Practical-01

Aim: Project Definition, Objective of the specified module & perform required engineering.

Project Name: Face Detection App

Definition: The **Face Detection App** is an application that leverages computer vision algorithms to detect and, optionally, recognize faces in images or videos. It will be extended to include features like content aggregation, search functionality, user interaction, security, and privacy, among others. This means users will be able to interact with a database of images, comment or rate these images, and search for specific content based on their preferences.

Objective of the Specified Module:

- **Content Aggregation:**
    - **Objective**: The app will gather content, such as images or video clips, from various sources to make them available to users for face detection.
    - The content can be tagged with metadata like image categories, emotions detected in faces, or specific people (if recognition is included).
- **Search Functionality:**
    - Objective: Users will be able to search for content based on specific criteria, such as:
    - Keywords (e.g., "happy face", "family photo").
    - Facial attributes (e.g., searching for images with smiling faces).
    - Date of upload or source (e.g., "Uploaded yesterday", "Photos from gallery")
- **User Interaction:**
    - **Objective:** The app will allow users to upload, view, and interact with images and video streams. Interaction features include**:**
    - Uploading images from their device
    - Viewing detected faces in real-time or after processing.

- Rating or commenting on detected faces.

➢ **Security & Privacy:**

- Security Measures:
- **User Data Encryption**: Encrypt sensitive data such as user information or facial recognition data.
- **Secure Image Upload**: Ensure uploaded images are transmitted via **HTTPS** to avoid man-in-the-middle attacks.
- **Face Data Protection**: For privacy, ensure that any facial data (e.g., from recognition) is anonymized unless explicit consent is obtained. It's essential to adhere to privacy standards like **GDPR** (General Data Protection Regulation) or **CCPA** (California Consumer Privacy Act).

➢ **Features:**

- **User Consent**: Ask for user consent before storing or processing any personal data, such as facial data.
- **Data Anonymization**: If recognition is implemented, ensure that the system doesn't store personal identifiable information unless explicitly needed.
- **Delete Functionality**: Users should be able to delete their uploaded images and any associated data from the system at any time.
- **Face Detection**: Detect faces in images and videos with high accuracy.
- **Face Recognition (Optional)**: Identify individuals from a set of known faces (optional for certain use cases).

➢ **Rating and Commenting:**

- **Objective**: Allow users to rate or comment on the images/videos processed by the app
- **Rating System**: Use a star rating system or thumbs up/down to rate the images.
- **Commenting**: Users can leave comments on images, which can be seen by other users.

➢ **Software Requirements for the Face Detection App**

- Frontend (User Interface)
- Mobile (Android/iOS)
- **Android**: Java/Kotlin using Android Studio

- **iOS**: Swift using Xcode.
- **Libraries/Tools**: OpenCV, TensorFlow Lite (for model inference), face-api.js (for web).
- **Backend Frameworks**
- **Flask/Django** (Python) or **Node.js** for building REST APIs.
- **Express.js** (for Node.js-based backend).
- **Database:**
- **Relational Database**: MySQL, PostgreSQL, or **Firebase** for storing images, user profiles, ratings, and comments.

➢ **Conclusion:**

The **Face Detection App** is designed to provide users with a powerful tool for detecting faces and analyzing emotions. The app offers various features such as real-time face detection, emotion recognition, and content management capabilities. With user-centric design, flexible access for different user roles, and a secure environment, the app aims to serve both personal and research purposes

Practical 02

Aim: Identify a Suitable Design and Implementation Model from Different Software Engineering Models for the development of a Face Detection App.

➢ **Introduction:**
Developing a **Face Detection App** involves creating software that can identify and process human faces from images or video streams. Such applications often leverage **machine learning algorithms**, **image processing techniques**, and **user interactions** (such as **camera integration** and **real-time processing**). Due to the dynamic nature of face detection technology and the frequent need to refine algorithms, an iterative approach is often preferred for building such apps. One highly suitable development model for this kind of project is the **Prototype Model** in software engineering.

➢ **Software Engineering Prototype Model:**
The **Prototype Model** is a software development approach that emphasizes creating early versions of the software to test functionality and gather user feedback. Unlike traditional waterfall methods, it enables iterative development, where a prototype is built, tested, and improved upon repeatedly until the final product is achieved. This process helps users visualize the app's features early on and provides insights into potential improvements.

