**1.INTRODUCTION**

**1.INTRODUCTION**

The purpose of this project is to provide a software solution that can use geometric code to create a 3D model prototype of an entity from a photo. Users will be able to input a snapshot of an entity and the software will build a 3D model that may be used for visualisation, simulation, or 3D printing. Overall, the entity to 3D model prototype program will be a useful tool for creating 3D models from images and will be used in a variety of sectors such as design, engineering, and construction.

The requirements for an Entity to 3D Model Prototype from a Photo are outlined. This system's goal is to automate the process of producing a 3D model of an entity from a snapshot. The algorithm will identify significant elements in the photo and construct a 3D model using computer vision and machine learning techniques. This method will be useful in a variety of industries, including building, gaming, and 3D printing.

The product is a piece of software that can generate 3D representations of items or entities from a single photograph. This program is intended for usage in a variety of industries, such as architecture, engineering, game development, and art. It is now possible to extract 3D information from 2D photos because to improvements in computer vision and machine learning. As a result, methods that can automatically construct 3D models from pictures have been developed.

The Entity to 3D Model Prototype from a Photo is one such tool that creates 3D models of items from a single shot using cutting-edge computer vision and machine learning techniques. It attempts to simplify 3D modelling, making it more accessible to a broader range of consumers and industries.

**2.PROBLEM STATEMENT**

**2.PROBLEM STATEMENT**

Creation of a technology that makes it possible to make 3D printed models of people using their faces. Multiple photos of human faces should be taken by the system, which should then be able to extract the necessary features and produce gcode, which a 3D printer uses to produce the 3D models.

Users should be able to compile a collection of images with human faces using the system. The user may have taken these images themselves or they may have come from pre-existing datasets. The system should use efficient methods to extract pertinent features from the gathered photographs. This can entail applying deep learning techniques or computer vision algorithms.

The system should be able to produce a variety of human face combinations from the acquired photographs. This might entail fusing distinct facial features or particular facial characteristics.

Gcode Generation: The features that were extracted should be processed to create the gcode instructions that 3D printers need to manufacture tangible objects. The desired set of faces should appear in the gcode. The system should connect to a 3D printer and supply the created gcode as the printing process's input. The 3D printed models that are produced should faithfully depict the intended combinations of human faces. The system should have an intuitive user interface that makes it simple for users to choose and mix faces, track the development of the 3D printing process, and control the workflow as a whole. The system should aim for high accuracy in feature extraction and make sure the 3D printed models produced show a suitable level of detail and quality, accurately depicting the selected combinations of human faces.

**3.LITERATURE REVIEW**

**3.LITERATURE REVIEW**

**3.1 FACE DETECTION AND RECOGNITION**

**3.1.1 Face Recognition from Low Resolution Images**

This research examines a real-time system for face identification using video monitoring photos. First, we will briefly outline the major elements of the biometric face image standards. The conformity of available scientific datasets with these biometric requirements has been verified. Following that, we will focus on the analysis of the prepared face recognition application using the eigenface approach. Finally, the results of our face recognition studies with low-resolution photos are shown. It was discovered that the proposed and tested algorithm is extremely robust to resolution changes. Even for low-resolution photographs (16x20 pixels), the identification scores are acceptable.

Face identification from a video stream is more complex since the system must be resistant to variations in lighting, picture scale (size), and face location. The usage of the OpenCV Face Detector, which implements the Viola-Jones approach. The detection procedure can likewise be realized by skin color detection.

Principal component analysis (PCA), Fisherfaces based on linear discriminant analysis (LDA), independent component analysis (ICA), support vector machine (SVM), and other approaches, such as nonnegative matrix factorization (NMF), are commonly used to solve face identification.

**3.1.2 Face Detection and Recognition System using Digital Image Processing**

The most crucial feature in recognizing somebody is their face. Face recognition aids in verifying any person's identity by using his particular traits because it acts as an individual identity for everyone. The entire technique for authenticating any face data is separated into two stages.

In the first phase, the face is detected fast except in circumstances when the object is put relatively far away, and then the second phase begins in which the face is recognized as an individual. The entire process is then repeated, assisting in the development of a face recognition model, which is considered to be one of the most meticulously planned biometric technologies.

In general, the Eigenface approach and the Fisherface method are the two techniques that are currently used in face recognition patterns. The Eigenface approach use PCA (Principal Component Analysis) to reduce the face dimensional space of the facial features. This work is concerned with developing a face recognition system utilizing digital image processing.

**3.1.3 Face Detection Techniques: A Review**

With the phenomenal growth of video and picture databases, there is a tremendous demand for intelligent systems to automatically comprehend and examine information, as doing so manually is becoming increasingly difficult. The face is important in social interaction because it conveys a person's identity and sentiments. Humans do not have the same ability as machines to recognize diverse faces. As a result, automatic face detection systems are crucial in face identification, facial expression recognition, head-pose estimation, human-computer interface, and so on. Face detection is a computer system that locates and sizes a human face in a digital image. Face identification has emerged as a notable topic in the computer vision literature.

