RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CB23332
SOFTWARE ENGINEERING LAB

Laboratory Record Note Book

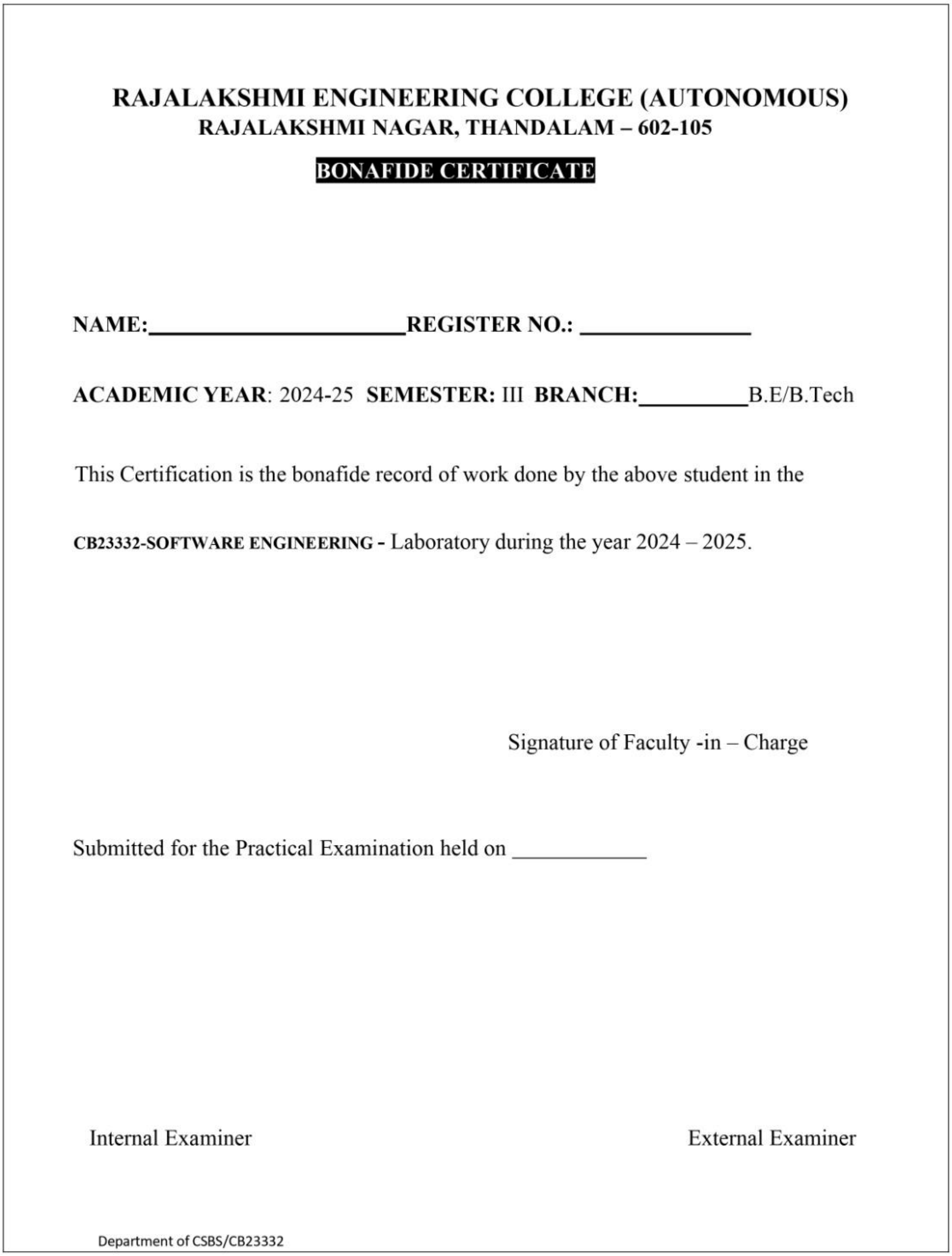
Name : ....

Year / Branch / Section : ....

Register No. : ....

Semester : ....

Academic Year : ....



RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

RAJALAKSHMI NAGAR, THANDALAM – 602-105

BONAFIDE CERTIFICATE

NAME:______________________**REGISTER NO.:** _______________

ACADEMIC YEAR: 2024-25 **SEMESTER:** III **BRANCH:**_________B.E/B.Tech

This Certification is the bonafide record of work done by the above student in the

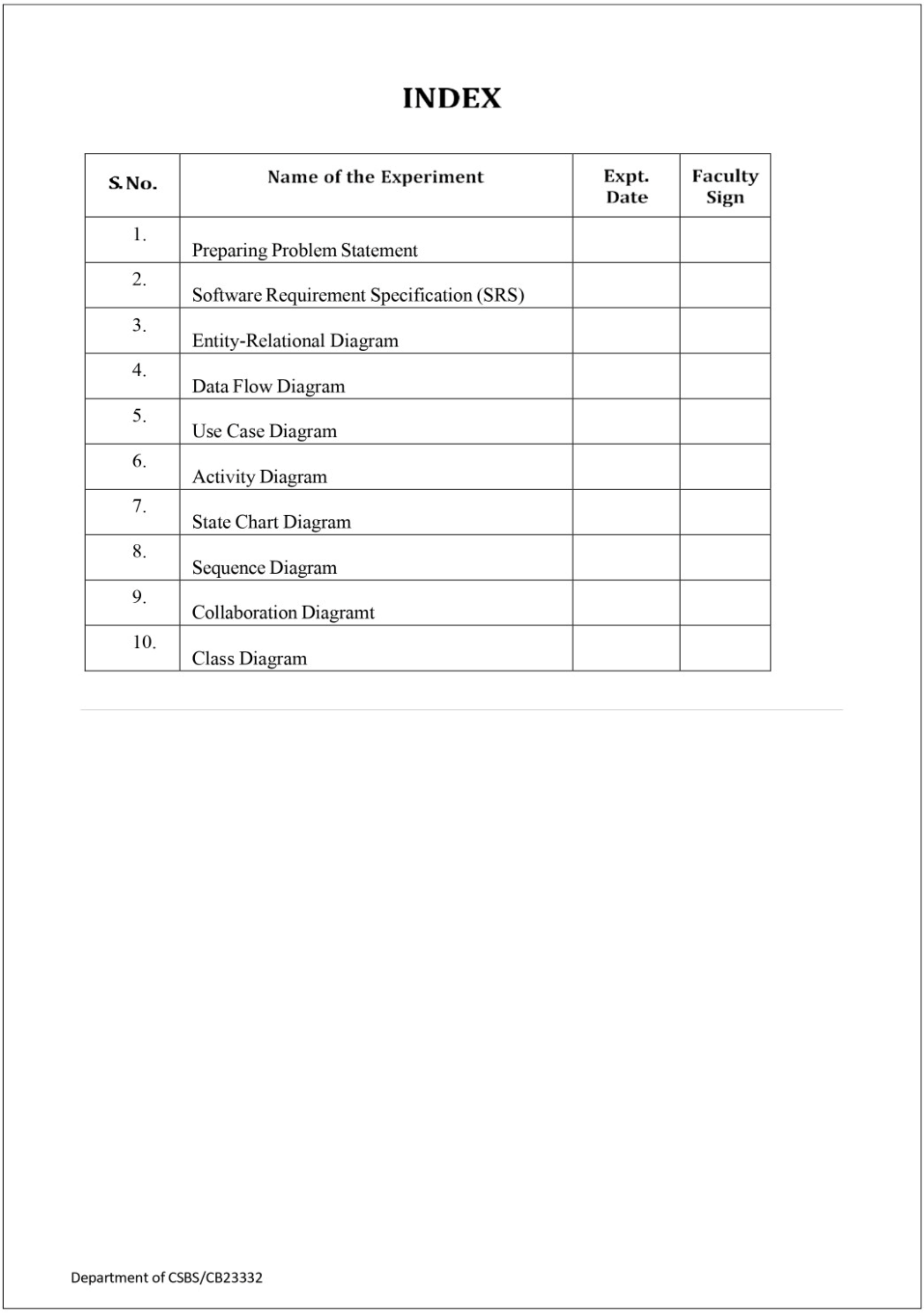
CB23332-SOFTWARE ENGINEERING - Laboratory during the year 2024 – 2025.

Signature of Faculty -in – Charge

Submitted for the Practical Examination held on ____________

Internal Examiner

External Examiner



INDEX

S.No.	Name of the Experiment	Expt. Date	Faculty Sign
1.	Preparing Problem Statement		
2.	Software Requirement Specification (SRS)		
3.	Entity-Relational Diagram		
4.	Data Flow Diagram		
5.	Use Case Diagram		
6.	Activity Diagram		
7.	State Chart Diagram		
8.	Sequence Diagram		
9.	Collaboration Diagramt		
10.	Class Diagram		

EX NO:1	**WRITE THE COMPLETE PROBLEM STATEMENT**
DATE:	

AIM:

To prepare a PROBLEM STATEMENT for the project Fake Review Detection System.

