PRACTICAL 1

AIM:

Compare different software process models and apply the appropriate model for the given definitions:

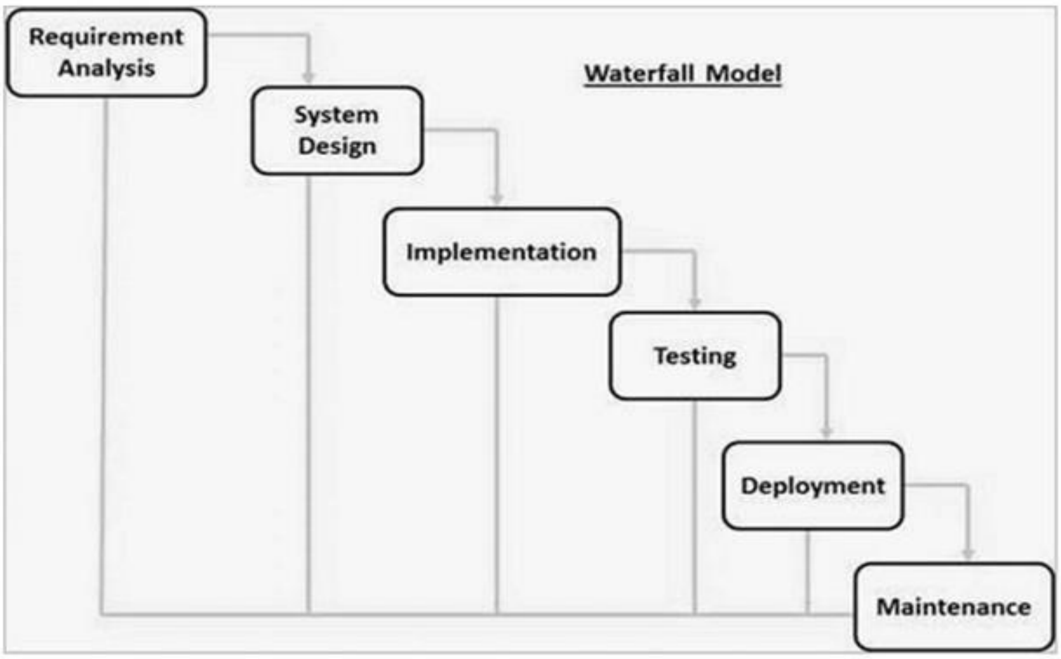
Examine and contrast various software development methodologies such as Waterfall, Agile, Spiral, and V-Model. Evaluate the strengths and weaknesses of each model in different project scenarios. Determine and justify which model best suits specific project requirements based on factors like project size, complexity, and team dynamics.

1. WATERFALL MODEL:

- **Overview:**

The Waterfall Model is a traditional and linear approach to software development that follows a sequential and phased process. It is one of the earliest and most straightforward methodologies in software engineering. The model is called "waterfall" because it progresses through distinct phases in a downward, linear fashion, similar to a waterfall flowing through different stages. Each phase must be completed before moving on to the next one, and changes made in one phase require revisiting previous phases.

- **Diagram:**



i. **Requirements Analysis and Definition:** In the initial phase, the project team works closely with stakeholders to gather and analyse project requirements. The goal is to clearly define the scope of the software system, identify functionalities, and establish a detailed understanding of user needs.

ii. **System Design:** Based on the gathered requirements, the system design phase involves creating a detailed blueprint for the software. This includes defining the overall architecture, specifying components and modules, and outlining data structures and interfaces. The result is a comprehensive system design document.

iii. **Implementation (Coding):** In this phase, the actual coding or programming of the software system takes place. The system design document is used as a reference, and the code is written according to coding standards and guidelines. The outcome is executable code that represents the desired functionalities.

iv. **Testing:** The testing phase focuses on verifying and validating the software. Different types of testing are conducted, including unit testing (testing individual components), integration testing (testing interactions between components), and system testing (testing the entire system). The goal is to identify and fix any defects or issues.

v. **Deployment (Installation):** Once the testing phase is successfully completed, the software is deployed to the production environment. This involves installing the software on the end users' systems or making it available for use. Deployment marks the transition from development to operational use.

vi. **Maintenance and Support:** The maintenance phase involves ongoing support and enhancements to the software. It includes addressing issues discovered in the live environment, fixing bugs, and making updates or improvements as necessary.

Maintenance ensures the long-term reliability and effectiveness of the software.

 Advantages:

i. **Clarity and Simplicity:** The Waterfall Model is straightforward and easy to understand. Its sequential nature provides a clear structure for both development teams and stakeholders.

ii. **Well-Defined Phases:** Each phase in the Waterfall Model has distinct deliverables and objectives, making it easier to manage and plan. This well defined structure aids in project management and documentation.

iii. **Stable Requirements:** It works well when the project requirements are well understood and unlikely to change. The emphasis on gathering comprehensive requirements at the beginning of the project can be an advantage if there is stability in requirements.

iv. **Controlled Development Process:** The linear and sequential nature of the Waterfall Model allows for better control over the project timeline, milestones, and resources.

Disadvantages:

i. **Inflexibility to Changes:** One of the main drawbacks is the model's lack of flexibility. Once a phase is completed, it is difficult to go back and make changes. Changes in requirements or design may be challenging to accommodate.

ii. **Late Customer Feedback:** Customer feedback is typically collected only at the end of the project during the testing phase. This can lead to misunderstandings or misalignments with customer expectations, as there are limited opportunities for adjustments during development.

iii. **Risk and Uncertainty:** If requirements are not accurately captured at the beginning, or if there are unforeseen challenges during development, it can result in significant issues later in the project. The risk is higher when dealing with complex or innovative projects.

iv. Limited Visibility Until Testing: Stakeholders might not get a tangible product until the testing phase, making it challenging to assess progress and ensure that the project is on the right track until late in the development process.

v. **Not Suitable for Large Projects:** For large and complex projects, the Waterfall Model may be impractical, as the time between project initiation and final delivery can be extensive, and changes may be required during such long durations.

Applications:

i. **Small to Medium-Sized Projects:** The Waterfall Model is well-suited for small to medium-sized projects with clear and well-defined requirements. In such cases, the sequential and structured approach of the Waterfall Model helps in managing and organizing the development process efficiently.

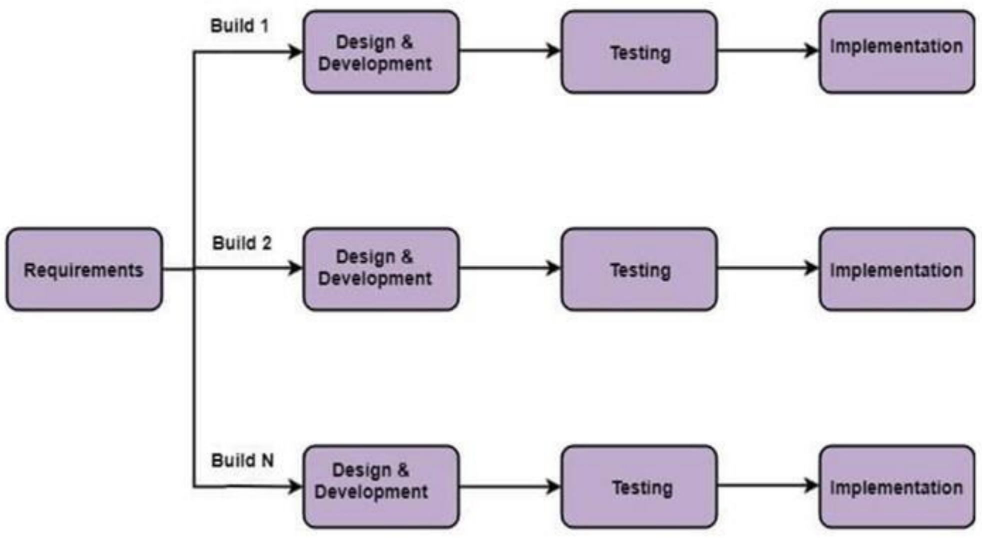
ii. **Simple Systems with Little Complexity:** Projects that involve the development of relatively simple and straightforward systems, where the architecture and design are not overly complex, can benefit from the Waterfall Model. The model's linear progression is effective in such uncomplicated scenarios.

2. INCREMENTAL MODEL:

- **Overview:**

Incremental model is an iterative approach to software development that involves dividing the project into smaller, manageable parts or increments. Each increment represents a portion of the overall system's functionality and is developed and delivered separately. The incremental model allows for the incremental build-up of the system, with each increment building upon the functionality of the previous one.

