**Product Design - Team 32, Fixing expression by stitching the best one**

**Team Members:**

1) P.V. Sakthidhar - 2022112005

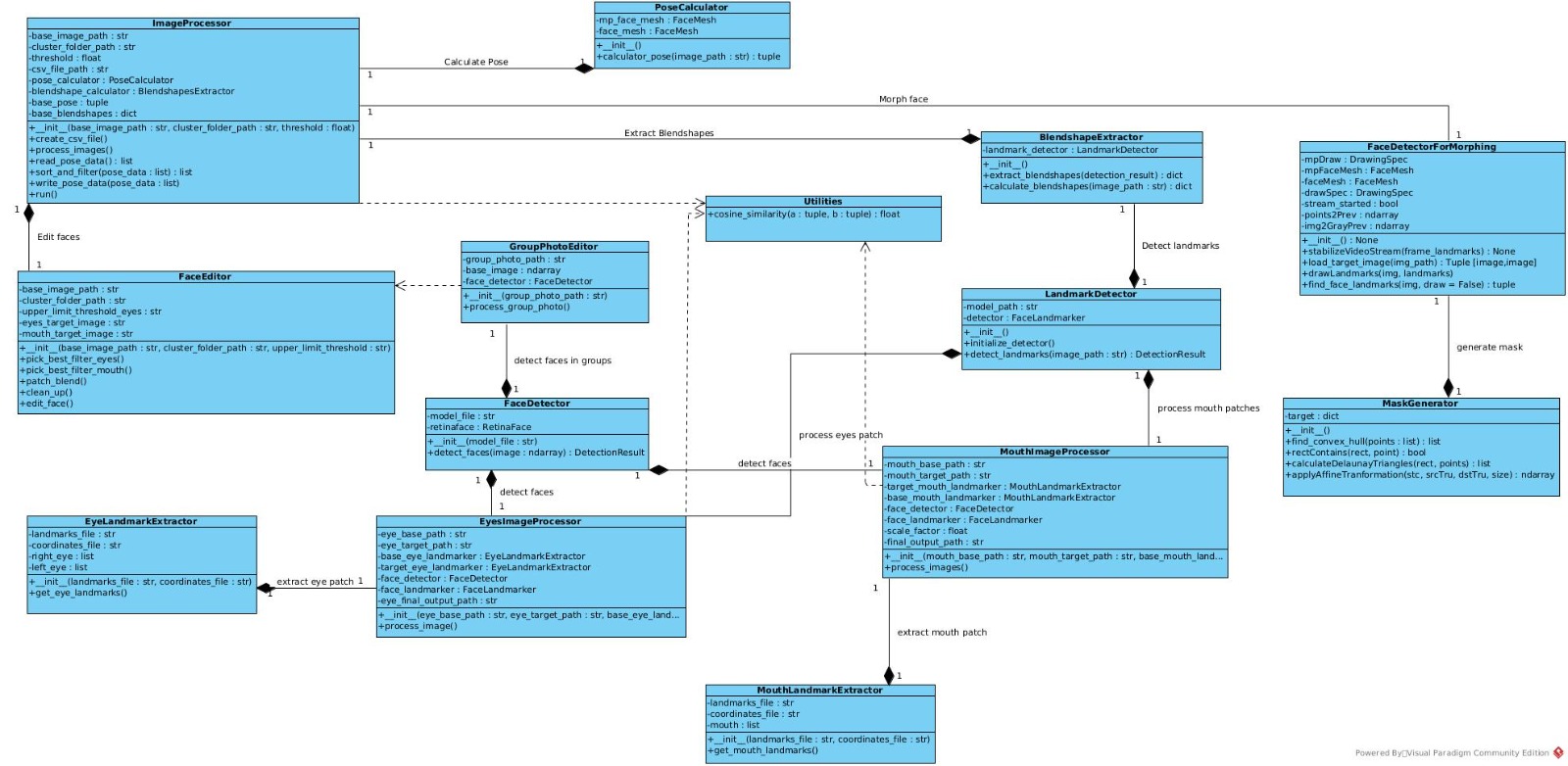
2) Saketh Reddy Vemula - 2022114014

3) Vaddineni Saideekshith - 2022101110

4) Yashas S B - 2022113001

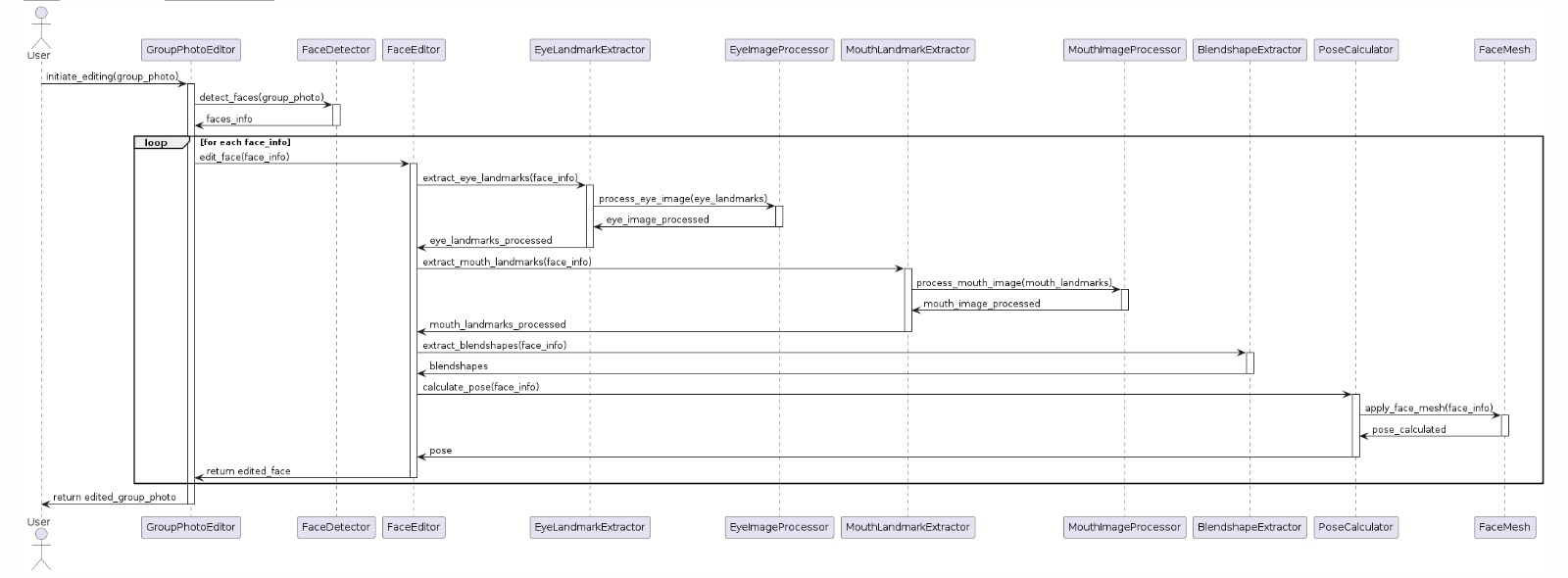
5) Ishita Bansal - 2022114004

# Design Model :



|  |  |
| --- | --- |
| GroupPhotoEditor | Class state :  > Responsible for editing group photos.  Class behavior :  > Initializes the class with the path to the group photo.  > process and edit the group photo. |
| FaceDetector | Class state  > responsible for detecting faces in an image.  Class behavior :  > Initializes the class with the path to the model file for face detection.  > Detects faces in the given image and returns the detection result. |
| FaceEditor | Class state  > responsible for editing individual faces.  Class behavior  > Initializes the class with the path to the base image, cluster folder, and the upper limit threshold for eyes.  > Selects the best filter for the eyes.  > Selects the best filter for the mouth.  > Blends the edited face with the original image.  > Performs cleanup tasks after editing the face.  > main function to edit a face is present |
| ImageProcessor | Class state:  > handle the overall image processing tasks.  Class behavior:  > Initializes the class with the path to the base image, cluster folder, and a threshold value.  > Creates a CSV file for storing data.  > Processes the images.  > Reads and returns the pose data as a list.  > Sorts and filters the pose data.  > Writes the pose data to a file or storage.  > The main function to run the image processing pipeline. |
| PoseCalculator | Class state  > Responsible for calculating the pose or orientation of a face.  Class behavior :  > Calculates the pose of the face in the given image and returns it as a tuple. |
| BlendshapeExtractor | Class state  > extracts blendshapes or facial expression representations from images.  Class behavior :  > Extracts the blendshapes from the given detection result and returns them as a dictionary.  > Calculates the blendshapes for the given image and returns them as a dictionary. |