ALGORITHM;

1. **The problem statement is the initial starting point for a project.**
2. **A problem statement describes what needs to be done without describing how.**
3. **It is typically a one-to-three-page document that all stakeholders agree on, describing the project goals at a high level.**
4. **The problem statement is intended for a broad audience and should be written in non-technical terms.**
5. **It helps technical and non-technical personnel communicate effectively by providing a clear description of the problem.**
6. **It does not describe the solution to the problem.**

INPUT:

1. **The input to requirement engineering is the problem statement prepared by the customer.**
2. **It may provide an overview of the existing system along with broad expectations from the new system.**
3. **The first phase of requirements engineering begins with requirements elicitation, i.e., gathering information about requirements.**
4. **Requirements are identified with the help of the customer and existing system processes.**

PROBLEM:

Online reviews significantly influence consumer decisions, from choosing products to selecting services. However, the growing prevalence of fake reviews—intentionally misleading or fabricated feedback—poses a serious challenge. These fake reviews mislead customers, erode trust in online platforms, and create unfair advantages or disadvantages for businesses. Traditional methods of manual review moderation are inadequate for the vast amount of online content, leading to inefficient detection and an inability to keep pace with evolving tactics. Hence, there is a pressing need for an intelligent, automated system to identify and combat fake reviews effectively.

BACKGROUND:

The explosion of e-commerce platforms and online services has amplified the importance of reviews for decision-making. However, many platforms lack robust systems to detect and remove fake reviews. Fake reviews are often generated by bots, purchased from paid reviewers, or created to manipulate consumer behavior. Current systems rely heavily on manual review moderation or basic keyword filters, which are neither scalable nor effective against sophisticated fake review methods. The lack of a reliable detection system undermines consumer trust and results in financial losses for businesses that depend on authentic feedback.

RELEVANCE:

Identifying fake reviews is essential to maintaining the integrity of online platforms and fostering consumer trust. Fake reviews not only harm customers but also impact honest businesses by skewing ratings and feedback. Developing a smart fake review detection system can promote fairness in online marketplaces, enhance customer satisfaction, and preserve the credibility of review-based decision-making. By leveraging advanced technologies, the proposed system can address this widespread issue and contribute to a more transparent digital ecosystem.

OBJECTIVES:

The primary objective of this project is to develop a Fake Review Detection System that utilizes advanced technologies to identify and mitigate fake reviews, enhancing trust in online platforms. Specific objectives include:

1. **Analyzing Current Review Systems: Assessing existing review mechanisms to identify gaps and vulnerabilities.**
2. **Dataset Collection and Preparation: Gathering and preprocessing datasets of genuine and fake reviews to train the system effectively.**
3. **Developing Machine Learning Models: Building models capable of detecting patterns and anomalies in reviews to distinguish fake ones.**
4. **Incorporating Natural Language Processing (NLP): Utilizing NLP techniques to analyze sentiment, writing patterns, and linguistic cues in reviews.**
5. **Detecting Behavioral Anomalies: Tracking unusual review posting patterns (e.g., repetitive reviews, suspicious timelines) to identify potential fraud.**
6. **Real-Time Detection: Implementing algorithms to flag fake reviews in real-time for rapid intervention.**
7. **Scalability: Designing the system to handle large-scale review data across multiple platforms efficiently.**
8. **User Notifications and Reporting: Creating mechanisms to alert platform administrators and users about suspected fake reviews.**
9. **Integration with Platforms: Ensuring seamless integration with e-commerce, hospitality, and service-based platforms.**
10. **Continuous Learning and Adaptation: Updating models to adapt to evolving tactics used by fake review generators.**
11. **End-User Education: Providing resources to educate users and businesses about recognizing and reporting fake reviews.**

This project aims to foster trust in online platforms by equipping them with an effective, scalable, and adaptive system to detect and combat fake reviews.

Result:

EX NO:2	**WRITE THE SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT**
DATE:	

AIM:

To do requirement analysis and develop Software Requirement Specification (SRS) for fake review system.

ALGORITHM:

1. **Define the Purpose**
   - Identify the core purpose of the SRS document (e.g., to provide a detailed description of the system's functional and non-functional requirements).
2. **Gather Requirements**
   - Conduct discussions with stakeholders to gather all requirements.
   - Define the requirements based on user needs, system functionality, and technical constraints.
3. **Draft the SRS Document Structure**
   - Establish the sections to include, such as Introduction, Overall Description, Functional and Non-Functional Requirements, etc.
4. **Write the Introduction Section**
   - Specify the purpose of the system.
   - Define the scope of the project, including high-level goals and objectives.
   - List key definitions, acronyms, and references used in the document.
5. **Describe the Overall System**
   - Outline the system context and operating environment (e.g., blockchain framework, smart contract functionality).
   - Identify the user classes and characteristics (e.g., buyers, sellers, agents).
   - List dependencies on other systems or technologies.

1. Introduction

- The aim is to build a secure, transparent, and efficient real estate platform leveraging blockchain and smart contracts.
- This document outlines the requirements for the development of the system.

- The project will automate real estate transactions, reduce intermediaries, and ensure contract enforceability.
- Target users include buyers, sellers, agents, and property managers.
- Key terms like blockchain, smart contracts, and immutability are defined.

2. Overall Description

- The system operates in a blockchain environment, ensuring decentralized and tamper-proof transactions.
- Users can list, buy, and sell properties via automated smart contracts.
- The platform supports multi-user roles and manages property information transparently.
- It interfaces with financial institutions for payment processing.
- Assumptions include blockchain stability and user familiarity with digital transactions.

3. System Features

- Property listing and management for verified users.
- Smart contract-based transactions to automate payments and ownership transfers.
- Secure storage of transaction history on the blockchain.
- User authentication and authorization for secure access.
- Real-time notifications and updates on transaction status.

4. External Interface Requirements

- Web-based UI for user interaction with responsive design for multiple devices.
- APIs to connect with third-party services for identity verification and payment.
- Integration with blockchain nodes to facilitate smart contract execution.
- Data exchange protocols for secure transmission of sensitive information.
- Compatibility with digital wallets for cryptocurrency transactions.

5. System Requirements

- Functional: Support for user registration, property listing, transaction tracking, and payment handling.
- Performance: Fast transaction processing and quick smart contract execution.
- Security: Encrypted data storage and secure blockchain transactions.
- Reliability: High uptime and fault tolerance for critical operations.
- Scalability: Ability to handle a growing number of users and transactions.

6. Use Case Diagrams

- Depicts interactions between users (buyers, sellers, agents) and the system.
- Shows primary use cases such as "List Property," "Buy Property," and "View Contract."
- Includes user authentication, property search, and transaction completion processes.

-

- Highlights system responses for each action initiated by users.
- Provides a visual outline of each user role’s permissions and interactions.

7. System Models

- Illustrates system architecture, showing data flow between components (UI, blockchain, database).
- Describes interactions between front-end, back-end, and smart contract layers.
- Displays data storage on the blockchain and interactions with external payment gateways.
- Shows component-level communication for real-time data updates.
- Includes sequence diagrams to visualize order of operations during transactions.

8. Security and Privacy

- Ensures user data encryption and privacy through blockchain immutability.
- Implements role-based access control for secure user interactions.
- Uses multi-factor authentication for account security.
- Includes periodic security audits to identify and address vulnerabilities.
- Protects sensitive information such as personal data and payment details.

9. Maintenance and Support

- Regular updates to incorporate user feedback and improve system performance.
- Scheduled blockchain node maintenance for system stability.
- Bug-tracking and resolution process to ensure smooth operations.
- Dedicated support team for user assistance and troubleshooting.
- Documentation and training for easy onboarding of new users and admins.

SAMPLE OUTPUT:

1. Introduction

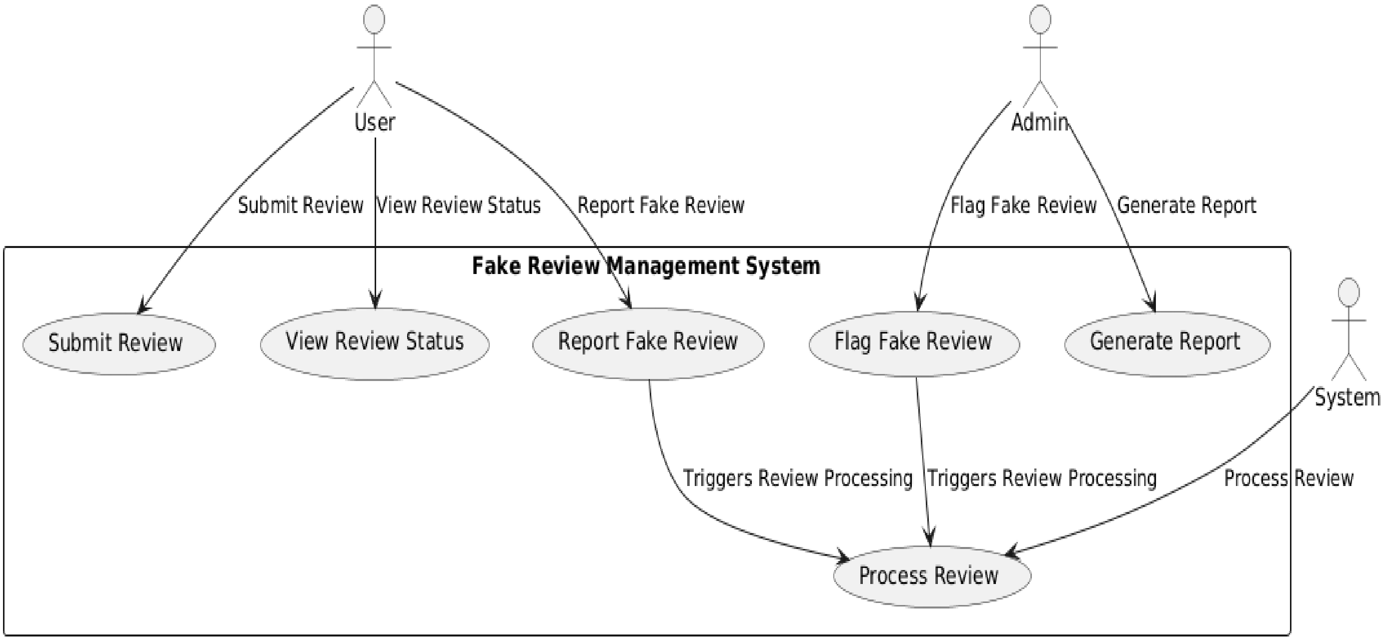
1.1 Purpose
The purpose of this document is to outline the requirements for a Fake Review Detection System, aimed at identifying fraudulent or deceptive reviews on online platforms. The system enhances transparency, protects user trust, and improves decision-making for consumers and businesses.

1.2 Goals

- **Accurately detect and flag fake reviews using machine learning (ML) and natural language processing (NLP) techniques.**
- **Provide actionable insights and tools for administrators to manage flagged reviews.**
- **Enhance the overall credibility and reliability of online platforms.**

1.3 Target Users

- **Platform Administrators: Monitor flagged reviews and manage system configurations.**
- **Business Owners: Gain insights into review authenticity and credibility.**
- **Consumers: Access reliable reviews for informed decision-making.**


User
Admin
System
Submit Review
View Review Status
Report Fake Review
Flag Fake Review
Generate Report
Fake Review Management System
Submit Review
View Review Status
Report Fake Review
Flag Fake Review
Generate Report
Triggers Review Processing
Triggers Review Processing
Process Review
Process Review

1.4 Project Scope

The system focuses on analyzing user-generated reviews, detecting patterns indicative of fraudulent activity, and providing actionable insights to stakeholders. It integrates with existing platforms and supports real-time detection and analysis.

1.5 Key Definitions

- **Fake Review: A review designed to mislead users by providing false or biased information.**
- **Machine Learning (ML): Algorithms that allow the system to improve detection accuracy over time.**
- **Natural Language Processing (NLP): Techniques for analyzing and understanding review content.**

2. Overall Description

2.1 System Environment

- **Operates on cloud-based infrastructure for scalability and real-time processing.**
- **Integrates with existing online platforms through RESTful APIs.**
- **Utilizes machine learning models for dynamic analysis of review patterns.**

2.2 User Roles and Characteristics

- **End Users: Access credibility scores for reviews when browsing products/services.**
- **Administrators: Review flagged content and make decisions on its validity.**
- **Business Owners: Monitor trends and address issues with reviews.**

2.3 Dependencies

- **Machine learning frameworks (e.g., TensorFlow, PyTorch) for model training.**
- **APIs for integration with platforms and external services.**
- **Access to training datasets of genuine and fake reviews.**

2.4 Assumptions

- **Platforms provide access to review data via APIs or data sharing agreements.**
- **Users have internet access and familiarity with basic digital tools.**

3. System Features

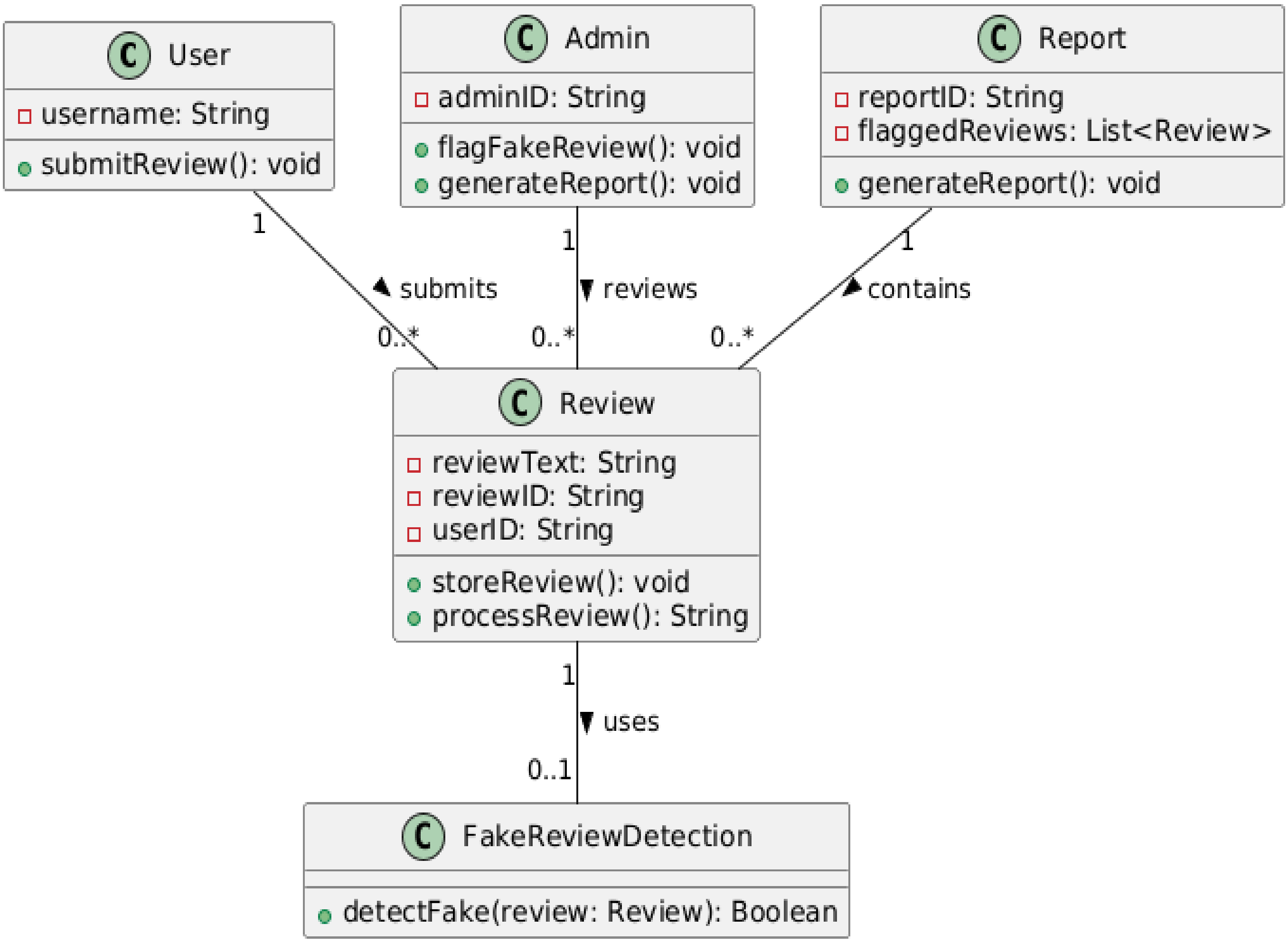
1. **Review Analysis:**
   - **NLP-based linguistic analysis to identify patterns and anomalies in review content.**
   - **Behavioral analysis of reviewer activity (e.g., posting frequency, account age).**
2. **Credibility Scoring: Assigns a trust score to each review based on authenticity likelihood.**
3. **Real-Time Detection: Processes new reviews instantly for fraudulent indicators.**
4. **Notification System: Alerts administrators about flagged reviews.**
5. **Data Visualization: Provides dashboards for monitoring flagged reviews and authenticity trends.**
6. **Integration Support: Seamlessly integrates with e-commerce platforms and digital marketplaces.**

4. External Interface Requirements

4.1 User Interfaces

- **Frontend:**

○


User
username: String
submitReview(): void
Admin
adminID: String
flagFakeReview(): void
generateReport(): void
Report
reportID: String
flaggedReviews: List<Review>
generateReport(): void
1
submits
0..*
1
reviews
0..*
1
contains
0..*
Review
reviewText: String
reviewID: String
userID: String
storeReview(): void
processReview(): String
1
uses
0..1
FakeReviewDetection
detectFake(review: Review): Boolean

  - **Web-based dashboard for administrators, built using frameworks like React.js or Angular.**
  - **Responsive design for cross-device compatibility.**
- **Consumer View: Displays credibility scores on platform reviews.**