- **Diagram:**



Lifecycle of incremental model:

i. **Requirements Analysis:** Similar to the Waterfall Model, the Incremental Model starts with the analysis of requirements. However, instead of gathering all requirements upfront, the project team identifies the core set of requirements that can form the basis for the first increment.

ii. **System Design:** In the system design phase, the architecture and overall structure of the system are defined based on the initial set of requirements. The design is modular, and decisions are made about which features will be included in the first increment. **iii. Implementation (First Increment):** The first increment is implemented, incorporating the essential features identified in the initial requirements and design phase. This increment represents a partial but functional version of the complete system.

iv. **Testing (First Increment):** Testing is performed on the features implemented in the first increment. This includes unit testing, integration testing, and system testing to ensure that the added functionality meets the specified requirements and does not introduce errors.

v. **Evaluation and Feedback:** Stakeholders, including end-users, evaluate the functionality implemented in the first increment. Feedback is gathered to identify improvements, changes, or additional features that may be required.

vi. **Iteration (Repeat Steps 1-5):** Steps 5 to 9 are repeated for each increment until the complete system is developed. The process of iteration allows for continuous refinement and adaptation based on ongoing feedback and changing requirements. **vii. Deployment of Final Increment:** Once all increments are implemented, tested, and validated, the final increment is deployed, and the complete system is delivered to the end-users or customers.

viii. Maintenance and Support: After deployment, the system enters the maintenance phase, where any issues discovered in the live environment are addressed, and updates or enhancements may be made as need

Advantages :

i. **Flexibility and Adaptability:** The Incremental Model allows for flexibility and adaptability to changes in requirements. New features or changes can be incorporated in subsequent increments based on feedback.

ii. **Early Detection of Defects:** Since testing is performed incrementally, defects can be identified and addressed early in the development process. This leads to higher software quality and reduces the cost of fixing issues later in the project.

iii. **Customer Feedback:** Continuous customer feedback is encouraged throughout the development process. This ensures that the delivered increments align with customer expectations and requirements.

iv. **Reduced Risk:** By breaking the project into manageable increments, the overall project risk is reduced. Each increment is a smaller and more manageable piece, making it easier to identify and mitigate potential issues

Disadvantages:

i. **Management Overhead:** Managing multiple increments and coordinating their development can add complexity and overhead. Proper project management is essential to ensure synchronization between teams working on different increments.

ii. **Higher Costs:** The cost of managing and maintaining multiple increments can be higher compared to other development models. Each increment may require its own set of resources, including testing and deployment efforts.

iii. **Compatibility Issues:** If not managed carefully, compatibility issues between different increments can arise. Ensuring that all increments work seamlessly together may require additional effort.

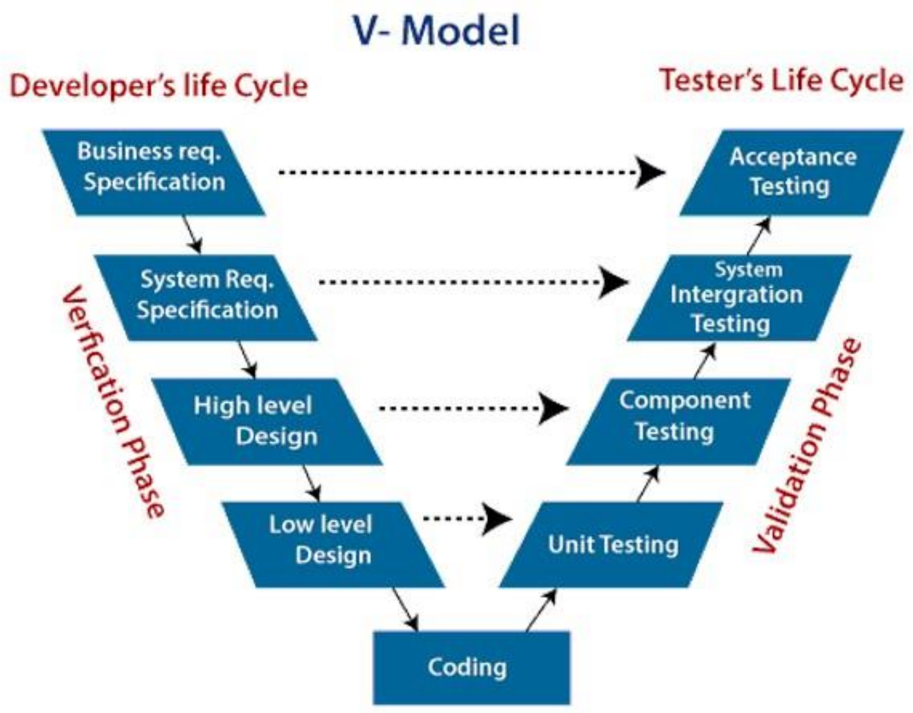
Applications:

i. Product Lines with Common Features: When developing a family of related products with common features and functionalities, the Incremental Model can be applied. Each increment can represent a version of the product with additional features or variations. **ii. Web Application Development:** Web development often involves continuous updates, feature additions, and improvements. The Incremental Model is suitable for developing web applications as it allows developers to release new features incrementally, responding to changing user needs and market demands.

iii. Product Lines with Common Features: When developing a family of related products with common features and functionalities, the Incremental Model can be applied. Each increment can represent a version of the product with additional features or variations.

3. V MODEL:

- **Overview:**
  The V-model is a type of SDLC model where the process executes in a sequential manner in a V-shape. It is also known as the Verification and Validation model. It is based on the association of a testing phase for each corresponding development stage. The development of each step is directly associated with the testing phase. The next phase starts only after completion of the previous phase i.e., for each development activity, there is a testing activity corresponding to it.
- **Diagram:**



Verification: It involves a static analysis method (review) done without executing code. It is the process of evaluation of the product development process to find whether specified requirements meet.

Validation: It involves dynamic analysis method (functional, non-functional), testing is done by executing code. Validation is the process to classify the software after the completion of the development process to determine whether the software meets the customer expectations and requirements.

- **Advantages:**

  i. **Clear and Simple:** The V-Model is straightforward and easy to understand. Its graphical representation with a V-shaped diagram makes it clear how each development phase corresponds to a testing phase.

  ii. **Early Detection of Defects:** Testing activities are performed in parallel with development, allowing for early detection and correction of defects. This can lead to a higher quality end product.

  iii. **Traceability:** The V-Model promotes traceability by establishing a clear relationship between each development phase and its corresponding testing phase. This helps in tracking the progress of the project and ensuring that all requirements are addressed.

  iv. **Structured Approach:** The model provides a structured and systematic approach to software development. This structure is beneficial for planning, organizing, and managing the development and testing processes.

Disadvantages:

i. **Rigidity and Inflexibility:** The V-Model is a sequential and rigid methodology, making it less adaptable to changes in requirements. If there are changes or additions to the requirements during the development process, it can be challenging to incorporate them without affecting the entire project plan.

ii. **Not Suitable for Large and Complex Projects:** The V-Model may not be well suited for large and complex projects where requirements are subject to frequent changes. Its rigid structure may hinder the flexibility required for such projects. **iii. Costly to Change:** Making changes to the project after it has progressed beyond the early stages can be expensive and time-consuming. This is because modifications may require revisiting and potentially redoing multiple phases of the development and testing process.

Applications:

i. **Small to Medium-sized Projects:** The V-Model is often applied to small to medium-sized projects where requirements are well-understood and unlikely to change significantly during the development process. Its structured approach helps manage such projects effectively.

ii. **Cost-Conscious Projects:** Organizations with budget constraints may find the VModel appealing due to its emphasis on early defect detection. Addressing issues early in the development life cycle can potentially reduce overall project costs.

4. AGILE MODEL:

Overview:

The Agile Model was primarily designed to help a project adapt quickly to change requests. So, the main aim of the Agile model is to facilitate quick project completion. Agility is achieved by fitting the process to the project and removing activities that may not be essential for a specific project. Also, anything that is a waste of time and effort is avoided. The Agile Model refers to a group of development processes. These processes share some basic characteristics but do have certain subtle differences among themselves. In the Agile model, the requirements are decomposed into many small parts that can be incrementally developed. The Agile model adopts Iterative development. Each incremental part is developed over an iteration. Each iteration is intended to be small and easily manageable and can be completed within a couple of weeks only. At a time one iteration is planned, developed, and deployed to the customers.

Long-term plans are not made.

- **Diagram:**

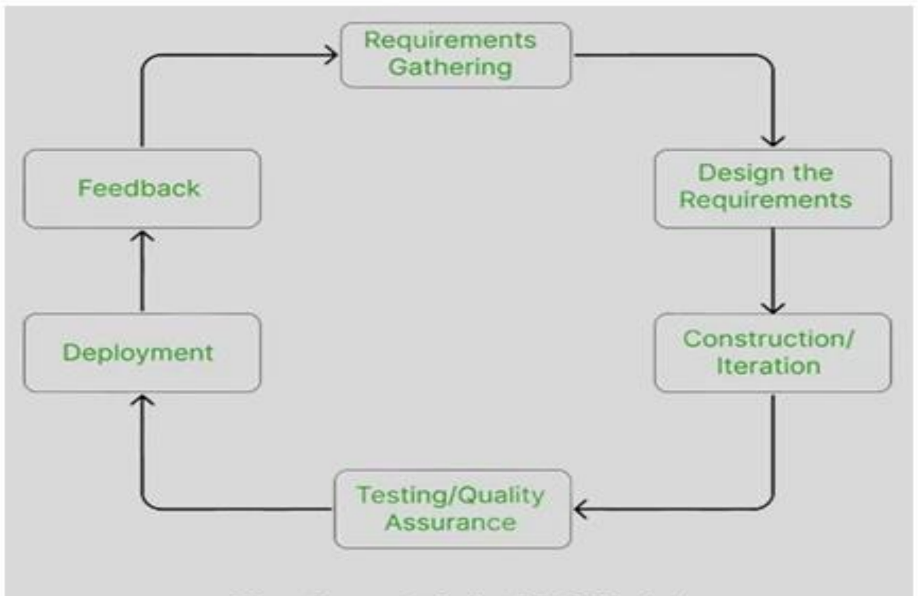


Fig:- Steps in Agile SDLC Model

- **Advantages:**

i. **Flexibility and Adaptability:** Agile is highly flexible and adaptable to changes in requirements, allowing teams to respond quickly to evolving customer needs and market conditions. This flexibility is particularly beneficial in dynamic and fast-paced environments.

ii. **Faster Time to Market:** The iterative and incremental nature of Agile allows for the delivery of working software in short cycles. This results in faster time to-market, enabling organizations to release valuable features sooner and gain a competitive edge.

iii. Customer Involvement and Satisfaction: Agile places a strong emphasis on customer collaboration throughout the development process. Regular feedback and involvement of stakeholders ensure that the delivered software aligns with customer expectations, leading to higher satisfaction.