➢ **Prototype Model Breakdown**

- **Build a Prototype**
- The first step in the prototype model is to create a **basic version** of the Face Detection App with **core functionality**—such as basic face detection capabilities using pre-built machine learning models, an interface for uploading or capturing images, and simple results displaying recognized faces
- **Get Feedback:**
- Once the prototype is developed, it's released to a **target group of users** for feedback. This feedback is invaluable for understanding user expectations,

uncovering issues with the face detection accuracy, the usability of the app, and the performance of the prototype.

- **Improve the prototype:**
- **Improving algorithms** for better face detection or incorporating new machine learning models. Enhancing the **user interface** to make the app more intuitive.

➢ **Final Version:**

- After several iterations and refinements, the prototype evolves into the **final version** of the Face Detection App. By this stage, the core face detection functionality is working optimally, and additional features like user feedback systems, emotional recognition, and integration with social media platforms might be added.

➢ **Maintain**:

Post-deployment, the app will enter the **maintenance phase**, where bug fixes, updates, and improvements will be made. This phase is essential in addressing any issues that arise after the app is live, such as bug fixes related to new face detection techniques or improving security features.

➢ **Advantages of the prototype model:**

- User-Centric Development:
- The prototype model allows for **active involvement of users** from the very beginning. By receiving feedback early and often, the development team can ensure that the app meets the needs and expectations of its users. For example, if users report that the app doesn't detect faces well in certain lighting conditions, the team can address that issue before proceeding to the final version.
- Early Problem Detection:
- **Face detection errors** due to low-quality images.
- Performance problems (e.g., app slowdowns in real-time video processing).
- **Reduced Risk of Failure**:
- By involving **users early** and getting regular feedback, the development team can reduce the risk of building an app that doesn’t meet user

expectations or business goals. If users don't like the prototype or if they experience major issues, the team can refine and adjust accordingly.

- **Flexibility in Incorporating Changes**:
- Unlike traditional methods (e.g., Waterfall), the Prototype Model is highly flexible. Developers can **add features** (such as additional facial recognition capabilities, security features, etc.) as the app progresses through iterations.

➢ **Disadvantages of the Prototype Model:**

- Changing Requirements:
- The Prototype Model encourages **continuous iteration**, which can result in **changing requirements** as the app evolves. While flexibility is a key advantage, it can also lead to **scope creep**, where the scope of the app expands beyond the original vision.
- Frequent changes based on user feedback might lead to **increased development time** and the need for more resources
- Resource Intensive:
- Building and testing prototypes can be **resource-intensive**, requiring substantial **time**, **personnel**, and **technology** to continuously improve the app.
- Lack of Comprehensive Documentation:
- The rapid changes and constant iteration can sometimes result in **insufficient documentation**. As new features and updates are added, documentation may become outdated or unclear, which can create difficulties for maintaining the app in the long term.

➢ **Features:**

- Face Detection Algorithm:
- Core feature that uses machine learning algorithms (such as **Haar Cascades**, **HOG**, or **Deep Learning-based models**) to identify faces from images or video streams
- User Interface:
- Simple and intuitive UI allowing users to upload images or take photos using the device's camera
- Real-Time Processing:

- Capability to detect faces in **real-time** video or camera feeds, providing immediate results.

- **Security & Privacy:**
- **Data encryption** for protecting user images and facial data
- **Privacy measures** to ensure that images are not stored or misused
- **Integration with Other Features:**
- **Emotion recognition** or **age estimation** based on facial features
- Option to save or share detected faces

➢ **Conclusion**

- For the development of a **Face Detection App**, the **Prototype Model** is a highly suitable approach. It supports **rapid development**, **continuous feedback**, and **early problem detection**, which are crucial for refining complex technologies like machine learning-based face detection. By building and improving prototypes iteratively, the development team can ensure that the app is user-centric, reliable, and flexible enough to incorporate new features as needed.

Practical 03

Aim: Prepare Software Requirement Specification (SRS) for the Face Detection App.

- **Introduction**
  - The **Face Detection App** is designed to allow users to upload images or use real-time video feeds to detect and recognize faces in various scenarios. The app utilizes advanced computer vision techniques and machine learning models to detect faces accurately, enabling users to interact with the application for various purposes like user authentication, emotion recognition, and more. The app will feature multiple levels of user access and provide content moderation and administrative control to maintain the integrity and privacy of the data.
- **Scope:**
  - The **Face Detection App** aims to provide a reliable platform for users to upload and process images or video in real-time, detecting human faces in the visual content. The application will offer a range of functionalities, including but not limited to face recognition, emotion detection, and real-time face tracking.
  - Key Features:
  - Upload and capture images for face detection
  - Real-time face detection from video feeds
  - Emotion recognition based on facial expressions
- **Description**
  - Production Functions:
  - Face Detection: The primary function of the app will be detecting faces in uploaded images or live video streams. The app will utilize advanced machine learning algorithms for accurate and fast face detection.
  - Case Study Upload
  - Users can upload images or videos to the app to analyze and detect faces. The uploaded media will be processed, and detected faces will

be displayed with additional data like emotion analysis and confidence levels.

