Bilkent University
Department of Computer Engineering

Senior Design Project

Group: T2317 - RoadVisor

Detailed Design Report

Group Members:
Ahmet Faruk Ulutaş - 21803717 - faruk.ulutas@ug.bilkent.edu.tr
Ammaar Iftikhar - 21901257 - ammaar.iftikhar@ug.bilkent.edu.tr
Arda İçöz - 21901443 - arda.icoz@ug.bilkent.edu.tr
Emin Berke Ay - 21901780 - berke.ay@ug.bilkent.edu.tr
Nurettin Onur Vural - 21902330 - onur.vural@ug.bilkent.edu.tr

Supervisor:
Asst. Prof. Dr. Hamdi Dibeklioğlu

Innovation Expert:
Cem Çimenbiçer

Course Instructors:
Erhan Dolak
Tağmaç Topal

13.03.2023

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

Table of Contents

1. Introduction

1.1 Purpose of the system

Navigation in unknown areas is a difficulty faced by automobilists. The automobilist may be an experienced taxi driver or a novice. Navigators don't only face the hurdle of driving through unknown streets, but also face the task of keeping track of road information like traffic lights and street signs simultaneously. The presence of so much information for the driver to process might lead to increased mistakes or accidents [1]. There are other issues that pertain to the road, like boundaries and crowd density, that play a crucial role to inform the driver. If a greater amount of information is presented to the driver with more ease, the driver will be able to perform better.

While there are many applications that attempt to present the driver with directions to the destination, there aren't many that help simultaneously inform the driver of other necessary information simultaneously. These applications mostly focus on delivering the queried information to the driver like the location of certain places and the distance to the next turn. Information about processing real-time information for the navigator on the road is not prioritized by advanced applications like Google Maps. Our application attempts to fill the disparity between the user view and the application. RoadVisor assists the drivers in real-time using Machine Learning and Augmented Reality.

The main motivation behind RoadVisor is to assist the driver in processing the real-time events on the street fetched using the mobile phone camera while also providing the user with an improved street view and directions on their mobile phones. The ability to process information and the information fetched by the application is constrained by the processing capabilities of the devices. The application will use different Machine Learning methodologies, like Deep Neural Networks, and APIs, like Google Maps Platform API, to continuously inform the driver about road information and directions. Augmented Reality will try to ensure that the driver does not miss the road information while consulting the mobile for directions or other information.

The application will mitigate traffic violations, like violations of street signs and traffic lights, by drivers. The application will try to fill the void in the navigation application industry by presenting a viable and better alternative to the ones available in the market. Our application will be an attempt to assist the driver without distracting or bugging to decrease the fatalities caused by road accidents or car crashes [1]. The application will be built incrementally and optimized for usage on mobile devices.

RoadVisor aims to have a place in the Offering section and be a Product Performance focused application according to the 10 Types of Innovation Wheel by Doblin [2]. We believe that the AR-supported navigation feature distinguishes RoadVisor from other applications and provides better and more comfortable functionality. With these distinguishable features, RoadVisor will be an incremental innovation project and optimization will be the key element as the goal is to provide a better user experience.

1.2 Design Goals

1.2.1. Reliability

- It's crucial for RoadVisor to give navigation information correctly. This means that the application has to be reliable in terms of fulfilling the expected necessities such as preparing the route accurately, placing the direction arrows correctly, showing true traffic light and sign information messages, and successfully sending the SOS messages with the location of the user.
- RoadVisor has to operate without significant interruptions and therefore provides navigation service and other promised services during the entire car trip duration starting from the moment that the destination is selected until the user arrives at it given that the necessary conditions are satisfied such as battery and connection.

1.2.2. Performance

- RoadVisor has to have a high standard in terms of performing well in frequent conditions and in a short duration. Since the application needs to constantly extract input from live camera feeds and analyze them frame by frame, the frame rate has to be above a tolerable level so that the application works without significant delay.
- The detection models used within RoadVisor have to perform with high accuracy rates for being considered reliable.
- The application needs to work smoothly with the touch screen which requires RoadVisor to have a fast response time.
- RoadVisor is designed mainly to work in a mobile environment. In this respect, the application has to perform well to satisfy its features without being overly affected by hardware limitations of mobile phones.

1.2.3. Usability

- RoadVisor is an application that aims to reach a very wide range of audience. Essentially, all drivers (especially ones who do not possess the latest technological tools in their cars) are targeted as potential users of our application. In this respect, the application needs to be understandable and easy to use by the target audience.
- RoadVisor needs to be compatible with a wide range of Android phones given that they are above a determined version.
- RoadVisor needs to have a highly responsive, simple-looking user interface. Given that the user will be busy with driving as well the UI has to be designed in such a manner that it will not have very complex operations for the user that take various steps. Instead, the parts that require user interaction must be large enough, easily locatable on the phone screen, and respond within a short amount of time (less than a minute).
- Since the user will be driving meanwhile, the application must not cause unnecessary distraction or irritate the user. This will be achieved by avoiding using irritating alert sounds, having a simple UI layout, and decreasing the need for user input (such as automatically closing pop-ups after a while).

1.2.4. Privacy

- Confidential information such as contact numbers, destination information, or exact route data will not be used with third parties to respect user privacy.
- To minimize the risk of exposure of personal information in a possible attack, confidential data will be kept encrypted.

1.3 Definitions, acronyms, and abbreviations

- Augmented Reality (AR)
- Machine Learning (ML)
- You Only Look Once (YOLO): “YOLO algorithm aims to predict a class of an object and the bounding box that defines the object location on the input image. It recognizes each bounding box using four numbers that are center of the bounding box, width of the box, height of the box. In addition to that, YOLO predicts the corresponding number for the predicted class as well as the probability of the prediction” [1].