This study provides an in-depth examination of the many strategies investigated for face detection in digital photographs. This study also discusses many challenges and uses of face detection. Finally, various standard databases for face detection are provided, along with their features. Furthermore, we organize special discussions on the practical aspects of developing a robust face detection system and conclude this paper with several promising future research directions.

Face detection is a computer system that identifies the position and size of a human face in a digital image. The facial features are recognized, and any other items in the digital image, such as trees, buildings, and bodies, are ignored. It is a subset of object-class detection in which the aim is to locate and size all items in an image that correspond to a specified class. Face detection is a more general case of face localization. The goal of face localization is to determine the locations and sizes of a known number of faces (typically one).

There are two techniques to detecting facial features in a digital image: feature-based and image-based approaches. The feature-based technique attempts to extract image features and compare them against knowledge of facial features. While the image-based approach seeks the best possible match between training and testing images.

**3.2 IMAGES TO 3D MODEL PROTOTYPE**

**3.2.1 From Images to 3D Models**

The emphasis of the paper is on computer graphics' significance for 3-D (3D) models. Rendering realistic visuals of the 3D environment is possible thanks to computer graphics. However, the world must first be represented graphically in order to generate these images. There are numerous strategies accessible. Probably the most well-known are laser range scanners and other tools that shine light onto the object and gather 3D data by observing the reflection. The easiest approach to comprehend how a 3D shape can be created is to think about how people naturally see the 3D world. By looking at a three-dimensional environment from two slightly different angles, depth can be felt.

First, it's necessary to recover the relative motion between successive images. Finding related image features—that is, image points that come from the same 3D feature—between these photos works hand in hand with this method. The process of recovering the camera's motion, calibration, and 3D structure of the features comes next. There are two stages to this process. The reconstruction initially exhibits a projective skew, in which parallel lines are not parallel, angles are incorrect, lengths are either too long or excessively short, etc. As a result, there is no a priori calibration.

In order to reconstruct these points as well, the following step involves making an attempt to match all of a picture's image pixels with pixels in nearby images. Only a sparse set of 3D points are present in the reconstruction created as described in the preceding sentence (only a few features are initially taken into account). Interpolation could be a solution however this usually results in models with subpar visual quality. Therefore, the next step entails making an effort to match all of an image's image pixels with pixels in nearby images so that these points can also be rebuilt. Knowing all of the camera parameters that we acquired in the earlier step substantially simplifies this operation. The images can be warped so that the horizontal scanlines and the search range line up. In (Pollefeys et al., 1999b), an algorithm that can accomplish this for any camera motion is provided. This enables us to employ a stereo approach that is effective and quickly computes an optimal match for the whole scanline (Van Meerbergen et al., 2002). As a result, for practically every pixel in an image, we can estimate the depth (i.e., the separation between the camera and the surface of the object). A whole dense 3D surface model is produced by combining the outcomes of all the photos. To get a final photo-realistic outcome, it is possible to texture map the photos utilized for the reconstruction.

**3.2.2 Image based 3D Modelling**

The primary issues and potential solutions for the creation of 3D models from terrestrial photographs are discussed in this research. For accurate 3D modelling, close-range photogrammetry has long dealt with either manual or automatic image measurements. In many application areas today, 3D scanners are also becoming a common source of data input, although image-based modelling is still the most thorough, affordable, portable, adaptable, and widely-used method. This study presents the complete pipeline for 3D modelling from terrestrial picture data, taking into account various methods and examining all the phases involved.

IBR, or image-based rendering. This excludes the creation of a geometric 3Dmodel, but it may be regarded as a useful technique for the creation of virtual views for particular objects and under particular camera motions and scene conditions (Shum and Kang, 2000). IBR generates fresh perspectives of 3D environments right from source photographs. The method depends on either having a precise understanding of the camera positions or using automatic stereo matching, which in the absence of geometric data, necessitates a significant number of closely spaced images.

Modelling with images (IBM). This approach is frequently used to simulate accurate topography and urban areas (Gru n, 2000) or the geometric surfaces of architectural items (Streilein, 1994; Debevec et al., 1996; van den Heuvel, 1999; Liebowitz et al., El-Hakim, 2002). The majority of the time, interactive approaches still produce the most striking and precise results.

Range-based modelling. This technique collects an object's 3D geometric information directly. It can produce a highly precise and detailed depiction of most shapes and is based on expensive (at least for now) active sensors. These sensors are nonetheless pricey, created for certain uses or ranges, and impacted by the surface's reflective properties. They need some understanding of the capabilities of each individual technology at the specified range, and the generated data must be filtered and altered. Most systems simply concentrate on the acquisition of 3D geometry, offering only a monochromatic intensity value for each range value.

**3.2.3 Creation of prototype 3D models using RAPID PROTOTYPING**

The topic of the paper is using RAPID PROTOTYPING to create 3D models. New technologies are currently being incorporated into everyday life. The usage of FDM (fused deposition modelling) technology, which predominantly use thermoplastics to produce 3D models, is a suitable example. A few years ago, only certain types of businesses, research organizations, and universities had access to fast prototyping technologies. For smaller organizations and individuals, recent years have seen the affordability of technologies like FDM and STL (Stereolithography). The replicating rapid prototype RepRap (replicating rapid prototype), of which the extended version is the Prusa i3 printer, is the focus of this segment.

