A MAJOR PROJECT REPORT

on

"ADVERTISEMENT SYSTEM BASED ON REAL TIME AGE AND GENDER DETECTION"

Submitted in partial fulfillment of the requirement for the award of the degree

of

MASTER OF COMPUTER APPLICATIONS

by

NEERAJA GOLI

UID: 111723039147

Under the Guidance of

Mr. A. Rajeshwar

Assistant Professor

DEPARTMENT OF MCA

LOYOLA ACADEMY DEGREE & PG COLLEGE

Alwal, Secunderabad 500010

(Autonomous and affiliated to Osmania University)

Re-accredited with 'A' Grade by NAAC

"College with Potential for Excellence" by UGC

2023-2025

LOYOLA ACADEMY
(DEGREE & PG COLLEGE)

Alwal, Secunderabad 500010

(Autonomous and affiliated to Osmania University)

Department

of

Master of Computer Applications

CERTIFICATE

This is to certify that this Major-project entitled **"ADVERTISEMENT SYSTEM BASED ON REAL TIME AGE AND GENDER DETECTION"** is Bonafide work carried out by **NEERAJA GOLI** bearing **UID:111723039147** in **LOYOLA ACADEMY DEGREE & PG COLLEGE** and Submitted to **OSMANIA UNIVERSITY** in partial fulfillment of the requirements for the award of Master of Computer Applications.

Mr. A. Rajeshwar
Internal Examiner

External Examiner

Dr. P Dayaker
Head of the Department

Rev. Fr. Dr. N B Babu SJ
Principal

MANAC Infotech (P) Limited
Development • Training • Staffing • Recruiting

Ref: MANAC/PROJ/24-25/1290

To
Head of the Department,
Master of Computer Applications,
Loyola Academy Degree & PG College,
Alwal Rd, Secunderabad-500015.

COMPLETION CERTIFICATE

This is to certify that **Ms. NEERAJA GOLI (111723039147)** from **LOYOLA ACADEMY, OLD ALWAL, SECUNDERABAD 500010,** has successfully completed the project titled "**ADVERTISEMENT SYSTEM BASED ON REAL TIME AGE AND GENDER DETECTION"** in Partial fulfilment of the **Masters of Computer Applications** course during the period **March 2025 to June 2025**

Yours Truly

Manac Infotech (P) Limited

MANAC Infotech (P) Limited
Development • Training • Staffing • Recruiting

Ref: MANAC/PROJ/24-25/1290

To
Head of the Department,
Master of Computer Applications,
Loyola Academy Degree & PG College,
Alwal Rd, Secunderabad-500015.

COLLABORATIVE CERTIFICATE

This is to certify that **NEERAJA GOLI (111723039147)**, from **Loyola Academy**, **OLD ALWAL, SECUNDERABAD, 500010**, has successfully done a "**COLLABORATIVE RESEARCH**" work with **Manac Infotech (P) Limited** on the topic titled "**ADVERTISEMENT SYSTEM BASED ON REAL TIME AGE AND GENDER DETECTION".**

This project was done during the period of **March 2025 to June 2025.**

Yours Truly

Manac Infotech (P) Limited

ACKNOWLEDGEMENT

This acknowledgment transcends the reality of formality, and I would like to express deep gratitude and respect to all those people behind this project who guided, inspired, and helped me complete it.

I express my profound gratitude to Rev. Fr. Dr N.B Babu SJ, the Principal of Loyola Academy Degree and PG College, and Dr. P. Dayaker, the Head of the MCA department, for giving me this opportunity to pursue this Major-project, which I was passionate about, and helping me add a feather to my hat of educational assets.

I extend my special thanks to my internal guide Mr. A. Rajeshwar for the time and efforts that he provided throughout the year. His guidance, advice, and suggestions were extremely helpful to me during the completion of the Major-project. In this aspect, I am eternally grateful to you.

I acknowledge that this Major-project was completed entirely by me and not by someone else.

Place: Hyderabad

NEERAJA GOLI
111723039147

DECLARATION

I, Neeraja Goli, a student of NMCA, hereby declare that the Major-Project titled "**ADVERTISEMENT SYSTEM BASED ON REAL TIME AGE AND GENDER DETECTION**" which is submitted by me to Mr. A. Rajeshwar, Loyola Academy Degree and PG College, Secunderabad, Alwal, in partial fulfillment of requirement for the award of the degree of computer science, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition. The Author attests that permission has been obtained for the use of any copyrighted material appearing in the Dissertation, or Major-Project report other than brief excerpts requiring only proper acknowledgment in scholarly writing, and all such use is acknowledged.

Place: Hyderabad

NEERAJA GOLI
111723039147

ABSTRACT

In the modern era of digital marketing, personalized advertising plays a crucial role in enhancing user engagement. This project, "Advertisement System Based on Real-Time Age & Gender Detection" leverages computer vision and deep learning to dynamically display targeted advertisements based on the demographics of viewers.

The system utilizes OpenCV's deep learning models for age and gender classification through a real-time webcam feed. A face detection model identifies individuals, and a CNN-based classifier predicts their gender and age group. The system ensures uniqueness by preventing duplicate counts of individuals within a short time frame. The detected data, including gender, age, timestamp, and the displayed advertisement, is stored in a JSON file for analytical purposes.

A Flask-based web interface serves as the front end, where advertisements are displayed based on real-time demographics. The default advertisement is shown for 30 seconds upon startup, after which the detection module activates. If the number of males exceeds females, a male-oriented advertisement is shown, and vice versa. The interface updates seamlessly, ensuring dynamic advertisement rotation based on the audience composition. This project finds applications in retail stores, shopping malls, digital billboards, and smart advertising kiosks, providing businesses with a data-driven approach to targeted marketing.

Future enhancements could include database integration, emotion detection, multi-person tracking, and integration with AI driven recommendation systems for further optimization.

Keywords: Real-time advertisement, age and gender detection, OpenCV, Flask, deep learning, targeted marketing.

TABLE OF CONTENT

LIST OF FIGURES:

LIST OF OUTPUTS:

1. INTRODUCTION

In the rapidly evolving world of digital marketing, personalized and targeted advertising has become an essential strategy to enhance user engagement and maximize advertisement effectiveness. Traditional advertising methods, which display the same content to every viewer regardless of their background, often lead to reduced interest and wasted promotional efforts. To overcome this limitation, intelligent systems that adapt to the viewer's demographic attributes such as age and gender are gaining traction. These systems enable advertisers to show more relevant content to the audience, increasing both visibility and conversion rates.

This chapter gives an overview of the system's purpose, aim, objectives, background and operation environment.

1.1 PURPOSE, AIM AND OBJECTIVES:

The purpose of this project is to design and implement a real-time, intelligent advertisement system that uses computer vision and deep learning techniques to detect the age group and gender of individuals in front of a webcam. Based on the detected demographic information, the system dynamically selects and displays targeted advertisements that are most relevant to the viewer. This approach enhances the effectiveness of digital advertising, increases viewer engagement, and demonstrates the power of AI in marketing automation.

Aim: To develop an AI-powered Smart Advertisement System that:

- Detects human faces in real-time using webcam input.
- Predicts the age group and gender of each detected individual using pre-trained deep learning models.
- Displays personalized advertisements based on detected demographic traits.
- Tracks and logs all demographic interactions for analytical and decision-making purposes.

Objectives are:

- Implement real-time face detection using OpenCV and face recognition.
- To integrate deep learning models from Hugging Face (rizvandwiki/gender-classification for gender and nateraw/vit-age-classifier for age) for accurate demographic classification in real-time.

- Matching the detected demographic profile to a corresponding advertisement from a pre-defined JSON mapping and display personalized advertisement dynamically.
- Provide a Web Interface Integration i.e.Flask-based web interface that Streams live webcam footage, Displays the currently shown advertisement and Shows real-time visitor count and demographic summary.
- Implement a dashboard to visualize demographics data filtered by gender, age group, and date range.

1.2 BACKGROUND OF PROJECT:

In the era of digital transformation, advertising has evolved beyond static billboards and generic television commercials to dynamic, data-driven strategies. Traditional advertisement systems often target broad audiences without considering the demographics of viewers, leading to lower engagement and reduced impact. With the growing availability of affordable computing resources and intelligent systems, there is a significant opportunity to enhance advertising effectiveness by integrating artificial intelligence. One such innovation involves tailoring advertisements based on real-time understanding of the audience's demographic attributes, such as age and gender.

This project aims to bridge the gap between traditional advertisement methods and intelligent content delivery by developing a Smart Advertisement System that leverages computer vision and deep learning. By utilizing a webcam, the system detects human faces in real time and classifies them using deep learning models into various age groups and gender categories. Once demographic details are determined, the system instantly displays a corresponding advertisement. This enables more targeted and meaningful interactions between advertisements and viewers, leading to potentially higher engagement and conversion rates.

1.3 SCOPE OF PROJECT:

The scope of this project encompasses a comprehensive smart advertisement in supermarkets and outlets, as to attract the customers and grab attention and implementation. The project will encompass various stages, including advertisement data acquisition, preprocessing webcam, model development, integrating with Deep Learning Models, and storing the details. Key aspects of the project scope include:

- **Real-Time Demographic Detection:** The system continuously captures video frames from a webcam and applies Hugging Face models to perform real-time face detection and demographic classification.
- **Advertisement Personalization and Display:** Upon successful demographic classification, the system dynamically selects and displays the most appropriate advertisement image based on the detected age and gender. This personalization aims to increase advertisement effectiveness and viewer attention.
- **Interactive Dashboard and Data Analytics:** A web-based dashboard is provided for visualizing demographic statistics. Users can filter records by gender, age group, and date range to analyze traffic patterns and demographic trends.
- **System Feedback and Robustness:** In scenarios where no face is detected due to poor lighting or obstructions, the system is capable of providing feedback to users.
- **Administrative Utilities:** The system includes tools for viewing stored data, resetting demographic records, and maintaining data integrity. This functionality supports operational oversight and ensures the system remains manageable over time.
- **Extensibility and Future Enhancements:** The modular design of the system allows for future upgrades, such as the addition of emotion recognition, cloud data synchronization, support for multiple camera feeds, and the integration of speech-based advertisement interaction mechanisms.

1.4 MODULES DESCRIPTION:

The Smart Advertisement System is architecturally divided into distinct yet interconnected modules, each responsible for handling specific functionality to ensure seamless user experience and effective advertisement delivery. The system integrates real-time image processing, machine learning, web development, and data analytics. The following are the primary modules of the project:

1. **Face Detection and Demographic Classification Module**

   This module is responsible for detecting human faces from a live webcam feed and classifying each detected face by gender and age group. It leverages Hugging Face models (Gender Classification - rizvandwiki/gender-classification and Age Classification - nateraw/vit-age-classifier).

2. **Advertisement Display Module**

   Based on the demographic classification result, this module selects an appropriate

advertisement image from a predefined adv.json file, which maps each age group and gender to a corresponding advertisement. The selected advertisement is displayed in real time alongside the webcam feed. If no face is detected or lighting is poor, a default message such as "Check Proper Light" or a placeholder ad is shown to guide user engagements.

3. **Data Logging and Storage Module**

   This module handles the recording of demographic information into both a local SQLite database and an Excel spreadsheet. Each entry includes attributes such as timestamp, age group, gender, and face encoding. This persistent data is later utilized for analysis, reporting, and visualization.

4. **Dashboard and Visualization Module**

   This module offers an interactive dashboard where users can view demographic trends in a visual format. Using Chart.js, it displays age and gender-wise bar charts that are dynamically updated based on database queries. **Filters** such as gender, age group, and date range are provided to allow deeper insights into visitor patterns over time and downloading the data in form of Excel file.

2. SYSTEM ANALYSIS

In this chapter, we will discuss and analyze the developing process of Audit Control including software requirement specification and comparison between existing and proposed systems. The functional and non-functional requirements are included in the SRS part to provide a complete description and overview of system requirements before the developing process is carried out. Besides that, existing vs. proposed provides a view of how the proposed system will be more efficient than the existing one.

