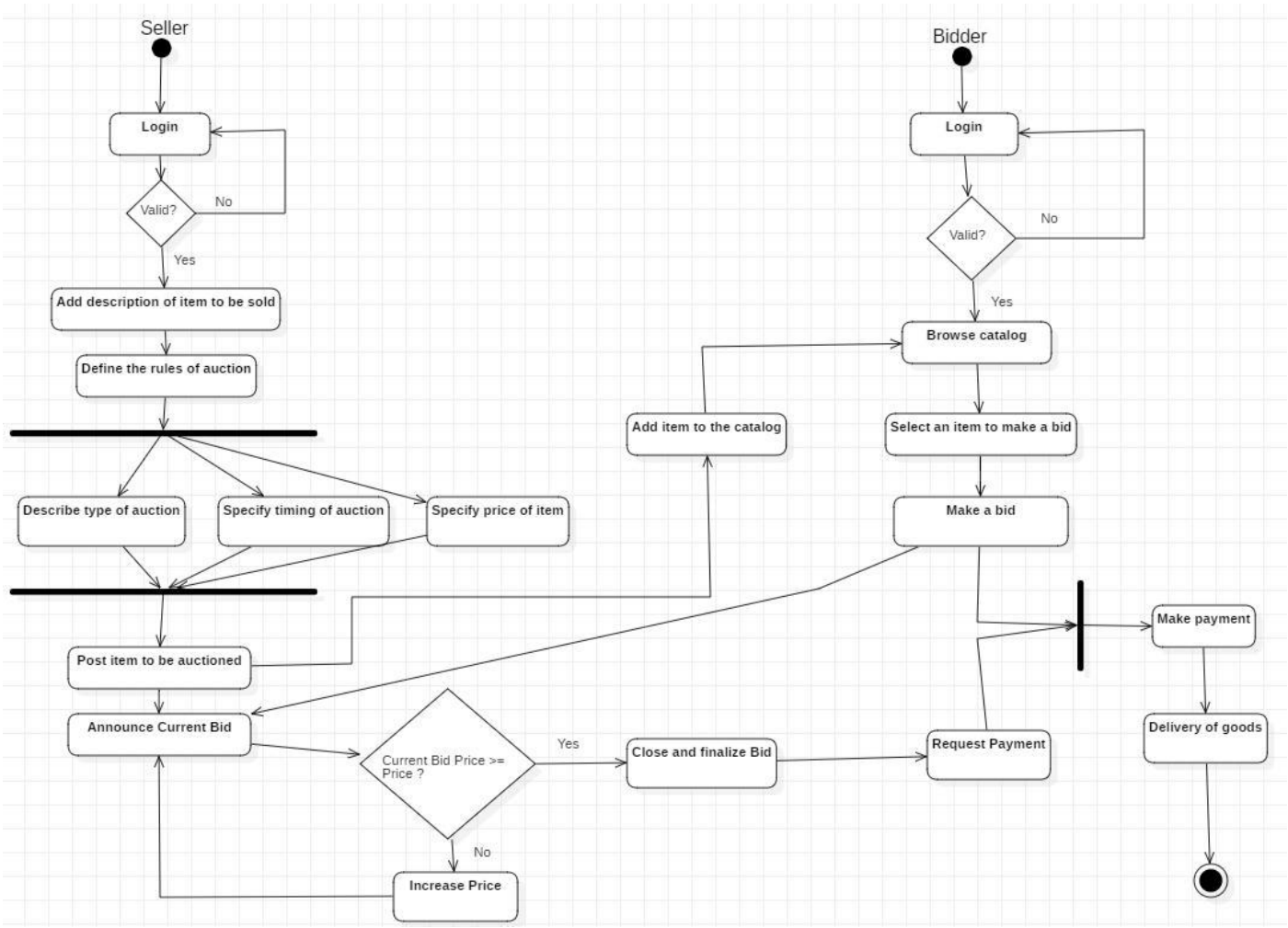
Use Case Diagram:



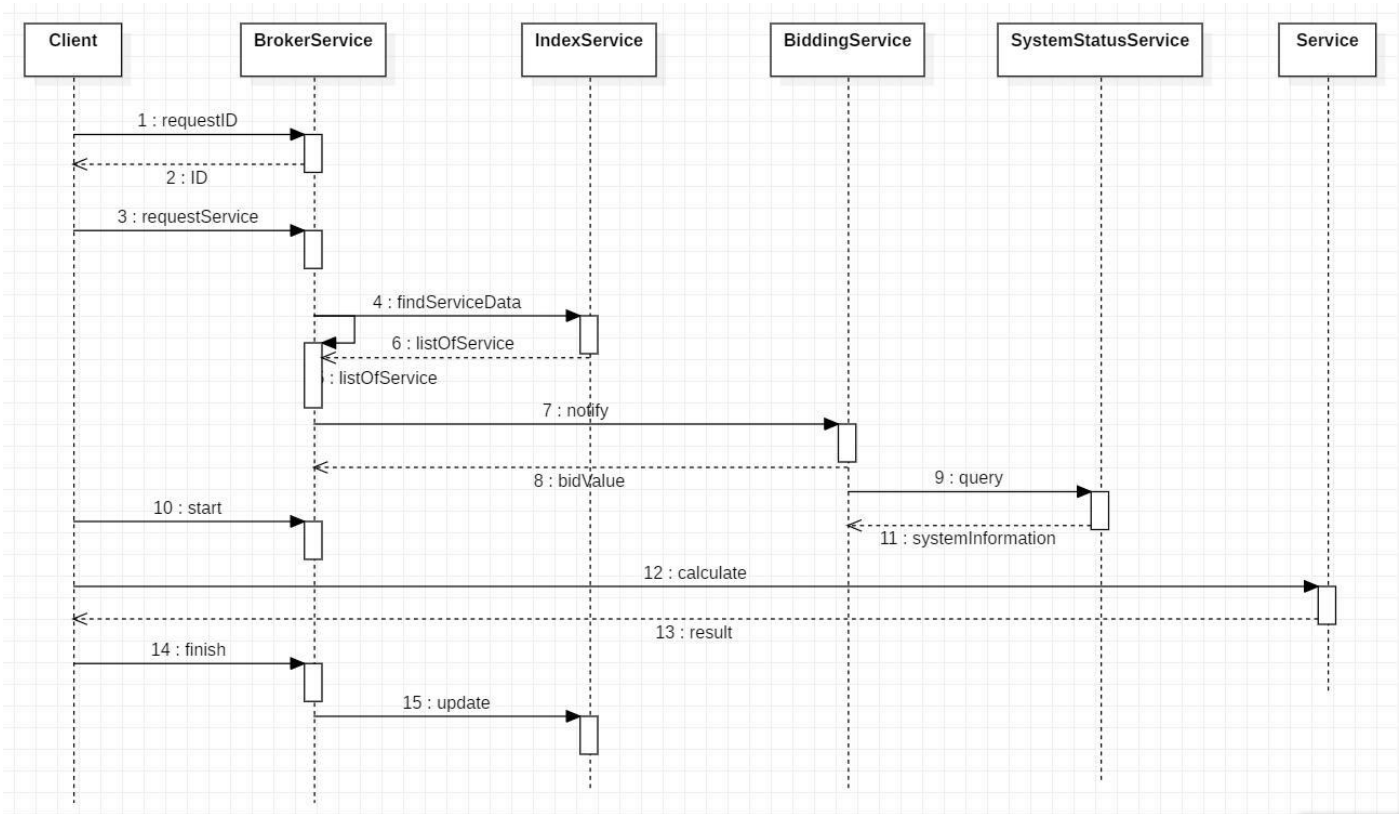
Class Diagram:



Activity Diagram:

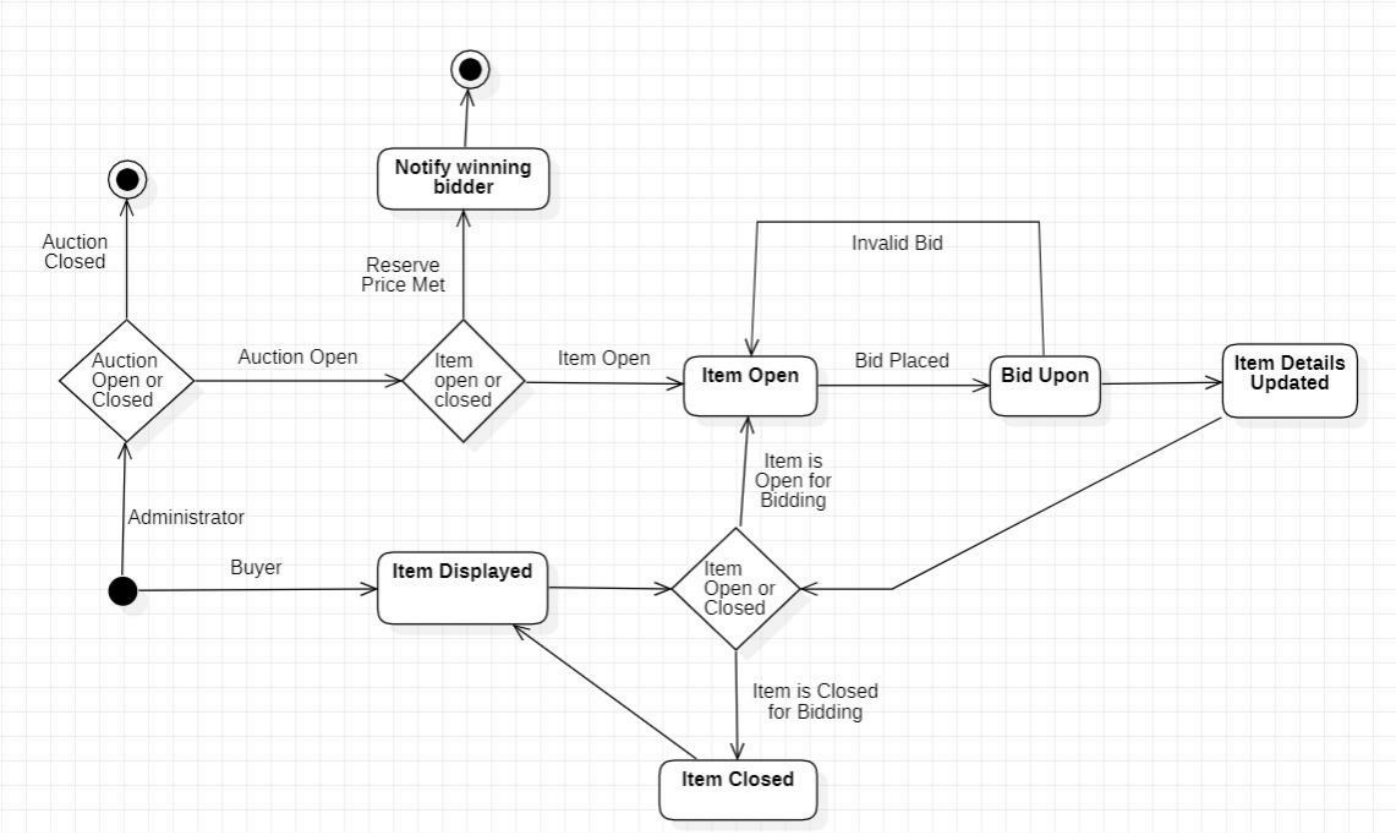


Sequence Diagram:



State Chart Diagram:

bnjm,.hh



Git Commands

1. Git Init:

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith
$ git init
Initialized empty Git repository in C:/Users/DIPANKAR/Desktop/projects/algozeni
h/.git/

DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (master)
$
```

2.Git Clone:

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (master)
$ git clone https://github.com/DIPANKAR-123/Ecomzy.git
Cloning into 'Ecomzy'...
remote: Enumerating objects: 96, done.
remote: Counting objects: 100% (96/96), done.
remote: Compressing objects: 100% (72/72), done.
remote: Total 96 (delta 30), reused 81 (delta 17), pack-reused 0
Receiving objects: 100% (96/96), 504.83 KiB | 1.19 MiB/s, done.
Resolving deltas: 100% (30/30), done.

DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (master)
$
```

3.Git Status

```
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Ecomzy/

nothing added to commit but untracked files present (use "git add" to track)
```

4.Git add .

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (master)
$ git add .

DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (master)
```

5.Git rm -f [file name]:

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (master)
$ git rm -r temp.txt
error: the following file has changes staged in the index:
    temp.txt
(use --cached to keep the file, or -f to force removal)
```

6.Git Commit:

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (master)
$ git commit -m "first commit"
[master (root-commit) 3a70fc5] first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 temp.txt
```

7.Git Branch -M:

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (master)
$ git branch -M main

DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (main)
$
```

8.Git Remote:

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (main)
$ git remote add origin https://github.com/DIPANKAR-123/test.git

DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (main)
$ git remote
origin
```

9.Git Push:

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 215 bytes | 215.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/DIPANKAR-123/test.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (main)
$
```

10.Git Checkout:

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (main)
$ git checkout -b testbranch
Switched to a new branch 'testbranch'
```

11.Git Log:

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (main)
$ git log
commit 3a70fc55865cc5e1f17efb7ca6af7d970dffa6f9 (HEAD -> main, origin/main, tes
branch)
Author: Dipankar <dipankaryadav1234@gmail.com>
Date:   Sat Nov 18 00:59:12 2023 +0530

    first commit
```

12.Git Diff:

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (testbranch)
$ git diff main testbranch
diff --git a/temp2.txt b/temp2.txt
new file mode 100644
index 0000000..e69de29
```

13.Git Merge:

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (main)
$ git merge testbranch
Updating 3a70fc5..c2c2aff
Fast-forward
 temp2.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 temp2.txt
```

14.Git Pull

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (main)
$ git pull
Already up to date.
```

15.Git Rebase:

```
DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (main)
$ git rebase testbranch
Successfully rebased and updated refs/heads/main.

DIPANKAR@LAPTOP-R3HT57DI MINGW64 ~/Desktop/projects/algozenith (main)
$
```