Stereolithography (SLA - Stereolithography) was the first technique employed. It gave rise to the subsequent iterations of 3D printing prototyping techniques. The STL format (Stereolithography) has been established as the standard format for model processing. It is currently supported by all CAD programs used in engineering practice and numerous commercial and non-commercial modelling tools. Resin that had been laser-cured served as the main component. Later, adhesive-treated laminated paper or film was employed, which was then laser-shaped to the required shape after application and curing.

The most used technology is SLS since it generates models in a variety of solid materials and consumes the material completely. The expensive price and enormous size of the item are a drawback. Consumption expenses are relatively expensive per kilogram of material. However, in a business, where strength and effective material use are the most important considerations, this is not deciding. If someone wants to use a 3D printer for personal use, they should take an entirely other route. Such a person has low purchase requirements, low energy requirements, excellent dependability, and low consumable material costs.

These specifications are particularly compatible with the FDM technology (Fused Deposition Modelling), in which the thermoplastic is extruded through the nozzle, cooled, and then the layers are created. The full model is made as a result. The models produced using this technique are generally reliable. The drawback is that the model's strength is decreased at the boundary of the layers because of the imperfect bonding. As a result of the poor design or severe cyclic loading, fractures may result.

When selecting the printer, the following factors were taken into account: printing area, cost of consumables, construction resistance, and potential for improvement.

**3.3 IMAGE TO G-CODE GENERATION**

**3.3.1 Image-based Contouring and G-code Generation for Additive Manufacturing**

In this paper, a novel method for producing G-codes for additive manufacturing (AM) using voxel-based models is proposed. The foreground voxels in the proposed method are split into skin and interior areas during the pre-processing stage. The model is then divided into layers along the z-axis, and each layer is then hatched one at a time in a bottom-up order to create the G-codes needed to fabricate the desired product. The suggested method first groups all non-zero pixels in a layer into clusters before processing it. The toolpaths are then traced and converted into G-codes for printing each cluster. The construction of skin and interior areas is accomplished by applying 3D image morphological operations by treating the input model as a 3D picture. As a result, the pre-processing procedure is substantially streamlined.

To implement the hatching of the 2D layers, the suggested solution uses a texture-mapping technique. The angles and resolutions of the infilling patterns can be adjusted by users using geometrical transformations. The hatching procedure thereby becomes more adaptable and effective. The suggested method also uses a novel approach to encode toolpaths during the hatching process, which results in less G-codes being generated. The efficiency of the AM process is raised as a result of the G-code program being compressed.

Techniques for additive manufacturing (AM) are frequently utilized for manufacturing and prototyping. They are more adaptable, inexpensive, and effective when compared to conventional subtractive manufacturing techniques. Thus, AM techniques increase productivity in a variety of fields of science, medicine, and industry. Typically, the input model for an AM process is expressed in a polygon-based representation, like an STL file. After processing this geometrical data with some helper software, a G-code program is created that directs the creation of the intended product. Since G-code generation is crucial for AM, a great deal of effort has been put into developing the necessary software and algorithms. As a result, there are already numerous G-code generators for polygon-based models available to consumers.

To create the toolpaths and G-codes, each cluster is treated individually. A cluster goes through two stages of hatching. The cluster's skin region pixels are rasterized in the first step to produce the G-codes that print the cluster border. The internal portion of the cluster is hatched in the second stage by creating G-codes using a texture-mapping method and filling it in according to specified patterns.

**3.3.2 A G-Code Generator for Volumetric Models.**

Slicers are used to translate input geometric models into G-codes in layered manufacturing (LM). Standard slicers only accept surface models as input data. In order to suit the data format of the slicers, volumetric models must be transformed into polygonal representations. Geometric mistakes and increased computational expenses follow. In this paper, we demonstrate an effective slicer that produces G-codes for volumetric models.

The suggested slicer has multiple phases. The input model is positioned in the first step so that its stability is maximized and its height is reduced. In order to keep the printed items from collapsing during the printing process, support structures are required for particular LM modalities. The second stage of the proposed slicer involves the detection and classification of overhangs as well as the growth of support structures to help the input model. Our slicer separates the input model's skin and internal areas, then slices the input model into a stack of 2D pictures for the subsequent computation. Following that, a contouring technique is used to produce toolpaths that fabricate a high-quality surface in the skin region.

In an LM process, the input model's skin should be densely printed to generate a smooth surface, while the inside space is filled in accordance with a specified pattern to shorten the printing process and lighten the weight of the final product. To carry out the remaining computations of the G-code generation process, it is therefore important to segregate the skin and interior areas. In order to build the skin region, our G-code generator continually erodes the foreground voxels. The thickness of the skin region is influenced by the number of erosions. A skin region gets thicker as there is more erosions. The number of erosions is left up to the users in this work. A foreground voxel's intensity changes to a predetermined value when it is eroded; this value is reserved for the skin region. Thus, following the creation of the skin region, the volume data set transforms into a triple-valued 3D image. Slices of the input model are created along the z axis for the calculations that follow. The interior pixels, skin, and resulting slices combine to form 2D images. Depending on the geometry of the input model, many clusters of skin and interior pixels may form in each slice. In order to cluster the foreground pixels, we must use a connected-component labelling approach. The toolpaths are then generated by hatching each of these clusters individually.

