Bilkent University

Department of Computer Engineering

Senior Design Project

Group No: 2332

Project Name: Cyclops

Final Report

21902348, Kaan Kurçer, kaan.kurcer@ug.bilkent.edu.tr

21902903, Osman Serhat Yılmaz, serhat.yilmaz@ug.bilkent.edu.tr

21903213, Ali Doğaç Urkaya, dogac.urkaya@ug.bilkent.edu.tr

21902358, Özgür Abi, ozgur.abi@ug.bilkent.edu.tr

21902035, Jankat Berslan Dinçer, berslan.dincer@ug.bilkent.edu.tr

Supervisor: Shervin Rahimzadeh Arashloo

Course Instructors: Erhan Dolak, Tağmaç Topal

19.05.2023

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS492.

Table of Contents

1. Introduction

Restaurants are visited by hundreds of customers every day. Different types of customers have different spending habits, visiting times, and preferences. The traffic of the restaurant also varies greatly for different hours of the day. Although restaurant owners already record the inside of their restaurants for security reasons, they are unable to collect and utilize the data produced by their customers. The information gained from monitoring the customers' count and preferences could be used to enhance both the user experience of the visitors and the profits of the restaurant owners. Cyclops aims to analyze the security feed of the restaurant to collect this information and provide it to the restaurant owner along with tips on how to improve their restaurant using said information.

Our system aims to keep track of customers in the restaurant while also providing detailed reports and statistics about customers. Therefore, the purpose of the system is to accurately track the number of customers while analyzing them at the same time.

The application will be able to track customer info of the restaurant and give precise and detailed information to restaurant owners. Also, since our system is meant to be used by restaurant owners or managers and not by engineers, UI will be easy to manage and navigate. Another aim of this application is to safely keep the customer data and protect their privacy.

While there already exists some apps that provide information about restaurant availability, these applications only provide limited availability data and not actual numbers, our application will be much more precise. For the restaurants, the app will provide them with useful insights on their customers.For this application computer vision, developed with python, will be utilized with cameras and it will be the base of our customer tracking and analysis systems. Our application will also be able to work even with crowded restaurants.

2. Requirements Details

Functional Requirements

- The user can sign up with their accounts.
- The user can choose between the customer and manager version of the site
- The program can generate a fullness amount of the restaurant
- The program can generate customer information

- The program can recognize individual customers
- The program can store information about tables in the restaurant
- Users can view information about various restaurants.
- Managers can view previous orders of customers
- Users can view graphs about restaurant availability by hour

Non Functional Requirements

i. Usability
   - System should be easily understandable by the restaurant staff
   - For the customers it should be easy to view restaurant info

ii. Performance
   - System should be able to keep track of customers pretty quickly
   - Stored information should be accessible without much lag

iii. Security
   - Data about customers should not be easily accessible
   - Each restaurants info should only be accessible by that restaurant

iv. Scalability
   - The app should work with decent size restaurants
   - Program should also be able to work with large number of customer info

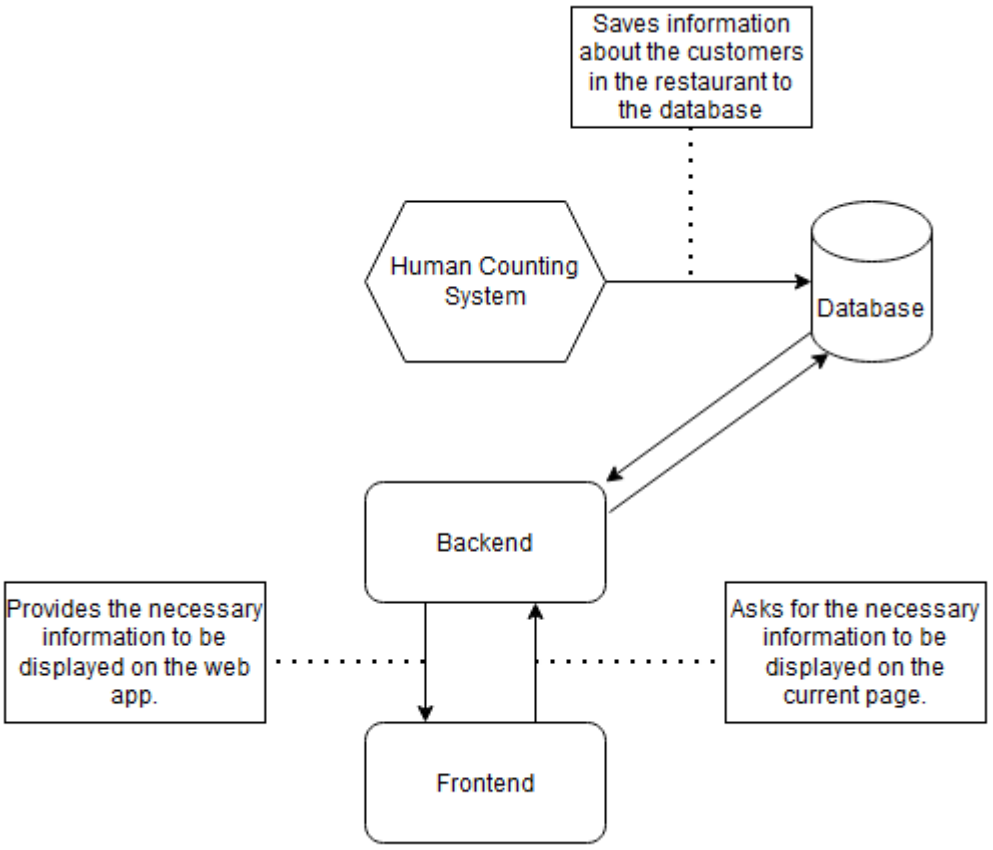
v. Reliability
   - The program should track customers with few or no errors

vi. Maintainability
   - Program and its components should be easy to keep track of and modify