4.2 APIs

- **RESTful APIs to exchange data with online platforms for review submission and retrieval.**
- **API endpoints for third-party verification and fraud detection services.**

4.3 Blockchain Integration (Optional)

- **To ensure the integrity of flagged reviews, maintain an immutable record of analysis results.**

4.4 Data Exchange

- **Secure protocols (e.g., HTTPS) for transmitting sensitive review data.**

5. System Requirements

5.1 Functional Requirements

- **Support for data ingestion from multiple platforms.**
- **Enable administrators to view and manage flagged reviews.**
- **Provide reports summarizing authenticity trends and system performance.**

5.2 Non-Functional Requirements

- **Performance: Detect and score reviews within 1 second of submission.**
- **Security: Encrypt all stored and transmitted data.**
- **Reliability: Ensure system uptime of 99.9% for continuous operation.**
- **Scalability: Handle large volumes of reviews across multiple platforms.**

6. Use Case Diagrams

6.1 Core Use Cases

- **Submit Review: User submits a review to the platform.**
- **Analyze Review: System processes the review for authenticity.**
- **Flag Review: System flags a suspicious review and notifies administrators.**
- **Admin Action: Administrators review flagged content and take necessary actions.**
- **Generate Reports: System provides periodic insights into review authenticity trends.**

7. System Models

7.1 Architecture

- **Frontend Layer: User interface for dashboards and reports.**
- **Backend Layer: APIs, ML models, and data processing.**
- **Data Storage: Database for storing reviews, results, and flagged items.**

7.2 Sequence Diagram

- **Illustrates the flow from review submission to detection and administrator action.**

8. Security and Privacy

- **Data Encryption: Encrypt review data during storage and transmission.**
- **Access Control: Implement role-based permissions for users and administrators.**
- **Multi-Factor Authentication (MFA): Enhance security for admin accounts.**
- **Regular Audits: Conduct periodic checks for vulnerabilities and biases in the ML model.**

9. Maintenance and Support

- **System Updates: Regular updates to ML models to adapt to evolving tactics.**
- **Technical Support: Available for troubleshooting integration and system issues.**
- **Bug Tracking: Maintain a bug-reporting system for continuous improvement.**
- **Documentation: Provide detailed guides for administrators and platform integrators.**
- **Training: Offer workshops for stakeholders to understand the system.**

Result:

EX NO:3	**DRAW THE ENTITY RELATIONSHIP DIAGRAM**
DATE:	

AIM:

To Draw the Entity Relationship Diagram for real estate booking system.

ALGORITHM:

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1:N Relationship Types.

Step 5: Mapping of Binary M:N Relationship Types.

Step 6: Mapping of Multivalued attributes.

INPUT:

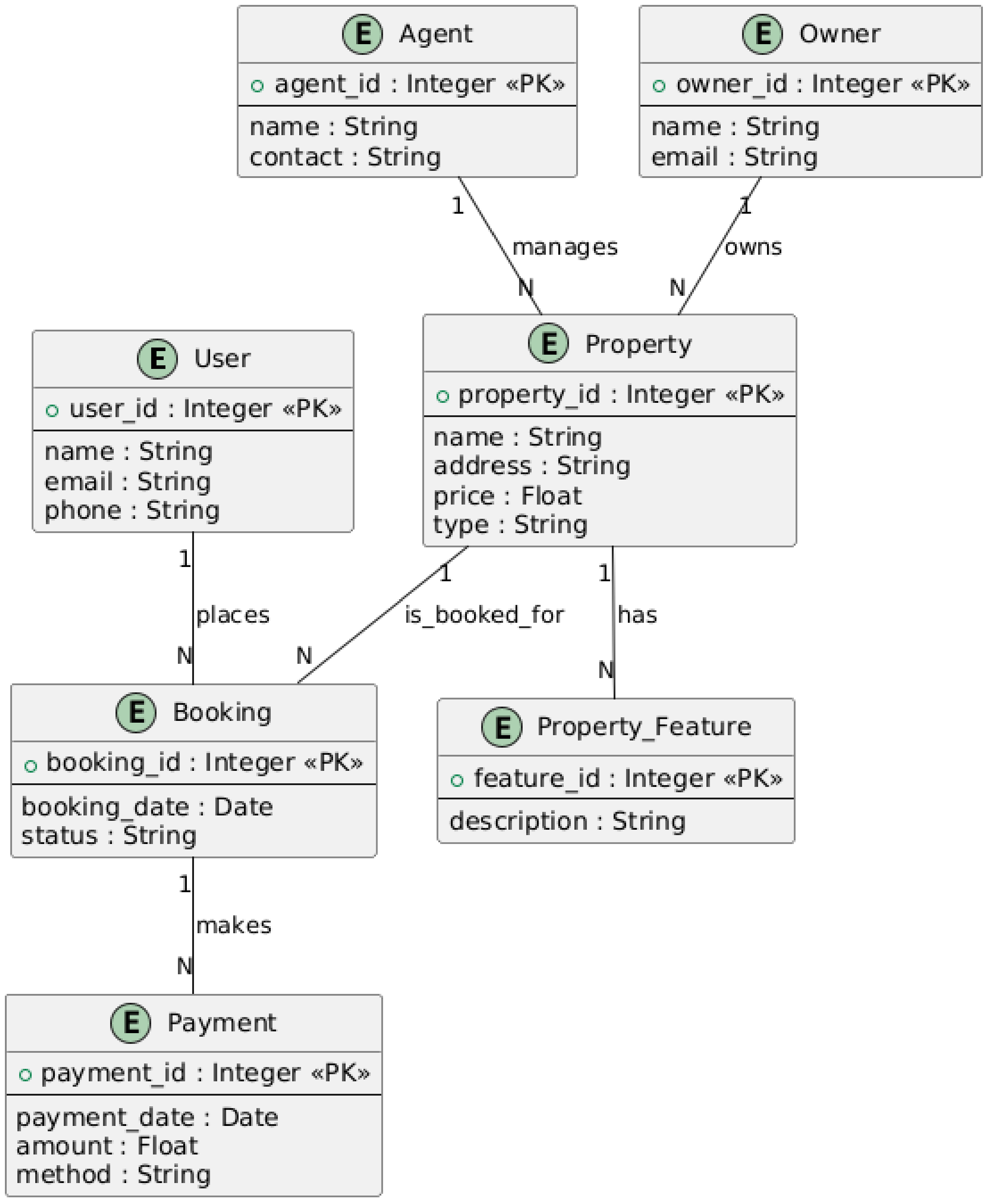
Entities

Entity Relationship Matrix

Primary Keys

Attributes

Mapping of Attributes with Entities


E Agent
○ agent_id : Integer «PK»
name : String
contact : String
E Owner
○ owner_id : Integer «PK»
name : String
email : String
1
manages
N
1
owns
N
E User
○ user_id : Integer «PK»
name : String
email : String
phone : String
E Property
○ property_id : Integer «PK»
name : String
address : String
price : Float
type : String
1
places
N
1
is_booked_for
N
1
has
N
E Booking
○ booking_id : Integer «PK»
booking_date : Date
status : String
E Property_Feature
○ feature_id : Integer «PK»
description : String
1
makes
N
E Payment
○ payment_id : Integer «PK»
payment_date : Date
amount : Float
method : String

Result:

EX NO:4	**DRAW THE DATA FLOW DIAGRAMS AT LEVEL 0 AND LEVEL 1**
DATE:	

AIM:

To Draw the Data Flow Diagram for real estate booking systemand List the Modules in the Application.

ALGORITHM:

1. Open the Visual Paradigm to draw DFD (Ex.Lucidchart)
2. Select a data flow diagram template
3. Name the data flow diagram
4. Add an external entity that starts the process
5. Add a Process to the DFD
6. Add a data store to the diagram
7. Continue to add items to the DFD
8. Add data flow to the DFD
9. Name the data flow
10. Customize the DFD with colours and fonts
11. Add a title and share your data flow diagram