- Search And Browse
- Users can search for previously uploaded content using various filters (e.g., date of upload, emotions, etc.). The search function will allow efficient retrieval of media by metadata associated with faces detected.
- Content Moderation and Admin Control
- Admin Control: The admin will have access to a backend interface to monitor content uploaded by users. Admins can approve, reject, or flag inappropriate content based on predefined criteria (e.g., nudity, offensive material).

➢ **User Characteristics**

- **Contributors**: These are users who contribute images or videos to the app for face detection and analysis. Contributors are typically researchers, students, or general users looking to analyze facial data for various purposes.
- **User** (Researchers, Students): These users may upload images and receive analysis results (e.g., face detection, emotion recognition). They can interact with the content and download results.
- **Moderators**: Moderators are responsible for reviewing the content uploaded by contributors to ensure it complies with app guidelines. They can flag inappropriate content and assist in the content moderation process.

➢ **Functional Requirements:**

- Face Detection Functionality
- The system should detect faces in uploaded images and video feeds using deep learning-based algorithms.
- The app should return the number of faces detected, along with each face's bounding box and confidence level.
- Emotion Recognition:
- The system should identify emotional expressions from faces (happy, sad, neutral, etc.) with high accuracy.
- The app should display the detected emotions alongside the faces.

- User Authentication:
- Users must be able to sign up and log in to the app using email or other authentication methods (social media integration).
- Admins should be able to manage user accounts and reset passwords if necessary
- Search and Retrieval
- Users should be able to search for previously uploaded images or videos based on metadata (e.g., facial emotion, date of upload).
- Feedback and User Interactions:
- Users should be able to provide feedback about the app’s performance and suggest improvements.

➢ **Content Management**

- **Search and Retrieval:**
- Efficient search functionality allowing users to filter uploaded content based on metadata such as the number of faces detected, emotions, or other relevant criteria.
- **User Feedback**
- A feedback mechanism where users can rate the app's face detection and emotion recognition accuracy. Feedback will be used to improve the app’s functionality.
- Security Requirements:
- **Data Encryption**: All sensitive data, including images and videos, will be encrypted both during transmission and at rest.
- **Authentication and Authorization**: Secure user authentication will be implemented to ensure that only authorized users can access certain app features.

➢ **Assumption and Dependencies:**

- The app assumes that the user has access to a **camera-enabled device** (smartphone, tablet, or computer) for face detection in real-time video
- The app relies on pre-trained **machine learning models** for face detection and emotion recognition. These models must be updated periodically to improve accuracy.

- The app depends on **cloud storage** for handling large media files (images and videos)

- **Conclusion:**
  - The **Face Detection App** is designed to provide users with a powerful tool for detecting faces and analyzing emotions. The app offers various features such as real-time face detection, emotion recognition, and content management capabilities. With user-centric design, flexible access for different user roles, and a secure environment, the app aims to serve both personal and research purposes.

Practical 04

Aim: Develop Software project management planning (SPMP) for the Face Detection App

- **Team Roles:**
  - **Project Manager(PM)**
  - **Responsibilities:** Oversee the entire project, including planning, resource allocation, timeline management, and risk mitigation. Ensure that the project stays on track and meets its objectives.
  - **Skills:** Strong leadership, communication, time management, risk management.
  - **Developers:**
  - **Responsibilities:** Write the code and implement the core features of the Face Detection App, including face detection algorithms, emotion recognition, real-time video processing, and data storage.
  - **UX designer:**
  - Responsibilities: Design the user interface of the app, ensuring that it is intuitive, accessible, and visually appealing. Conduct user testing and iteratively improve the design based on user feedback.
  - **Engineers:**
  - **Responsibilities:** Work with the development team to implement the backend infrastructure, ensuring the app runs efficiently. They will also assist in integrating machine learning models and ensuring app scalability.
  - **Business Analyst (BA):**
  - **Responsibilities**: Act as a liaison between stakeholders (users, admins, business owners) and the development team. Define business requirements, gather feedback, and ensure that the project aligns with the overall goals and vision.
- **Project Duration:**
  - **Planning & Requirement Gathering**: 1 Month
  - **Design**: 2 Months