3. Final Architecture and Design Details

3.1. Overview

The architecture of Cyclops is split into three main modules: human counting system, the frontend that users interact with, and the backend which ties the other two together. The human counting system and the backend communicate via the database(the human counting system updates the data there) whereas the backend and frontend communicate via HTTP requests. We've followed an MVC(Model-View-Controller) approach throughout the application.



3.2. Human Counting System

The human counting system is responsible for keeping track of the amount of people that are in the restaurant, as well as doing face recognition when they enter the restaurant. It uses a human detection model which is trained specifically for each restaurant(as the camera angles and image quality differ from restaurant to restaurant). The data that this system generates is written into the database, where it can be read by the backend.

3.3. Backend

The backend of the application is where the data related to the app is kept and manipulated. The backend is responsible for reading the data written by the human counting system and sending the relevant data to the frontend so that it can be displayed. Relevant information about the restaurants and users as well as customers are modeled as classes within the backend. These data are written into the database and modified as needed.

3.4. Frontend

The frontend of the application is where the user interacts with the app. Depending on which pages the user is displaying at that moment, necessary data is queried to the backend. The data is then displayed on the frontend for the user to see.

4. Development/Implementation Details

This part will be explained in 4 subsections, these subsections are frontend of the web project, backend of the web project, the human counting system and the face recognition system.

4.1. Frontend

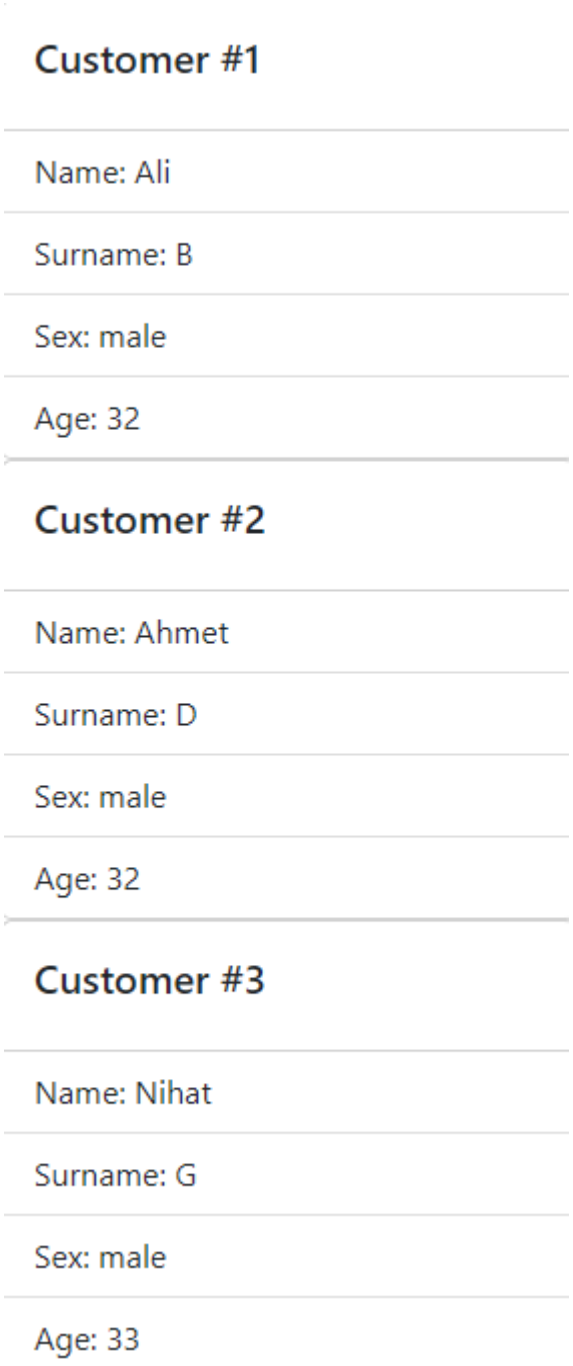
The frontend part of our project is divided into 2 parts for different users. The first one is for the restaurant owners and the staff to use, this project's data contains sensitive information therefore it can only be seen by the restaurant owner. The other part of the frontend is created for customers to see the number of people in any restaurant, this web project has data from many restaurants and displays them.