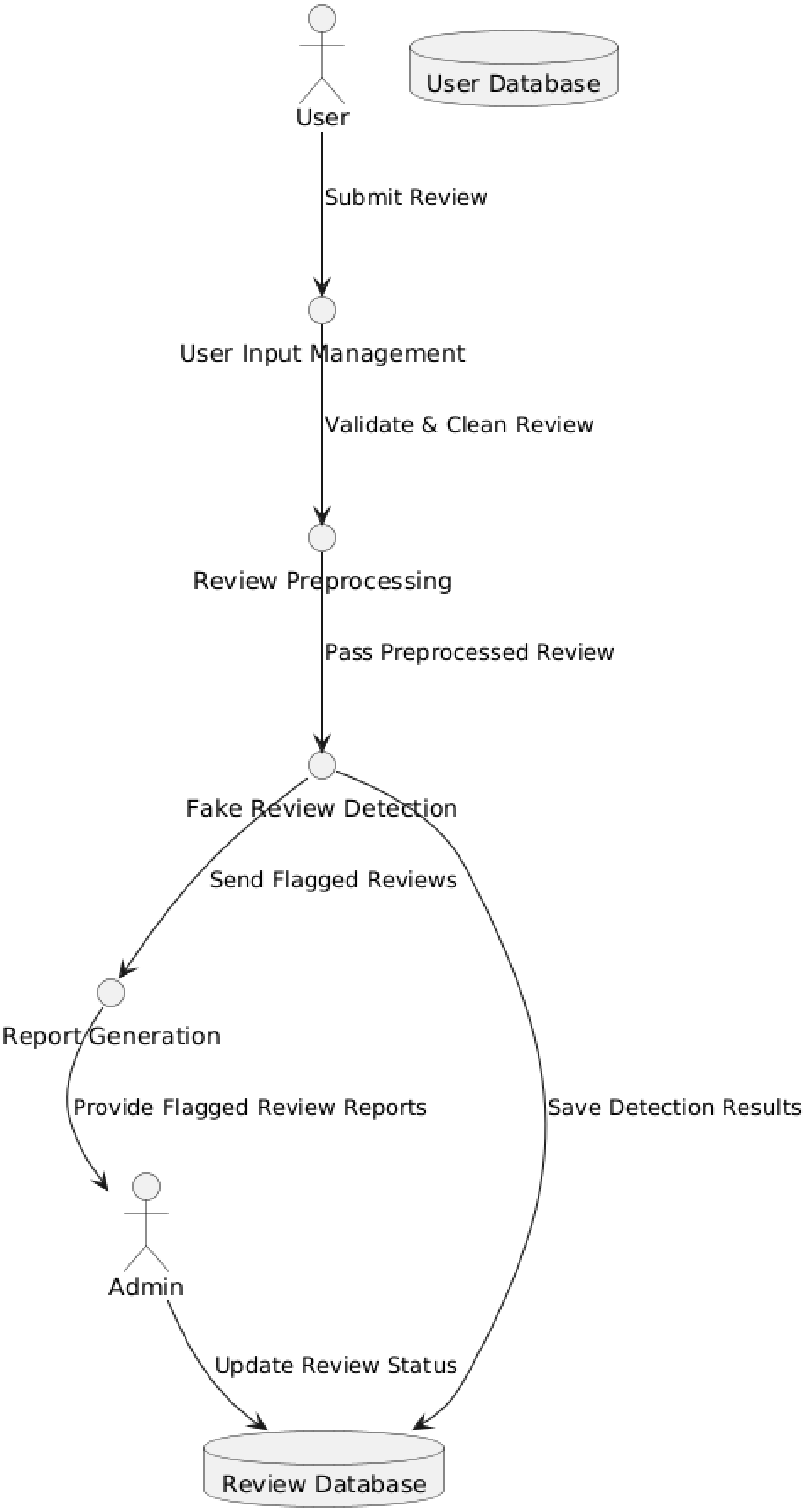
INPUT:

Processes

Datastores

External Entities

Result:


User
User Database
Submit Review
User Input Management
Validate & Clean Review
Review Preprocessing
Pass Preprocessed Review
Fake Review Detection
Send Flagged Reviews
Save Detection Results
Report Generation
Provide Flagged Review Reports
Admin
Update Review Status
Review Database

EX NO:5	**DRAW USE CASE DIAGRAM**
DATE:	

AIM:

To Draw the Use Case Diagram for real estate booking system.

ALGORITHM:

Step 1: Identify Actors

Step 2: Identify Use Cases

Step 3: Connect Actors and Use Cases

Step 4: Add System Boundary

Step 5: Define Relationships

Step 6: Review and Refine

Step 7: Validate

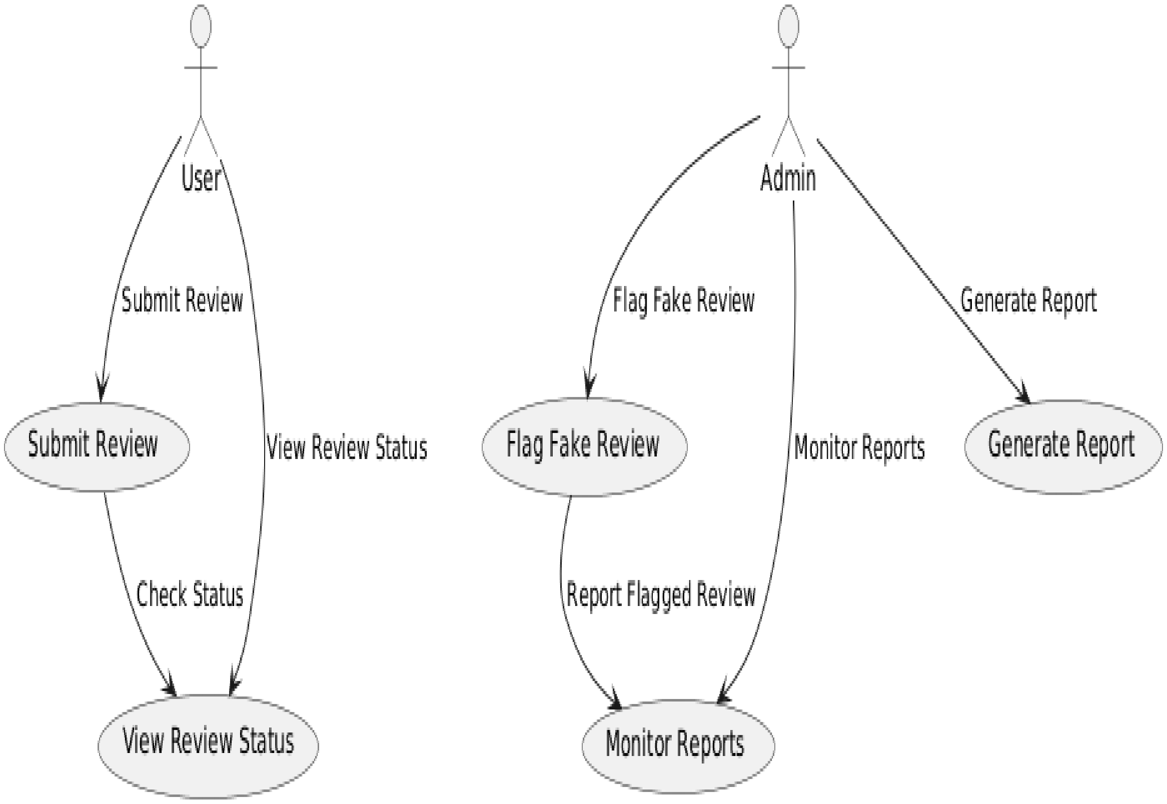
INPUTS:

Actors

Use Cases

Relations

Result:


User
Submit Review
View Review Status
Submit Review
Check Status
View Review Status
Admin
Flag Fake Review
Monitor Reports
Generate Report
Flag Fake Review
Generate Report
Report Flagged Review
Monitor Reports

EX NO:6	DRAW ACTIVITY DIAGRAM OF ALL USE CASES.
DATE:	

AIM:

To Draw the activity Diagram for real estate booking system.

ALGORITHM:

Step 1: Identify the Initial State and Final States

Step 2: Identify the Intermediate Activities Needed

Step 3: Identify the Conditions or Constraints

Step 4: Draw the Diagram with Appropriate Notations

INPUTS:

Activities

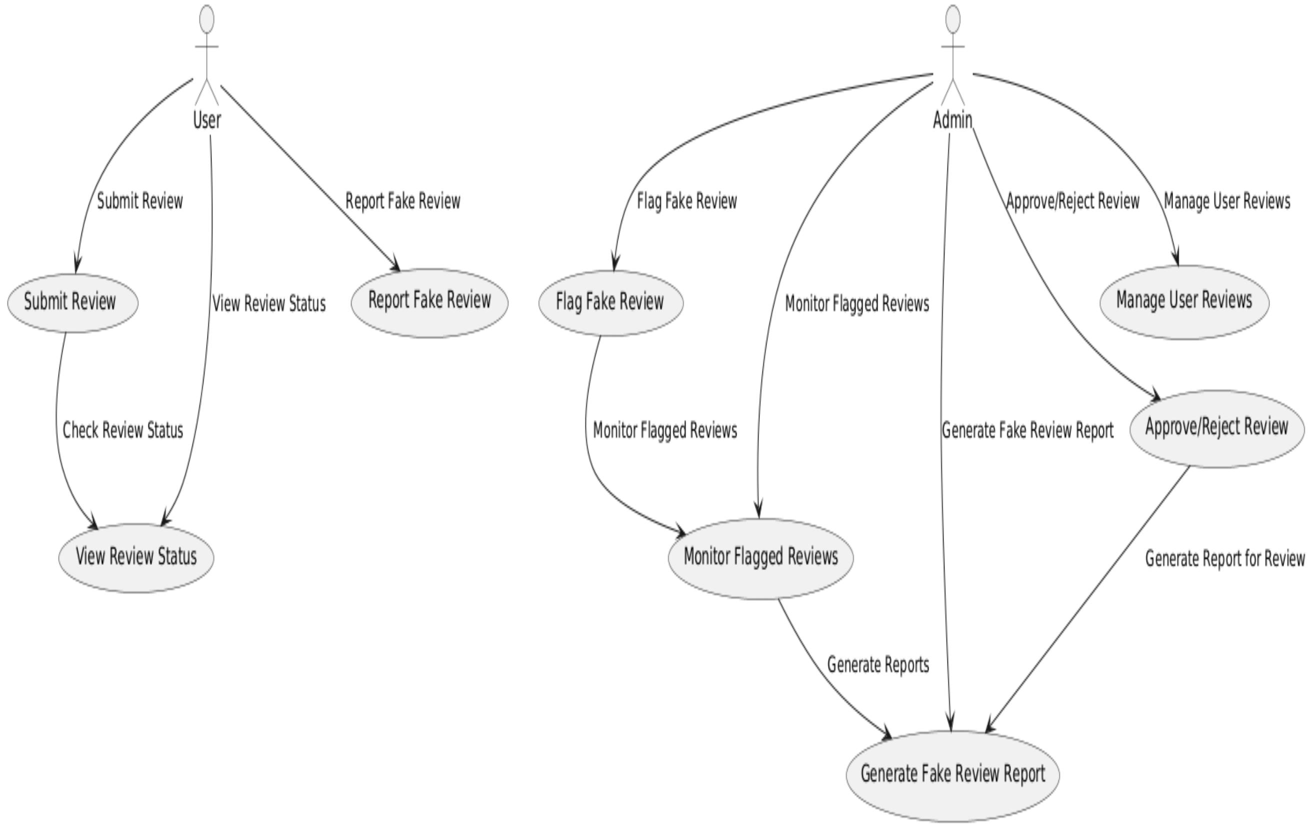
Decision Points

Guards

Parallel Activities

Conditions

Result:


User
Submit Review
View Review Status
Report Fake Review
Submit Review
Report Fake Review
Check Review Status
View Review Status
Admin
Flag Fake Review
Monitor Flagged Reviews
Approve/Reject Review
Manage User Reviews
Flag Fake Review
Manage User Reviews
Generate Fake Review Report
Approve/Reject Review
Monitor Flagged Reviews
Generate Report for Review
Generate Reports
Generate Fake Review Report

EX NO:7	**DRAW STATE CHART DIAGRAM OF ALL USE CASES.**
DATE:	

AIM:

To Draw the State Chart Diagram for real estate booking system.

ALGORITHM:

STEP-1: Identify the important objects to be analysed.

STEP-2: Identify the states.

STEP-3: Identify the events.

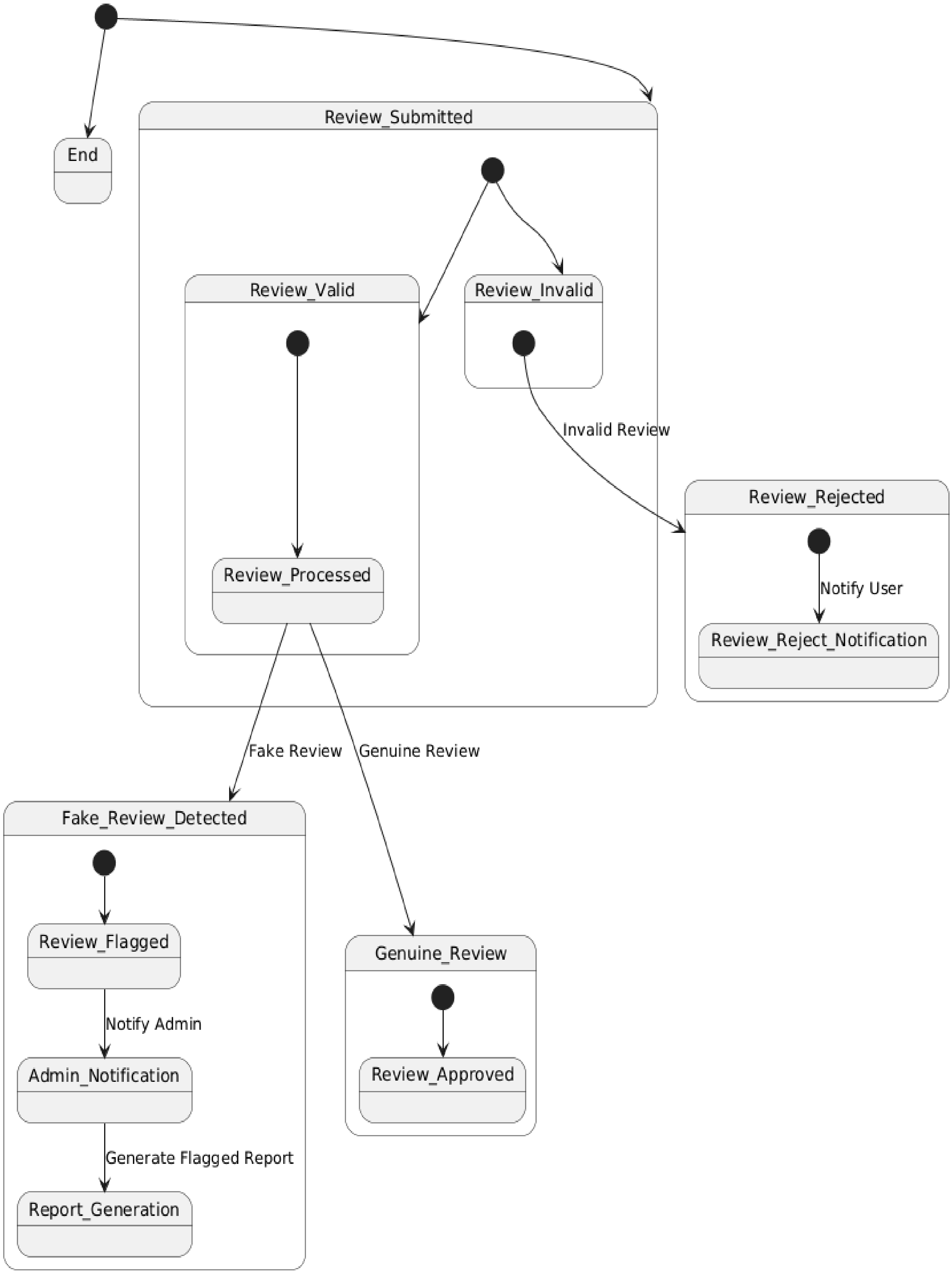
INPUTS:

Objects

States

Events

Result:


End
Review_Submitted
Review_Valid
Review_Invalid
Review_Processed
Invalid Review
Review_Rejected
Notify User
Review_Reject_Notification
Fake Review
Genuine Review
Fake_Review_Detected
Review_Flagged
Notify Admin
Admin_Notification
Generate Flagged Report
Report_Generation
Genuine_Review
Review_Approved

EX NO:8	**DRAW SEQUENCE DIAGRAM OF ALL USE CASES.**
DATE:	

AIM: To Draw the Sequence Diagram for real estate booking system.

ALGORITHM:

1. Identify the Scenario

2. List the Participants

3. Define Lifelines

4. Arrange Lifelines

5. Add Activation Bars

6. Draw Messages

7. Include Return Messages

8. Indicate Timing and Order

9. Include Conditions and Loops

10. Consider Parallel Execution

11. Review and Refine

12. Add Annotations and Comments

13. Document Assumptions and Constraints

14. Use a Tool to create a neat sequence diagram

INPUTS:

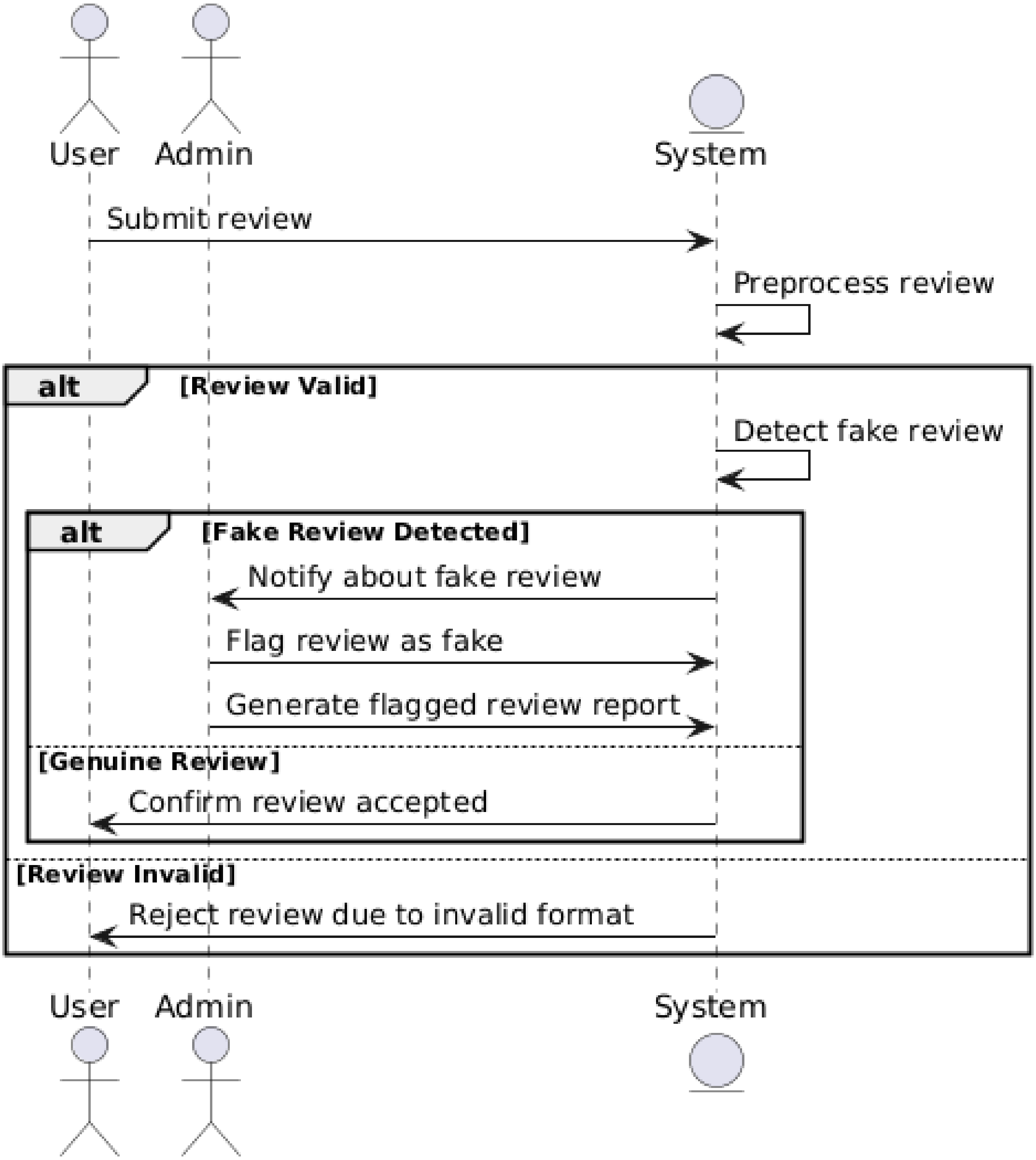
Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:


User
Admin
System
Submit review
Preprocess review
alt
[Review Valid]
Detect fake review
alt
[Fake Review Detected]
Notify about fake review
Flag review as fake
Generate flagged review report
[Genuine Review]
Confirm review accepted
[Review Invalid]
Reject review due to invalid format
User
Admin
System

EX NO:9	**DRAW COLLABORATION DIAGRAM OF ALL USE CASES**
DATE:	

AIM:

To Draw the Collaboration Diagram for real estate booking system.

ALGORITHM:

Step 1: Identify Objects/Participants

Step 2: Define Interactions

Step 3: Add Messages

Step 4: Consider Relationships

Step 5: Document the collaboration diagram along with any relevant explanations or annotations.

INPUTS:

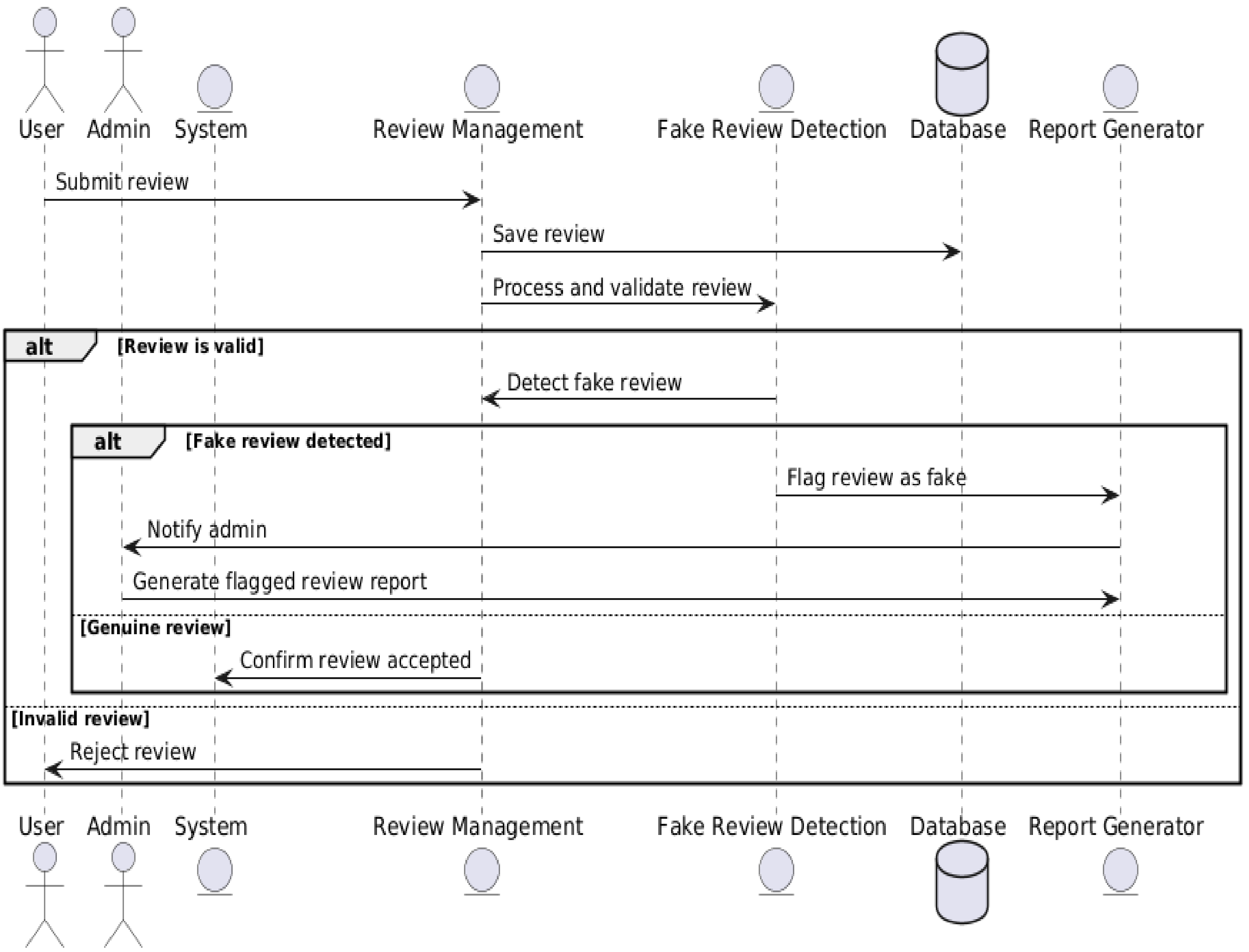
Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:


User
Admin
System
Review Management
Fake Review Detection
Database
Report Generator
Submit review
Save review
Process and validate review
alt
[Review is valid]
Detect fake review
alt
[Fake review detected]
Flag review as fake
Notify admin
Generate flagged review report
[Genuine review]
Confirm review accepted
[Invalid review]
Reject review
User
Admin
System
Review Management
Fake Review Detection
Database
Report Generator

EX NO:10	**ASSIGN OBJECTS IN SEQUENCE DIAGRAM TO CLASSES AND MAKE CLASS DIAGRAM.**
DATE:	

AIM:

To Draw the Class Diagram for real estate booking system.

ALGORITHM:

1. Identify Classes
2. List Attributes and Methods
3. Identify Relationships
4. Create Class Boxes
5. Add Attributes and Methods
6. Draw Relationships
7. Label Relationships
8. Review and Refine
9. Use Tools for Digital Drawing

INPUTS:

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

RESULT:

EX NO:11	**MINI PROJECT FOR FAKE REVIE MANAGEMENT SYSTEM**
DATE:	

Aim:

The aim of the Fake Review Management System is to design a mechanism that efficiently detects fake reviews submitted by users, processes them for authenticity, and allows admins to flag, manage, and generate reports for fake reviews. The system ensures that only genuine reviews are approved and published, enhancing the credibility of the platform.