- **Development (Iteration 1)**: 3 Months
- **Testing (Iteration 1)**: 1 Month
- **Development (Iteration 2)**: 3 Months
- **Testing (Iteration 2)**: 1 Month
- **Deployment & Maintenance**: 1 Month
- **Post-Deployment Review and Closure**: 1 Month

➢ **Phases if the Project:**

- Planning and requirement gathering:
- **Objective:** Understand and document all project requirements, including functional and non-functional aspects.
- **Activities:**
- Gather requirements from stakeholders (users, admins, moderators).
- Define scope and objectives of the Face Detection App.

- **Design:**
- **Objective:** Develop a design for the app, including UI/UX, architecture, and database schema.
- **Activities:**
- Create wireframes and user interface designs.
- Design the app architecture and backend infrastructure.
- **Development (iteration 1):**
- **Objective:** Implement core features and functionalities, such as face detection, emotion recognition, and image uploading.
- **Testing (Iteration 1):**
- **Objective:** Perform initial testing to identify bugs and issues in face detection and emotion recognition.
- **Activities:**
- Conduct unit testing, integration testing, and user acceptance testing (UAT).
- **Development (Iteration 2)**
- **Objective:** Enhance the app with additional features like search, browse, content moderation, and administrative control.
- **Activities:**

- Implement user feedback features.
- **Testing (Iteration 2):**
- **Objective:** Validate the new features and ensure the app functions as expected.
- **Activities:**
- Perform system and regression testing.
- Deployment & Maintenance
- **Objective:** Deploy the app to production and ensure smooth operation.
- **Activities:** Deploy the app to cloud services or app stores.
- Monitor app performance and address issues promptly.
- Post-Deployment Review and Closure
- **Objective:** Final project evaluation and closure.
- **Activities:**
- Gather feedback from stakeholders.
- Review performance metrics (e.g., user adoption, face detection accuracy

➢ **Resources (People and tools):**

- People:
- Project Manager
- Developers
- US designer
- Engineers
- Business
- Moderators
- Tools
- Version Control: Git, GitHub
- Development Tools
- **Backend**: Node.js, Django
- **Frontend**: React Native or Flutter
- **Machine Learning**: TensorFlow, OpenCV, Keras
- **Database**: PostgreSQL or MongoDB
- **Design Tools**: Figma , Adobe XD, Sketch

- **Testing Tools**: Selenium, Jest, Postman
- **Project Management**: Jira, Trello

➢ **Risk Management**

- Identified Risks
- Algorithm Accuracy: The machine learning models may not perform well under varying real-world conditions (e.g., low lighting, occluded faces).
- **Data Privacy Concerns**: Users might have concerns about facial data being processed and stored.
- **Scope Creep**: As the app develops, additional features may be requested, leading to increased complexity.

➢ **Quality Management**

- **Code Reviews**: Conduct regular code reviews to ensure quality, security, and maintainability.
- **Testing**: Implement automated and manual testing to detect and fix defects.
- **Quality Metrics**
- **Defect Density**: Track the number of bugs or issues discovered during the testing phase.
- **User Satisfaction**: Measure user feedback through surveys or app ratings

➢ **Communication**

- Weekly team meetings to review progress and address roadblocks
- Daily standups to discuss immediate tasks.
- Use Slack for instant communication and Zoom for meetings.
- Daily standups to discuss immediate tasks.

➢ **Change Management**

- **Change Request**: Any changes to the project (features, timelines, or budget) must be submitted as a formal change request by stakeholders.
- **Impact Analysis**: Analyze the impact of the requested changes on the project (time, cost, scope).
- **Approval Process**: The project manager and key stakeholders must approve the changes before implementation.

➢ **Project Closure**

- Finalizing and archiving project documentation.
- Reviewing and confirming all project objectives were met.
- Conducting a **lessons learned** session with the team
- Transitioning the app to maintenance mode and providing support for users.

- **Conclusion**

  This **Software Project Management Plan (SPMP)** outlines the steps, resources, and strategies necessary to deliver the **Face Detection App** successfully. By adhering to these plans and processes, the project will progress in an organized and efficient manner, ensuring a quality, user-friendly product that meets business and user requirements.