2.1 HARDWARE AND SOFTWARE REQUIREMENTS

2.1.1 HARDWARE REQUIREMENTS:

- **Processor** **:** Intel(R) Core(TM) i3 CPU M 330 @ 2.13GHz 2.13 GHzCore
- **RAM** **:** 8 GB RAM
- **Hard Disk** **:** 64GB required
- **Webcam** **:** In-built or External HD Webcam
- **Hard disk** **:** 40 GB or above**.**
- **GPU** **:** 4GB or above**.**

2.1.2 SOFTWARE REQUIREMENTS:

- **Language** : Python
- **Database** : SQLite, JSON
- **Operating System** : Windows 10/11, Mac.
- **IDE** : Microsoft Visual Studio Code
- **Libraries** : OpenCV, Hugging Face Modules, Flask, NumPy, face recognition

2.2 SOFTWARE REQUIREMENT SPECIFICATION:

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.)

The SRS phase consists of two basic activities:

1) **Problem/Requirement Analysis:**

The process is order and more nebulous of the two, deals with understand the problem, the goal and constraints.

2) **Requirement Specification:**

Here, we focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity.

The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

2.2.1 ROLE OF SRS:

The role of the SRS in this project is to provide clarity and direction throughout the software development lifecycle. By establishing well-defined functional requirement and non-functional requirements, the SRS ensures that the development process is both structured and goal-oriented.

In the context of testing and validation, the SRS acts as a benchmark against which the final product is evaluated. Test cases and user acceptance criteria are derived directly from the documented requirements, allowing both the developers and reviewers to assess whether the system meets its intended goals.

The SRS provides a foundation for future enhancements, as it documents current features, constraints, and design decisions. As the system evolves, the SRS can be updated to reflect new requirements or extended functionality, ensuring continued maintainability and scalability.

2.2.2 Functional Requirements:

- **Webcam Integration for Real-Time Input**

  The system must continuously access the laptop's webcam to capture live video frames as input for face detection and demographic analysis.

- **Face Detection and Extraction**

  The application should accurately identify human faces from each frame and extract relevant facial features for further processing.

- **Age and Gender Prediction**

  The system must employ Hugging Face models to determine the age group and gender of each detected face in real time.

- **Demographic-Based Advertisement Display**

  Based on the predicted gender and age range, the system should dynamically display a corresponding advertisement image from the predefined dataset stored in the adv.json file.

- **Data Storage in Database**

  All unique demographic detections (age, gender, timestamp) should be saved to a local SQLite database (advertisement.db) for record-keeping and analysis.

- **Advertisement Fallback Mechanism**

  If the webcam is unable to detect a face due to poor lighting or obstruction, a default image should be displayed along with a user-friendly message like "Ensure proper lighting".

- **Live Dashboard Visualization**

  A web-based dashboard should be available, allowing real-time monitoring of demographic counts using graphical charts and statistics powered by Chart.js.

- **API Endpoints for Data Access**

  The backend should expose REST endpoints (e.g. /dashboard) to provide real-time data to the front-end components.

- **Demographic Filter Functionality**

  The dashboard must allow filtering demographic data based on gender, age group, and date range to analyze visitor trends effectively.

- **User Interface Navigation**

  A button must be available on the main page to redirect to the dashboard, enabling smooth navigation between detection and analytical views.

- **Efficient Advertisement Switching**

  The advertisement image must automatically refresh at regular intervals based on the most recent demographic input.

2.2.3 Non-Functional Requirements:

- **Performance Efficiency**

  The system must deliver near real-time age and gender predictions from the live webcam feed, with minimal delay to ensure a seamless and dynamic advertisement display experience.

- **Reliability**

  The application should reliably detect and classify age and gender under normal operating conditions and must consistently display advertisements without crashing or halting.

- **Accuracy**

  The system must use Deep learning models (face recognition and classifying for male and female of different age groups) that ensure acceptable levels of accuracy for predictions across diverse faces.

- **Scalability**

  The architecture should be modular so that it can be extended to support multiple camera feeds, additional age/gender categories, or larger advertisement datasets in the future.

- **Usability**

  The interface should be intuitive and user-friendly, requiring no technical expertise to operate, with clearly labeled buttons, real-time counts, and accessible visuals.

- **Maintainability**

  The codebase should be clean, well-commented, and organized into logical modules for ease of debugging and future enhancements.

- **Portability**

  The application must be executable on any standard Windows/Linux system with Python and OpenCV support, without dependency on external hardware.

- **Responsiveness**

  The web dashboard and main page should update every few seconds using Fetch requests, offering near-instant feedback on demographic changes and ad switching.

- **Fault Tolerance**

  In case of poor lighting or detection failure, the system should gracefully fall back to a default message or advertisement without terminating operations.

- **Compatibility**

  The system must function correctly across standard web browsers (Chrome, Edge) and be compatible with Flask's web framework and SQLite database.

2.3 SCOPE:

This document is the only one that describes the requirements of the system. It is meant for the use by the developers, and will also be the basis for validating the final delivered system. Any changes made to the requirements in the future will have to go through a formal change approval process. The developer is responsible for asking for clarifications, where necessary, and will not make any alterations without the permission of the client.

2.4 EXISTING SYSTEM:

The real-world advertisement landscape traditionally relies on broad demographic targeting through mediums like television, radio, billboards, and static digital displays. Techniques often

involve market research and segmentation to identify general audience characteristics for a given channel or time slot. For instance, children's products are advertised during cartoons, and luxury goods in high-end magazines. While effective to a degree, this approach lacks granular personalization and real-time adaptability to individual viewers.

More recently, digital advertising has shifted towards online behavioral targeting, utilizing Browse history, search queries, and social media activity to deliver somewhat personalized ads. However, these systems primarily operate within the digital realm and don't directly address physical spaces. These existing systems often fall short in providing truly context-aware and individualized advertisements in physical environments, leading to potential ad fatigue and reduced engagement.

2.4.1 DRAWBACKS OF EXISTING SYSTEM:

- Traditional advertisement systems display the *same content to all viewers*, regardless of their age, gender, or preferences, leading to ineffective marketing.
- Most existing systems do not include real-time analysis of the viewer’s demographics, which limits their ability to adapt dynamically to the audience.
- Advertisement content is often scheduled in advance based on assumptions or generalized data, which has more branding but often people think to buy based on their requirement reduces the impact.
- There is no integration of computer vision technologies to recognize or analyze faces for demographic insights.
- Advanced intelligent advertisement platforms are typically expensive, and out of reach for small businesses or local retailers.
- Existing intelligent ad systems may require cloud infrastructure, continuous internet access, and technical expertise to operate and maintain.

2.5 PROPOSED SYSTEM:

The proposed system aims to revolutionize traditional advertising methods by introducing an intelligent, real-time advertisement display system that dynamically adapts to the demographic profile of viewers captured through a webcam.

The proposed “*Advertisement System based on Real time Age and Gender Detection*” system introduces a novel approach to in-store advertising by leveraging real-time computer vision and machine learning to deliver highly personalized and dynamic content. Unlike static displays or

broad demographic targeting, it reacts instantly to individual viewers, providing relevant advertisements on the fly.

2.5.1 ADVANTAGES OF PROPOSED SYSTEM:

1. **Real-time Video Feed Integration:**

   A. **Continuous Capture**: The system continuously captures video frames from a live camera feed. This ensures constant monitoring of the immediate viewing environment.

   B. **High-Speed Processing:** Frames are used to maintain a fluid and responsive user experience, crucial for dynamic Advertisement content changes.

2. **Advanced Facial Recognition and Demographic Inference:**

   A. **Precise Face Detection:** Utilizes robust Haar Cascade Classifiers, an advanced deep learning models to accurately detect human faces within the video stream.

   B. **Deep Learning-Based Age Prediction:** A CNN model is employed to estimate the age group of the detected individual with high accuracy.

   C. **Machine Learning-Based Gender Classification:** A ViT(Visual Transformer) model, combined with ResNet, is used to classify the detected individual's gender as 'male', 'female'. This provides a reliable and efficient gender classification.

3. **Dynamic Advertisement Matching and Display:**

   A. **Intelligent Advertisement Mapping:** A core component is the advertisement mapping mechanism. This mapping is fully customizable and allows for a rich variety of targeted content.

   B. **Contextual Ad Switching:** As New individual's is observed, their gender and age are identified, the system dynamically switches the displayed advertisement to the most relevant one from the mapping.

   C. **"No Face" State Handling:** The system can also revert to a default or generic advertisement when no face is detected as displaying general advertisements.

4. **Real-time Data Logging and Analytics:**

   A. **Demographic Data Collection:** The system logs each instance of detected gender, age group, and the corresponding ad path served into a local SQLite database.
   B. **Performance Monitoring:** Real-time demographic counts are maintained in memory to provide an immediate overview of the audience distribution.

5. **Enhanced User Engagement and Marketing Effectiveness:**

   A. **Hyper-Personalization in Physical Space:** By tailoring advertisements to the specific individual standing in front of the display, AdVision aims to significantly increase ad relevance, capture attention, and drive purchase intent.

This proposed system differentiates itself by moving beyond traditional, generalized advertising to a granular, real-time, and data-driven approach, maximizing the impact of advertisements in physical retail and public environments.

3. TECHNOLOGIES USED

The proposed Smart Advertisement System leverages a variety of modern technologies, frameworks, and tools to ensure real-time performance, reliable detection, and user-friendly interaction. The combination of these technologies allows for the seamless integration of machine learning, web development, and data visualization components.

3.1 Python Programming Language:

Definition: Python is a high-level, interpreted, general-purpose programming language known for its readability, simplicity, and versatility. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python was created by Guido van Rossum and first released in 1991. It emphasizes code readability and reduces the cost of program maintenance.

Importance: In the development of the Smart Advertisement System Based on Real-Time Age and Gender Detection, Python serves as the backbone of the entire application. The language facilitates the integration of computer vision, machine learning models, web development, database handling, and real-time data visualization. Its ability to interface with hardware, pre-trained models, and web frameworks makes it indispensable for this project.

Process: The core processes handled using Python in this system include:

- Using OpenCV to access webcam feed and detect faces in real-time.
- Loading and processing Hugging Face models for demographic classification.
- Inserting demographic data into an SQLite database and ensuring uniqueness of faces using the face recognition library.
- Building RESTful APIs and dynamic pages with Flask.
- Data Visualization using Chart.js with Flask APIs to display filtered demographic charts.

Types: Python is an interpreted, dynamically-typed, and high-level language. It supports both procedural and object-oriented paradigms, which were both utilized in modularizing various parts of the system.

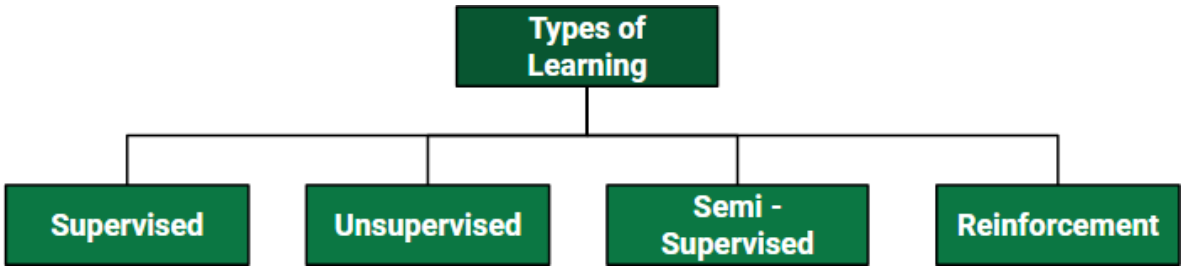
3.2 Machine Learning:

Machine Learning is a subset of Artificial Intelligence that enables computer systems to learn from data and improve their performance on tasks without being explicitly programmed. It involves the development of algorithms that can recognize patterns, make decisions, and adapt to new information. Machine Learning models are trained using historical data to identify relationships and generate predictive outcomes.