 Disadvantages:

i. **Lack of Predictability:** Agile's adaptability can make it challenging to predict project scope and timelines accurately.

ii. **Dependency on Customer Availability:** Frequent collaboration requires consistent engagement from customers and stakeholders.

iii. **Team Inexperience:** Teams unfamiliar with Agile practices may initially struggle with implementation.

iv. **Potential for Scope Creep:** Agile's flexibility can result in scope creep if not managed carefully.

v. **Difficulty in Scaling:** Scaling Agile to large projects or organizations can be complex.

vi. **Documentation Challenges:** Agile prioritizes working software over extensive documentation, which may be a challenge in certain environments.

Applications:

vii. **Marketing Campaigns:** Agile principles can be applied to marketing activities, allowing teams to iterate on campaigns, respond to changing market conditions, and measure the effectiveness of different strategies.

viii. **Non-Profit and Government Projects:** Agile is applied in non-profit organizations and government projects to improve collaboration, responsiveness to stakeholder needs, and the delivery of public services. **ix. Consulting Services:** Agile principles are used in consulting services to deliver projects in a collaborative and adaptive manner, ensuring that client needs are met effectively.

AIM:

Prepare a Case Study of the Agile Model based on an IT company comprising 3to 4 pages: Develop a comprehensive case study on how an IT company effectively implemented the Agile Model for their software development processes. Highlight the key principles of Agile, such as iterative development,collaboration, and flexibility. Discuss real-life examples and outcomes, focusingon how Agile improved project delivery, team communication, and customersatisfaction within the company.

Case Study: Implementation of the Agile Model in an IT Company

Introduction:

As the software industry continues to evolve, companies need to adopt adaptive and efficient processes for product development. The Agile model has emerged as a popular framework that allows businesses to respond quickly to market demands, improve team collaboration, and meet client expectations more effectively. This case study explores how an IT company successfully integrated the Agile model into their software development practices, resulting in significant improvements in project delivery, communication, and client satisfaction.

Company Background:

The IT company in this case specializes in creating customized web applications for a diverse client base. Initially, the company employed the Waterfall development model, which led to several challenges, such as project delays, difficulties in accommodating changes in client requirements, and issues with inter-team communication. These obstacles hampered the company's ability to deliver projects on time and caused friction with customers.

The company recognized the need to pivot to a more dynamic and customer-focused approach, leading to the decision to adopt the Agile model to streamline their processes and improve overall project performance.

Adoption of Agile Model

Core Agile Principles:

1. **Iterative Development**: Focus on delivering small, working components of the product in regular intervals.
2. **Collaboration**: Foster continuous communication among team members and maintain regular client feedback.
3. **Adaptability**: Encourage flexibility in responding to new requirements or unexpected changes during the project lifecycle.

Transition Process:

To ensure a smooth transition, the company adopted **Scrum**, a well-known Agile framework. The following practices were introduced as part of the transition:

5. **Cross-functional Teams**: Teams composed of developers, testers, and product managers worked together, enhancing collaboration.
6. **Daily Standups**: Teams held brief meetings to review progress, identify challenges, and set daily goals.

Strengthened Communication

Internal communication improved significantly with Agile practices. In the past, team members—developers, testers, and managers—often worked in isolation, leading to communication breakdowns. With Agile, daily standups promoted transparency within teams and enabled everyone to stay informed about the project's status.

During the development of an e-commerce platform, the testing team detected a critical bug and reported it during a standup meeting. This allowed the developers to resolve the issue quickly, within the same sprint. As a result, the company reduced overall development time by **30%**, demonstrating the positive impact of Agile's emphasis on communication.

Enhanced Customer Satisfaction

Agile's emphasis on frequent client interaction allowed the company to address customer needs more effectively. Clients participated in regular sprint reviews, providing feedback on features and ensuring the product aligned with their expectations. This frequent collaboration greatly improved the client's experience.

For example, during the development of a financial services platform, the client requested several new features mid-project. Agile's flexibility enabled the development team to incorporate these features quickly, ensuring the final product met all client requirements. The client expressed appreciation for the company's **adaptability**, which led to future collaboration opportunities.

Challenges and Solutions

Despite its many benefits, the transition to Agile came with its own set of challenges. Some team members found it difficult to adjust to shorter sprint cycles and the rapid pace of Agile development. Additionally, there was a learning curve when it came to estimating tasks and using Agile tools such as Jira.

To overcome these challenges, the company implemented several strategies:

1. **Agile Training Programs:** The company provided training to familiarize teams with Agile principles and tools.
2. **Agile Coaches:** Coaches were brought in to assist teams in understanding Agile practices during their first few sprints.
3. **Project Management Tools:** Tools like Jira were introduced to help teams plan and manage their sprints effectively.

PRACTICAL 2

Introduction:

Creating a facial mood detection model is a complex, evolving project. The success of such a project depends heavily on choosing the right software development methodology. Given the need for flexibility and continuous improvements in emotion recognition technology, the Agile model stands out as the most suitable process. This document will discuss why Agile is a better choice for this project compared to more rigid approaches like the Waterfall model.

Managing Project Complexity:
Facial mood detection involves multiple layers of complexity, such as gathering data, training machine learning models, testing on various facial datasets, and continuous refinement of the model. As new techniques or data emerge, it is essential to adapt the project scope in a way that allows for ongoing changes.

Agile's Iterative Development is ideal for managing such complexity, as it allows the project to be broken into smaller, manageable sprints. Each sprint can focus on refining particular aspects of the model, such as detecting a specific emotion. This enables the team to address complexity step-by-step, while regularly testing and refining the work done so far.

Incorporating User Feedback and Collaboration:
Since facial emotion detection models must align with real-world needs, incorporating user feedback throughout the development process is crucial. Understanding how the model performs in diverse environments and scenarios helps ensure it accurately recognizes various emotional expressions.

Agile emphasizes ongoing customer collaboration, ensuring that users and stakeholders provide feedback throughout the project. This ensures that the model evolves based on actual use cases and improves with each iteration.

Delivering Incremental Features:
Facial mood detection can be applied to various expressions, and it is essential to release functional components, such as detecting basic emotions like happiness or sadness, early in the development cycle. More complex features, like recognizing subtle emotional shifts, can be built over time.

Agile's Incremental Development allows for the delivery of early working versions of the model. This gives users access to a functional product while additional features are added over subsequent iterations.

Continuous Testing and Refinement:
Constant testing is a vital part of ensuring that the facial mood detection model remains accurate and effective, especially as new data and facial expressions are introduced.

Agile promotes continuous testing throughout the project. With each sprint, new data can be tested and integrated into the system, allowing the team to refine the model's accuracy and performance regularly.

Risk Management:
Facial emotion detection models face unique challenges, such as variations in expressions across different cultures. Delaying feedback until the end, as done in a Waterfall model, could lead to issues that are difficult to resolve.

Agile reduces risk through short development cycles and regular feedback. This allows the team to identify and address risks early, minimizing the likelihood of encountering major issues at the end of the project.

Software Project Management Plan (SPMP) for Facial Mood Detection Model

Project Scope and Objectives:

- Project Scope: The goal of this project is to build a system capable of detecting human emotions based on facial expressions. The primary focus will be on recognizing common emotions such as happiness, sadness, anger, and neutral expressions, excluding more complex psychological states.
- Objectives:
    1. Develop a reliable and scalable emotion recognition system.
    2. Complete the project within the established timeline and budget.
    3. Ensure the model meets performance and accuracy targets.

Project Milestones:

1. Requirements gathering and analysis.
2. Completion of the design phase.
3. Development of core emotion detection features.
4. Initial testing (unit and integration testing).
5. Final model development and integration.
6. User acceptance testing.
7. Project delivery and deployment.

Resource Allocation:

- Team Composition:
    - Software developers (Frontend and Backend)
    - UI/UX Designers
    - Quality Assurance engineers

  - Project Manager
  - Data Scientists specializing in model training
- Technologies & Tools:
  - Development: Git, Docker, IDEs
  - Project Management: Jira, MS Vision
  - Testing: Selenium, JUnit
- Budget Allocation:
  - Salaries for team members
  - Technology stack and cloud infrastructure for training and testing

Schedule:

- Gantt Chart: Create a Gantt chart that outlines timelines for phases such as design, development, testing, and deployment. Ensure task dependencies are well-defined to facilitate a smooth workflow.

Risk Management:

- Risk 1: Changing project requirements due to evolving emotion detection technology.
  - Mitigation: Establish a clear scope, and use a change management process to handle adjustments.
- Risk 2: Difficulty in recognizing subtle or culturally diverse facial expressions.
  - Mitigation: Use a diverse dataset and conduct ongoing testing with real-world scenarios.
- Risk 3: Budget overruns due to unanticipated challenges.
  - Mitigation: Closely monitor resource allocation and implement financial checkpoints.

Communication Plan:

- Internal Communication: Daily stand-ups and weekly progress reviews to ensure team alignment.
- External Communication: Regular reports and updates to stakeholders.
- Tools: Slack or Microsoft Teams for internal collaboration and Zoom for meetings with clients.

Quality Assurance:

- Quality Standards: Follow established coding practices, ensure security protocols, and meet accuracy benchmarks.
- Testing Procedures:

- Unit and Integration Testing to verify code quality.
- System Testing to validate model performance.
- User Acceptance Testing to confirm that the model meets end-user expectations.