| LandmarkDetector | Class state  > detects facial landmarks in images.  Class behavior :  > Initializes the class with the path to the model for landmark detection.  > Initializes the landmark detector.  > Detects facial landmarks in the given image and returns the detection result. |
| EyesImageProcessor | Class state  > processes eye regions in images.  Class behavior :  > Initializes the class with paths to the base and target eye images, instances of EyeLandmarkExtractor, FaceDetector, FaceLandmarker, and the path for the final output.  > Processes the eye regions in the images. |
| EyeLandmarkExtractor | Class state:  > extracts eye landmarks from files. Class behavior:  > Initializes the class with paths to the landmark and coordinate files.  > Extracts and returns the mouth landmarks. |
| MouthImageProcessor | Class state:  > processes mouth regions in images. Class behavior:  > Initializes the class with paths to the base and target mouth images, instances of MouthLandmarkExtractor, FaceDetector, FaceLandmarker, a scale factor, and the path for the final output.  > Processes the mouth regions in the images. |
| Utilities | Class state:  > contains utility functions. Class behavior:  > Calculates the cosine similarity between two tuples and returns a float value. |
| FaceDetectorForMorphing | Class state:  > Detects the faces and identifies the landmarks. It draws Landmarks, Stabilizes the video(if webcam is used). Also does image loading and preprocessing for web input. Class behavior  > Stabilizes video stream using optical flow.  > Loads a target image from a specified path for face morphing.  > Finds face landmarks in the given image.  > Draws landmarks on the given image. |
| Mask Generator for Morphing | Class state:   > Generates and applies a facial mask to an image based on detected facial landmarks. Class behavior:  > Finds the convex hull for a set of points.  > Checks if a point is inside a given rectangle. |