Importance:

- **Demographic Classification:** Machine Learning enables automatic classification of human faces into specific age groups and genders using deep learning models.
- **Targeted Advertisement:** By analyzing the detected demographic, ML allows the system to display personalized advertisements tailored to the audience.
- **Real-time Inference:** Machine Learning models are capable of processing webcam feeds in real-time to provide immediate demographic insights.
- **Data Collection and Learning:** The system records demographic trends which can be used for future training of advanced recommendation models.
- **Automation of Manual Processes:** Instead of manually selecting advertisements, ML automates the display process based on visual analysis of the audience.

Types of Machine Learning:



Supervised Machine Learning: Supervised learning involves training a model on labeled data, where the input features are associated with corresponding output labels. The goal is to learn a mapping function from input to output that can be used to make predictions on new, unseen data. Supervised learning tasks include classification, regression, and ranking.

Unsupervised Machine Learning: Unsupervised learning involves training a model on unlabeled data, where the algorithm tries to find patterns, structures, or relationships in the

data without explicit supervision. Unsupervised learning tasks include clustering, dimensionality reduction, and anomaly detection.

Reinforcement Learning: Reinforcement learning involves training an agent to interact with an environment and learn optimal behavior through trial and error. The agent receives feedback or rewards based on its actions and adjusts its strategy to maximize cumulative reward over time. Reinforcement learning is used in various applications such as game playing, robotics, and autonomous systems.

Supervised learning is characterized by the presence of input data paired with correct output labels, enabling a model to learn from examples and predict future outcomes on unseen data.

In the context of the project, Hugging Face models were utilized to detect gender and age of individuals captured through a live webcam feed. These models had been previously trained on comprehensive facial image datasets, where each image was annotated with its corresponding gender and age group. The training process involved exposing the model to a variety of facial features and associated demographic labels, allowing it to identify patterns and correlations that distinguish different age ranges and gender identities.

The reason for employing supervised learning lies in the nature of the task as the goal was to generalize the mapping from facial features to demographic categories.

3.3 Frameworks:

1. **Flask**
    - **Definition:** Flask is a lightweight and micro web framework written in Python. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. Flask provides tools, libraries, and technologies that allow developers to build web applications and APIs efficiently.
    - **Overview:** Flask acts as the core backend web server that manages routing, request handling, API development, and integration with computer vision modules. It

connects the UI with the demographic detection system and database storage.

- **Architecture:** Flask follows the Model-View-Controller (MVC) architecture Model handles the data logic, View renders the frontend interface, Controller contains the application logic that connects inputs to the model and views.
- **Features:**
    1. Minimal and flexible design.
    2. Built-in development server and debugger.
    3. RESTful request handling.
    4. Extension support for forms, authentication.

- **Advantages:**
    1. **Lightweight and Simple-** Flask is minimalistic, with a small core and easy-to-understand codebase.
    2. **Flexible and Easy to Customize-** Developers can add only the components they need, avoiding unnecessary complexity.
    3. **Fast Development-** Ideal for building prototypes and MVPs quickly due to its simplicity.
    4. **Large Community and Extensions-** Flask has a supportive community and many extensions for adding features like authentication, databases, etc.
    5. **Easy Integration with Other Tools-** Easily integrates with front-end libraries, databases (like SQLite, PostgreSQL), and APIs.
    6. **Modular Design:** Supports modular code through Blueprints, making it easier to scale applications.
    7. **Clear and Readable Code:** Promotes clean and readable code, helpful for beginners and teams.

3.4 Packages and Libraries:

1. NumPy (Numerical Python):

- **Overview:** NumPy is a fundamental package for scientific computing with Python, providing support for multi-dimensional arrays and matrices.
- **Features:**
    - Array manipulation: NumPy offers a wide range of functions and operations for manipulating arrays, including indexing, slicing, reshaping, and

concatenation.

- Mathematical operations: NumPy provides mathematical functions for performing element-wise operations, linear algebra routines, Fourier analysis, and random number generation.
- Broadcasting: NumPy's broadcasting mechanism allows for efficient computation of operations on arrays with different shapes and sizes.

- **Performance:**
  - NumPy's efficient implementation of array operations and memory management makes it ideal for handling large datasets and numerical computations in machine learning projects.
  - It is optimized for speed and memory usage, with core routines implemented in C and Fortran, ensuring high performance and scalability for computationally intensive tasks.

2. **OpenCV:**

- **Overview:**

  OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides a comprehensive suite of tools for image and video processing, object detection, face recognition, and real-time computer vision applications. It serves as the **primary interface for capturing real-time video frames from the webcam**, detecting faces within those frames, and performing preprocessing.

- **Features:**

1. **Real-Time Webcam Video Capture:**

- Utilized `cv2.VideoCapture()` to capture continuous video feed from the laptop's front camera.
- Ensures frame-by-frame reading for real-time processing.

- **Performance:**

1. Speed: OpenCV processes frames in real time (~25–30 FPS possible), ensuring that age and gender predictions are fast enough for user-facing applications.
2. Accuracy Aid: Although OpenCV alone doesn't predict age or gender, its accurate face detection and preprocessing play a critical role in ensuring the downstream Hugging Face model receive clean, standardized input, improving their prediction reliability.

2. **Hugging Face Model:**

- **Overview:** Hugging Face is a leading open-source platform and AI company known for its extensive collection of state-of-the-art machine learning models, particularly in natural language processing, computer vision, and audio. It provides a collaborative model-sharing hub called the Hugging Face Model Hub, where developers and researchers can host, download, and fine-tune pre-trained models.

  In this project, Hugging Face plays a critical role by providing two pre-trained deep learning models:

  A) nateraw/vit-age-classifier for age group classification

  B) rizvandwiki/gender-classification for gender prediction

  These models are integrated into the system to perform real-time demographic analysis using webcam input.

- **Features:**

1. **Pre-trained Deep Learning Models:** Enables quick deployment without the need to train models from scratch.
2. **Transformers and CNN-Based Architectures:** Utilizes advanced neural networks like Vision Transformers (ViT) and CNN.
3. **Easy Integration with Python and PyTorch:** Supports seamless integration using the transformers and torch libraries.
4. **Model Documentation and Versioning:** Each model comes with detailed descriptions, usage examples, and version control.
5. **Cloud-Based Hosting:** Models are hosted on the Hugging Face Hub, ensuring availability and scalability.

- **Performance:**

1. **Speed:** The Hugging models offer fast inference, which allows them to operate in real time alongside OpenCV's face detection.
2. **Accuracy:** Their robustness across multiple lighting and pose variations makes them ideal for a retail or smart kiosk environment.

3. **Face Recognition**

- **Overview:** The face recognition library is a powerful and user-friendly Python library built on top of dlib, which is known for its highly accurate face recognition capabilities using deep learning. In this project, face recognition plays a critical role in ensuring unique face identification during real-time webcam feed

processing.

- **Features:**

  1. Seamless Integration with SQLite and OpenCV: `face recognition` works in tandem with OpenCV's face detection module to capture the face and then encodes it for uniqueness checking.

- **Performance:**
  1. **Real-Time Capability:** The library offers fast encoding and comparison, typically within milliseconds per face, allowing it to run in real time alongside video capture and age-gender prediction.
  2. **Efficient Memory and CPU Usage:** Even without GPU acceleration, face recognition performs reliably on standard hardware.

4. **SQLite:**

- **Overview:** SQLite is a lightweight, embedded, serverless relational database management system. It stores the entire database as a single .db file on the disk, making it an ideal choice for small to medium-scale applications like your Smart Advertisement System. In your project, SQLite serves as the backend database for storing demographic data (age, gender, timestamps), tracking unique face entries, and enabling dashboard analytics.
- **Features:**

  1. **Demographic Data Storage:**

     - Every time a new unique face is detected, the predicted age and gender along with the timestamp are stored in the `demographics` table of the `advertisement.db` SQLite database.
     - This data is later used for filtering statistics and generating dashboard charts.

  2. **Efficient Data Filtering for Dashboard:**

     - Through SQL queries executed from Flask (e.g., /filter_data route), the application retrieves filtered counts based on gender, age group, and time range.

- **Performance:** Lightweight & Fast Execution

4. SYSTEM DESIGN & UML DIAGRAMS

System design is transition from a user-oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

4.1 SOFTWARE DESIGN:

In designing the software following principles are followed:

- **Modularity and partitioning**: Software is designed such that, each system should consist of hierarchy of modules and serve to partition into separate function.
- **Coupling:** Modules should have little dependence on other modules of a system.
- **Cohesion:** Modules should carry out in a single processing function.
- **Shared use:** Avoid duplication by allowing a single module be called by other that need the function it provides.

4.2 ARCHITECTURE:

Architecture diagram is a diagram of a system, in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation.

A SMS user for who the application looks like a user interface actually consists of a database called as SQLite that comes along with Android SDK and need no other installation. This is the database that is used to store and retrieve information. This is an application that is developed in java and hence all its features apply here as well such as platform independence, data hiding.

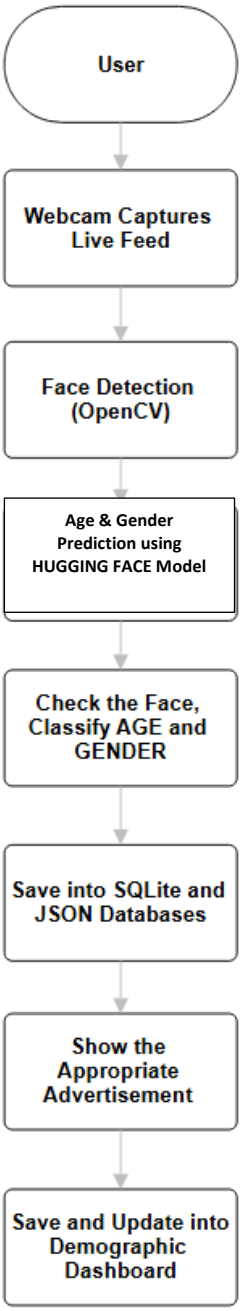


FIGURE 4.2: SYSTEM FLOW DIAGRAM

4.3 UNIFIED MODELING LANGUAGE (UML):

The unified modeling is a standard language for specifying, visualizing, constructing and documenting the system and its components is a graphical language which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure and control information about the systems.

Depending on the development culture, some of these artifacts are treated more or less formally than others. Such artifacts are not only the deliverables of a project; they are also critical in controlling, measuring, and communicating about a system during its development and after its deployment.

The UML addresses the documentation of a system's architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UML provides a language for modeling the activities of project planning and release management.

4.3.1 BUILDING BLOCKS OF UML:

The vocabulary of the UML encompasses three kinds of building blocks:

- ✓ Things.
- ✓ Relationships.
- ✓ Diagrams.

4.3.1.1 Things in the UML:

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

There are four kinds of things in the UML:

- ✓ Structural things.
- ✓ Behavioral things.
- ✓ Grouping things.
- ✓ Annotational things.

1. **Structural things** are the nouns of UML models. The structural things used in the

project design are:

- ✓ First, a **class** is a description of a set of objects that share the same attributes, operations, relationships and semantics.

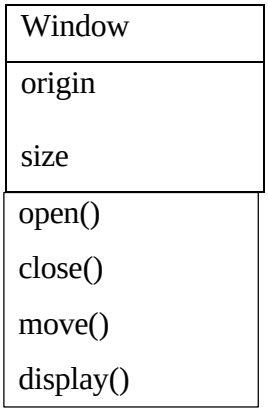


Fig: Classes

- ✓ Second, a **use case** is a description of set of sequence of actions that a system performs that yields an observable result of value to particular actor.

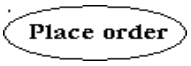


Fig: Use Cases

- ✓ Third, a node is a physical element that exists at runtime and represents a computational resource, generally having at least some memory and often processing capability.