1.4 Overview

RoadVisor uses Augmented Reality to assist the driver to navigate. The view provided to the user includes the road view and additional information that improves the information output to the driver. The features RoadVisor offers are:

Augmented Reality Supported Navigation: The feature provides the user the directions to navigate by showing arrows in the augmented reality view. It also displays road boundaries and turns. The user can view the road as well as the directions. The feature will be implemented using image processing and computer vision techniques along with map api's. We will use Google Geospatial API to place AR elements as arrows onto the bounded road. Mapbox Navigation API fetches and returns the turn-by-turn navigation information.

Traffic Light Detection: The feature informs the driver about traffic light signals. The feature helps to improve driver's awareness of traffic lights. The information utilized by the feature will be fetched using the back camera of the device. The feature is in line with our motive to help drivers process road information with ease. The feature will be implemented using Computer Vision, particularly YOLO object detection methods. The dataset selected for this feature is Bosch Small Traffic Lights Dataset [2]. This particular dataset has 13427 road images with a resolution of 1280x720 pixels and has around 24000 annotated traffic lights that are marked with bounded boxes.

Sign Detection: The application will help users be aware of the traffic signs that might be on the road like 'school ahead" or "danger ahead." The signs the feature will recognize are expected to follow the conventions of the selected dataset, namely Traffic Signs Dataset in YOLO format [3]. The implementation of this feature will also utilize YOLO algorithm for object detection. The feature assists the user to prevent violation of signs that might lead to other consequences like fines.

Pedestrian Detection/Crowd Density Analysis: The application will provide information to the user about the presence of pedestrians and crowd in a particular location. The application will use crowd-counting techniques to implement this feature [6]. We can implement this feature using YOLO object detection as well. When it comes to the dataset, Caltech Pedestrian Dataset is the choice having approximately 10 hours of 640x480 30Hz video taken from a vehicle driving through regular traffic in an urban environment. About 250,000 frames (in 137 approximately minute long segments) with a total of 350,000 bounding boxes and 2300 unique pedestrians were annotated. The annotation includes temporal correspondence between bounding boxes and detailed occlusion labels [4].

Emergency Alert: Assisting our users during times of danger is a feature that we want to include. In a risky situation such as attempt of breaking into the car, breakdown of the car, attacks on the vehicle or any other problematic situation that involves the potential risk of danger, the user will be able to activate the S.O.S feature that will notify the selected close contacts. The feature might be able to save the lives of people by immediately contacting the concerned authorities.

2. Current Software Architecture

Using machine learning-backed features in mobile applications is currently a common theme. Therefore it is possible to find decent applications that work on detecting pedestrians, traffic lights/signs and implementing augmented reality navigation. However, after our search in application markets, to our knowledge, there is currently no application that provides all these features in one complete package. Hence, we thought it is best to give some example applications that implement these features and discuss the similarities and differences.

The first application is called *Car Assistant Android [5],* and it detects traffic signs. This application uses the MobileNet SSD model, which utilizes the TensorFlow object detection feature [6], and it uses its own dataset. In the provided demo video, the confidence level of detected signs is between 75%-90%. Our application implements the YOLO algorithm for sign detection, and we use the European sign dataset because there are very few differences between it and Turkish signs. The feature is still in development, and therefore the confidence level results will not be discussed here.

The following application is called *TrafficLights Detector For Android [7],* and it is used to detect traffic lights. This application mainly uses Caffee [8] framework along with TensorFlow, and the dataset is provided by David Brai [9]. Along with detecting traffic lights, it also detects which color the lights are currently at. This application itself can be called very similar to our traffic light detection feature because currently, we are also developing this feature to detect both lights and colors.

Next, we have the pedestrian detection feature. Machine Learning models for mobile devices can be found easily, but finding an example application was relatively more challenging. This application [10] has a demo video where they demonstrate car and pedestrian detection features. It is developed with OpenCV and is only available for Android. Our application only focuses on pedestrian detection as cars are already relatively easy to detect by drivers. This feature is also in development, but we believe that RoadVisor has the potential to be a preferred application for this feature, considering that the application markets are far away from being saturated with similar-featured applications.

Lastly, it is possible to reach the repositories of augmented reality navigation applications for mobile phones. When we did a market search for these applications, we observed that lots of them utilized augmented reality features for indoor navigation [11]. RoadVisor's inspiration to use augmented reality comes from Mercedes-Benz's navigation feature [12]. However, this feature is only available on high-tier Mercedes cars. Additionally, even if Phiar's AR Navigation application [13] is similar to RoadVisor, it is not published for the use of the general public. Therefore we believe that RoadVisor has the potential to be the candidate application for these needs.

3. Proposed Software Architecture

In this section, proposed software architecture will be discussed and explained in detail.

3.1. Overview

In this section, our system's proposed software architecture and subsystem decomposition is explained in detail. We have several design goals to aim and lead us through the project.The architecture has been designed to meet the functional and non-functional requirements of the software, such as performance, security, reliability, and usability, while also being flexible and adaptable to future changes and enhancements. For this reason our project consists of several layers. The layers are the user interface layer, the application layer and the data layer. The user interface layer is the layer where the user interacts with the system and it is the part where the users will see when they open up the application on their Android mobile phone. The application layer is the layer where the business logic is handled and the models are incorporated into the system. The data layer deals with the data our application collects and handles.

This section also explains the hardware/software mappings of the project. However, since our project does not have any hardware components, we will leave this section blank. Additionally, how we handle persistent data is explained in detail and the access control and security of our system will be explained. Our project does not deal with big data and we do not require any special access or authentication from our users, therefore these sections will be rather short.