PRACTICAL 3

AIM:

Software Requirements Specification (SRS) for Facial Mood Recognition System

1. Introduction

The **Facial Mood Recognition System** is designed to detect and classify the emotions conveyed through facial expressions in videos or images. The system will utilize advanced machine learning models to recognize a range of emotions such as happiness, sadness, anger, surprise, fear, and neutrality. It will serve users such as mental health professionals, HR departments, customer service, and general users who need to assess emotional states from facial expressions.

This document outlines the functional and non-functional requirements for developing a system that allows users to upload media (images/videos), process them through a trained emotion detection model, and receive results detailing the mood identified in the media.

Definitions, Acronyms, and Abbreviations:

- **Facial Mood Recognition:** Detecting human emotions based on facial expressions.
- **SRS:** Software Requirements Specification.
- **ML:** Machine Learning, used to identify emotional patterns.
- **API:** Application Programming Interface.
- **NLP:** Natural Language Processing, used for potential extensions (chatbot integration).
- **GDPR:** General Data Protection Regulation.

2. Overall Description

2.1 Product Perspective

The system will be a cloud-based platform designed for users to upload images or videos to detect emotions from facial expressions. It will be web-accessible and function independently, but it could integrate with third-party tools and services, such as external ML models or data visualization tools.

2.2 Product Functions

The system will offer the following functionalities:

- **User Registration and Authentication:** Users can sign up, log in, and manage their accounts.
- **Media Upload:** Users can upload images or videos in supported formats (JPEG, PNG, MP4, AVI) for analysis.
- **Facial Mood Detection:** The system will process media using ML algorithms to detect emotions.

- **Result Reporting:** The results will display the detected emotions along with confidence scores.
- **History Management:** Users can access a history of their uploads and analysis reports.
- **Premium Features:** Subscribed users will get access to additional features like larger file uploads, detailed analysis, and more emotions detected.
- **Payment Integration:** Premium services can be purchased through integrated payment gateways.

2.3 User Classes and Characteristics

- **General Users:** Individuals who want to detect emotions in media for personal or professional reasons. They will have access to standard features.
- **Premium Users:** Users who pay for enhanced features like larger file uploads, faster processing, and advanced emotion analytics.
- **Administrators:** Manage platform performance, monitor user activities, and handle system configurations.

2.4 Operating Environment

- **Platform:** Accessible via any major web browser (Chrome, Firefox, Safari) on desktops, tablets, and smartphones.
- **Cloud Infrastructure:** The system will use cloud-based servers like AWS or Azure to handle workload scalability.

2.5 Design and Implementation Constraints

- The platform must comply with GDPR and other applicable data protection regulations.
- File upload size will be limited to 1 GB for general users, with premium users allowed up to 5 GB.
- Algorithms must be optimized to handle varying image or video quality and lighting conditions.

3. Functional Requirements

3.1 User Registration and Authentication

- **User Sign-Up:** Users can create accounts using their email and a password. A verification email will be sent to confirm account registration.
- **User Login:** Users log in with their credentials. Secure sessions will be maintained using tokens.
- **Forgotten Password:** A password reset link will be sent to the registered email if the user forgets their password.

3.2 Media Upload

- **Supported Formats:** The system will accept common media formats (JPEG, PNG, MP4, AVI).

- **Upload Limit:** 1 GB for general users, up to 5 GB for premium users.
- **Error Handling:** If an unsupported file format or file size is uploaded, the system will notify the user.

3.3 Facial Mood Detection Analysis

- **Media Processing:** The system will use ML models to analyze media and detect facial expressions that correspond to emotions.
- **Report Generation:** The system will generate reports showing:
  - Detected emotions such as happiness, sadness, anger, fear, surprise, and neutrality.
  - A confidence score indicating the accuracy of the emotion detected.
- **Premium Analysis:** Premium users will have more detailed reports that include heatmaps highlighting the regions contributing to the detected emotions.

3.4 Results Management

- Users can view and download results in their dashboard.
- Reports can be exported in PDF format for record-keeping.

3.5 Payment Integration

- **Payment Gateways:** Payments for premium services will be processed through gateways such as PayPal or Stripe.
- **Subscription Model:** Monthly or yearly subscriptions can be selected for premium features.

4. Non-Functional Requirements

4.1 Performance Requirements

- The system must process media and return results within 5 minutes for a 500 MB file (standard users).
- Premium users should experience faster processing, under 2 minutes for the same file size.

4.2 Usability Requirements

- The platform will have a user-friendly interface with drag-and-drop upload functionality and clear navigation.
- **Accessibility:** The system must follow WCAG 2.1 guidelines for accessibility, supporting users with disabilities.

4.3 Reliability and Availability

- **Uptime:** The system must guarantee 99.9% uptime, with scheduled maintenance periods announced in advance.
- **Backups:** Daily backups will be performed to ensure data integrity.

4.4 Scalability

- The system should handle up to 10,000 concurrent users and scale dynamically as user traffic increases.

4.5 Security

- **Data Encryption:** All data transfers will be encrypted using SSL/TLS.
- **Password Protection:** User passwords will be securely hashed and stored.
- **Sensitive Data Handling:** Credit card details will be managed by secure payment processors.

5. External Interface Requirements

5.1 User Interface

- **Dashboard:** A clean and simple dashboard for users to manage uploads, view reports, and access their account settings.
- **Upload Interface:** A drag-and-drop feature for uploading media with a progress bar.
- **Results Display:** Reports showing detected emotions with visual indicators.

5.2 API Interface

- A RESTful API will be available for third-party developers to integrate the emotion detection service into their applications.

6. Security Requirements

6.1 User Data Protection

- All personal data (names, emails) will be encrypted and stored securely.
- The system will undergo regular security audits to ensure that user data is not compromised.

6.2 Multi-Factor Authentication (MFA)

- Premium and admin users will be required to enable multi-factor authentication for added security during login and payment processing.

7. Appendices

- **Appendix A:** UI/UX wireframes of the platform.
- **Appendix B:** API documentation for developers looking to integrate the emotion detection service into external platforms.

PRACTICAL 4

AIM :

Calculate cost and effort estimation for the given project definition using FP calculation (Simple, Average, and Complex) and the COCOMO model. Consider Object Oriented Programming as the language for:

Deepfake video detection:

Estimate the cost and effort required to develop an e-commerce website by applying Function Point (FP) analysis for different complexity levels (Simple, Average, and Complex). Additionally, use the COCOMO model to further refine the estimation. This includes considering various factors like project size, team experience, and development environment, specifically focusing on object-oriented programming techniques.

Cost and Effort Estimation for Facial Mood Recognition Model

1. Introduction

This document outlines the cost and effort estimation for developing a **facial mood recognition model** using Function Point (FP) analysis and the COCOMO model. The project aims to implement a tool that can recognize and classify facial moods using object-oriented programming (OOP). We will apply FP analysis to calculate the effort required for simple, average, and complex functionalities, followed by the COCOMO model to further refine the cost and effort estimates based on factors such as project size, developer experience, and the development environment.

2. Project Definition

The facial mood recognition model will include the following features:

- **User authentication and profile management**.
- **Image or video analysis using machine learning algorithms**.
- **Real-time facial mood detection**.
- **Reporting and analytics for detected moods**.
- **Integration with external APIs for image/video processing**.
- **Security features like data encryption and privacy compliance**.

The project will follow object-oriented programming principles, emphasizing maintainability, scalability, and reusability.

3. Function Point (FP) Analysis

FP analysis estimates the size and complexity of a software project by considering the number of inputs, outputs, data files, user interactions, and external interfaces. Each component is categorized as simple, average, or complex.

Function Points by Complexity Level

Category	**Simple**	**Average**	**Complex**
External Inputs	3 FP	4 FP	6 FP
External Outputs	4 FP	5 FP	7 FP
External Queries	3 FP	4 FP	6 FP
Internal Logical Files	7 FP	10 FP	15 FP
External Interface Files	5 FP	7 FP	10 FP

FP Calculation Example

The facial mood recognition model includes the following estimated counts for each category:

- **External Inputs**: 5 (average complexity)
- **External Outputs**: 4 (complex complexity)
- **External Queries**: 3 (average complexity)
- **Internal Logical Files**: 6 (simple complexity)
- **External Interface Files**: 3 (average complexity)

Based on the table above, we calculate:

- **External Inputs**: 5 * 4 FP = 20 FP
- **External Outputs**: 4 * 7 FP = 28 FP
- **External Queries**: 3 * 4 FP = 12 FP
- **Internal Logical Files**: 6 * 7 FP = 42 FP
- **External Interface Files**: 3 * 7 FP = 21 FP

Total FP Calculation

Total FP = 20 + 28 + 12 + 42 + 21 = **123 FP**

4. COCOMO Model

The Constructive Cost Model (COCOMO) is used to estimate the cost and effort based on project size. For this analysis, we use the **Basic COCOMO model**, which calculates effort (in person-months) and development time based on lines of code (LOC).

Basic COCOMO Formula

- **Effort (E)** = a * (KLOC)^b
- **Development Time (Tdev)** = c * (Effort)^d

Where:

- **KLOC** = Thousands of Lines of Code
- **a, b, c, d** are constants based on project type:
  - **Organic** (small, simple projects): a = 2.4, b = 1.05, c = 2.5, d = 0.38
  - **Semi-detached** (medium-sized): a = 3.0, b = 1.12, c = 2.5, d = 0.35
  - **Embedded** (complex, large projects): a = 3.6, b = 1.20, c = 2.5, d = 0.32

Effort Estimation (Example)

Assuming the facial mood recognition model is a semi-detached project with an estimated **20 KLOC**, we apply the formula:

- **Effort (E)** = 3.0 * (20)^1.12 = 3.0 * 25.30 ≈ **75.9 person-months**
- **Development Time (Tdev)** = 2.5 * (75.9)^0.35 = 2.5 * 4.57 ≈ **11.4 months**

Cost Estimation

To estimate the cost, we multiply the effort by the average salary of a developer. Assuming the average developer cost is **$8,000 per month**:

- **Cost** = Effort * Average Monthly Salary = 75.9 * $8,000 = **$607,200**

5. Final Estimations

FP-Based Effort Estimation

Based on FP analysis, the project requires approximately **123 function points**. With a rough estimate of **10 FP per person-month**, the effort is:

- **Effort** = 123 / 10 = **12.3 person-months**

COCOMO-Based Estimation

The COCOMO model provides a refined estimate based on the project size. The calculated effort is **75.9 person-months**, with a development time of **11.4 months**.