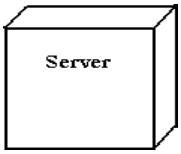


Fig: Nodes

2. **Behavioral things** are the dynamic parts of UML models. The behavioral thing used is:
   - ✓ Interaction: An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves a number of other elements, including messages, action sequences (the behavior invoked by a message, and links (the connection between objects).

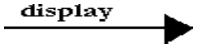


Fig: Messages

4.3.1.2 Relationships in the UML:

There are four kinds of relationships in the UML:

- ✓ Dependency.
- ✓ Association.
- ✓ Generalization.
- ✓ Realization.

✓ A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).

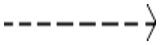


Fig: Dependencies

✓ An **association** is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.

Fig: Association

✓ A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element (the parent).

Fig: Generalization

✓ A **realization** is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.

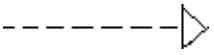


Fig: Realization

4.3.2 UML DIAGRAMS:

4.3.2.1 CLASS DIAGRAM:

A class is a representation of an object and, in many ways; it is simply a template from which objects are created. Classes form the main building blocks of an object-oriented application. Although thousands of students attend the university, you would only model one class, called Student, which would represent the represent the entire collection of students.

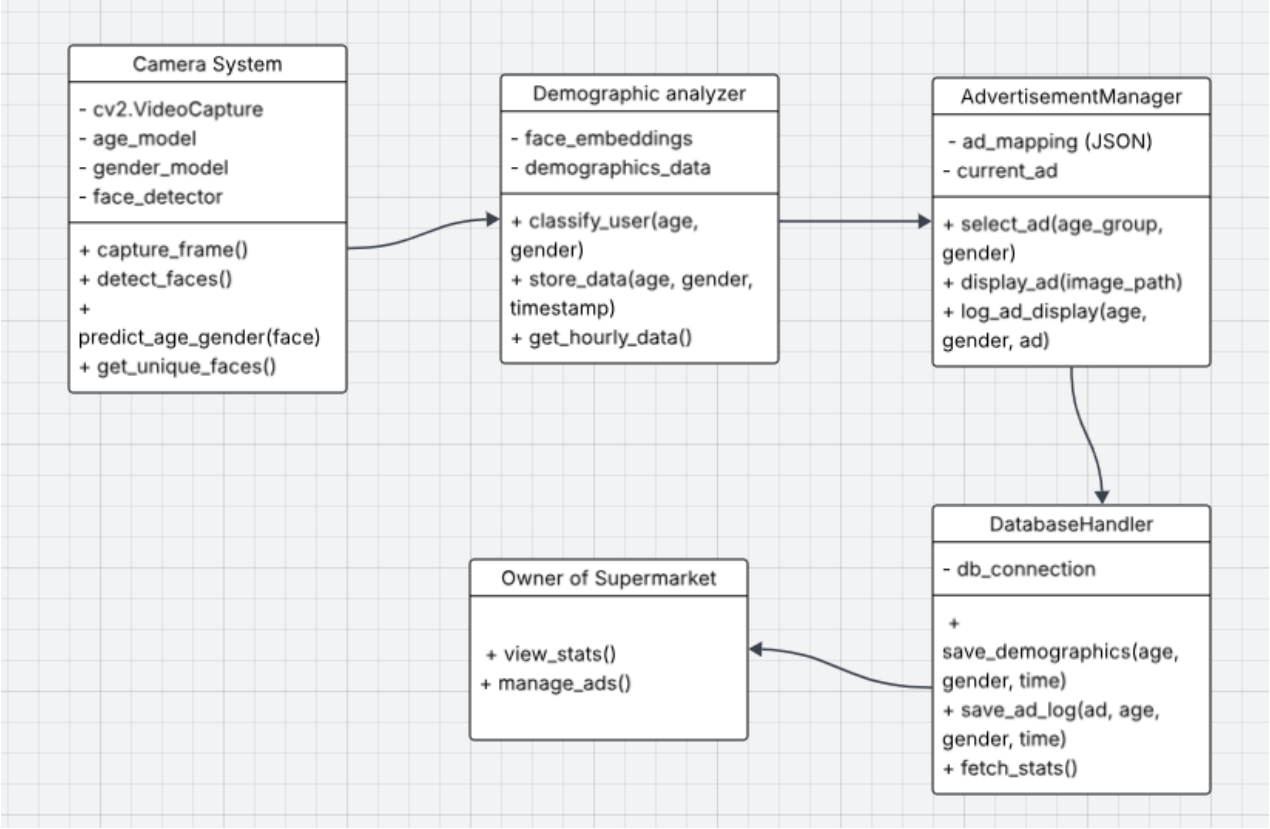


FIGURE 4.3.2.1: CLASS DIAGRAM

4.3.2.2 USE CASE DIAGRAM:

A use case diagram is a graph of actors set of use cases enclosed by a system boundary, communication associations between the actors and users and generalization among use cases. The use case model defines the outside (actors) and inside (use case) of the system's behavior.

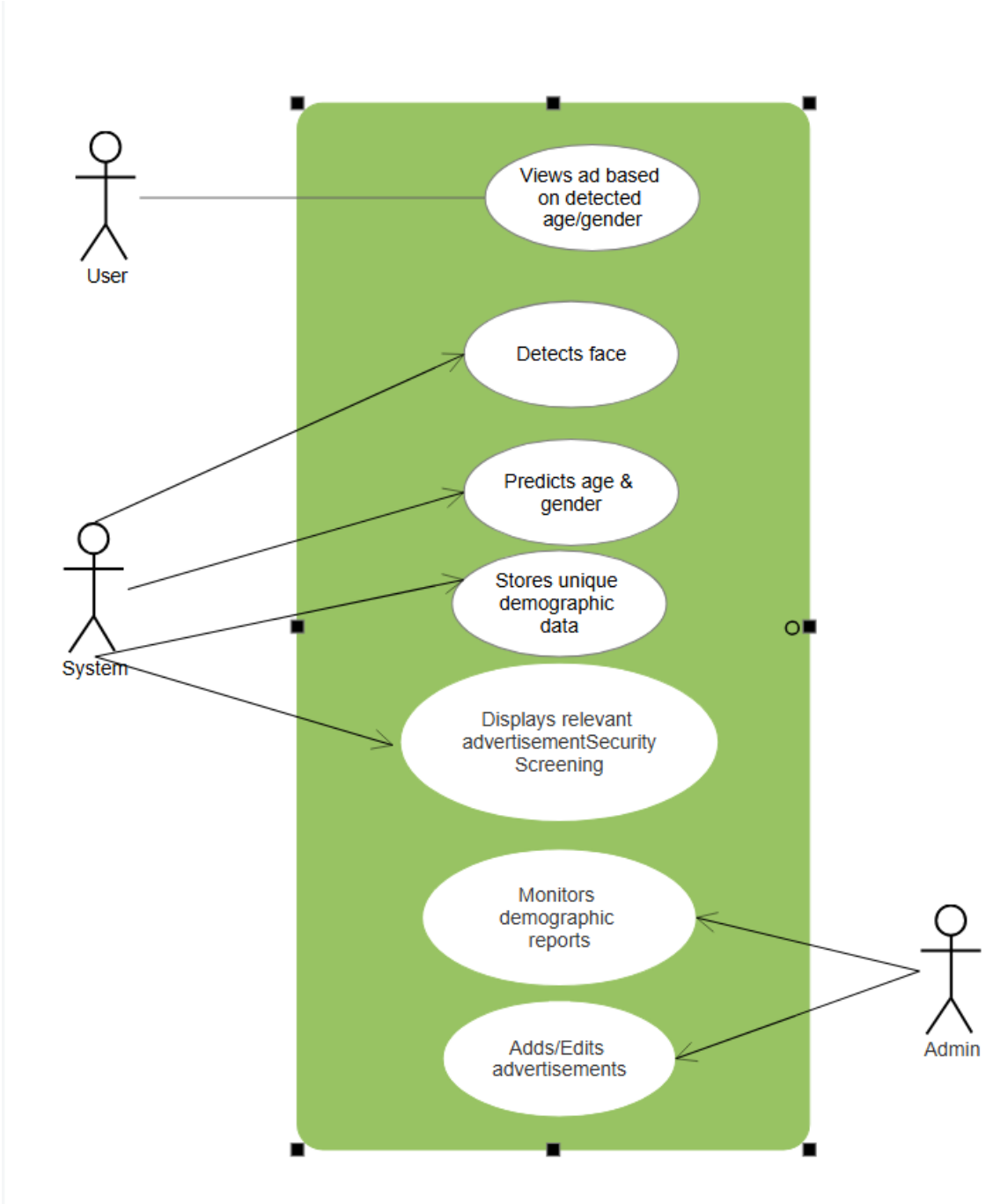


FIGURE 4.3.2.2: USE CASE DIAGRAM

4.3.2.3 SEQUENCE DIAGRAM:

Sequence diagram are used to represent the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interactions of the header elements, which are displayed horizontally at the top of the diagram.

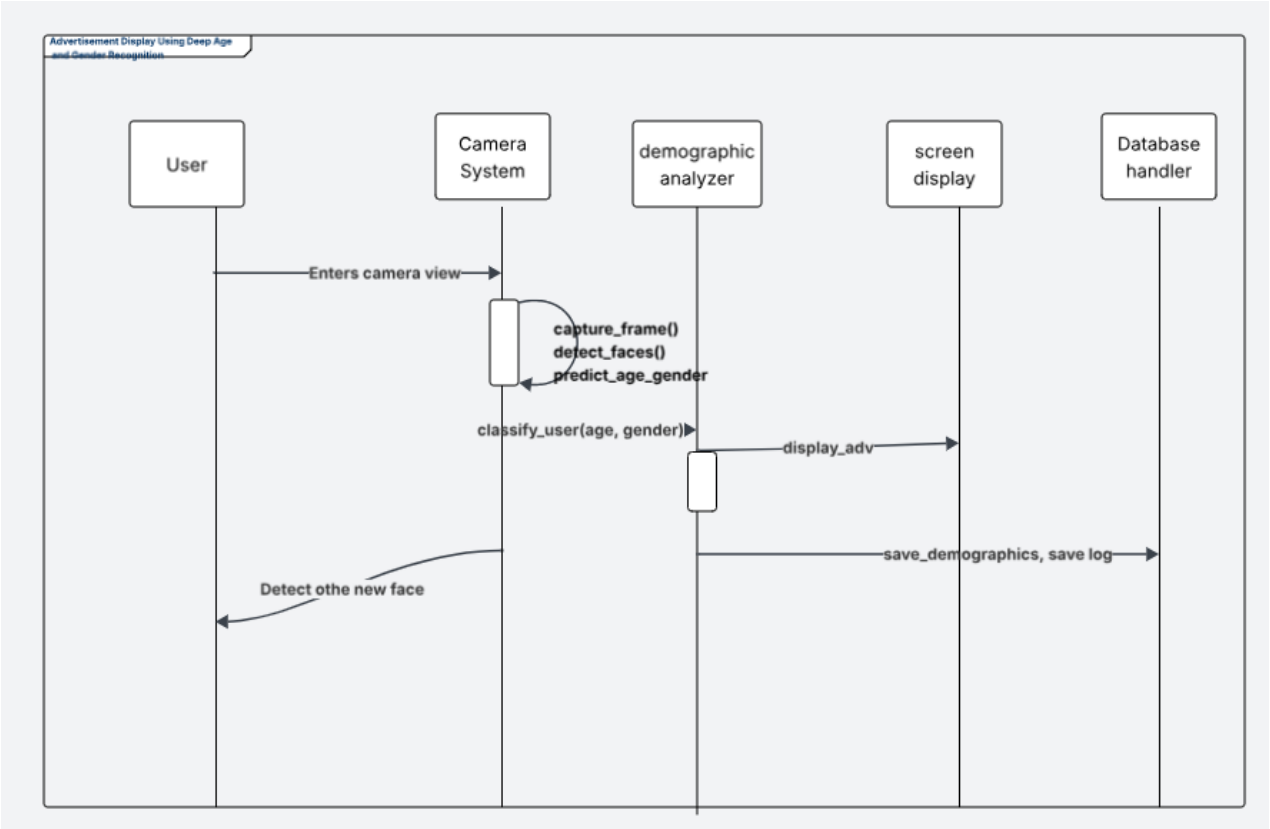


FIGURE 4.3.2.3: SEQUENCE DIAGRAM

4.3.2.4 ACTIVITY DIAGRAM:

Activity diagram represent the business and operational workflows of a system. An Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state.