3.2. Subsystem Decomposition

We will follow three-tier architecture to divide the subsystem into three different layers:

1. Interface
2. Application Logic
3. Server

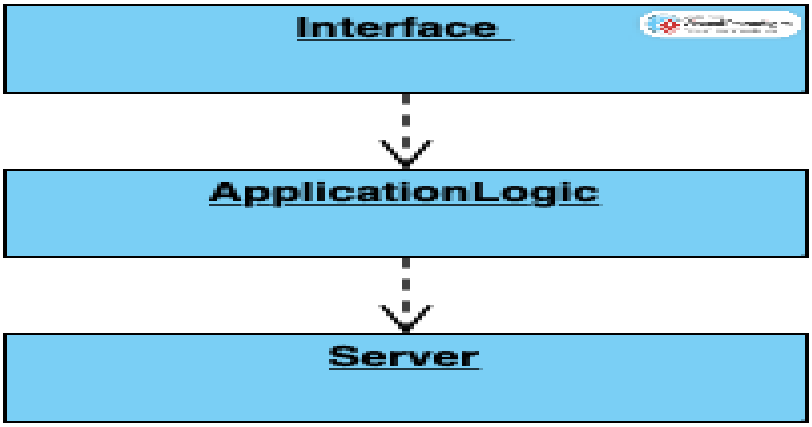

Figure 3.1: Three-tier decomposition of our Application

This particular style of subsystem decomposition architecture has been selected based on the needs of our application. Other architectures, like four-tier or simple server-client architectures, can be possible options, but will either create an unnecessary layer or prevent logical hierarchical separation of different systems. The Interface subsystem layer is based on the user device, in our case, Android phones. The Application logic is also based on the user device, but it serves as a mechanism for the interface layer to interact with the server. The lowest layer of our architecture is the Server layer which is responsible for storing information in the database and fetching data from there. More details about the subsystems is in the subsystem services section.

3.3. Hardware/Software Mappings

As our project does not have any hardware components, this section is left blank.

3.4. Persistent Data Management

The amount of persistent data that we manage is limited to functionalities that we provide. We don't store any additional or dynamic data. The data that is stored is the user information for logging in and emergency requests. A user can add this information during the sign up. We don't store any data regarding the user's location or behavior. Our application intends to respect user privacy by not collecting unnecessary sensitive or behavioral user information.

3.5. Access Control and Security

The application can only be accessed by registered users. In addition to this, we only give a user access to his/her information. No user information is shared between two users. For access to a user's emergency contacts, the user needs to be logged into his account which happens using verification of user credentials which is managed in the server. There are no security risks for the user due to the limited amount of information that is stored. The requests during an emergency are made on the server. This prevents security issues like interception of emergency contacts information by third parties.

4. Subsystem Services

4.1 Interface Subsystem

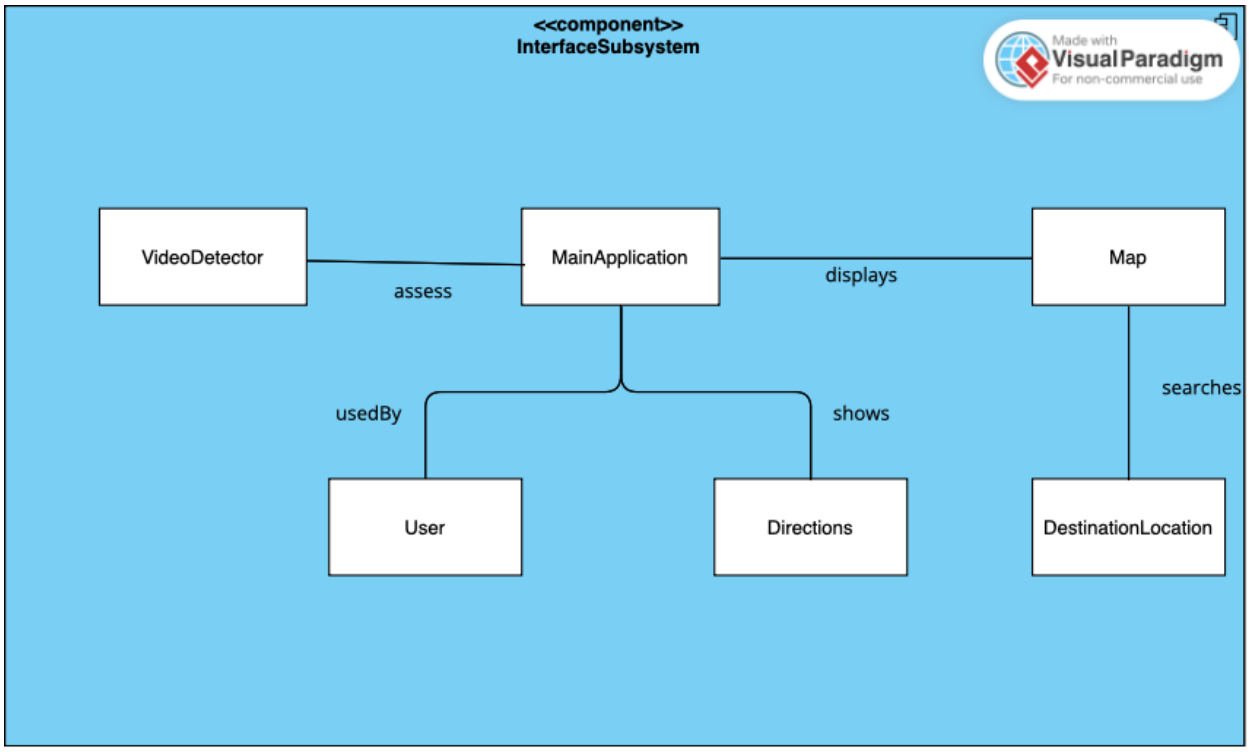

Figure 4.1: The Interface Layer Subsystem

The interface layer subsystem contains necessary sections to display the maps and navigation information. It also contains a VideoDetector system that helps display detected traffic lights, traffic signs, and pedestrians. The VideoDetector fetches the detector information from the Application Logic level subsystem. The User system is responsible for managing the user information like username. It also helps the application fetch emergency contacts during emergency requests. The Directions system is responsible for showing the user the arrows after the destination has been

selected using the Map. The User will select the destination he/she is traveling to using the map. The Map and other systems interact directly with the Application Logic Layer. The DestinationLocation selected is then used by the Application logic to make requests for the directions to the destination. The directions fetched are than displayed by the using the Directions system.