Comparison of FP and COCOMO Estimations

- **FP-Based Effort**: 12.3 person-months
- **COCOMO-Based Effort**: 75.9 person-months
- **Estimated Development Cost (COCOMO)**: **$607,200**

These estimations provide insight into the development effort and cost, with the COCOMO model accounting for more detailed factors such as project scale and complexity.

PRACTICAL 5

AIM:

To perform the project management task for a given software project application Deepfake video detection using the JIRA Tool . Identify and describe five project management tools, emphasizing their features and use cases. Provide an in-depth explanation of JIRA, supported by relevant figures, detailing its capabilities in project management, agile methodologies, and customization options.Free Trial Setup: Set up a free trial of JIRA to familiarize yourself with its interface and features. Create an Epic: Learn how to create an epic, which is a large body of work thatcan be broken down into smaller tasks.Create an Issue: Understand the process of creating an issue, which represents a single unit of work within JIRA.Create Story: Explore how to create user stories, which are short descriptions of a feature or functionality from the end-user perspective. Create Subtask: Break down stories or tasks into smaller, manageable subtasks. Create Sprint: Set up a sprint, a time-boxed period during which a specific set of tasks must be completed, and manage the sprint within JIRA to track progress andcompletion.
Project Management Tools Overview

1.JIRA

Features:

JIRA is a powerful tool widely used for issue tracking and project management, particularly in Agile environments. It supports Scrum and Kanban boards, backlog prioritization, sprint planning, and detailed reporting.

Use Case: Ideal for managing software development projects, JIRA helps teams track bugs, stories, tasks, and epics. It integrates with CI/CD tools, enhancing its capabilities in DevOps pipelines.

2. Trello

Features: Trello uses boards, lists, and cards to manage tasks visually. It allows easy dragand-drop task management, task assignments, due dates, checklists, and file attachments.

Use Case: Suitable for smaller teams and simpler project management needs, Trello excels in task tracking, brainstorming, and individual task management.

3. Asana

Features: Asana provides task assignments, project timelines, workspaces, and customizable workflows. It supports task dependencies, milestones, and workload management, making it easy to track project progress.

Use Case: Great for cross-functional teams, Asana is used in project planning, resource management, and task coordination across various industries.

Microsoft Project o Features: A robust tool for project planning, resource allocation, and budgeting. It offers Gantt charts, task dependencies, and project tracking.

6.Monday.com

Features: Offers a highly customizable interface with boards, timelines, charts, and automation rules. It integrates with other tools like Slack, Google Drive, and JIRA.

Use Case: Suitable for project management, marketing campaigns, sales pipelines, and team collaboration, providing a visual and intuitive way to manage tasks.

In-Depth Analysis of JIRA

1.Capabilities in Project Management

Issue Tracking: JIRA's core capability is issue tracking. It allows the creation of tasks, bugs, stories, and epics, which can be assigned, tracked, and prioritized.

Sprint Planning: For Agile teams, JIRA provides robust sprint planning tools, enabling teams to set up sprints, manage backlogs, and visualize workload.

Roadmaps: JIRA's roadmaps feature provides a high-level view of project progress, helping teams plan, track, and communicate with stakeholders.

Reporting and Analytics: JIRA offers detailed reporting features such as burn-down charts, sprint reports, velocity charts, and cumulative flow diagrams, which help track team performance and identify areas for improvement.

2. Agile Methodologies

Scrum Support: JIRA allows teams to create Scrum boards, manage sprint cycles, and visualize sprint progress, making it an essential tool for Agile teams focusing on iterative development.

Kanban Boards: JIRA's Kanban boards offer a visual representation of work in progress, helping teams manage workflows, set WIP limits, and identify bottlenecks in real-time.

Backlog Grooming: Product owners and managers can easily prioritize backlogs, reorder tasks, and plan upcoming sprints to ensure a steady flow of work.

3. Customization Options

Custom Workflows: JIRA allows the creation of customized workflows that match the team's unique processes. It supports custom states, transitions, and automation rules.

Custom Fields and Screens: Users can define custom fields to capture specific data and create screens to control how issues are displayed.

Plugins and Integrations: JIRA integrates with a wide array of tools, such as Confluence, Bitbucket, Slack, and GitHub, expanding its capabilities and fitting into existing tech stacks

Automation: JIRA's automation engine allows teams to set up rules for repetitive tasks, reducing manual effort and minimizing errors. Relevant Figures Illustrating JIRA's **Capabilities:**

Figure 1: Scrum Board in JIRA: Displays how tasks are organized into sprints, with columns representing stages of work.

Figure 2: Roadmap View: Shows the timeline of epics and tasks, providing a visual representation of project progress.

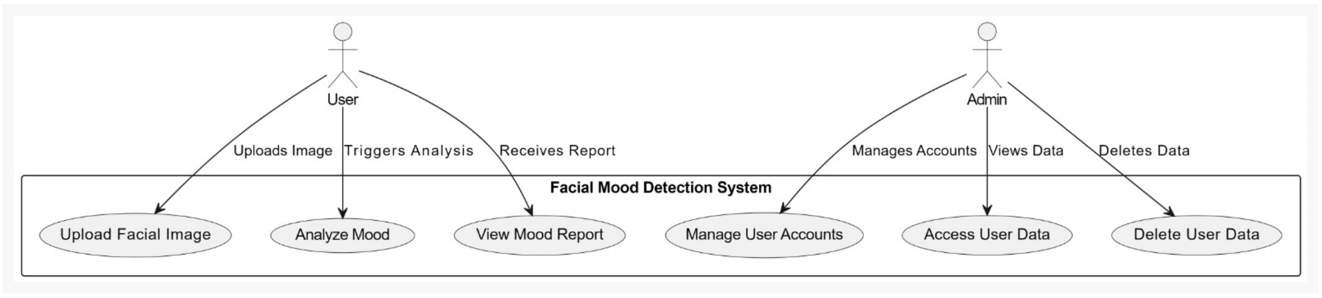
Figure 3: JIRA Dashboard: A customizable interface where team members can track key metrics, monitor sprint progress, and get quick insights.

PRACTICAL-6

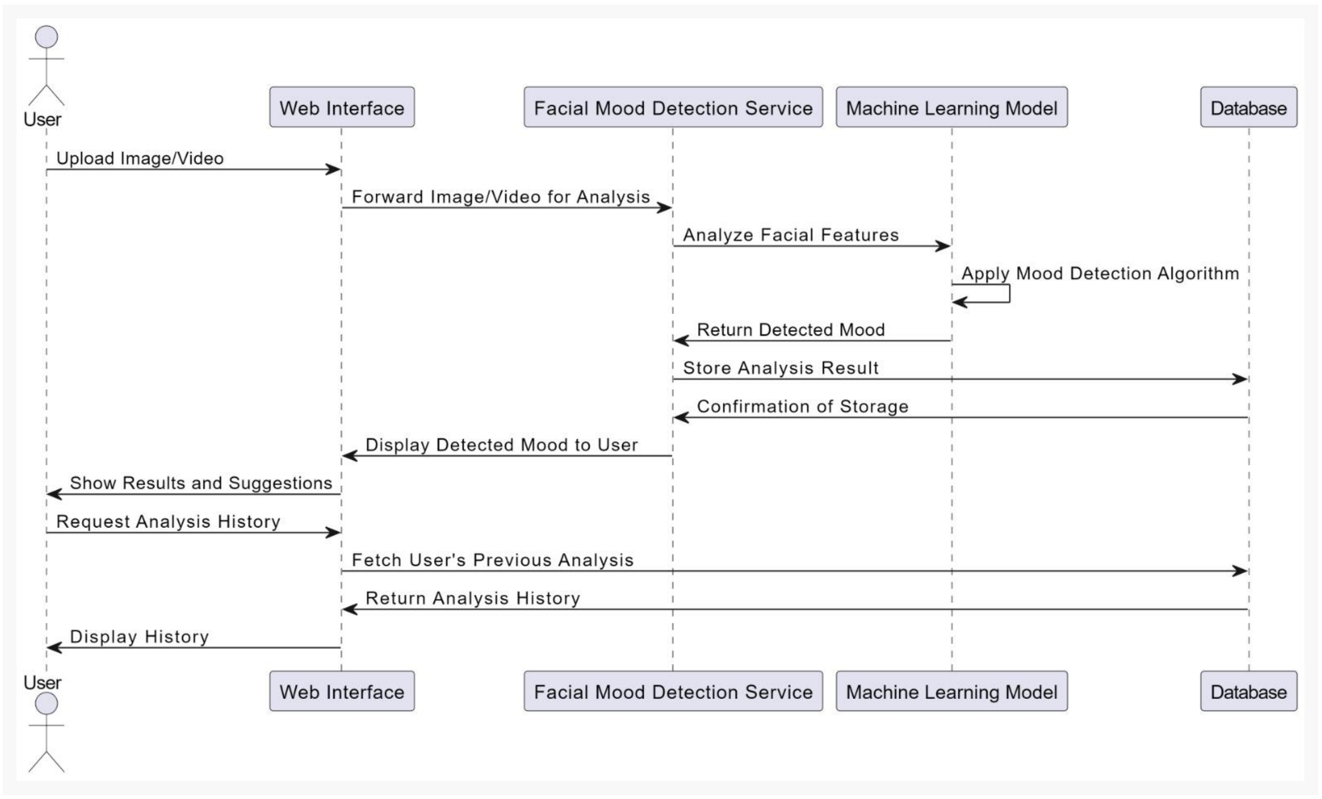
AIM:

Draw the UML diagrams for the deepfake video detection using object-oriented concepts with MS VISIO, MS Project,or any other online tool:

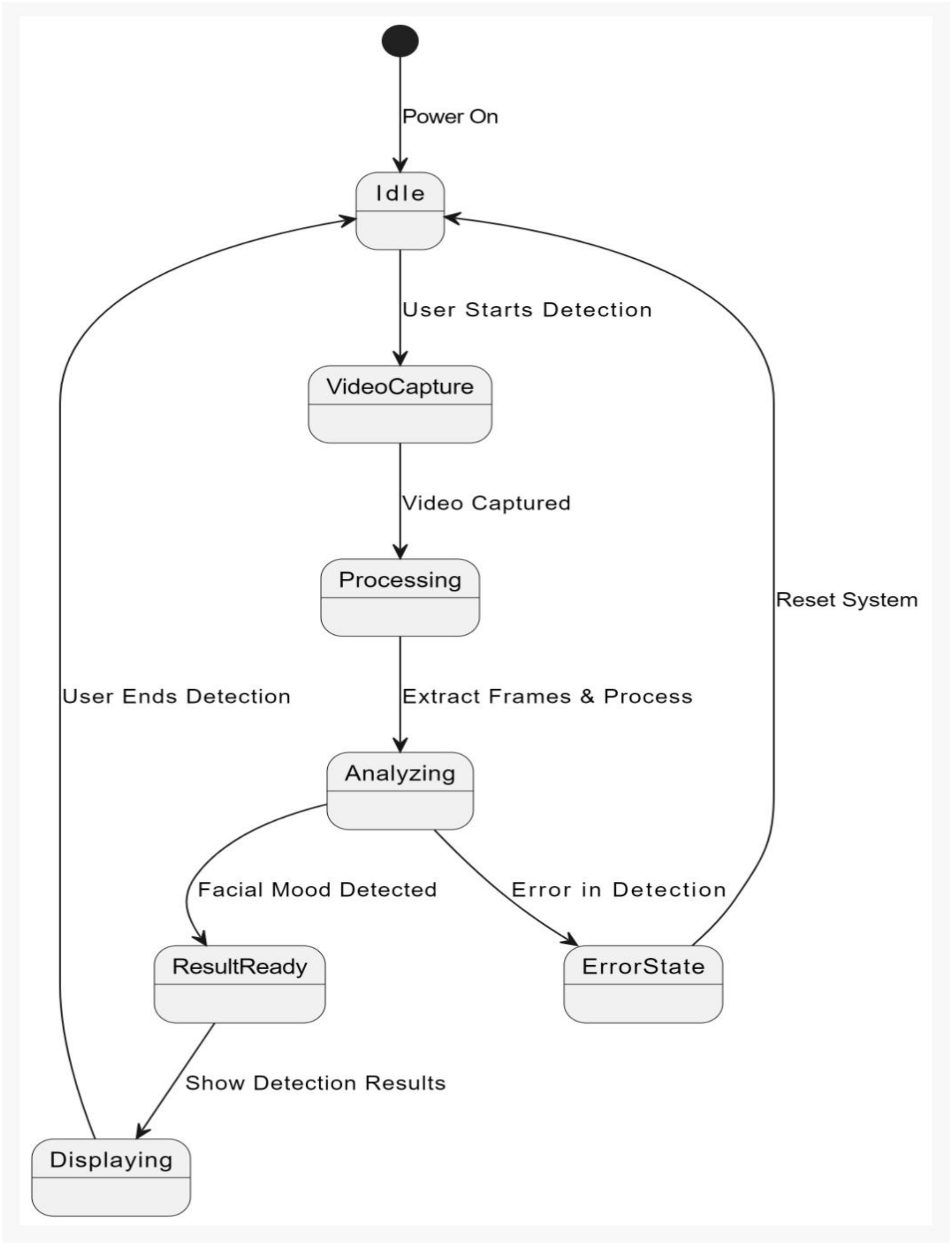
Use Case Diagram:



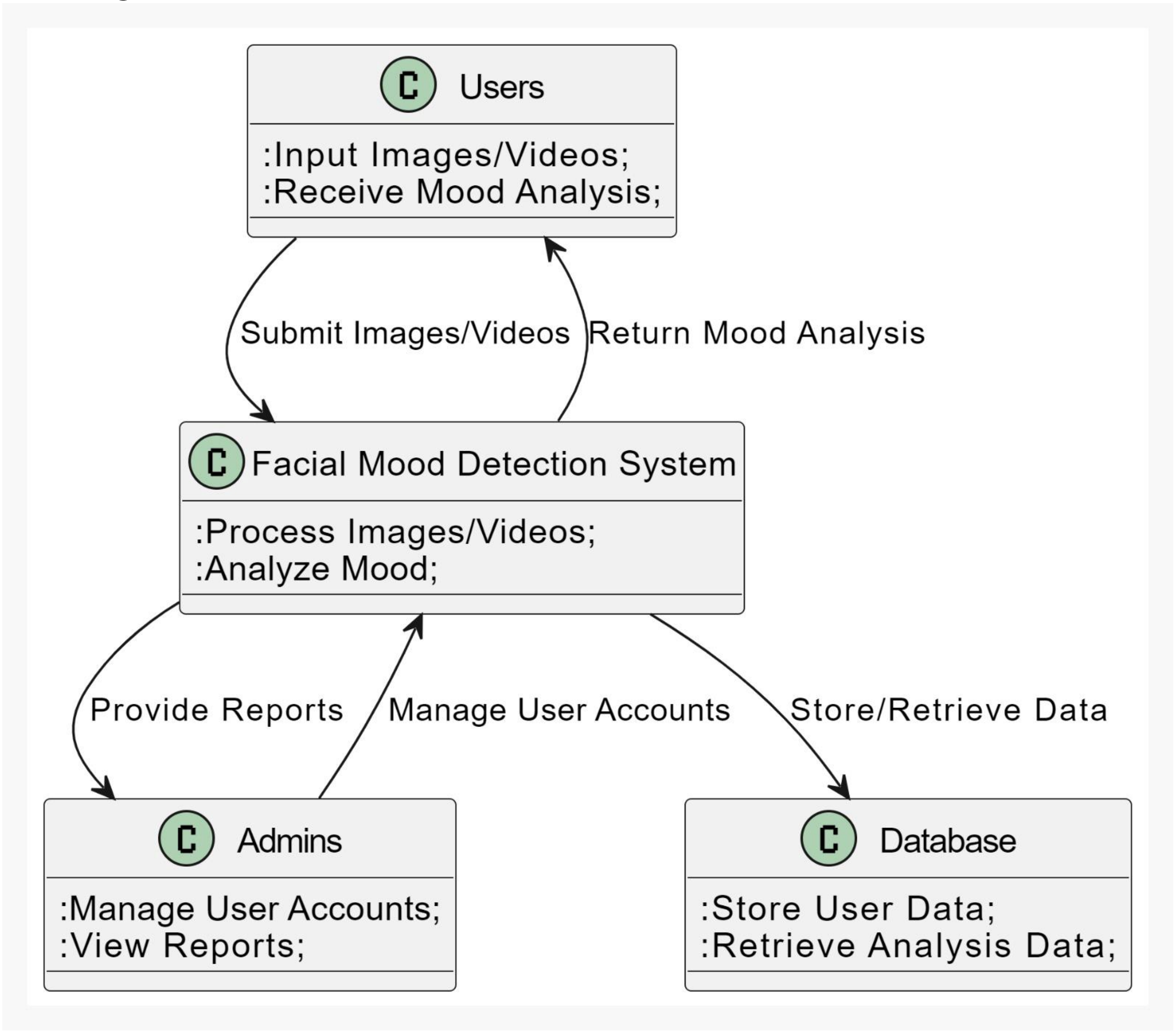
Sequence Diagram:



StateChart or Activity Diagram:



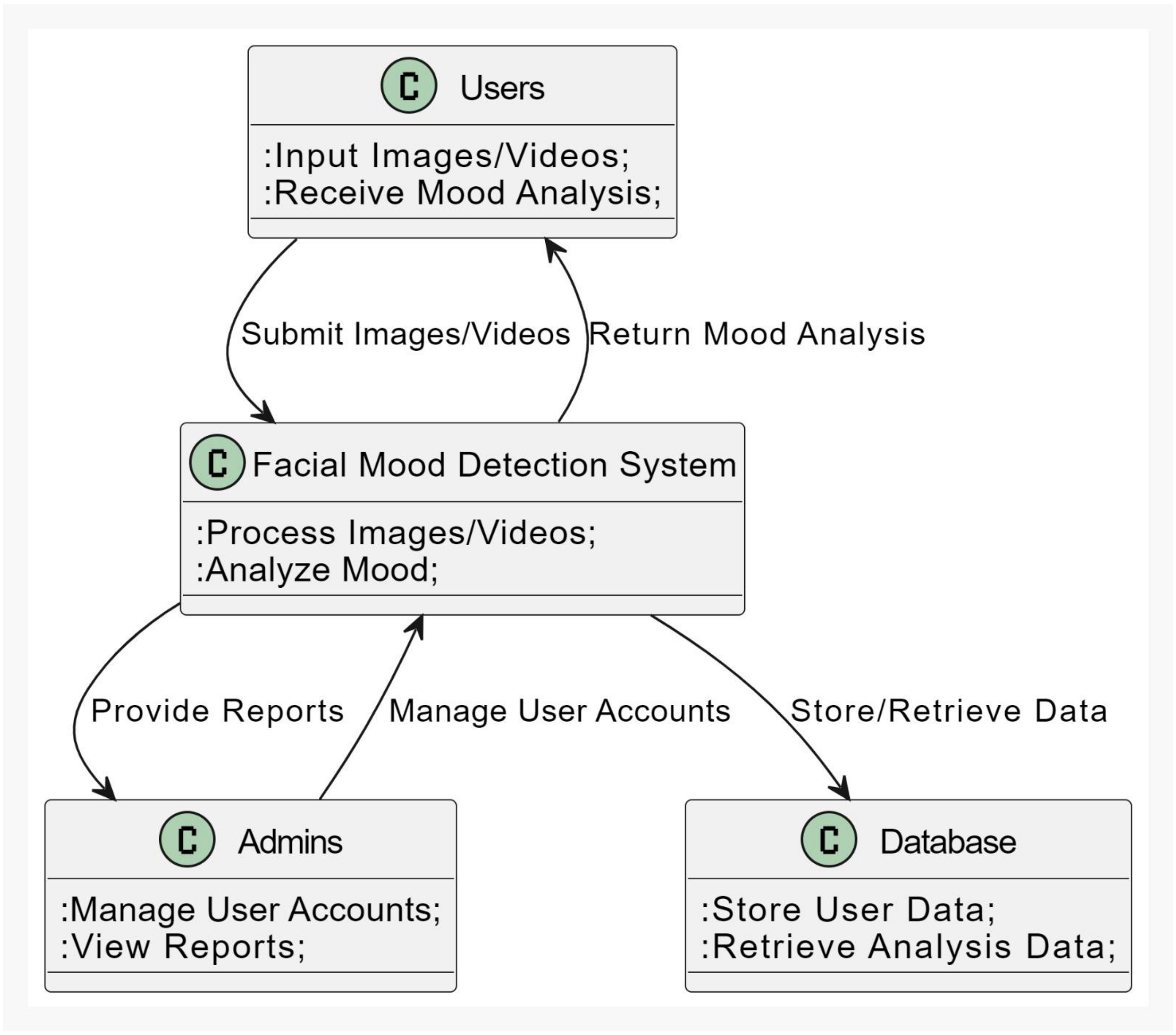
Class Diagram:



PRACTICAL-7

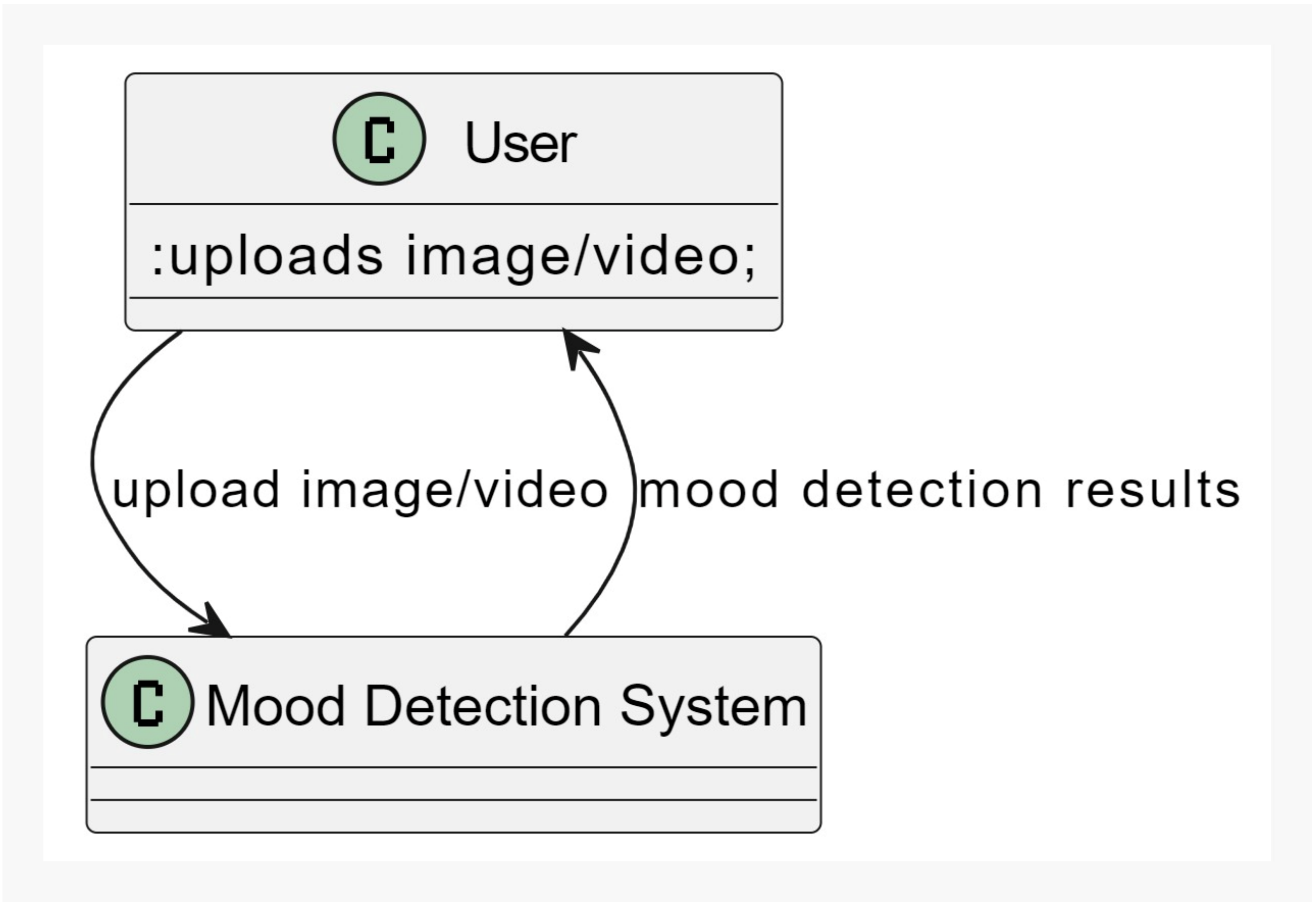
Draw diagrams for the given definition deepfake video detection using a Functional Oriented Approach:

Context Level Diagram:

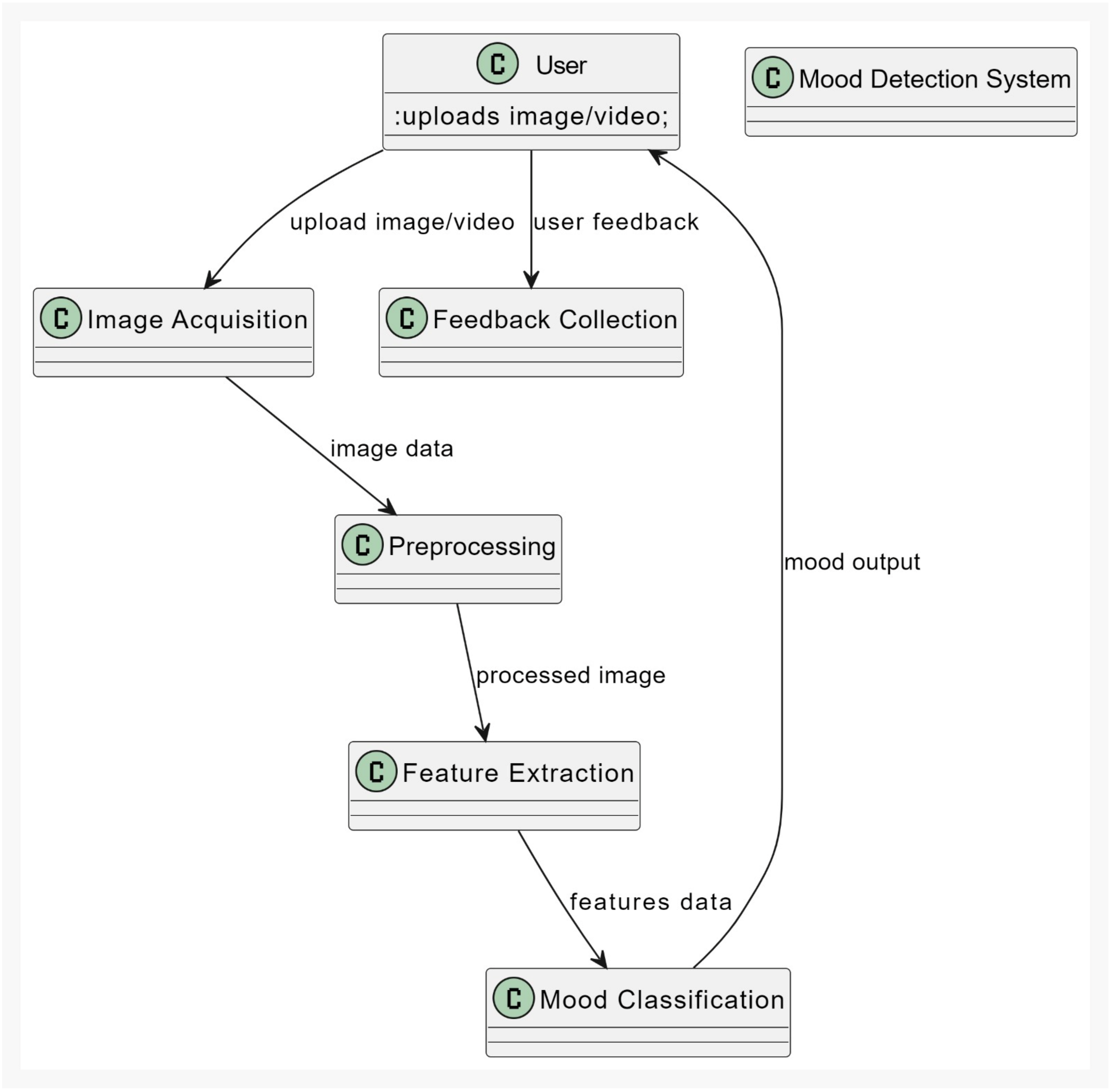


All Level DFD (Data Flow Diagrams):