So, what is the importance of an Activity diagram, as opposed to a State diagram? A State diagram shows the different states an object is in during the lifecycle of its existence in the system, and the transitions in the states of the objects. These transitions depict the activities causing these transitions, shown by arrows.

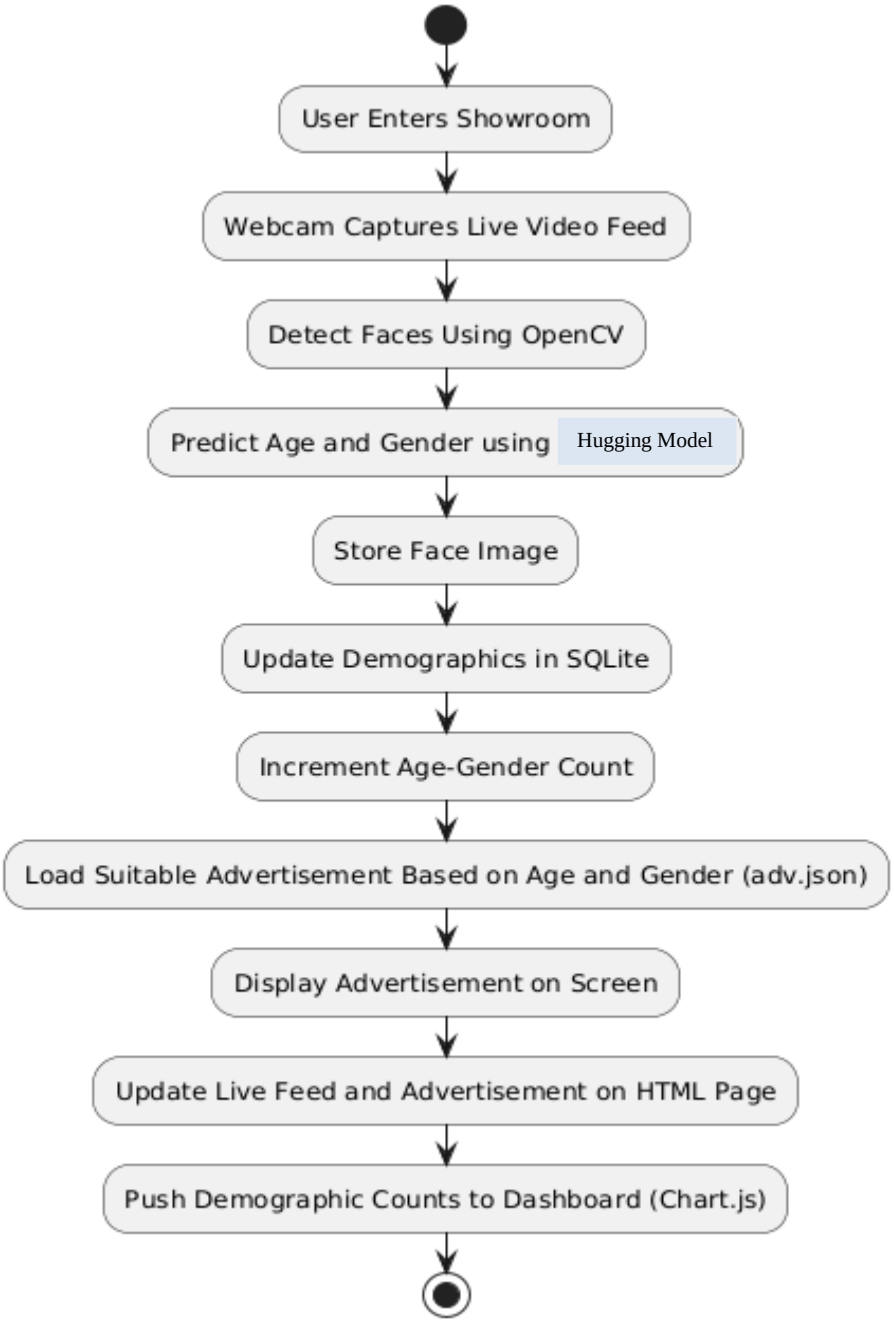


FIGURE 4.3.2.4: ACTIVITY DIAGRAM

4.3.2.6 COMPONENT DIAGRAM:

In the Unified Modeling Language, a **Component diagram** depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

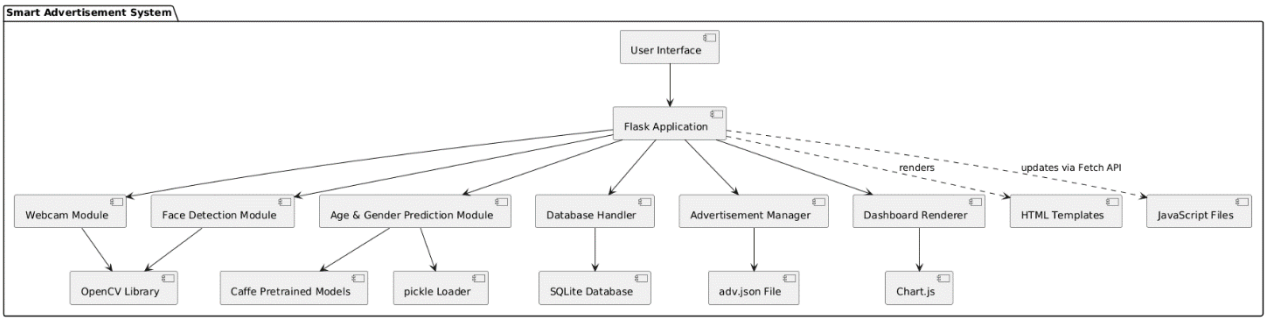


FIGURE 4.3.2.5: COMPONENT DIAGRAM

4.3.2.7 DATA FLOW DIAGRAM:

Data flow diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. So deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