4.2 Application Logic Subsystem

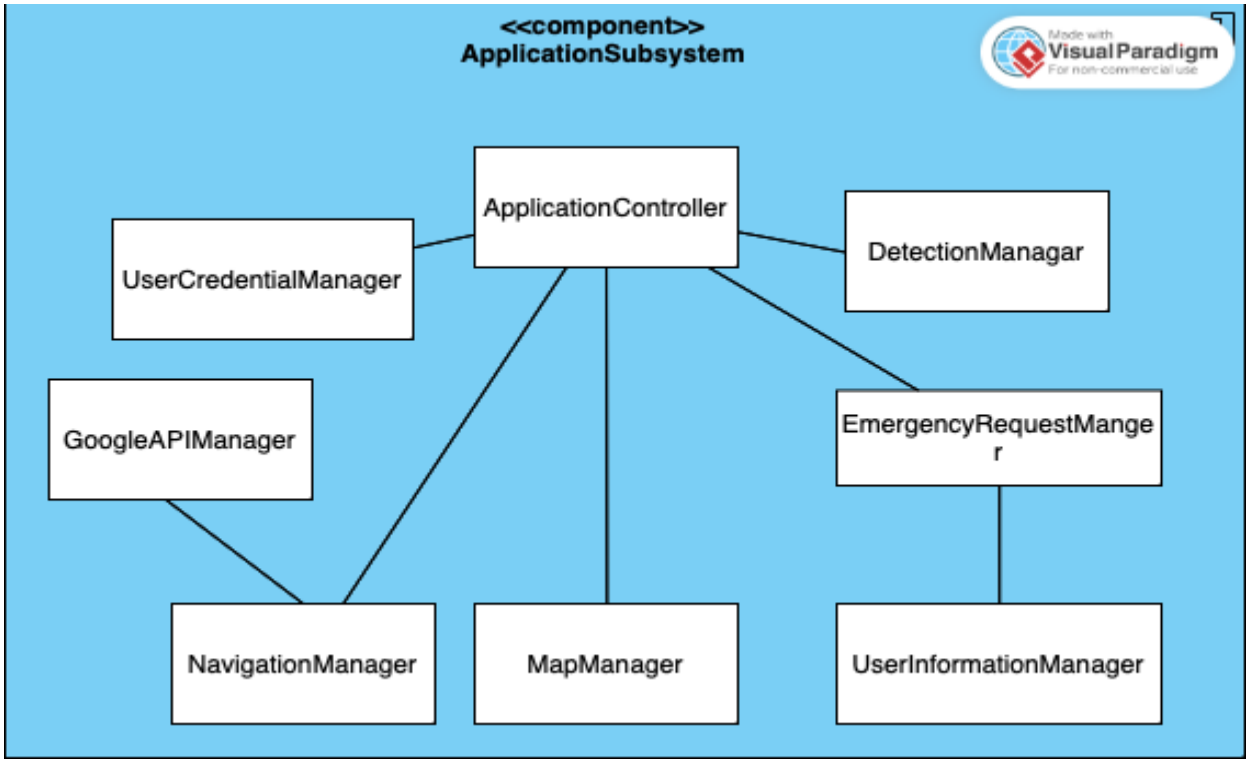

Figure 4.2: The Application Logic Layer

The Application Logic Layer is responsible for the control of the application, response to user requests, and connecting the interface and server. This layer acts as a buffer between the two layers. All the requests made by the interface layer are controlled by the ApplicationController which in turn calls other managers based on specific functions required by the user requests. The Map Manager is responsible for fetching the map and managing the selection of the destination by the user. The MapManager sends the map to the interface. On user interaction with the map, the Interface layer makes requests to the ApplicationController which in turn based on the application status responds to the request. The EmergencyRequestManager is responsible for managing the requests made by the user for emergency help during an accident. The EmergencyRequestManager uses the UserInformationManager to fetch the information

of the user's emergency contacts from the Server Layer, and then send a notification to the user regarding the emergency.

The DetectionManager is responsible for generating the detection of lights, signs, and pedestrians. The manager runs Yolo detection algorithms on the images sequentially sampled from the camera of the Android device. The information about the detections made by the device are then sent to the interface by the application controller. The NavigationManager is responsible for making the requests to the MapAPIManager for directions based on the user's location and the final destination. The requests will be made by the Application layer based on the user's location. We will try to optimize the amount of requests that are being made to decrease unnecessary API requests and computation on the Android device.