4.1.1. Website for the Restaurant Owner and Staff

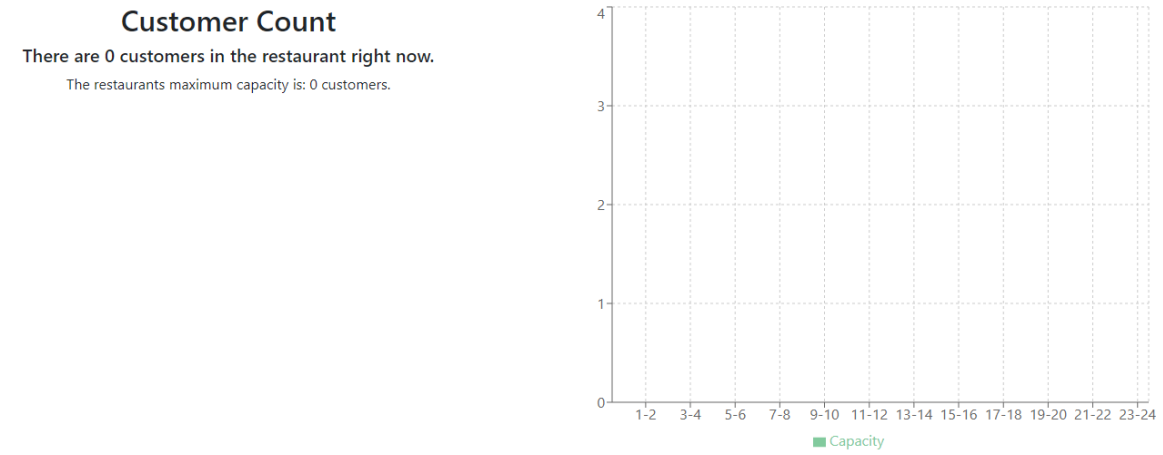
Home Page: A simple page that allows managers to choose which restaurant they will manage

Login/Register: Page for logging in and registering. Without a login users can not access the website for the managers. Customer website does not require users to login/register.

Customers Page:In the customer page, managers can see each customers present in the restaurant. Customers name and surname are available. All the customers that are present at the restaurant at that time are listed.



Restaurant Page: In this page customer count and live graph of the customer count is displayed. This tracks the precise number of customers.



Orders Page: This page display orders given by the customers. Customer name, order and table id will be displayed along with the order.

4.1.2. Website for Customers

4.2. Backend

The backend part of our project is divided into 2 parts similar to the frontend part. They will be explained in their respective subsections. Both parts are implemented using Java Spring, and have entities, repositories, services and controllers used for data manipulation.

4.2.1. Website for the Restaurant Owner and Staff

The application has entity classes for modeling all the data mentioned in the database part. It has the following controllers for managing communication between React and database:

- **CustomerController**: Has 2 endpoints for managing customer entities.
    1. getById: Returns a customer entity which has the specified id.
    2. all: Returns all customers in the database.
- **LoginController**: Has 3 endpoints for managing authentication.
    1. getEmail: Returns current users registration email. Returns “notAuthenticated” if user is not logged in.
    2. register: Registers a user with the specified information (email, username, password).
    3. perform_login: Not part of the LoginController but part of Spring Security, authenticates user with credentials specified at the login form.
- **OccupiedDataController**: Has 3 endpoints for managing table occupation data.
    1. gettoday: Returns all table occupation data in the current day.
    2. gettodaycount: Get today’s customer count.
    3. create: Creates an occupation data with specified table id and customer count.
- **OrderController**: Has 2 endpoints for managing food order data.
    1. getById: Returns a food order entity which has the specified id.
    2. all: Returns all orders.

4.2.2. Website for Customers

The application has entity classes for modeling all the data mentioned in the database part. It has the following controllers for managing communication between React and database:

- **ResInfoController**: Has 2 endpoints for managing restaurant information.

1. /{id}: Returns a restaurant information data which has the specified restaurant id.
2. all: Returns all restaurant information data.

- **LoginController**: Has 3 endpoints for managing authentication.
    1. getEmail: Returns current users registration email. Returns “notAuthenticated” if user is not logged in.
    2. register: Registers a user with the specified information (email, username, password).
    3. perform_login: Not part of the LoginController but part of Spring Security, authenticates user with credentials specified at the login form.

4.3. Human Counting System

One of the main functionalities of our project is to count the number of people that are in the restaurant at that moment and send this data to the backend for the web project to display it. In this part, the generation process of the human detection model and how these data are sent to the backend will be explained.

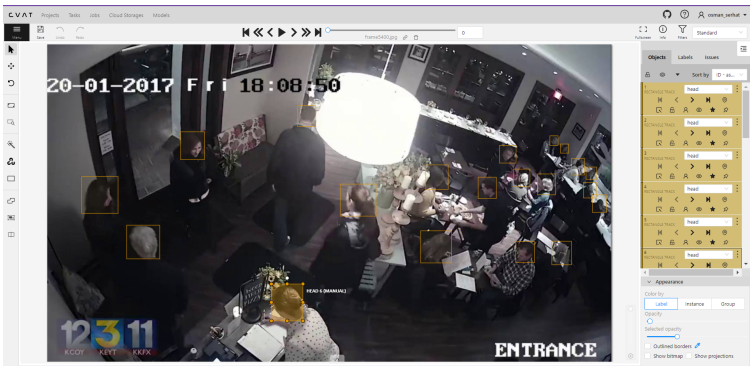
4.3.1. Human Detection Model

The model that detects humans in a restaurant is custom trained for every restaurant because it increases the accuracy of the model. In this process, the sample frames taken from the restaurant footage are labeled using the image labeling tool CVAT (Computer Vision Annotation Tool)

4.3.1.1. Image Labeling using CVAT

CVAT, which stands for Computer Vision Annotation Tool, is a platform that facilitates the annotation and labeling of images for various computer vision tasks. It provides a user-friendly interface and a range of annotation tools, making it suitable for both experts and non-experts in computer vision.