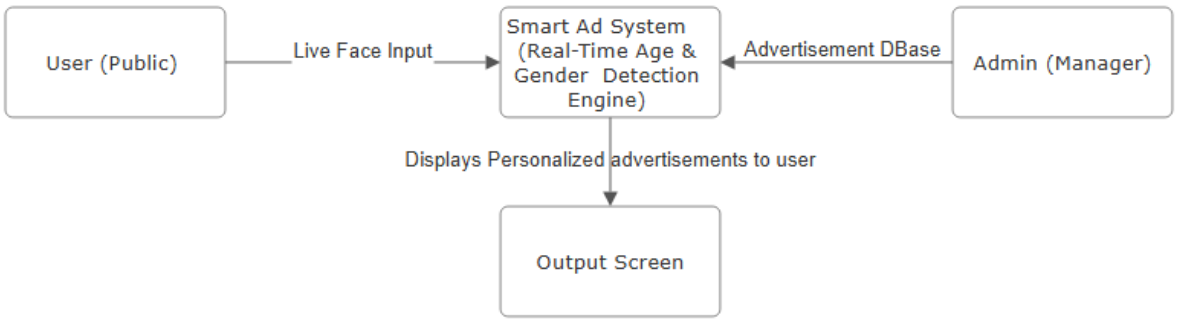


FIGURE 4.3.2.6: DFD -0 DIAGRAM

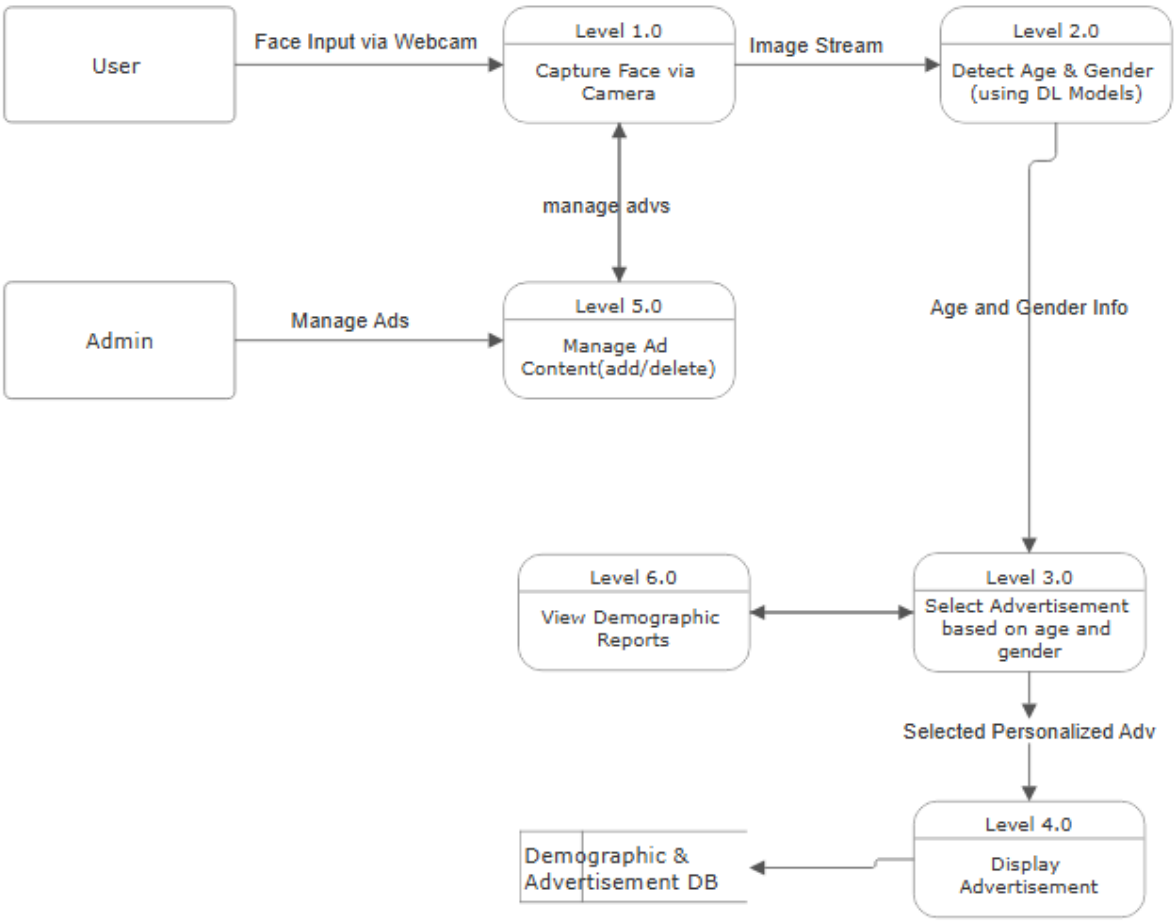


FIGURE 4.3.2.7: DFD -1 DIAGRAM

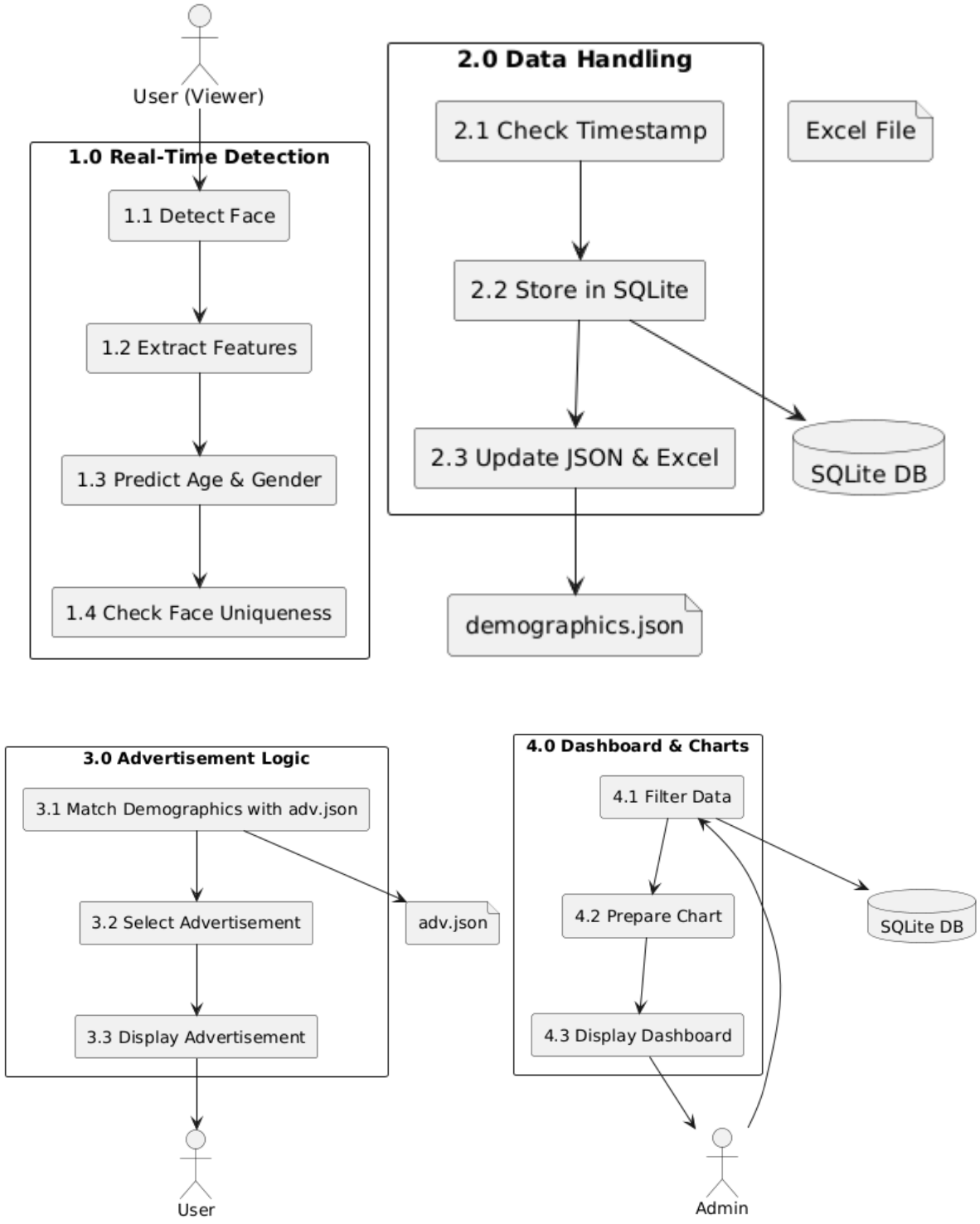


FIGURE 4.3.2.8: DFD -2 DIAGRAM

5. INPUT/OUTPUT DESIGN

5.1 INPUT DESIGN:

The input screen of the Smart Advertisement System serves as the initial interaction point for the system's core functionality—real-time age and gender detection using webcam feed. As soon as the system is launched, the user is greeted with a visually organized webpage (index.html) that presents a live camera feed on the left side. This input feed is captured directly from the system's front-facing webcam, continuously scanning for human faces. The webcam acts as the primary input device, feeding video frames into the detection pipeline, where faces are localized using OpenCV's Haar cascade classifiers or deep learning-based detectors. The input screen also contains system information such as the total number of unique visitors detected in the last hour, making it an intelligent, dynamic entry point into the system.

In addition to webcam-based input, the system supports backend data handling triggered automatically when a new unique face is recognized. The system does not require any manual data entry from the user; instead, it relies entirely on passive visual input and internal logic to detect, classify, and store demographic data. This hands-free input mechanism is critical for real-time advertisement systems, especially in commercial environments like malls or showrooms, where user interaction must be minimal and non-intrusive. The input screen is designed to be seamless and intuitive, ensuring that the user or passerby experiences no delay or complexity in engaging with the system.

5.2 OUTPUT DESIGN:

The output screen of the system provides the dynamic response generated from the analyzed input data. Once a face is detected, and its age and gender are classified, the corresponding advertisement is immediately retrieved from the adv.json mapping file. The selected advertisement image is then displayed on the right side of the index.html screen in real time, reflecting personalized content tailored to the detected demographic. This output screen also visually updates every few seconds with smooth transitions to accommodate new inputs and maintain user engagement. The real-time advertisement display ensures the system adapts instantly to environmental changes and new visitor entries.

Apart from the advertisement display, the output includes a visual count of unique visitors and updates demographic statistics in the backend. A separate output view is available on the /dashboard page, where stakeholders can analyze demographic trends over time using interactive bar charts powered by Chart.js. Filters allow users to narrow down demographic views by gender, age group, or date range.

6. IMPLEMENTATION

6.1 app.py:

```
import cv2
import numpy as np
import threading
import json
import time
import os
import sqlite3
import logging
from flask import Flask, render_template, Response, jsonify, request
from datetime import datetime, timedelta
from io import StringIO
import csv
from utils.prediction_utils import predict_age_gender
# Configure logging to file and console
logging.basicConfig(   level=logging.DEBUG,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[       logging.FileHandler('app.log'),
        logging.StreamHandler()    ])
app = Flask(__name__)
# Configurations
DB_PATH = "advertisement.db"
ADS_JSON_PATH = "ads.json"
VIDEO_WIDTH, VIDEO_HEIGHT = 640, 480
FRAME_SKIP = 3
# Video Capture
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    logging.error("Failed to open webcam")
cap.set(cv2.CAP_PROP_FRAME_WIDTH, VIDEO_WIDTH)
```

```
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, VIDEO_HEIGHT)
# Global States
lock = threading.Lock()
is_detecting = False
detection_thread = None
current_ad = "static/default_ad.jpg"
current_gender = None
current_age = None
last_detections = {}
# Ensure database exists
def init_db():
    try:
        conn = sqlite3.sqlite3.connect(DB_PATH)
        c = conn.cursor()
        c.execute("""
            CREATE TABLE IF NOT EXISTS demographics (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                gender TEXT,
                age TEXT,
                timestamp TEXT
            )       """)
        c.execute("CREATE INDEX IF NOT EXISTS idx_demographics ON demographics (gender,
age, timestamp)")
        conn.commit()
        logging.info("Database and index initialized successfully")
    except sqlite3.Error as e:
        logging.error(f"Database initialization error: {e}")
    finally:
        conn.close()
# Load ads from JSON
def load_ads():
    try:        with open(ADS_JSON_PATH, 'r') as f:
            ads = json.load(f)
```

```
            logging.debug(f"Loaded {len(ads)} ads from {ADS_JSON_PATH}")
            return ads
        except FileNotFoundError:
            logging.error(f"{ADS_JSON_PATH} not found")
            return []
        except json.JSONDecodeError:
            logging.error(f"Invalid JSON in {ADS_JSON_PATH}")
            return []
    def detect_faces(frame):
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
    'haarcascade_frontalface_default.xml')
        if face_cascade.empty():
            logging.error("Failed to load haarcascade_frontalface_default.xml")
            return []
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
        return faces
    def save_demographics(gender, age):
        with lock:
            detection_key = f"{gender}_{age}"
            current_time = datetime.now()
            if detection_key in last_detections:
                if (current_time - last_detections[detection_key]).total_seconds() < 2:
                    logging.debug(f"Skipped duplicate detection: {detection_key}")
                    return
            last_detections[detection_key] = current_time
            for key in list(last_detections.keys()):
                if (current_time - last_detections[key]).total_seconds() > 3600:
                    del last_detections[key]
            try:
                conn = sqlite3.connect(DB_PATH)
                c = conn.cursor()
                timestamp = current_time.strftime("%Y-%m-%d %H:%M:%S")
```

```
            c.execute("INSERT INTO demographics (gender, age, timestamp) VALUES (?, ?, ?)", (gender, age, timestamp))
            conn.commit()
            logging.debug(f"Saved demographics: gender={gender}, age={age}, timestamp={timestamp}")
        except sqlite3.Error as e:
            logging.error(f"Error saving demographics: {e}")
        finally:
            conn.close()
def detect():
    global current_gender, current_age
    frame_count = 0
    while is_detecting:
        success, frame = cap.read()
        if not success:
            logging.warning("Failed to read frame from webcam")
            continue
        if frame_count % FRAME_SKIP == 0:
            faces = detect_faces(frame)
            for (x, y, w, h) in faces:
                face_img = frame[y:y+h, x:x+w]
                if face_img.size == 0:
                    continue
                try:
                    gender, age = predict_age_gender(face_img)
                    with lock:
                        current_gender = gender
                        current_age = age
                    logging.debug(f"Detected: gender={gender}, age={age}")
                    save_demographics(gender, age)
                    color = (255, 0, 0) if gender == "male" else (147, 20, 255)
                    cv2.rectangle(frame, (x, y), (x+w, y+h), color, 2)
                    label = f"{gender}, {age}"
```

```
                cv2.putText(frame, label, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, color, 2)
            except Exception as e:
                logging.error(f"Error in predict_age_gender: {e}")
                continue
        frame_count += 1
        time.sleep(0.1)
def gen_frames():
    while True:
        success, frame = cap.read()
        if not success:
            logging.warning("Failed to read frame for video feed")
            break
        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/dashboard')
def dashboard():
    return render_template('dashboard.html')
@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
@app.route('/toggle_detection', methods=['POST'])
def toggle_detection():
    global is_detecting, detection_thread
    data = request.get_json()
    is_detecting = data.get('detecting', False)
    if is_detecting:
        detection_thread = threading.Thread(target=detect)
        detection_thread.start()
```

```
    logging.debug(f"Detection toggled: is_detecting={is_detecting}")
    return jsonify({'success': True, 'detecting': is_detecting})
@app.route('/get_demographic')
def get_demographic():
    with lock:
        data = {"gender": current_gender, "age": current_age}
        logging.debug(f"Returning demographic: {data}")
        return jsonify(data)
@app.route('/get_counts')
def get_counts():
    try:
        conn = sqlite3.connect(DB_PATH)
        c = conn.cursor()
        query = "SELECT gender, age, COUNT(*) FROM demographics GROUP BY gender, age"
        logging.debug(f"Executing query: {query}")
        c.execute(query)
        rows = c.fetchall()
        data = {"male": {}, "female": {}}
        for gender, age, count in rows:
            if gender in data:
                data[gender][age] = count
            else:
                logging.warning(f"Unexpected gender value: {gender}")
        conn.close()
        logging.debug(f"get_counts returned: {data}")
        return jsonify(data)
    except sqlite3.Error as e:
        logging.error(f"Error in get_counts: {e}")
        return jsonify({"male": {}, "female": {}}), 200

@app.route('/get_ad')
def get_ad():
    global current_gender, current_age
```

```
    with lock:
        gender = current_gender
        age = current_age
    if gender and age:
        ads = load_ads()
        for ad in ads:
            if ad["gender"].lower() == gender.lower() and ad["age"] == age:
                ad_image = ad["image_path"]
                if os.path.exists(ad_image):
                    logging.debug(f"Selected ad: {ad_image} for {gender}, {age}")
                    return jsonify({"ad_image": ad_image})
                else:
                    logging.warning(f"Ad image not found: {ad_image}")
        logging.warning(f"No ad found for {gender}, {age}")
    else:
        logging.debug("No demographic detected, returning default ad")
    ad_image = "static/default_ad.jpg"
    return jsonify({"ad_image": ad_image})
@app.route('/status')
def status():
    global current_gender, current_age
    with lock:
        no_face = current_gender is None or current_age is None
    logging.debug(f"Status: no_face={no_face}")
    return jsonify({"no_face": no_face})
@app.route('/get_unique_count')
```

7. TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation.

7.1 TESTING OBJECTIVES:

- To ensure that during operation the system will perform as per specification.
- To make sure that system meets the user requirements during operation.
- To make sure that during the operation, incorrect input, processing and output will be detected.
- To see that when correct inputs are fed to the system the outputs are correct.
- To verify that the controls incorporated in the same system as intended.
- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.