4.3 Server Subsystem

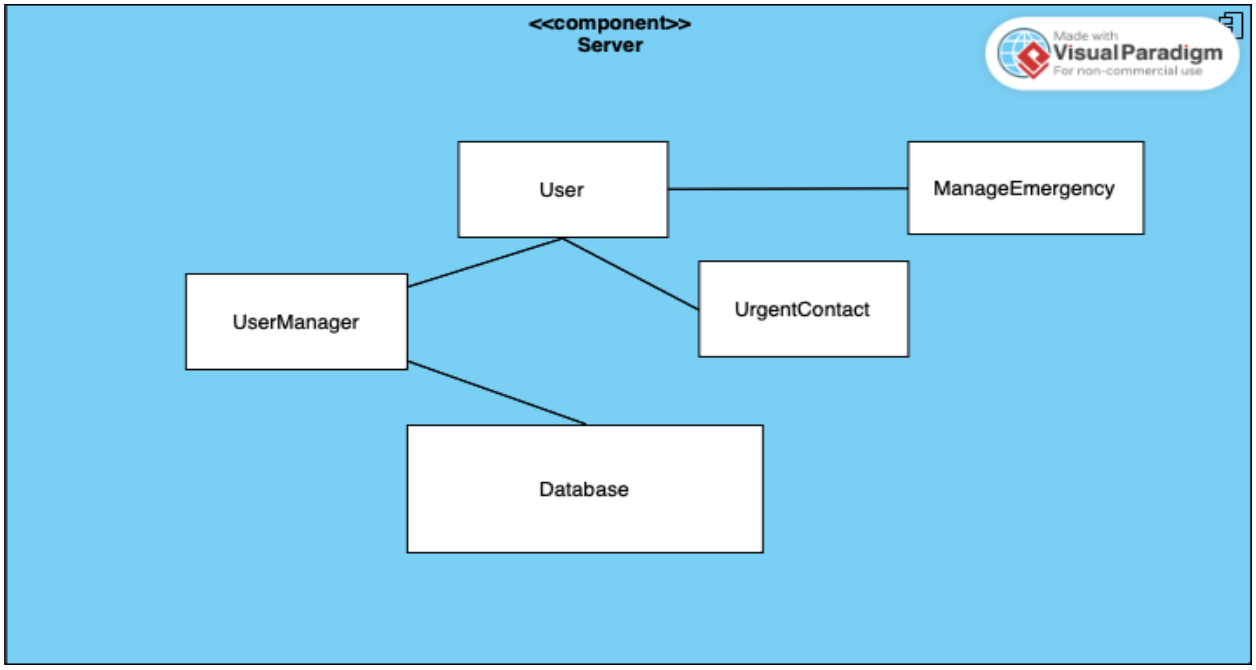

Figure 4.3: Server Subsystem Layer

The Server subsystem layer is not as extensive as the other two layers due to the small dependency of the application on stored data. Most of the processing will take place on the client device. The emergency functionality, login, and account creation are the only dependencies of the application that are directly dependent on the server. Our database is based on the server and the requests that are dependent on the database take place on the server. We are using a MySQL database based on an Azure database. We store information regarding username, password, and emergency contacts on the database.

5. Test Cases

Functional Tests

Register API Tests

Test ID: TS-01-1
Title: Successful User Registration

- Procedure:
    1. Send a POST request to the API endpoint /api/v1/accounts/register/ with correct user data.
    2. Verify that the response status code is 201 CREATED.
    3. Verify that the response message is Please check your email to activate your account.
    4. Verify that the confirmation email is sent to the user's email address.
    5. Verify that the email contains the correct activation link.
- Outcome:
    1. The user should be successfully registered and a confirmation email should be sent to the user's email address. The activation link in the email should be correct and the user should be able to activate their account.
- Severity: High

Test ID: TS-01-2
Title: User Registration with Duplicate Email

- Procedure:
    1. Send a POST request to the API endpoint /api/v1/accounts/register/ with user data containing an email address that already exists in the database.
    2. Verify that the response status code is 400 BAD REQUEST.
    3. Verify that the response message is User with this email already exists.
- Outcome:
    1. The user should not be registered and an error message should be returned stating that a user with that email address already exists.
- Severity: High

Test ID: TS-01-3
Title: Token Creation

- Procedure:

1. Send a POST request to the API endpoint /api/v1/accounts/register/ with correct user data.
2. Verify that the response status code is 201 CREATED.
3. Verify that the user's token is a unique value.

- Outcome:
    1. The user's token should be a unique value.
- Severity: Medium

Test ID: TS-01-4
Title: Confirmation Email Sent

- Procedure:
    1. Send a POST request to the API endpoint /api/v1/accounts/register/ with correct user data.
    2. Verify that the response status code is 201 CREATED.
    3. Verify that the confirmation email is sent to the user's email address.
    4. Verify that the email subject is Account Activation.
    5. Verify that the email message contains the correct activation link.
- Outcome:
    1. The confirmation email should be sent to the user's email address with the correct activation link.
- Severity: Medium

Update API Tests
Test ID: TS-02-1
Title: Successful User Update

- Procedure:
    1. Authenticate as a user by sending a POST request to the API endpoint /api/v1/token/ with correct user credentials.
    2. Send a PATCH request to the API endpoint /api/v1/accounts/update/ with correct user data to update.
    3. Verify that the response status code is 200 OK.
    4. Verify that the response message is {'success': True}.
    5. Verify that the user data has been successfully updated in the database.
- Outcome:
    1. The user data should be successfully updated in the database and a success message should be returned.
- Severity: High

Test ID: TS-02-2
Title: User Update with Invalid Data

- Procedure:
    1. Authenticate as a user by sending a POST request to the API endpoint /api/v1/token/ with correct user credentials.
    2. Send a PATCH request to the API endpoint /api/v1/accounts/update/ with user data containing invalid data.
    3. Verify that the response status code is 400 BAD REQUEST.
    4. Verify that the response message contains the errors returned by the serializer.
- Outcome:
    1. The user data should not be updated in the database and an error message should be returned containing the errors returned by the serializer.
- Severity: Medium

Test ID: TS-02-3
Title: User Update with Missing Fields

- Procedure:
    1. Authenticate as a user by sending a POST request to the API endpoint /api/v1/token/ with correct user credentials.
    2. Send a PATCH request to the API endpoint /api/v1/accounts/update/ without providing any of the required fields.
    3. Verify that the response status code is 400 BAD REQUEST.
    4. Verify that the response message is {'error': 'At least one of the following fields must be provided: email, first_name, last_name, password, old_password.'}.
- Outcome:
    1. The user data should not be updated in the database and an error message should be returned stating that at least one of the required fields must be provided.
- Severity: Medium

Activation API Tests

Test ID: TS-03-1
Title: Account Activation Process

- Procedure:

1. After a user signs up, they receive the correct URL to activate their account.
2. The user sends a GET request to the API by opening the correct URL.
3. The API checks the incoming request.
4. If the user does not provide a valid user_id and token value, the API returns a 400 Bad Request error.
5. If the user provides a valid user_id and token value, the API activates the user's account and returns a "Account activated successfully" message with a 200 OK status.