**3.3.3 G-code Modeling for 3D Printer Quality Assessment**

This study outlines our efforts to analyze infill mistakes since the quality of the infill has an effect on the component's structural integrity. The infill of a portion can make it structurally sound even if the part looks good from the outside. Furthermore, the infill is typically hidden once a part has finished printing. So, while the part is being printed, the infill must be monitored-code, by instructing the motors of computerized machine tools where to go, how quickly to move, and what path to take, Parser is a numerical control (NC) programming language that enables the production of goods. In this instance, the G-code is made up of a variety of commands that are mostly utilized for setting up the printer and are not necessary for producing the part. We have no need for these commands. Out of the possible G-code commands, the OpenGL parser only used two of them. G0 and G1 are the two commands. The rest of the commands are either used to configure the printer, such as changing the bed temperature or height, or to change the coordinate system from absolute positioning to relative positioning.

The first step is to delete each and every comment. G-code includes two distinct ways to represent comments. The first approach uses a semi-colon. A line's semi-colon will be searched for by the parser. If it does, everything following the semicolon will be removed. The use of brackets is the second method for designating a comment. After locating a starting parenthesis, the parser searches the rest of the line for the closing parenthesis. It will eliminate the substring containing the comment after locating the closing parenthesis. Identifying if a command is a G0 or G1 is the second phase in the extraction stage. It checks the line's initial character to do this. The following character in the line is tested if it is a "G." It continues on to the following line if the first character is not a "G." If the second character is a "0," the remaining characters on t

line is sent to a function that executes the G0 command. A function for the G1 command receives the rest of the line if the second character is a "1."

**4.DATA**

**4.DATA**

**4.1 Overview**

We need to Capture the voice of student and sent it to server and Server will do the following steps

* Preprocessing Voice to remove noise from it.
* Convert it to text.
* Convert it into query, execute and store the result into database. Cache if necessary

**4.2 Types of data required for the system**

**4.2.1 Input Data**

* We mainly Require Data of the students in some kind of format (JSON is Preffered ).
* And Voice Command of Students. The size depends mainly on the Command and No. of Queries (ex: - Get me my latest GPA and What were my ISA Marks and their average)

**4.2.2 Model Architecture**

* **Tokenization: - Spacy, NLTK.**
* **POS: - HMM Model**
* **Named Entity: - CRF (Conditional Random Field)**

**4.2.3 Training Details**

* **We’re Going to use Pre-trained Models.**

**4.2.4 Performance Metrics**

* Accuracy of models are roughly 84% of all models

**5. SYSTEM REQUIREMENTS SPECIFICATION**

**5. SYSTEM REQUIREMENTS SPECIFICATION**

**5.1 Product Perspective**

The product is a software application that can generate 3D representations of things or entities from a single photograph. This program is intended for usage in a variety of industries, such as architecture, engineering, game development, and art. It is now possible to extract 3D information from 2D photos because to improvements in computer vision and machine learning. As a result, methods that can automatically construct 3D models from pictures have been developed. One such tool is the Entity to 3D Model Prototype from a Photo, which employs cutting-edge computer vision and machine learning algorithms to construct 3D models of items from a single photograph. It attempts to simplify the 3D modelling process, making it more accessible to a broader range of users and industries.

* + 1. **Product Features**
* Photo Upload: Users will be able to input photographs of entities to be modelled.
* Entity identification: The system will identify the entity in the photo using computer vision techniques.
* Feature extraction: The system will employ computer vision techniques to extract essential properties from the recognized thing, such as shape, texture, and colour.
* 3D modelling: The system will construct a 3D model of the discovered entity using machine learning techniques.
* User Interface: The system will offer an easy-to-use interface for photo upload and 3D model display.
  + 1. **Operating Environment**
* Hardware platform: A desktop or laptop computer with at least an Intel Core i5 processor, 8GB RAM, and a dedicated graphics card.
* Operating System: Windows 10 or macOS 10.14 or later is required.
* Software Components: Python 3.7 or later, OpenCV, TensorFlow, NumPy, and the Flask web framework are required.

# Functional Requirements

# Validation of input photos: The tool should be able to validate the input photos to guarantee that they are of adequate quality and have the necessary features for 3D modelling. This includes inspecting the photo's resolution, lighting, and perspective.

# Feature extraction: Using computer vision techniques, the tool should be able to extract features from the input photo, such as edges, textures, and colors. This is the most important phase in determining the shape and structure of the entity being modelled.

# 3D reconstruction: The tool should reconstruct a 3D model of the entity using the features retrieved from the supplied photo. The depth and shape of the object are estimated using machine learning methods, and a digital representation of it is created.

# Output: The program should generate an output file containing the G-code for the entity's 3D model.

# External Interface Requirements

# Hardware Requirements

# Computer: The computer will be the key component that will run the program and process the data needed to generate the 3D prototype.

# Camera: The camera will collect the image that will be utilized to generate the 3D prototype.

# Graphic Card: The graphics card will be in charge of rendering the 3D pictures that will be used to develop the prototype. It is critical to remember that the graphics card must be capable of rendering high-quality images with a high level of detail and resolution.