Algorithm:

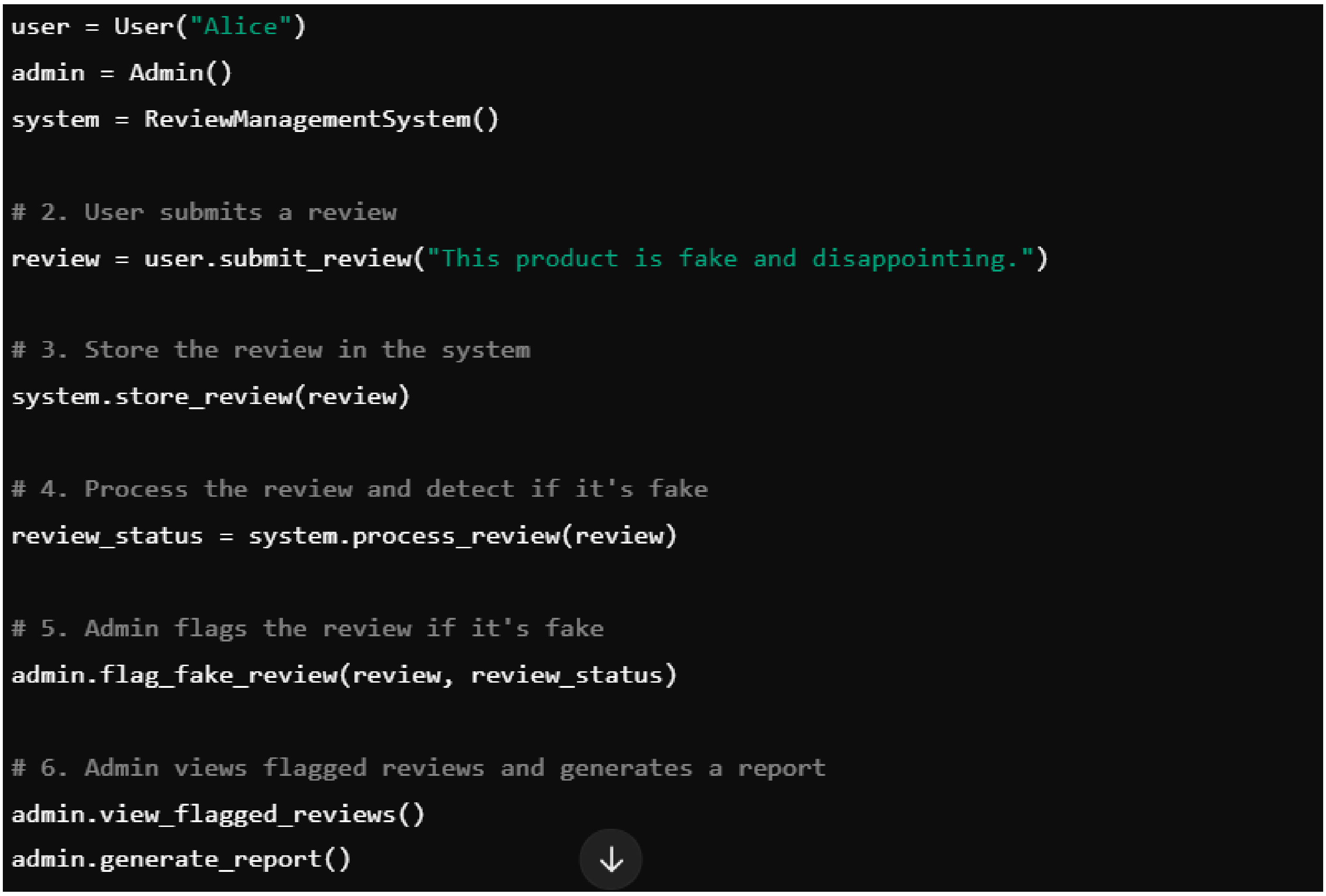
1. Start the system.
2. User submits a review:
   - The user submits a review via the platform.
3. Store the review:
   - The system stores the submitted review in a Database for processing.
4. Process the review:
   - The review is validated for correct formatting and content.
5. Detect fake review:
   - The review is passed through a Fake Review Detection model that checks for suspicious patterns such as:
     - Overuse of certain keywords.
     - Repetitive patterns across multiple reviews.
     - Inconsistent user behavior (e.g., multiple reviews in a short time from a new account).
6. Classify the review:
   - If the system detects the review as fake, flag it for admin review.
   - If the review is genuine, approve it and notify the user.
7. Admin reviews flagged reviews:
   - The admin views the flagged reviews, which are either approved or deleted.
8. Generate fake review reports:
   - Admin generates periodic reports on the flagged reviews.
9. End the process.

PROGRAM:

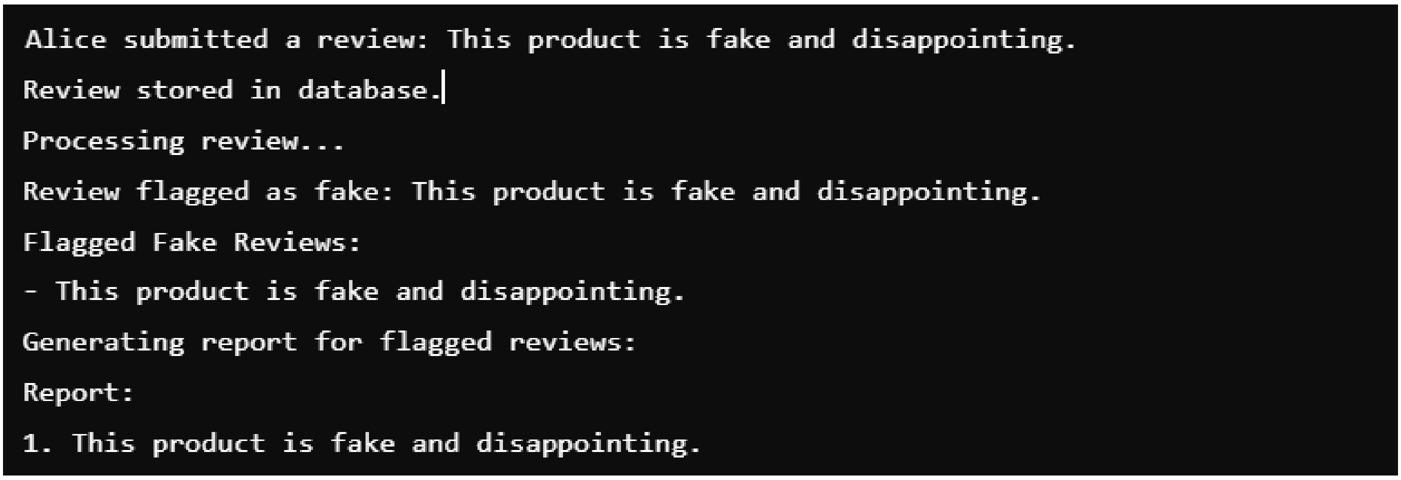
```
# Class to represent a User
class User:
    def __init__(self, username):
        self.username = username

    def submit_review(self, review_text):
        print(f"{self.username} submitted a review: {review_text}")
        return review_text

# Class to handle review management
class ReviewManagementSystem:
    def __init__(self):
        self.database = []  # List to store reviews

    def store_review(self, review):
        self.database.append(review)
        print("Review stored in database.")

    def process_review(self, review):
        print("Processing review...")
        return self.fake_review_detection(review)

    def fake_review_detection(self, review):
        # Fake review detection logic (simplified for demonstration)
        suspicious_keywords = ["fake", "scam", "unreal", "disappointing"]
        if any(keyword in review.lower() for keyword in suspicious_keywords):
            return "Fake"
        else:
            return "Genuine"

# Class to represent the Admin's actions
class Admin:
    def __init__(self):
```

```
 self.flagged_reviews = []  # List to store flagged fake reviews

    def flag_fake_review(self, review, review_status):
        if review_status == "Fake":
            self.flagged_reviews.append(review)
            print(f"Review flagged as fake: {review}")

    def view_flagged_reviews(self):
        print("Flagged Fake Reviews:")
        for review in self.flagged_reviews:
            print(review)

    def generate_report(self):
        print("Generating report for flagged reviews:")
        for review in self.flagged_reviews:
            print(f"Report: {review}")

# Main simulation

# 1. Create a user and an admin
user = User("Alice")
admin = Admin()
system = ReviewManagementSystem()

# 2. User submits a review
review = user.submit_review("This product is fake and disappointing.")

# 3. Store the review in the system
system.store_review(review)

# 4. Process the review and detect if it's fake
review_status = system.process_review(review)
```

OUTPUT:



```
user = User("Alice")
admin = Admin()
system = ReviewManagementSystem()

# 2. User submits a review
review = user.submit_review("This product is fake and disappointing.")

# 3. Store the review in the system
system.store_review(review)

# 4. Process the review and detect if it's fake
review_status = system.process_review(review)

# 5. Admin flags the review if it's fake
admin.flag_fake_review(review, review_status)

# 6. Admin views flagged reviews and generates a report
admin.view_flagged_reviews()
admin.generate_report()
```



```
Alice submitted a review: This product is fake and disappointing.
Review stored in database.
Processing review...
Review flagged as fake: This product is fake and disappointing.
Flagged Fake Reviews:
- This product is fake and disappointing.
Generating report for flagged reviews:
Report:
1. This product is fake and disappointing.
```

```
# 5. Admin flags the review if it's fake
admin.flag_fake_review(review, review_status)


# 6. Admin views flagged reviews and generates a report
admin.view_flagged_reviews()
admin.generate_report()
```

Conclusion:

The Fake Review Management System provides an automated approach to identify, flag, and report fake reviews, improving the trustworthiness of the platform. By automating the detection process and allowing admin oversight, the system ensures that only authentic reviews are showcased, enhancing the credibility and quality of user feedback. The system can be further optimized with advanced machine learning models for more accurate detection and scalability in handling large volumes of reviews.