- Outcome:
  1. When a GET request is sent with a valid user_id and token, the account is successfully activated and the API returns a "Account activated successfully" message with a 200 OK status.
  2. When a GET request is sent with an invalid user_id or token, the API returns a 400 Bad Request error.
- Severity: High

Login API Tests
Test ID: TS-04-1
Title: Test the API functionality for user login.

- Procedure:
  1. Send a POST request to the '/api/v1/login/' endpoint with valid email and password in the request body.
  2. Verify that the response status code is 200 OK.
  3. Verify that the response body contains user details including user id, email, first name, last name, and token.
- Outcome:
  1. The API should return a response with status code 200 OK and user details in the response body when the user credentials are correct.
  2. The API should return a response with status code 400 Bad Request and an error message in the response body when the user credentials are incorrect or missing.
- Severity: Medium

Logout API Tests
Test ID: TS-05-1, TS-05-2
Title: Test logout API

1. Test that the user can successfully log out of the system.
2. Test that an authenticated user is required to access the logout API.

- Procedure:
    1. Authenticate a user and obtain the token.
    2. Make a POST request to the logout API with the token in the Authorization header.
    3. Verify that the response status code is 200 OK.
    4. Verify that the response message is "Logged out successfully."
    5. Verify that the user is logged out of the system.
    6. Attempt to access a protected resource with the token.
    7. Verify that the response status code is 401 Unauthorized.
- Outcome:
    1. The user should be able to log out of the system successfully.
    2. An authenticated user is required to access the logout API.
- Severity: Medium

Password Reset Request API Tests

Test ID: TS-06-1
Title: Test Password Reset Request API

- Procedure:
    1. A POST request is made with a valid email address.
    2. A successful response is received for an existing user with the email address.
    3. An unsuccessful response is received for a non-existing user with the email address.
    4. The returned response status code should be 200 OK.

- Outcome:
    1. When a POST request is made with a valid email address and the email is registered, a message "If the email you entered is valid, you will receive instructions on how to reset your password in your email shortly." is returned along with a 200 OK status code.
    2. When a POST request is made with a valid email address and the email is not registered, the same message is returned along with a 200 OK status code.
    3. When a POST request is made with an invalid email address, an error message and a 400 Bad Request status code are returned.
- Severity: High

Password Reset Confirm API Tests

Test ID: TS-07-1
Title: Test Password Reset Confirm API

- Procedure:
    1. Set up the user and generate a password reset token and encoded_pk.
    2. Hit the "Reset Password API" endpoint with the token and encoded_pk in the URL, and provide a new password and its confirmation in the request body.
    3. Check that the response has a status code of 200 and the message 'Password reset complete'.
    4. Verify that the user's password has been successfully reset in the database.
- Outcome:
    1. The user should be successfully set up with valid credentials.
    2. The password reset token and encoded_pk should be generated and retrieved successfully.
    3. The "Reset Password API" endpoint should accept the token and encoded_pk in the URL and the new password in the request body.
    4. The response from the API should have a status code of 200 and the message 'Password reset complete'.
    5. The user's password should be successfully updated in the database with the new password.
- Severity: High

Urgent Contacts API Tests

Test ID: TS-08-1
Title: Get All Urgent Contacts

- Procedure:
    1. Send a GET request to 'api/v1/urgent-contacts/' endpoint with a valid authentication token.
- Outcome:
    1. Response status code should be 200.
    2. Response body should contain all urgent contacts in the database serialized as a JSON array.
- Severity: Low

Test ID: TS-08-2
Title: Create a New Urgent Contact

- Procedure:

1. Send a POST request to 'api/v1/urgent-contacts/' endpoint with a valid JSON payload containing a new urgent contact.

- Outcome:
    1. Response status code should be 201.
    2. Response body should contain the serialized JSON representation of the newly created urgent contact.
    3. The created urgent contact should be stored in the database.
- Severity: High

Test ID: TS-08-3
Title: Create a New Urgent Contact with Invalid Data

- Procedure:
    1. Send a POST request to 'api/v1/urgent-contacts/' endpoint with an invalid JSON payload.
- Outcome:
    1. Response status code should be 400.
    2. Response body should contain an error message indicating the validation error.
- Severity: Medium

Test ID: TS-08-4
Title: Get Urgent Contacts without Authentication

- Procedure:
    1. Send a GET request to 'api/v1/urgent-contacts/' endpoint without authentication token.
- Outcome:
    1. Response status code should be 401.
    2. Response body should contain an error message indicating that authentication is required.
- Severity: High

Test ID: TS-08-5
Title: Try to Create Urgent Contact without Authentication

- Procedure:
    1. Send a POST request to 'api/v1/urgent-contacts/' endpoint without authentication token.
- Outcome:
    1. Response status code should be 401.

2. Response body should contain an error message indicating that authentication is required.

- Severity: High

Urgent Contacts Send Email API Tests

Test ID: TS-09-1
Title: Test the Notification System for Urgent Contacts

- Procedure:
  1. Create a user.
  2. Create at least one urgent contact for the user.
  3. Make a POST request to the API with the following payload:
     {
     "location": "123 Main St, Anytown USA"
     }
  4. Verify that the response status code is 200.
  5. Verify that the response message is "Urgent contacts have been informed."
  6. Verify that the email is sent to the correct recipients with the correct subject and message.
- Outcome:
  1. The response status code should be 200.
  2. The response message should be "Urgent contacts have been informed."
  3. The email should be sent to the correct recipients with the correct subject and message.
- Severity: High

Test ID: TS-09-2
Title: Test Required Location Field Validation for Urgent Contacts

- Procedure:
  1. Create a user.
  2. Create at least one urgent contact for the user.
  3. Make a POST request to the API with the following payload:
  4. {}
  5. Verify that the response status code is 400.
  6. Verify that the response error message is "Location is required."
  7. Verify that the email is not sent.
- Outcome:
  1. The response status code should be 400.
  2. The response error message should be "Location is required."