# 3D printer: The prototype will be printed using a 3D printer. It is critical to ensure that the 3D printer supports the right materials for printing the prototype, such as plastic or metal, and that it has a high enough resolution to print the prototype precisely. The hardware components will communicate using conventional protocols such as USB, HDMI, or Ethernet.

# Software Requirements

* Databases:  Mongo DB
* Operating Systems: Compatible OS
* Tools and libraries:  OpenCV (Open-Source Computer Vision Library), Python 3.x NumPy, TensorFlow, PyTorch, SciPy, STL
* Source: Photo
  1. **Non-Functional Requirements**

**5.4.1 Performance Requirement**

* The 3D model prototype should be generated within a reasonable time frame, with a maximum processing time of 5 minutes per entity.
* The model should be able to handle a large number of entities and photos, with a minimum of 5 photos.
* The system should be able to handle a variety of photo types, excluding low-resolution and blurry photos, without affecting the accuracy of the 3D model.
  1. **Other Requirements**
     1. **Compatibility Requirements**
* The system should be compatible with different operating systems, including Windows, macOS, and Linux.
* The system should support various input file formats, such as JPG, PNG, and TIFF.
* The system should be compatible with different 3D modeling software, such as Autodesk Maya, Blender, and SketchUp.
  + 1. **Usability Requirements**
* The system should have a user-friendly interface that is easy to navigate.

**6. SYSTEM DESIGN**

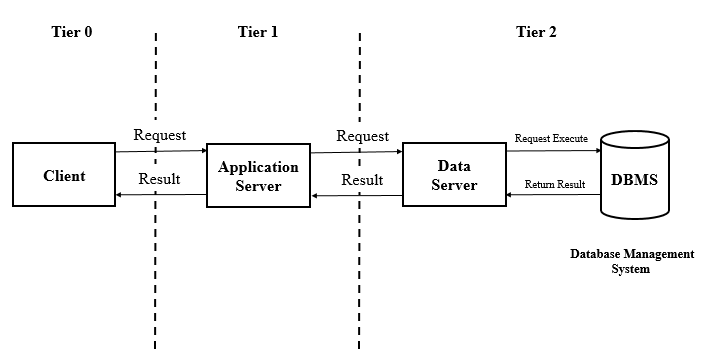
**6.SYSTEM DESIGN**

**6.1 Design Considerations**

**6.1.1 Design Goals**

The main goal of designing a project is to describe the working process which will ease the work of developer. We are giving an analogy of how the components of product will be working.

Although the following model is predicated on one, it is not actually based on a client-server architecture.



As Analogizing the above, we are assuming User will work as a client [Tier 0], User Interface [Application] will be working as an Application Server [Tier 1], Whereas ML/AI model takes the role of Data Server [Tier 2] which will communicate with the actual Database where the extracted dataset will be stored.

**6.1.2 Architecture Choices**

The following architecture would be used to develop our AI/ML-based model:

* Convolutional Neural Networks (CNNs): These are a sort of neural network design that is often used for image processing applications including image recognition and classification. CNNs employ convolutional layers to extract information from input images and pool the results to reduce data dimensionality.
* Generative Adversarial Networks (GANs): These are a sort of neural network design that is made up of two parts: a generator and a discriminator. GANs are extensively used for image synthesis and may build 3D models from 2D photos.

# 6.1.3 Constraints, Assumptions and Dependencies

There is no Design Constraint.

But Assumptions to be made are: -

* User will be uploading the photo which contains at least one human face.
* The output may not be accurate.

This Model will be highly dependent on the Quality of Input image, and Number of faces found in the image.

# 6.2 High Level System Design

# A high-level design document, or HLDD, augments the current project description with the required elements to represent an acceptable model for construction. This document contains a high-level architecture diagram that depicts the system's structure, including the hardware, database architecture, application architecture (layers), application flow (navigation) and technology architecture.

# A high-level design summarizes a system, product, service, or process.

# Such an overview makes it easier for supporting components to be interoperable with one another.

# The highest-level design should include a concise description of all platforms, systems, goods, services, and processes on which it is dependent, as well as any significant changes that must be made to them.

# Furthermore, all significant commercial, legal, environmental, security, safety, and technical risks, as well as any issues and assumptions, should be briefly considered.