To use CVAT, the first step is to upload the images that require labeling. These images were obtained by dividing the restaurant footage and taking a sample every 20 frames. Once these images are uploaded, the annotators can access them through the CVAT interface. CVAT offers various annotation tools, such as bounding boxes, polygons, and key points, which can be used to label objects of interest in the images. The annotators can draw bounding boxes around the objects, assign labels to them, and fine-tune the annotations as needed. We used the tracker annotator to label the human heads in the example footage.



Once the images are labeled using CVAT, the next step is to generate YOLO (You Only Look Once) data from these labeled images. YOLO is a popular object detection algorithm that requires specific data formats for training. CVAT offers an export functionality that allows users to export the annotations in YOLO-compatible formats, such as YOLOv3 which is the one that we used. By exporting the annotations in these formats, the labeled images are used as training data for training YOLO models.

4.3.1.2. Training the model with Darknet

After YOLOv3 formatted data is obtained, to train an object detection model we used the Darknet framework. It begins by cloning the Darknet repository from GitHub and making necessary configurations in the Makefile to enable GPU and OpenCV support. Pre-trained weights for the Darknet model 53 is used. The training process is initiated using the modified configuration file and pre-trained weights. The Darknet framework sets up and trains a YOLOv3 model, making it suitable for object detection tasks.

4.3.2 Detection with OpenCV

After pre-trained YOLOv3 weights are obtained, we use OpenCV and Python for real-time object detection on a video using the YOLOv3 algorithm and save the average hourly count of a specific object in a database. The code uses the OpenCV library to load the pre-trained YOLOv3 weights and configuration files. It captures frames from the specified video file and resizes them for processing. The YOLOv3 model is then used to detect objects within each frame, and the detected objects are annotated with bounding boxes. The code

calculates the number of detected objects and tracks the hourly count by incrementing a frame counter. After every hour (3600 frames), the average hourly count is calculated and saved to a MySQL database which is connected to the backend of both web projects. The code continues this process until the end of the video. Overall, this code provides a practical solution for real-time object detection and automated counting of objects in videos.

4.4. Face Recognition System

Introducing our cutting-edge face recognition system, designed to revolutionize the way we cater to our valued customers at our restaurant. This system has been specifically developed to predict and identify our customers, enabling us to provide a personalized experience. With the ability to recognize and remember individual faces, our face recognition system ensures that each customer receives tailored service, allowing us to anticipate their preferences and deliver a level of satisfaction. We used OpenCV and Python to implement this functionality. Restaurants should place a camera near face height to the entrances for this process to work because face recognition requires high-quality images to distinguish the people properly and not mix in predictions.

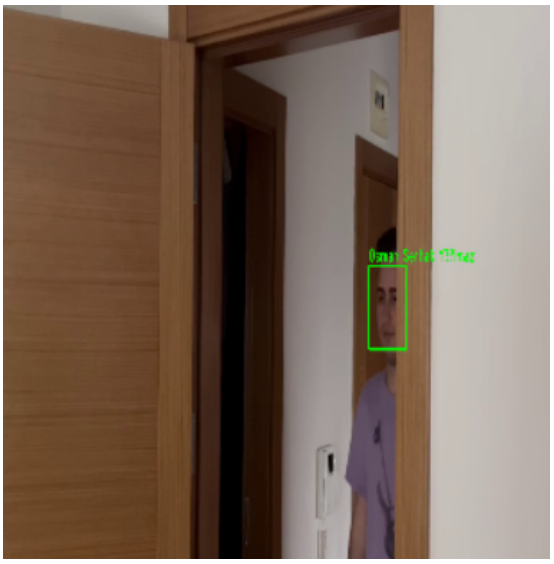
Our code implements a face recognition system that can detect and recognize faces in a video. It applies face detection on each frame using a Haar Cascade classifier. If a frame containing a detected face is found, it is returned. This way, we can get the customer's face when he/she enters the restaurant and use our model to predict it.

A model is created to predict the customer names using the previous entrances of the customer. When a customer enters to the restaurant for the first time, our code cannot label it since her/his face is not saved previously. So the code creates a new folder and gives it an id, saves the new customer to the backend, and puts the images of the customer into a folder to use it in the future predictions of that customer.

The creation of the model includes creating and training a face recognizer. In this case, it uses the LBPH (Local Binary Patterns Histograms) face recognizer from OpenCV. The training data, consisting of faces and labels, is passed to the face recognizer for training.

Below is the frame that the program passes to the model to predict, this frame is the first moment that our code sees a face in the video:

After this frame is passed to the model for prediction, it gives the below output:



As it can be seen, our program captures the moment a customer enters the restaurant and successfully predicts if that customer entered that restaurant before. After the prediction, the system saves this customer to the database with its entrance time and id, later the web project uses this information.

4.5. Database

4.5.1. Website for the Restaurant Owner and Staff

The database for restaurants is hosted on the restaurant's own hardware and the data is not shared with other restaurants. It holds restaurant specific information and information about customers who have visited the restaurant. It has the following tables:

- Customer: Stores customer name, surname, age and sex.