3. The email should not be sent.

- Severity: Medium

Test ID: TS-09-3
Title: Test Notification System Error Handling for Missing Urgent Contacts

- Procedure:
    1. Create a user.
    2. Make a POST request to the API with the following payload:
       {
       "location": "123 Main St, Anytown USA"
       }
    3. Verify that the response status code is 400.
    4. Verify that the response error message is "Urgent Contact not found. Please add Urgent Contact to use this feature."
    5. Verify that the email is not sent.
- Outcome:
    1. The response status code should be 400.
    2. The response error message should be "Urgent Contact not found. Please add Urgent Contact to use this feature."
    3. The email should not be sent.
- Severity: Low

Test ID: TS-09-4
Title: Test Notification System Error Handling for Failed Email Sending

- Procedure:
    1. Create a user.
    2. Create at least one urgent contact for the user.
    3. Make a POST request to the API with the following payload:
       {
       "location": "123 Main St, Anytown USA"
       }
    1. Mock the send_mail function to raise an exception.
    2. Verify that the response status code is 500.
    3. Verify that the response error message is the exception message.
    4. Verify that the email is not sent.
- Outcome:
    1. The response status code should be 500.
    2. The response error message should be the exception message.
    3. The email should not be sent.
- Severity: High

Specific Urgent Contact API Tests

Test ID: TS-10-1
Title: Test Creation and Retrieval of Urgent Contacts

- Procedure:
    1. Create a user and log in.
    2. Add a new urgent contact using POST method to the '/api/v1/urgent-contacts/' endpoint.
    3. Retrieve the created urgent contact using GET method to the '/api/v1/urgent-contacts/<id>/' endpoint.
- Outcome:
    1. A new urgent contact should be created with the given details.
    2. The response of the GET request should contain the details of the created urgent contact.
- Severity: High

Test ID: TS-10-2
Title: Test Update and Retrieval of Urgent Contacts

- Procedure:
    1. Create a user and log in.
    2. Add a new urgent contact using POST method to the '/api/v1/urgent-contacts/' endpoint.
    3. Update the urgent contact using PUT method to the '/api/v1/urgent-contacts/<id>/' endpoint.
    4. Retrieve the updated urgent contact using GET method to the '/api/v1/urgent-contacts/<id>/' endpoint.
- Outcome:
    1. The urgent contact should be updated with the new details.
    2. The response of the GET request should contain the updated details of the urgent contact.
- Severity: High

Test ID: TS-10-3
Title: Test Retrieval of User's Urgent Contacts

- Procedure:
    1. Create a new user object.
    2. Login with the newly created user's credentials.
    3. Send a GET request to /api/v1/urgent-contacts/my-urgent-contacts/.
    4. Check that the response status code is 200.
    5. Check that the response contains a list of urgent contact objects.

6. Check that the urgent contact objects in the response belong to the logged-in user.

- Outcome:
    1. The response status code should be 200.
    2. The response should contain a list of urgent contact objects.
    3. The urgent contact objects in the response should belong to the logged-in user.
- Severity: Medium

Test ID: TS-10-4
Title: Test deleting an urgent contact

- Procedure:
    1. Create a new user object.
    2. Login with the newly created user's credentials.
    3. Create a new urgent contact object for the logged-in user.
    4. Send a DELETE request to /api/v1/urgent-contacts/{urgent_contact_id}/delete_urgent_contact/, where urgent_contact_id is the ID of the urgent contact object created in step 3.
    5. Check that the response status code is 204.
    6. Send a GET request to /api/v1/urgent-contacts/my-urgent-contacts/.
    7. Check that the response status code is 200.
    8. Check that the response does not contain the deleted urgent contact object.
- Outcome:
    1. The response status code after the DELETE request should be 204.
    2. The response status code after the GET request should be 200.
    3. The response after the GET request should not contain the deleted urgent contact object.
- Severity: High

Test ID: TS-10-5

Title: Test if a user can add an urgent contact for themselves

- Procedure:
    1. Create a user and get their authentication token.
    2. Send a POST request to the endpoint with the authentication token in the header and the urgent contact data in the body.

3. Check that the response has a status code of 201 and that the urgent contact data in the response body matches the data sent in the request.

- Outcome:
    1. If the test passes successfully, the system should return a status code of 201 indicating that the urgent contact was successfully created, and the urgent contact data in the response body should match the data sent in the request.
- Severity: Medium

Test ID: TS-10-6
Title: Test if a user can update an urgent contact they added

- Procedure:
    1. Create a user and get their authentication token.
    2. Add an urgent contact for the user.
    3. Send a PATCH request to the endpoint with the urgent contact ID, the authentication token in the header, and the updated urgent contact data in the body.
    4. Check that the response has a status code of 200 and that the urgent contact data in the response body matches the updated data sent in the request.
- Outcome:
    1. If the test passes successfully, the system should return a status code of 200 indicating that the urgent contact was successfully updated, and the urgent contact data in the response body should match the updated data sent in the request.
- Severity: Medium

Test ID: TS-10-7

Title: Test updating an urgent contact with invalid data

- Procedure:
    1. Create a new urgent contact object with the user's account details.
    2. Attempt to update the urgent contact object with invalid data.
    3. Check that the response status code is 400 and the response body contains an error message indicating the invalid data.
- Outcome:

1. The response status code should be 400 and the response body should contain an error message indicating the invalid data.