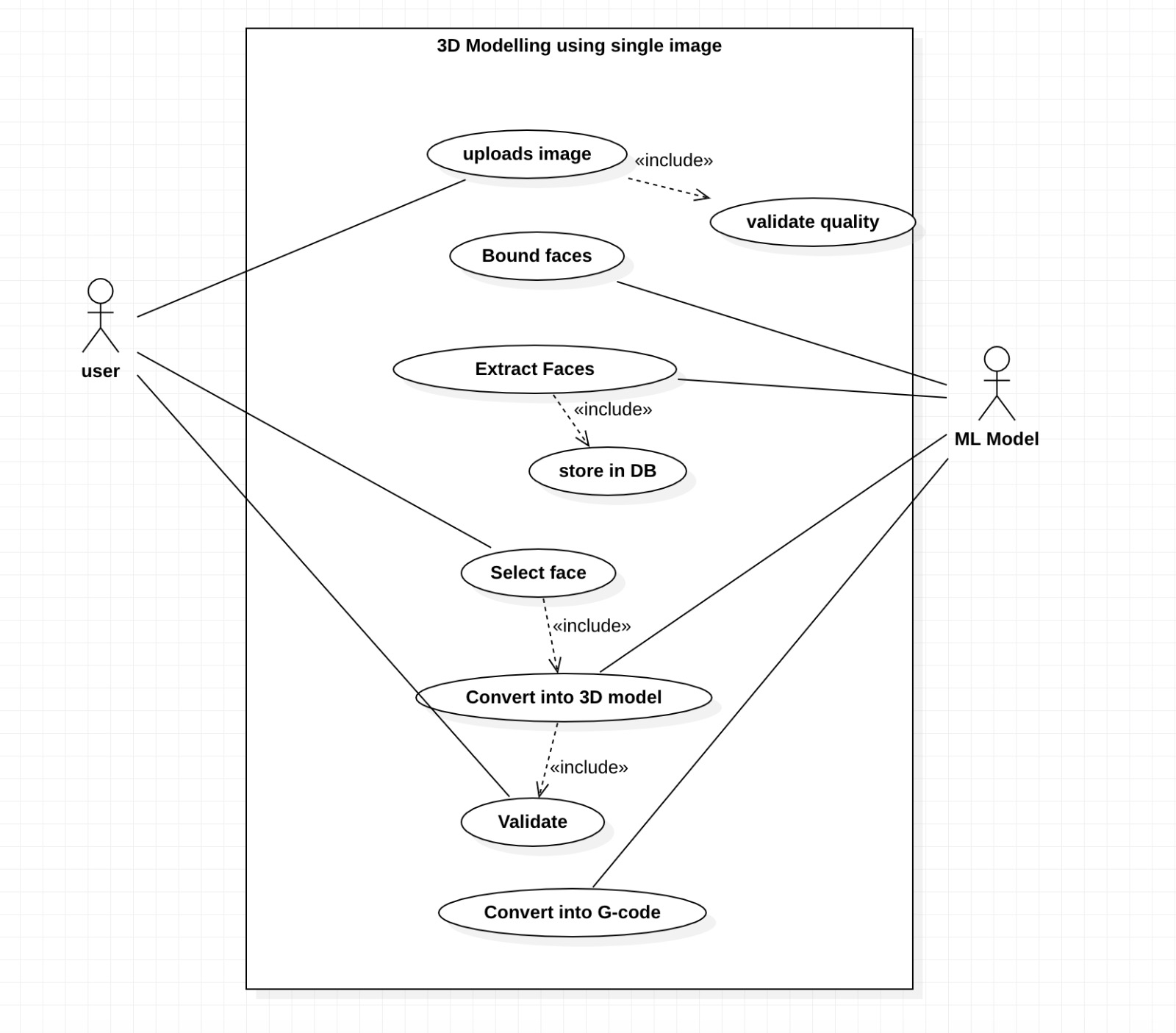
# The goal is to briefly discuss each work area while explicitly transferring ownership of more comprehensive design activity and fostering good communication among the various project teams.