The software developed has been tested successfully using the following testing strategies and any errors that are encountered are corrected and again the part of the program or the procedure or function is put to testing until all the errors are removed. A successful test is one that uncovers an as yet undiscovered error.

Note that the result of the system testing will prove that the system is working correctly. It will give confidence to system designer, users of the system, prevent frustration during implementation process etc.

7.2 TESTING METHODOLOGIES:

- ✓ White box testing.
- ✓ Black box testing.
- ✓ Unit testing.
- ✓ Integration testing.
- ✓ User acceptance testing.

- ✓ Output testing.
- ✓ Validation testing.
- ✓ System testing.

1) White Box Testing:

White box testing is a testing case design method that uses the control structure of the procedure design to derive test cases. All independents path in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity. Here the customer is given three chances to enter a valid choice out of the given menu. After which the control exits the current menu.

2) Black Box Testing:

Black Box Testing attempts to find errors in following areas or categories, incorrect or missing functions, interface error, errors in data structures, performance error and initialization and termination error. Here all the input data must match the data type to become a valid entry.

3) Unit Testing:

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

4) Integration Testing:

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

- ✓ **Top Down Integration:**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module.

- ✓ **Bottom Up Integration:**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated.

5) User acceptance Testing:

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

6) Output Testing:

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

7) Validation Testing:

Validation testing is generally performed on the following fields:

- ✓ **Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

- ✓ **Preparation of Test Data:**

Taking various kinds of test data does the above testing. Preparation of test data plays a

vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

- ✓ **Using Live Test Data:**

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

- ✓ **Using Artificial Test Data:**

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

7.3 USER TRAINING:

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose, the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

7.4 MAINTAINENCE:

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future.

The coding and designing are simple and easy to understand which will make maintenance easier.

7.5 TESTING STRATEGY:

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding.

7.5.1 SYSTEM TESTING:

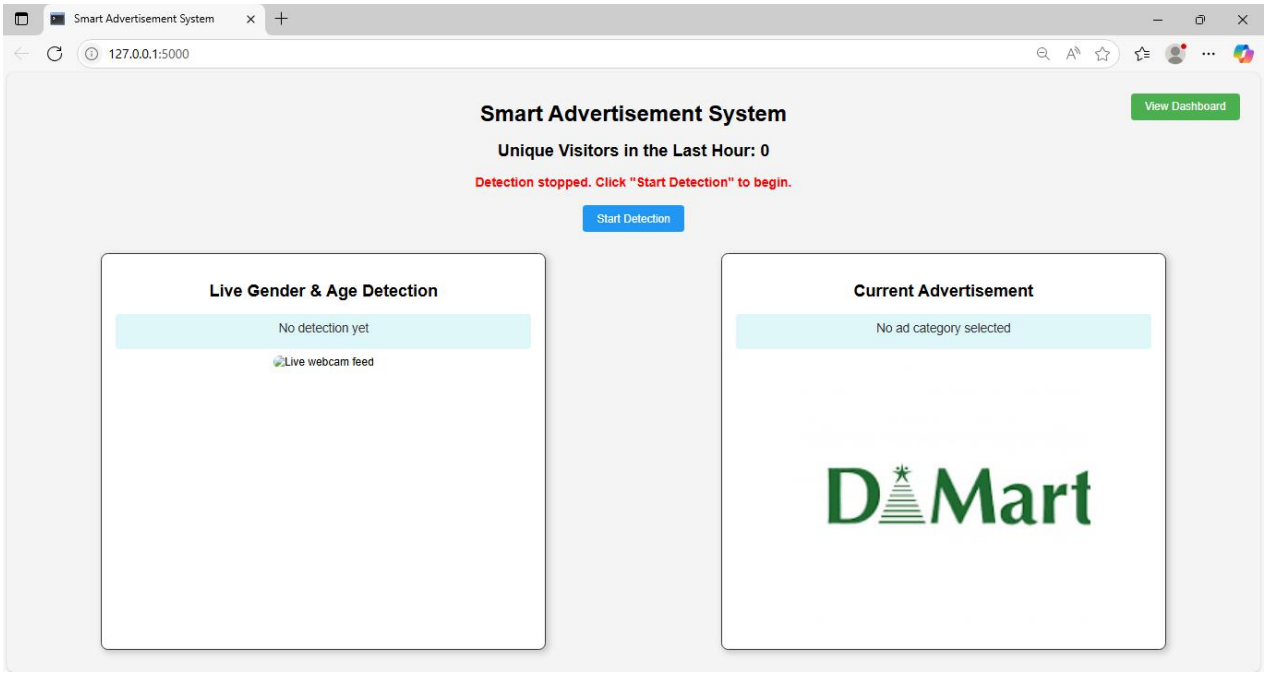
Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

7.5.2 UNIT TESTING:

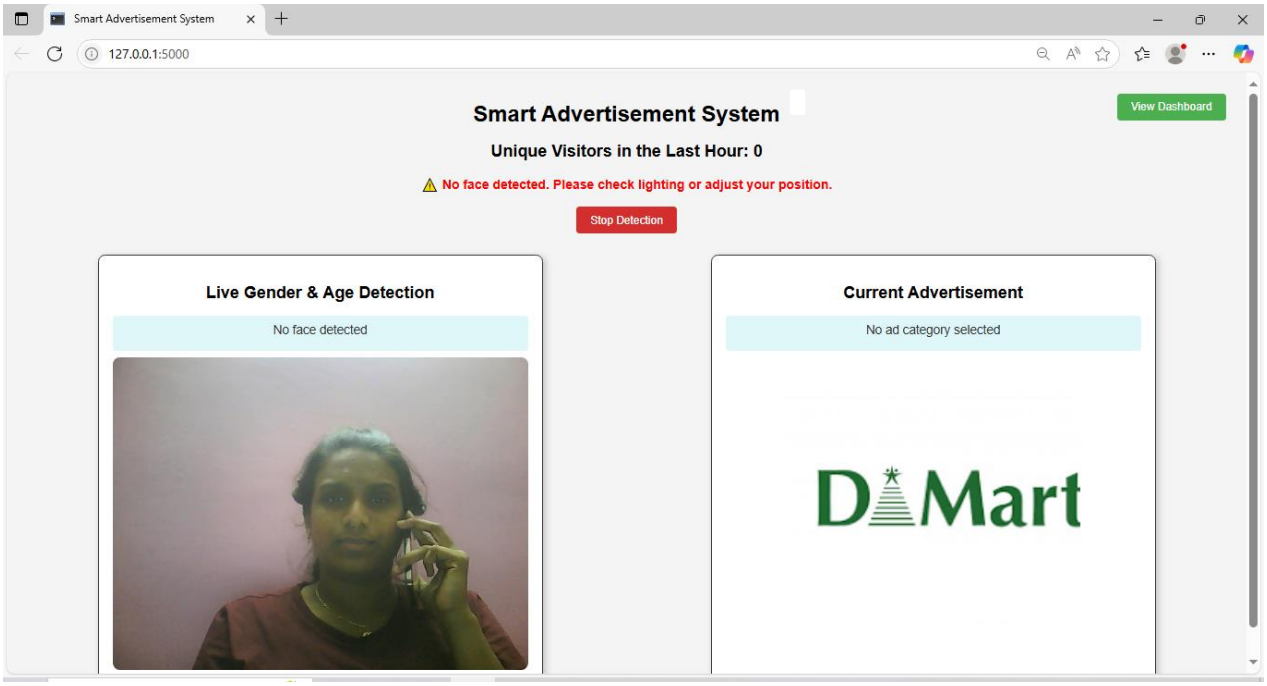
In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module. In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this.

7.6 TEST CASES:

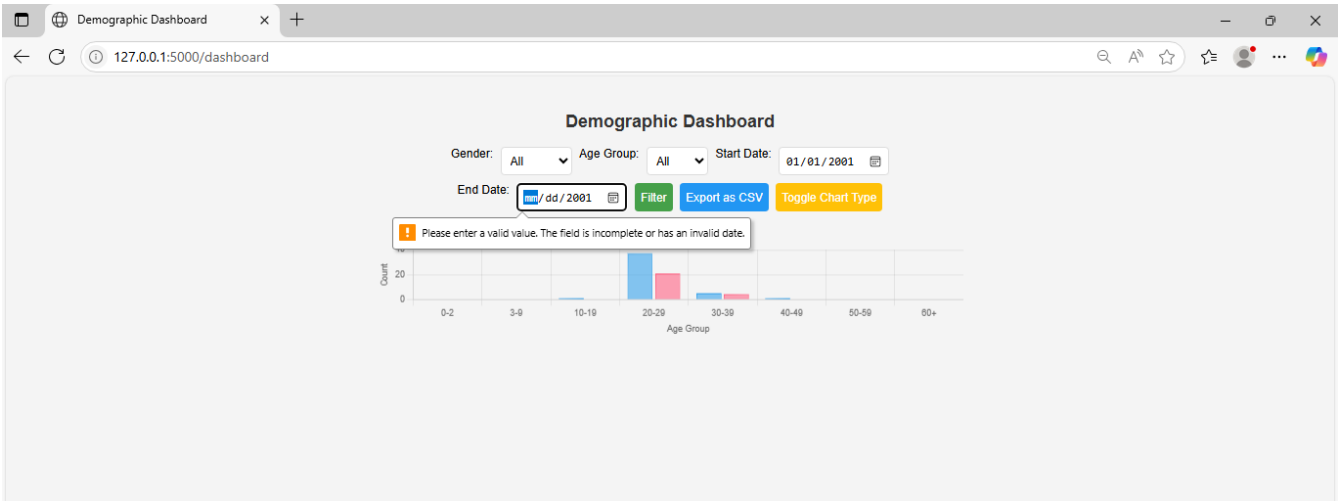
7.6.1 Test case 1: Checking for Start of Webcam.