- Customer_count: Stores customer count at a specific time.
- Food_order: Stores customer order, customer id and table id. The staff enters the information using ur system when a food is ordered.
- Occupied_data: Similar to customer count, but stores customer count at a specific time and table.
- Session: Represents the session of a customer or multiple customers in the restaurant. Stores customer count, customer enter and exit time, table id and money spent by customer.
- Session_customer_list: Binds sessions with customer ids. Stores session id and customer id list.
- User: Holds login credentials for restaurant owner and staff. Stores user id, email, password, username and role.

4.5.2. Website for the Customers

The database for restaurants is hosted on our own servers. It holds user authentication information and various information for all restaurants using our application, and has the following tables:

- Res_info: Holds restaurant information. Stores restaurant id, restaurant name, current customer count, address, telephone number and whether the restaurant is open or not.
- User: Holds login credentials for our users. Stores user id, email, password, username and role.

5. Test Cases and Results

Test ID	1
Test Type/Category	Functional, component, usability.
Summary/Title/Objective	Check if the login and register buttons correctly redirect the user to their respective pages.
Procedure of testing steps	Try the buttons on multiple platforms and verify that they successfully redirect users to the correct pages.
Expected results/Outcome	The login and register buttons correctly redirect the user to their respective pages.
Priority/Severity	Critical
Date Tested - Test Result	19.05.2023 - Pass

Test ID	2
Test Type/Category	Security, component.
Summary/Title/Objective	Check if the password is hidden while entering.
Procedure of testing steps	Try entering the password in the login screen.
Expected results/Outcome	The password is hidden while entering, and the letters are replaced by stars (*).
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	3
Test Type/Category	Security, installation, compatibility.
Summary/Title/Objective	Make sure the username does not contain invalid characters.
Procedure of testing steps	Try entering invalid characters into the username box while creating an account.
Expected results/Outcome	The user is blocked from choosing a username that contains invalid characters.
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	4
Test Type/Category	Security, installation.
Summary/Title/Objective	Make sure the user cannot have a password that does not fit predefined criteria.
Procedure of testing steps	Try entering a password that does not fit predefined criteria into the password box while creating an account.
Expected results/Outcome	The user is blocked from choosing a password that does not fit predefined criteria.
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	5
Test Type/Category	Usability, component.
Summary/Title/Objective	Check if closed restaurants are listed after open restaurants in the drop down list.
Procedure of testing steps	Check if the restaurants are listed correctly.
Expected results/Outcome	The restaurants that are currently closed show up after the restaurants that are currently open in the drop down list.
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	6
Test Type/Category	Usability, non-functional, component.
Summary/Title/Objective	Check if restaurants are displayed with the correct images.
Procedure of testing steps	Check different restaurants from the drop down list and see if their images are correct.
Expected results/Outcome	Each restaurant displays its logo or image next to its name in the drop down list.
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	7
Test Type/Category	Usability, functional, component.
Summary/Title/Objective	Check if the drop down list disappears and the user is correctly redirected to the chosen restaurant when a restaurant is selected from the list.
Procedure of testing steps	Check if the user is correctly redirected to the chosen restaurant.
Expected results/Outcome	The user is taken to the page of the restaurant of their choice.
Priority/Severity	Critical
Date Tested - Test Result	19.05.2023 - Pass

Test ID	8
Test Type/Category	Component, functional, usability, compatibility.
Summary/Title/Objective	Check if the restaurant list functions correctly at different window sizes.
Procedure of testing steps	Resize the window and check if the restaurant list works.
Expected results/Outcome	The restaurant list works at different window shapes and sizes.
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Pass

Test ID	9
Test Type/Category	Component, functional, usability..
Summary/Title/Objective	Check if the forgot password text is clickable.
Procedure of testing steps	Try clicking the forgot password text.
Expected results/Outcome	The forgot password text is clickable and redirects users to the password reset page.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	10
Test Type/Category	Component, non-functional, security.
Summary/Title/Objective	Make sure an account cannot be created without having pressed the “I agree to the Terms of Use and Privacy Policy” button.
Procedure of testing steps	Try and create an account without pressing the button.
Expected results/Outcome	Users that try to create accounts without pressing the aforementioned button are denied.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	11
Test Type/Category	Component, functional
Summary/Title/Objective	Check if the correct restaurant is displayed when the customer selects a restaurant.
Procedure of testing steps	Try to choose a restaurant from the restaurant list.
Expected results/Outcome	The correct restaurant is displayed when the customer selects a restaurant.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	12
Test Type/Category	Component, functional
Summary/Title/Objective	Check if the restaurant list is correctly displayed to the customer.
Procedure of testing steps	Navigate to the restaurant list screen to see the restaurant list.
Expected results/Outcome	The restaurant list is correctly displayed to the customer.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	13
Test Type/Category	Component, Functional
Summary/Title/Objective	Check if the current customer count is correctly displayed in the info page after choosing a restaurant.
Procedure of testing steps	Choose a restaurant and check if the current customer count matches with the data given to the backend via the monitoring system.
Expected results/Outcome	The current customer count is correctly displayed in the restaurant info page after choosing a restaurant.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	14
Test Type/Category	Component, Functional
Summary/Title/Objective	Check if the maximum capacity of a restaurant is correctly displayed in the restaurant info page after choosing a restaurant.
Procedure of testing steps	Choose a restaurant and check if the maximum capacity is displayed correctly in reference to the data in the database.
Expected results/Outcome	The maximum capacity of a restaurant is correctly displayed in the restaurant info page after choosing a restaurant.
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Pass