# 

# 6.3 Design Description

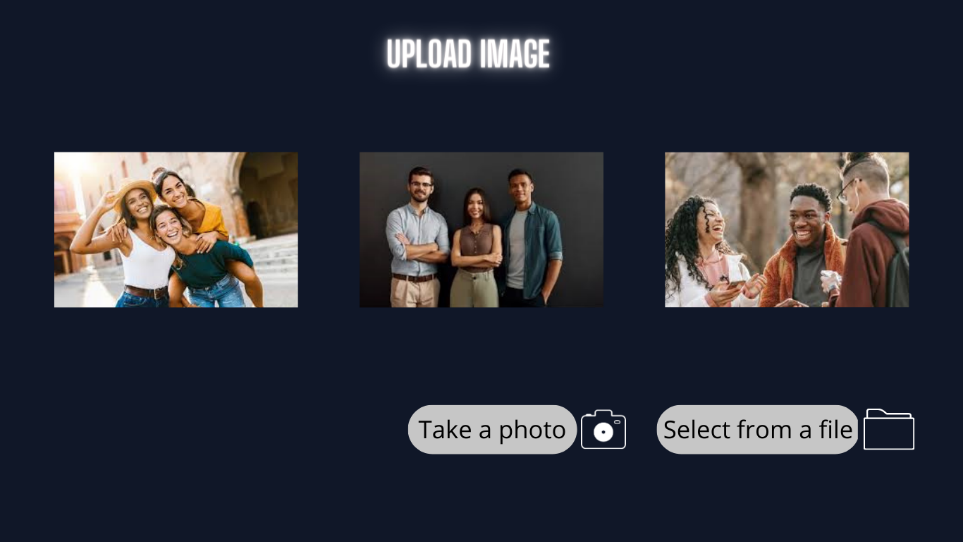
# 6.3.1 Use case Diagram

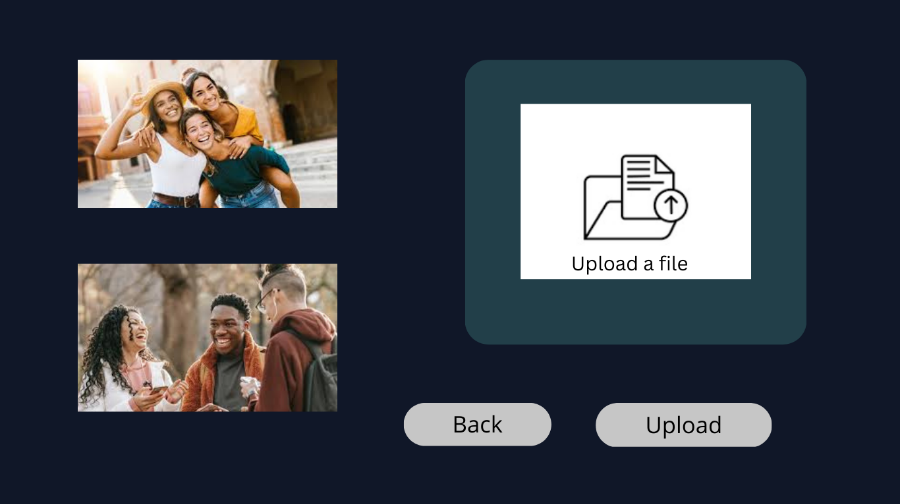
A use case is an approach for identifying, clarifying, and arranging system needs in system analysis. A use case is a collection of desirable sequences of interactions between systems and users in a specific environment that are tied to a specific purpose. The function generates a document that details all of the steps a user takes to accomplish an activity.



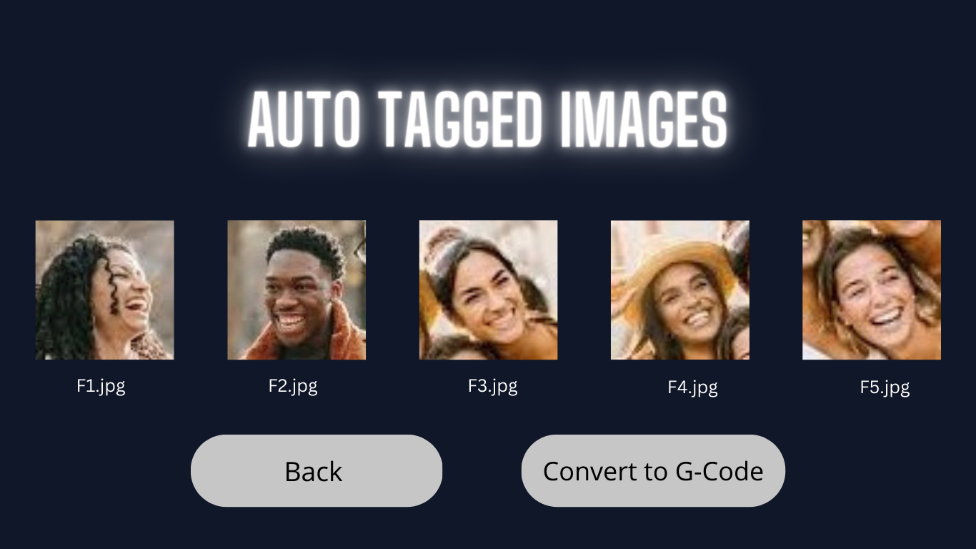
**6.3.2User Interface Diagrams**

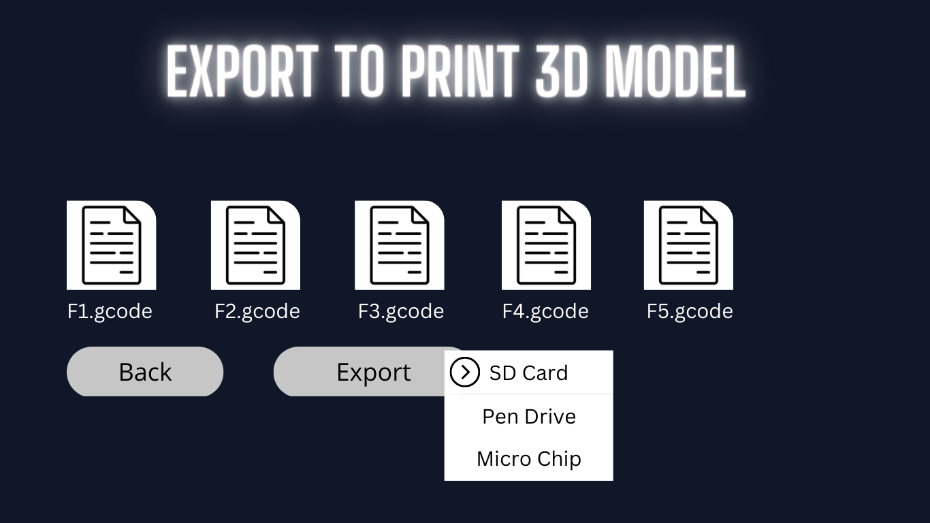












**7.IMPLEMENTATION AND**

**PSEUDOCODE**

**7.IMPLEMENTATION AND PSEUDOCODE**

**7.1 Algorithm**

High-level algorithm for extracting and bounding faces from an image and creating a database:

1. Load the input image.

2. Convert the image to grayscale for better contrast and easier processing.

3. Detect faces in the image using a face detection algorithm, such as Haar cascades, which can be implemented using OpenCV or other computer vision libraries.