7.6.2 Test case 2: Checking for Lighting and adjustment of user position.

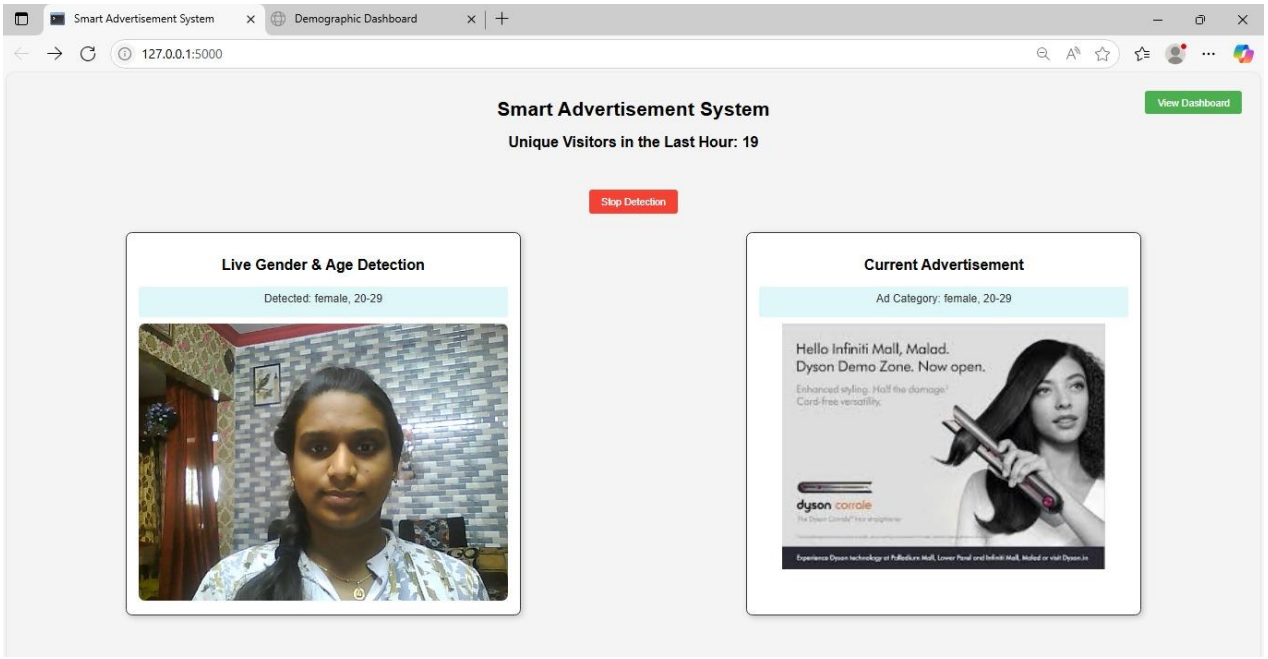


7.6.3 Test Case 3: Demographic date validation.

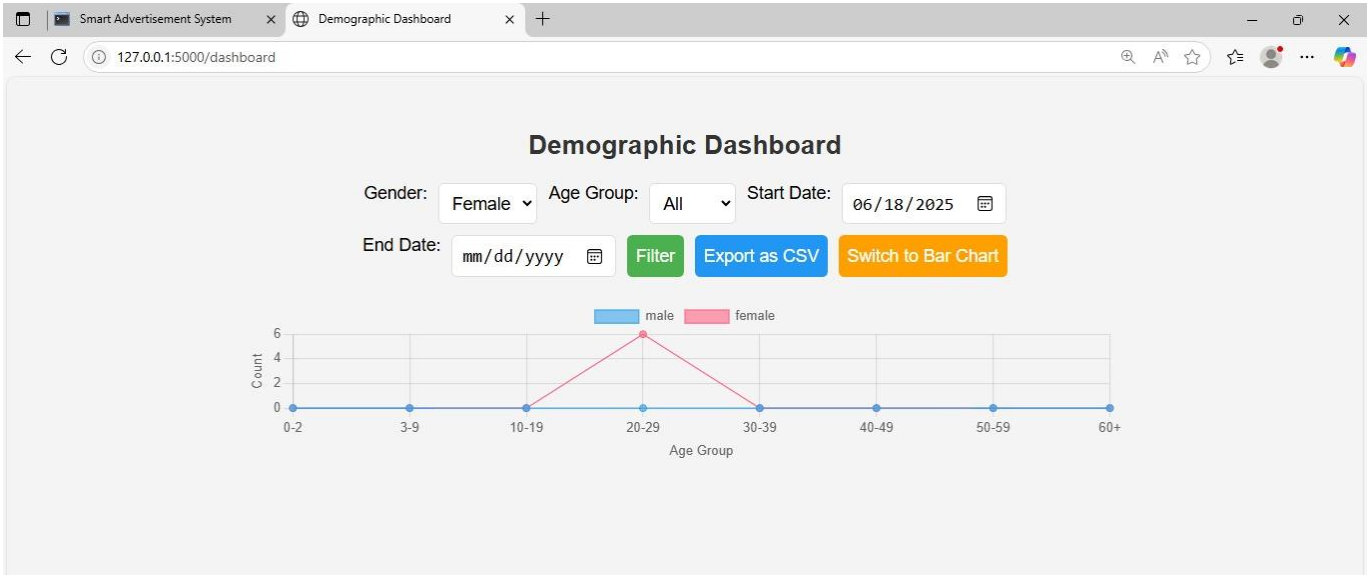


8. OUTPUT SCREENS

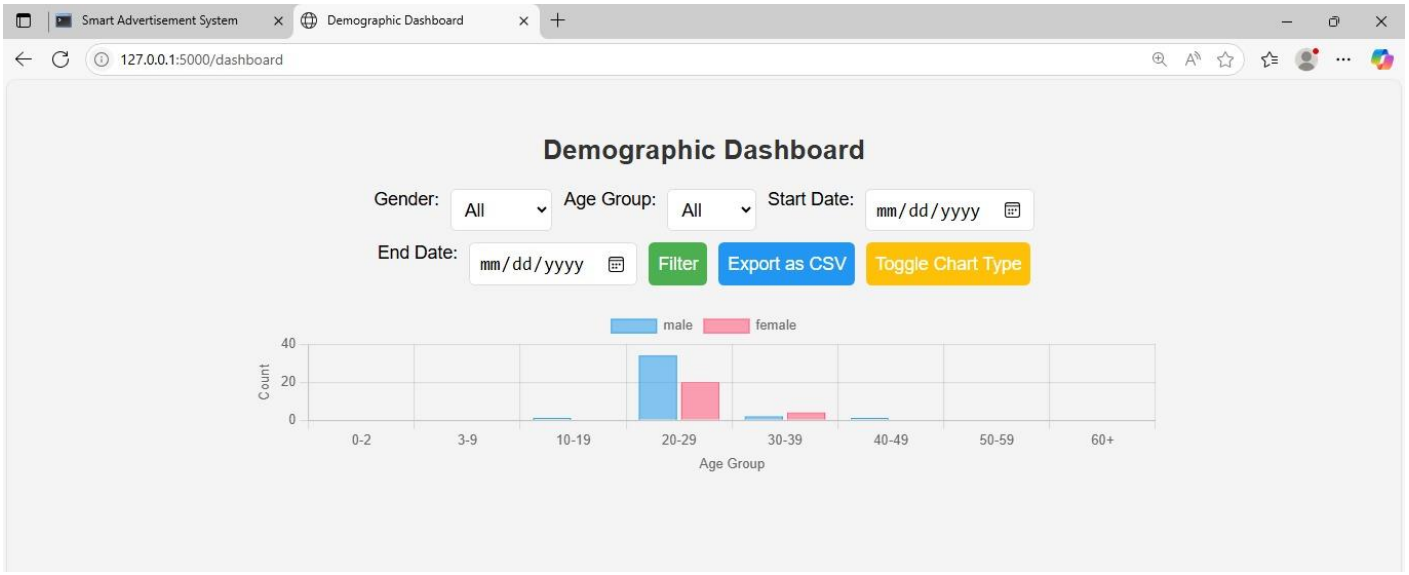
8.1 Advertisement Display Screen



8.2 Dashboard Processing Screen using Toggle Chart



8.3 Dashboard Processing Screen using Bar Chart



9. CONCLUSION AND FUTURE SCOPE

9.1 CONCLUSION:

The Smart Advertisement System leveraging real-time age and gender detection presents a significant step forward in the personalization of digital marketing and in-store customer engagement. Through the integration of computer vision, machine learning, and a Flask-based web framework, the project demonstrates a seamless blend of artificial intelligence with traditional advertising strategies. By identifying demographic traits of individuals through webcam-based input, the system dynamically displays targeted advertisements that are relevant to the observed audience segment enhancing both the user experience and marketing effectiveness.

In conclusion, the project serves as a scalable and practical prototype of AI-enhanced advertising systems. It can be further extended to include sentiment analysis, emotion recognition, or crowd analytics. As organizations and retailers move towards personalized customer interactions, this system lays a solid foundation for the future of smart, responsive advertising bridging the gap between technology and human behavior.

9.2 FUTURE SCOPE:

The future scope of the Smart Advertisement System is vast and promising. It can be enhanced by integrating emotion detection, eye-gaze tracking, or facial expression analysis to further refine advertisement targeting. The system could also incorporate voice recognition, behavioral analytics for deeper personalization. Expanding the database to store long-term visitor trends can help in predictive marketing. Integration with IoT-based display systems or mobile notifications can extend its reach. It can also be integrated in Residential Marts by storing entire community data, as analyze the products using advertisement, which has privacy and security for individual. Furthermore, implementing cloud-based analytics and real-time remote monitoring will make the system scalable across multiple locations, empowering businesses to deliver dynamic, data-driven advertisements with maximum customer engagement and efficiency.

10. BIBLIOGRAPHY

10.1 WEBSITES:

[1]. https://opencv.org/ - Official website of OpenCV - Library used for computer vision and image processing.

[2]. https://flask.palletsprojects.com/ - Documentation and usage of Flask web framework used for backend implementation.

[3]. https://github.com/spmallick/learnopencv - Practical OpenCV tutorials including age and gender detection models.

[4]. https://github.com/serengil/deepface - DeepFace library that combines multiple facial recognition and analysis models.

[5]. https://huggingface.co/nateraw/vit-age-classifier

[6]. https://www.pyimagesearch.com/ - Comprehensive tutorials on computer vision, OpenCV, and deep learning.

[7]. https://face-recognition.readthedocs.io/ - Documentation of the face_recognition Python library used in your project.

[8]. https://www.tensorflow.org/ - Machine learning framework often used in real-time detection systems.

[9]. https://scikit-learn.org/stable/ - Machine learning in Python - used with pickle for storing trained models.

[10]. https://chartjs.org/ - JavaScript library used for drawing charts in the dashboard.

[11]. https://sqlite.org/index.html - Official documentation and usage of SQLite, your database for storing demographics.

[12]. https://docs.python.org/3/ - Python programming language official documentation.

[13]. https://www.geeksforgeeks.org/ - Various articles and code samples on OpenCV, Flask, and database handling.

[14]. https://realpython.com/ - Professional tutorials and real-world Python web development with Flask.

[15]. https://towardsdatascience.com/ - Articles on machine learning, face detection, and smart ad systems.

[16]. https://huggingface.co/rizvandwiki/gender-classification

10.2 REFERENCES:

[1]. "Automated Attendance System Using Deep Learning" by Neeraja Goli, Shiva Kumar, Yashwanth and Madhav "https://github.com/Neeraja2303/AUTOMATED-ATTENDANCE-SYSTEM-USING-DEEP-LEARNING"

[2]. "Mirror Advertising — Mirror Advertising Screen—Pro Display", Pro Display, 2021. [Online]. Available:https://prodisplay.com/products/mirror-advertising-screen.

[3]. M. Ghazal, T. Al Hadithy, Y. Al Khalil, M. Akmal and H. Hajjdiab, "AMobile-Programmable Smart Mirror for Ambient IoT Environments,"2017 5th International Conference on Future Internet of Things andCloud Workshops (FiCloudW), 2017, pp. 240-245, doi: 10.1109/Fi-CloudW.2017.106.

[4]. T. Vrountas, "5 Examples That Show How Machine Learning isChanging Advertising", Instapage.com, 2021. [Online]. Available:https://instapage.com/blog/machine-learning-in-advertising.

[5]. M. Bonomo, G. Ciaccio, A. De Salve and S. E. Rombo, "CustomerRecommendation Based on Profile Matching and Customized Cam-paigns in On-Line Social Networks," 2019 IEEE/ACM InternationalConference on Advances in Social Networks Analysis and Mining(ASONAM), 2019, pp. 1155-1159, doi: 10.1145/3341161.3345621.

[6]. A. Kumar, M. Sharma, S. Gautam, R. Kumar and S. Raj, "AttendanceManagement System using Facial Recognition", 2020 InternationalConference on Decision Aid Sciences and Application (DASA), 2020.Available: 10.1109/dasa51403.2020.9317104

[7]. H. Leon, "How 5 brands have used facial recognitiontechnology - Digiday", Digiday, 2021. [Online]. Available:https://digiday.com/marketing/5-campaigns-used-facial-recognition-technology/.

[8]. R. Gautam and P. Mahara, "Perceptive advertising using standardisedfacial features", 2019 International Conference on Digitization (ICD),2019. Available: 10.1109/icd47981.2019.9105818.

[9]. S. Liew, M. Khalil-Hani, S. Ahmad Radzi and R. Bakhteri, "Gen-der classification: a convolutional neural network approach", TurkishJournal of Electrical Engineering & Computer Sciences, vol. 24, pp.1248-1264, 2016. Available: 10.3906/elk-1311-58.

[10]. W. Jiang et al., "An Empirical Study of Pre-Trained Model Reuse in the Hugging Face Deep Learning Model Registry," 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia, 2023, pp. 2463-2475, doi: 10.1109/ICSE48619.2023.00206.