Test ID	15
Test Type/Category	Component, Functional
Summary/Title/Objective	Check if the date selection menu under the Restaurant Density tab is displaying the dates correctly.
Procedure of testing steps	Click on the date selection menu and check if the dates are correctly ordered and displayed.
Expected results/Outcome	The date selection menu under the Restaurant Density tab is displaying the dates correctly.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	16
Test Type/Category	Component, Functional
Summary/Title/Objective	Check if the correct hour intervals are displayed under the Restaurant Density tab.
Procedure of testing steps	Select a day from the date selection menu and check if the bar chart displays the hour intervals correctly.
Expected results/Outcome	The correct hour intervals are displayed under the Restaurant Density tab.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	17
Test Type/Category	Component, Functional
Summary/Title/Objective	Check if the date selection menu under the restaurant density tab allows the user to change the year, month and day.
Procedure of testing steps	Try to change the year, month and day in the date selector.
Expected results/Outcome	The date selection menu under the restaurant density tab allows the user to change the year, month and day.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	18
Test Type/Category	Component, Non-functional
Summary/Title/Objective	Check if the bar charts are distinct enough from each other for different densities.
Procedure of testing steps	In the restaurant density tab check if the size of the chart is big enough to distinguish the bars from each other.
Expected results/Outcome	The bar charts are distinct enough from each other for different densities.
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Pass

Test ID	19
Test Type/Category	Component, Integration, Functional
Summary/Title/Objective	Check if changing the date changes the bar chart density representation.
Procedure of testing steps	Change the date from the date selection tab and check if the bar chart changes accordingly.
Expected results/Outcome	Changing the date changes the bar chart density representation.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	20
Test Type/Category	Component, functional
Summary/Title/Objective	Make sure the user can only enter a date (in the form of DD/MM/YYYY) in the date selection tab.
Procedure of testing steps	Try to enter text into the date selection tab to check if it allows text into the input field.
Expected results/Outcome	The user can only enter a date (in the form of DD/MM/YYYY) in the date selection tab.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	21
Test Type/Category	Functional, Integration
Summary/Title/Objective	Check if the prediction of human count in restaurant is correct
Procedure of testing steps	Compare the predicted customers by human counting system and compare it with the real results.
Expected results/Outcome	Error rate is expected to be below %10
Priority/Severity	Critical
Date Tested - Test Result	19.05.2023 - Pass

Test ID	22
Test Type/Category	Integration, Usability
Summary/Title/Objective	Check if the prediction of the human counting system is displayed correctly on the UI.
Procedure of testing steps	Compare the value in the database with the value in the UI.
Expected results/Outcome	Both values should be the same.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	23
Test Type/Category	Integration
Summary/Title/Objective	Check if the past human counts is saved
Procedure of testing steps	Compare the prediction of human counting system for a past time footage with the value saved to the database for that time.
Expected results/Outcome	Both values should be the same.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	24
Test Type/Category	Integration, Compatibility
Summary/Title/Objective	Check if the graph of density of restaurant is correct
Procedure of testing steps	Compare the values displayed in the graph with the real values saved in database.
Expected results/Outcome	Values should be the same.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	25
Test Type/Category	Integration, Compatibility
Summary/Title/Objective	Check if the favorite meal information of customer is correct
Procedure of testing steps	Compare the most ordered food by customer and favorite meal of customer in the database.
Expected results/Outcome	Both values should be the same
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	26
Test Type/Category	Security
Summary/Title/Objective	Check if logout is made successfully
Procedure of testing steps	Login to see if session started and then logout and check the session information to see whether it is finished or not.
Expected results/Outcome	Session should be finished after logout is made.
Priority/Severity	Critical
Date Tested - Test Result	19.05.2023 - Pass