# Sequence Diagram: Sequence diagram for the use case when user edits a group photo:



# Design Rationale:

1. **Seperation of Concerns**:

* The design separates different concerns into individual classes, such as face detection (FaceDetector), landmark detection (LandmarkDetector), pose calculation (PoseCalculator), blendshape extraction (BlendshapeExtractor), and image processing for specific regions like eyes (EyesImageProcessor) and mouth (MouthImageProcessor).
* This separation promotes modularity, code reusability, and maintainability, as each class has a specific responsibility.

1. **Reusability**:

* Some classes, like FaceDetector, LandmarkDetector, PoseCalculator, and BlendshapeExtractor, are designed to be reusable components that can be utilized in different parts of the system or even in other projects.
* This reusability promotes code sharing and reduces duplication of effort.

1. **Flexibility**:

* The design separates the concerns of processing different facial regions (eyes and mouth) into separate classes (EyesImageProcessor and MouthImageProcessor), allowing for independent development, testing, and potential replacement or enhancement of these components.

1. **Abstraction and Encapsulation**:

* The classes encapsulate their internal state and behavior, providing well-defined interfaces (methods) for interacting with them.
* For example, FaceDetector encapsulates the logic for detecting faces in an image, while LandmarkDetector encapsulates the logic for detecting facial landmarks.

1. **Composition and Dependency**:

* The design utilizes composition relationships to ensure tight coupling between related classes. For example, GroupPhotoEditor has a composition relationship with FaceDetector, indicating that the FaceDetector instance is a part of the GroupPhotoEditor.
* Dependency relationships are used when a class requires functionality from another class without owning it. For instance, EyesImageProcessor and MouthImageProcessor depend on the Utilities class for the cosine\_similarity function.