4. For each detected face, create a bounding box around it to localize the face in the image.

5. Crop the image using each bounding box to extract the face as a separate image.

6. Save each face image to a separate file in a database/folder.

7. Generate 2^n combinations of face images by selecting all possible combinations of n faces from the database/folder.

8. Store each combination as a separate file in a database/folder.

9. Allow users to upload images by providing a file upload feature or allowing them to enter the image name manually.

10. Repeat steps 1-8 for each uploaded image.

**8.CONCLUSION OF CAPSTONE PROJECT PHASE-1**

**8.CONCLUSION OF CAPSTONE PROJECT PHASE-1**

**9.PLAN OF WORK FOR CAPSTONE PROJECT PHASE-2**

**REFERENCE/ BIBLIOGRAPHY**

[[1]](https://www.researchgate.net/publication/286992675_Face_Recognition_from_Low_Resolution_Images/related) Marciniak, Tomasz & Dabrowski, Adam & Chmielewska, Agata & Weychan, Radoslaw. (2012). Face Recognition from Low Resolution Images. Communications in Computer and Information Science. 287. 220-229. 10.1007/978-3-642-30721-8\_22.

[[2]](https://ieeexplore.ieee.org/document/9074838) G. Singh and A. K. Goel, "Face Detection and Recognition System using Digital Image Processing," 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bangalore, India, 2020, pp. 348-352, doi: 10.1109/ICIMIA48430.2020.9074838.

[[3]](https://www.researchgate.net/publication/326667118_Face_Detection_Techniques_A_Review) Kumar, Ashu & Kaur, Amandeep & Kumar, Munish. (2019). Face Detection Techniques: A Review. Artificial Intelligence Review. 52. 10.1007/s10462-018-9650-2.

[[4]](https://www.researchgate.net/publication/220423006_From_images_to_3D_models) Pollefeys, Marc & Van Gool, Luc. (2002). From images to 3D models. Commun. ACM. 45. 50-55. 10.1145/514236.514263.

[[5]](https://www.researchgate.net/publication/227786426_Image-based_3D_Modelling_A_Review) Remondino, Fabio & El-Hakim, Sabry. (2006). Image‐based 3D Modelling: A Review. The Photogrammetric Record. 21. 269 - 291. 10.1111/j.1477-9730.2006.00383.x.

[[6]](https://www.researchgate.net/publication/330391341_Creation_of_prototype_3D_models_using_RAPID_PROTOTYPING) Harušinec, Jozef & Suchánek, Andrej & Loulová, Mária. (2019). Creation of prototype 3D models using RAPID PROTOTYPING. MATEC Web of Conferences. 254. 01013. 10.1051/matecconf/201925401013.

[[7]](https://www.researchgate.net/publication/331800424_Image-based_Contouring_and_G-code_Generation_for_Additive_Manufacturing) Ueng, Shyh-Kuang & Huang, Hsuan-Kai & Liu, Zen-Yu. (2019). Image-based Contouring and G-code Generation for Additive Manufacturing.

[[8]](https://www.researchgate.net/publication/335847757_A_G-Code_Generator_for_Volumetric_Models) Ueng, Shyh-Kuang & Huang, Hsuan-Kai & Huang, Hsin-Cheng. (2019). A G-Code Generator for Volumetric Models. Applied Sciences. 9. 3868. 10.3390/app9183868.

[[9]](https://www.researchgate.net/publication/331688882_G-code_Modeling_for_3D_Printer_Quality_Assessment) Welander, Tyler & Marsh, Ronald & Amin, Md Nurul. (2018). G-code Modeling for 3D Printer Quality Assessment.

[**APPENDIX A DEFINITIONS, ACRONYMS AND**](https://docs.google.com/document/d/1sxLzRaxGh0fTgzlXiH-5RgoxpmOZgtX9RD5OH3Dfigw/edit#heading%3Dh.qzpp4n2gssp1) **ABBREVIATIONS**

|  |  |  |
| --- | --- | --- |
| Sl.no | Value | Descroption |
| 1. | Entity | An object or thing that exists in the real world. |
| 2. | 3D Model | A three-dimensional representation of an object or space created using computer software. |
| 3. | Prototype | A preliminary version of a product, often used for testing and evaluation before final production. |
| 4. | Photo | A visual image captured using a camera or other device. (Image consisting of human face) |
| 5 | 3D | 3-Dimensional |
| 6. | GUI | Graphical User Interface |
| 7. | LDA | Linear Discriminant Analysis |
| 8. | ICA | Independent Component Analysis |
| 9. | SVM | Support Vector Machine |
| 10. | NMF | Non-negative matrix Factorization |
| 11. | PCA | Principal Component Analysis |
| 12. | STL | Stereolithography |
| 13. | FDM | Fused Deposition Modelling |
| 14. | CAD | Computer Aided Design |
| 15. | G-code | Geometric Code |