Test ID	27
Test Type/Category	Integration, Functional
Summary/Title/Objective	Check if the table numbers are unique for every table
Procedure of testing steps	Check the database to see if there are 2 tables that have the same id.
Expected results/Outcome	Every table should have different table number.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	28
Test Type/Category	Integration, Functional
Summary/Title/Objective	Check if the same table is not shown occupied at the same time by different customers.
Procedure of testing steps	Check the session time for tables and see if 2 different sessions intersect.
Expected results/Outcome	Sessions should not intersect for the same table.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	29
Test Type/Category	Usability, Functional
Summary/Title/Objective	Check if users can see empty tables at that moment correctly.
Procedure of testing steps	Compare the state of tables at the UI with the real state of tables.
Expected results/Outcome	Both values of empty tables should be the same.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	30
Test Type/Category	Functional
Summary/Title/Objective	Check if the session duration of the customer is predicted correctly.
Procedure of testing steps	Compare the value recorded by human counting system with the real duration of session of customer.
Expected results/Outcome	Values should not have more difference than 5 minutes.
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	31
Test Type/Category	Functional
Summary/Title/Objective	Check if the transition between restaurant information and customer information works correctly
Procedure of testing steps	Click the buttons to check if transition happens
Expected results/Outcome	The app should be able to switch from one tab to the other
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	32
Test Type/Category	Functional
Summary/Title/Objective	Check if today's customers menu displays all the customers
Procedure of testing steps	Check the todays customers menu and compare it to actual customers in a location
Expected results/Outcome	Menu lists all the customers
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	33
Test Type/Category	Functional
Summary/Title/Objective	Check if the pictures of today's customers are listed correctly
Procedure of testing steps	Look at the displayed pictures and compare them to real pictures
Expected results/Outcome	Pictures are displayed correctly
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	34
Test Type/Category	Functional
Summary/Title/Objective	Check if registered customers are recognized properly
Procedure of testing steps	See if the system can recognize a registered customer
Expected results/Outcome	System properly recognizes registered customers
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	35
Test Type/Category	Functional
Summary/Title/Objective	Check if entry time of customers are tracked correctly
Procedure of testing steps	Check in real life if the customers that enter are tracked properly
Expected results/Outcome	Customer entry times are correct
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	36
Test Type/Category	Functional
Summary/Title/Objective	Check if departure time of customers are tracked correctly
Procedure of testing steps	Check in real life if the customers that leave are tracked properly
Expected results/Outcome	Customer departure times are correct
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Pass

Test ID	37
Test Type/Category	Functional
Summary/Title/Objective	Check if the table no is properly listed
Procedure of testing steps	Check if the listed table no and real table no are matching
Expected results/Outcome	Table no is tracked correctly
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	38
Test Type/Category	Functional
Summary/Title/Objective	Check if the customer no is correct
Procedure of testing steps	Check to see if when a customer enters their no is properly assigned
Expected results/Outcome	Customer no is correctly tracked
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Pass

Test ID	39
Test Type/Category	Functional
Summary/Title/Objective	Check if customer visits are correct
Procedure of testing steps	Check if a customer is recorded when they visit
Expected results/Outcome	Customer visits are listed correctly
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Pass

Test ID	40
Test Type/Category	Functional
Summary/Title/Objective	Check if customer purchases are correct
Procedure of testing steps	Check if purchases can be entered and stored correctly
Expected results/Outcome	Purchases are valid
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	41
Test Type/Category	Functional
Summary/Title/Objective	Check if favorite meals are correct
Procedure of testing steps	Check the prior purchases and compare them to the favorite meal
Expected results/Outcome	Favorite meal is listed correctly
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	42
Test Type/Category	Functional
Summary/Title/Objective	Check if current customer count of the table is displayed correctly
Procedure of testing steps	Check the record in the database and compare it with the UI
Expected results/Outcome	Current customer count of the table is displayed correctly
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	43
Test Type/Category	Functional
Summary/Title/Objective	Check if customer count of the table at the selected time interval is displayed correctly
Procedure of testing steps	Check the record in the database and compare it with the UI
Expected results/Outcome	Customer count of the table at the selected time interval is displayed correctly
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	44
Test Type/Category	Functional
Summary/Title/Objective	Check if all tables are listed on the restaurant density page
Procedure of testing steps	Check the record in the database and compare it with the UI
Expected results/Outcome	All tables are listed on the restaurant density page
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	45
Test Type/Category	Functional
Summary/Title/Objective	Check if customers at the table are identified correctly
Procedure of testing steps	Check the records in the database and compare it with the UI
Expected results/Outcome	Customers at the table are identified correctly
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	46
Test Type/Category	Functional
Summary/Title/Objective	Check if the most preferred time interval for the table is displayed correctly
Procedure of testing steps	Check the record in the database and compare it with the UI
Expected results/Outcome	The most preferred time interval for the table is displayed correctly
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	47
Test Type/Category	Functional
Summary/Title/Objective	Check if there are no duplicate records in the customer count of tables database
Procedure of testing steps	Check the record in the database and compare it with the UI
Expected results/Outcome	There are no duplicate records in the customer count of tables database
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	48
Test Type/Category	Functional
Summary/Title/Objective	Check if the most preferred tables of the restaurant are displayed in correct order
Procedure of testing steps	Check the record in the database and compare it with the UI
Expected results/Outcome	The most preferred tables of the restaurant are displayed in correct order
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	49
Test Type/Category	Functional
Summary/Title/Objective	Check if average customer count of the table is displayed correctly
Procedure of testing steps	Check the record in the database and compare it with the UI
Expected results/Outcome	Average customer count of the table is displayed correctly
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	50
Test Type/Category	Functional
Summary/Title/Objective	Check if average customer count of the restaurant is displayed correctly
Procedure of testing steps	Check the record in the database and compare it with the UI
Expected results/Outcome	Average customer count of the restaurant is displayed correctly
Priority/Severity	Minor
Date Tested - Test Result	19.05.2023 - Unimplemented

Test ID	51
Test Type/Category	Functional
Summary/Title/Objective	Check if there are no unnecessary records in the customer count of tables database
Procedure of testing steps	Check the record in the database and compare it with the UI
Expected results/Outcome	There are no unnecessary records in the customer count of tables database
Priority/Severity	Major
Date Tested - Test Result	19.05.2023 - Unimplemented

6. Maintenance Plan and Details

Our application for restaurant owner and staff, and its database are hosted on the restaurant's own hardware, so it doesn't require any maintenance on our part.