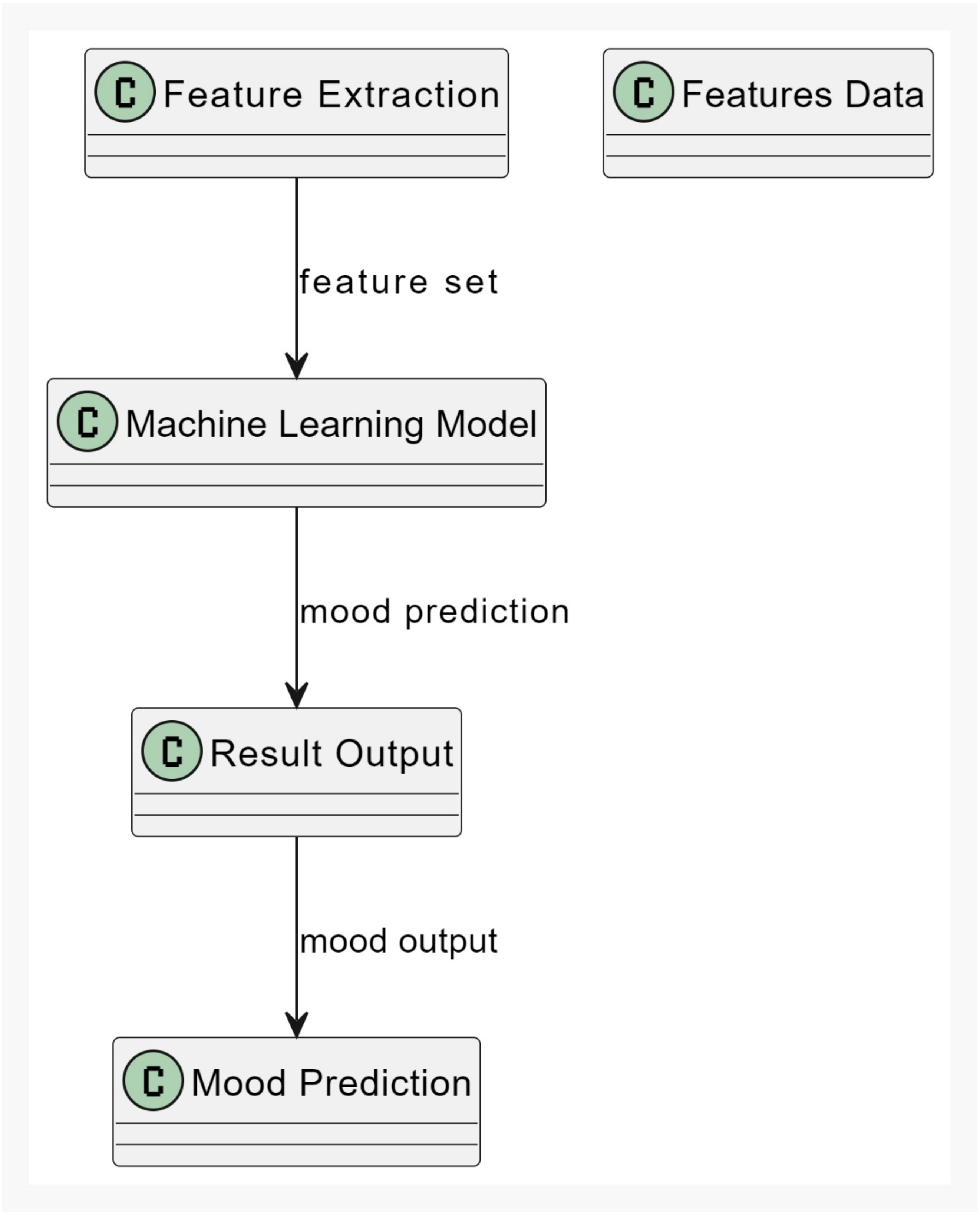
Level 0 DFD


User
:uploads image/video;
upload image/video
mood detection results
Mood Detection System

Level 1 DFD:



Level 2 DFD:


Feature Extraction
Features Data
feature set
Machine Learning Model
mood prediction
Result Output
mood output
Mood Prediction

PRACTICAL-8

Test Plan for Project: Facial Mood Recognition

Test Case
Test Case ID: TC001
Test Case Name: User Image Upload Functionality
Objective: Verify that users can successfully upload an image for facial mood analysis.
Preconditions: The user must be authenticated and have a valid image file (e.g., jpg, png).
Steps:

1. Log into the system using valid credentials.
2. Navigate to the "Upload Image" section.
3. Select a valid image file from local storage.
4. Click the "Upload" button.
5. Wait for the system to confirm successful upload.
   Expected Result:

- The image is uploaded to the system.
- The system displays a success message confirming the image was successfully uploaded.
- The uploaded image is stored in the designated database.
  Postconditions:
- The uploaded image file is saved for mood analysis.
- The system proceeds with the mood analysis automatically.

Test Case ID: TC002
Test Case Name: Facial Mood Detection Accuracy
Objective: Ensure that the system accurately detects and classifies the mood from the uploaded image.
Preconditions: An image has already been uploaded and processed for analysis.
Steps:

1. Upload an image with a known mood expression.
2. Start the mood detection analysis.
3. Wait for the analysis to complete.
4. Review the system's results to check the detected mood.
   Expected Result:

- The system should correctly identify the mood (e.g., happy, sad, angry).
- The detection accuracy should meet the predefined performance metrics (e.g., 90% accuracy).
  Postconditions:
- The result should be stored for future reference and reporting.

Test Suite
Test Suite Name: Image Processing Suite
Objective: Test all functionalities related to image uploading, analysis, and result retrieval in the facial mood recognition project.
Test Cases Included:

- TC001: User Image Upload Functionality
- TC002: Facial Mood Detection Accuracy
- TC003: Mood Result Retrieval
- TC004: Image Deletion by Admin

- TC005: Error Handling for Invalid Image Formats

Execution Sequence:

1. Start by testing the image upload functionality (TC001).
2. Proceed with facial mood detection accuracy (TC002).
3. Test result retrieval and viewing (TC003).
4. Test admin functionality for deleting inappropriate images (TC004).
5. Finally, check error handling for unsupported image formats (TC005).

Overall Test Suite Status: Complete when all test cases pass.

Testing Strategy

1. Scope of Testing:
   The primary goal is to validate all features of the *facial mood recognition* system. This includes:
   - User authentication and image upload.
   - Accurate mood detection from images.
   - Display and storage of mood results.
   - Admin functions, including system settings and image moderation.
2. Testing Levels:
   - Unit Testing: Test individual components (e.g., functions for mood analysis, user authentication).
   - Integration Testing: Ensure different components (e.g., image analysis, result display, and user authentication) work together seamlessly.
   - System Testing: Test the complete system by simulating real-world use cases.
   - Acceptance Testing: Validate that the system meets the overall requirements and is ready for deployment.
3. Test Environment:
   The system will be tested in a controlled environment that closely resembles the production environment. This includes:
   - A test database populated with sample images.
   - A web interface for user and admin access.
   - Backend processes (e.g., AI-driven facial mood analysis).
4. Testing Tools:
   - Unit Testing Framework: JUnit (for Java-based code) or PyTest (for Python-based code).
   - Performance Testing Tools: Apache JMeter to test system load and response times.
   - Defect Tracking Tools: JIRA for bug tracking and test case management.
5. Roles and Responsibilities:
   - Test Manager: Oversees the testing process and ensures deadlines are met.
   - Test Engineers: Write and execute test cases, identify bugs, and document results.
   - Developers: Collaborate with test engineers to resolve identified defects.
   - Product Owner: Reviews the test results and approves the final release of the system.

Unit Testing

Unit testing focuses on verifying individual components of the *facial mood recognition* system to ensure they function correctly.

Example Unit Test Case 1: Image Upload Function

- Objective: Test the function responsible for uploading image files.
- Test Method:
  - Mock the file upload request.

  - Pass valid and invalid image formats (e.g., jpg, png, pdf).
  - Ensure the function handles both cases correctly.
- Expected Output:
  - For valid formats, the function should store the file and return a success message.
  - For invalid formats, it should return an error message.

Example Unit Test Case 2: Mood Detection Algorithm

- Objective: Test the function responsible for analyzing facial moods in images.
- Test Method:
  - Pass pre-analyzed images (with known mood outcomes).
  - Ensure the function correctly identifies the mood (e.g., happy, sad, angry).
- Expected Output:
  - The function should return results that match the known outcomes with a high level of accuracy.

Execution of Unit Testing:

1. Set up a mock environment where individual components can be tested in isolation.
2. Execute the tests using the unit testing framework (e.g., JUnit or PyTest).
3. Validate outputs against the expected results for each unit.
4. Log any defects or errors for developer review.