Our application for customers and its database is hosted on DigitalOcean. It uses Ubuntu 22.10 as OS and has the following hardware properties:

- 2 vCPUs
- 4GB RAM
- 25 GB Disk

Both our frontend and backend applications use the same server, and the application can be accessed from 104.248.26.143:3000.
The database uses MySQL 8 and has the following properties:

- 1 GB RAM
- 10 GB Disk

Any updates on the customer application should be done at hours where most restaurants are not active (preferably at night) to ensure that the users are able to access our application without interruptions.

7. Other Project Elements

7.1. Consideration of Various Factors in Engineering Design

We took a comprehensive approach to engineering design in our software project, taking into consideration various factors that were crucial to its success. Firstly, we prioritized the user experience and ensured that our software was user-friendly, intuitive, and accessible to both restaurant owners and customers. Additionally, we placed a strong emphasis on the software's reliability and scalability, ensuring that it could handle large amounts of data and traffic and work with different sizes of restaurants. Security was also a major concern, and we implemented robust security measures to protect sensitive user data and prevent unauthorized access. Finally, we ensured that our software was compatible with a wide range of devices and operating systems, allowing for maximum flexibility and accessibility for our users.

7.2. Ethics and Professional Responsibilities

During the implementation of Cyclops, ethical and professional responsibilities were observed. Our application makes use of customer data such as facial recognition data, user login credentials (email, password etc.) and personal information (name, age etc.). Customer facial recognition data is obtained through security cameras of the restaurant and does not pose an ethical issue, while personal information is obtained by restaurant staff with customer consent. Customer data obtained in a restaurant is not shared with other restaurants, which reduces the security risks and ensures that restaurants are not able to exploit customer data.

7.3. Teamwork Details

7.3.1 Contributing and functioning effectively on the team

We were able to satisfy the requirements for contributing and functioning effectively on the team by prioritizing communication and teamwork. We made sure to communicate effectively with each other, actively listening to each other's ideas and feedback. We also worked together closely to achieve our common goals, and we were always accountable and responsible for our own tasks and deadlines. In addition, we were always willing to step up and help out when a team member needed assistance.

7.3.2 Helping creating a collaborative and inclusive environment

To help create a collaborative and inclusive environment, we actively sought out diverse perspectives and ideas, and encouraged each other to share our thoughts and experiences. We fostered open communication and feedback, creating a space where everyone felt valued and supported. We were always respectful and considerate of each other's unique backgrounds, experiences, and perspectives.

7.3.3 Taking lead role and sharing leadership on the team

In taking a lead role and sharing leadership on the team, we set clear goals and expectations for the team and communicated them effectively. We led by example, being accountable and responsible for our own tasks, demonstrating a strong work ethic and commitment to the project. Additionally, we shared leadership on the team, empowering and encouraging each other to take on leadership roles as well, and actively seeking out each other's input and feedback. By doing so, we were able to create a more collaborative and inclusive environment and ensure that everyone felt valued and supported.

7.3.4. Meeting objectives

As a collaborative team, we achieved most of our project objectives successfully. By using our diverse skill sets and expertise, we planned and executed our tasks to ensure data obtained from cameras for people counting is sent to our backend and shown to users on our frontend systems . Our team's technical prowess was utilized through the development and integration of Java Spring and Node.js, which enabled us to build a successful application. We made use of open communication channels, conducted regular meetings and shared progress updates, which resulted in effective coordination. Each team member played a vital role in this accomplishment, brought innovative ideas and provided constructive feedback. We embraced a flexible approach, accommodating adjustments as we updated project requirements throughout the project. Through our strong teamwork, we delivered a successful restaurant management system that met our project objectives.

7.4. New Knowledge Acquired and Applied

Although we had all worked on numerous projects in the last years, none of them required us to apply concepts and technologies from such different fields into a single project as this one. We were required to build 2 web applications, setup a server to host the project, train multiple image recognition systems, setup and manage a database, and to integrate all

these parts into a single project seamlessly. We had taken classes that helped us with each aspect of the project, but we had to do our own research to figure out how exactly these systems could be integrated into one. We also had to do our own research on how to set up a server for the customer web application. We applied what we had learned from our Machine Learning class into training the image recognition software using the YOLOv3 algorithm and the Darknet framework. Although we were acquainted with the concepts, we had to learn more on our own to be proficient with image recognition.

8. Conclusion and Future Work

In conclusion, our project addresses the challenge of counting the number of people in a restaurant and displaying it to users for assessing the density. We have developed an efficient solution using the YOLOv3 algorithm and the Darknet framework for real-time object detection in video footage. By leveraging computer vision techniques, our system accurately detects and tracks individuals within the restaurant environment. The average hourly count of people is calculated and stored in a database, allowing users to monitor the crowd density over time. This information can help restaurant owners and managers make informed decisions regarding capacity management, customer service, and overall safety measures. Our solution offers a practical and automated approach to crowd monitoring, eliminating the need for manual counting and providing real-time insights. Moving forward, further improvements and optimizations can be made to enhance the accuracy and performance of the system. Also, we aim to present this project to investors and possibly get an investment for our idea. Overall, our project showcases the potential of computer vision technology in facilitating crowd management.