- Severity: Medium

Test ID: TS-10-8

Title: Test deleting an urgent contact

- Procedure:
    1. Create a new urgent contact object with the user's account details.
    2. Send a DELETE request to the urgent contact detail endpoint with the urgent contact object's id.
    3. Check that the response status code is 204 and the urgent contact object is no longer in the database.
- Outcome:
    1. The response status code should be 204 and the urgent contact object should no longer exist in the database.
- Severity: High

Database Tests

Test ID: TS-11-1
Title: Verify that a new user can be created in the database

- Procedure:
    1. Create a new User object with valid data.
    2. Save the object to the database.
    3. Retrieve the object from the database.
    4. Verify that the retrieved object has the same data as the original object.
- Outcome:
    1. The new User object should be saved to the database and its data should match the original object.
- Severity: Medium

Test ID: TS-11-2
Title: Verify that a user can be updated in the database

- Procedure:
    1. Create a User object in the database.
    2. Update the object with new data.
    3. Save the object to the database.
    4. Retrieve the object from the database.
    5. Verify that the retrieved object has the updated data.

- Outcome:
    1. The User object should be updated in the database with the new data.
- Severity: Medium

Test ID: TS-11-3
Title: Verify that a user can be deleted from the database

- Procedure:
    1. Create a User object in the database.
    2. Delete the object from the database.
    3. Attempt to retrieve the object from the database.
    4. Verify that the object cannot be retrieved.
- Outcome:
    1. The User object should be deleted from the database and cannot be retrieved.
- Severity: High

Test ID: TS-11-4
Title: Verify that a relationship between two models is created in the database

- Procedure:
    1. Create a User object in the database.
    2. Create a Profile object in the database, associated with the User object.
    3. Retrieve the Profile object from the database.
    4. Verify that the retrieved object has the correct User object associated with it.
- Outcome:
    1. The Profile object should be created in the database and associated with the correct User object.
- Severity: Medium

Non-functional Tests

Performance Tests

Test ID: PCU-001
Title: Evaluate the maximum number of concurrent users the system can handle

- Procedure:
    1. Define a range of concurrent users to test, such as 10, 50, 100, and 500 users.
    2. Use a load testing tool like Apache JMeter to simulate the specified number of concurrent users accessing the system.

3. Increase the number of concurrent users until the system starts to show performance degradation.
4. Measure the response time, throughput, and resource utilization at different levels of concurrent users.
5. Repeat the test multiple times for each level of concurrent users to ensure consistent results.
6. Analyze the data to determine the maximum number of concurrent users the system can handle before showing signs of performance degradation.
7. Use the data to optimize the system and improve its scalability and performance.

- Outcome:
    1. Determine the maximum number of concurrent users that the system can handle before showing performance degradation.
    2. Identify the performance bottlenecks related to the system's concurrent user handling capabilities.
    3. Ensure that the system can handle a sufficient number of concurrent users under expected usage conditions.
- Severity: High

Test ID: API-001
Title: Evaluate the response time of a single API request under different load conditions

- Procedure:
    1. Define a range of load conditions to test, such as 10, 50, 100, and 500 requests per second.
    2. Use a load testing tool like Apache JMeter to simulate the specified number of requests per second and send requests to the API endpoint.
    3. Measure the response time for each request under each load condition and record the results.
    4. Repeat the load testing multiple times for each load condition to ensure consistent results.
    5. Analyze the data to identify any trends or issues in API performance under different load conditions.
    6. Determine the optimal load conditions for the system.
    7. Use the data to optimize the API and improve its scalability and performance.
- Outcome:
    1. Identify the optimal load conditions for the system's API.

2. Determine the maximum load that the API can handle before showing signs of performance degradation.
3. Identify the performance bottlenecks related to the API's response time under different load conditions.
4. Ensure that the API can handle expected usage conditions with acceptable response times.

- Severity: Medium

Test ID: DBL-001
Title: Evaluate the impact of database load on system performance

- Procedure:
    1. Define a range of database load conditions to test, such as 10, 50, 100, and 500 database queries per second.
    2. Use a load testing tool like Apache JMeter to simulate the specified number of database queries per second and concurrent users performing database-intensive operations on the system.
    3. Measure the response time, throughput, and resource utilization of the system under different levels of database load.
    4. Repeat the load testing multiple times for each load condition to ensure consistent results.
    5. Analyze the data to determine the impact of database load on system performance and identify any performance bottlenecks related to the database.
    6. Use the data to optimize the database and improve its scalability and performance.
- Outcome:
    1. Determine the impact of database load on system performance.
    2. Identify the performance bottlenecks related to the database under different load conditions.
    3. Ensure that the database can handle expected usage conditions with acceptable response times.
    4. Optimize the database for improved scalability and performance.
- Severity: High

Test ID: STC-001
Title: Evaluate the performance of the system under stress conditions

- Procedure:
    1. Define a range of stress conditions to test, such as 1000, 5000, and 10000 concurrent users and requests per second.

2. Use a load testing tool like Apache JMeter to simulate a heavy load on the system.
3. Measure the response time, throughput, and resource utilization of the system under heavy load.
4. Monitor the system for errors, crashes, or other signs of performance degradation.
5. Repeat the load testing multiple times for each stress condition to ensure consistent results.
6. Analyze the data to determine the maximum load the system can handle before showing signs of performance degradation and identify any performance bottlenecks under stress conditions.
7. Use the data to optimize the system and improve its scalability and performance.

- Outcome:
  1. Determine the maximum load that the system can handle before showing signs of performance degradation.
  2. Identify the performance bottlenecks related to the system's resource utilization under heavy load.
  3. Ensure that the system can handle expected usage conditions under stress conditions without crashes or errors.
  4. Optimize the system for improved scalability and performance.
- Severity: High

Security Tests

Test ID: ST-1
Title: SQL injection

- Procedure:
  1. Identify a form or input field that accepts user input that will be used in a database query.
  2. Enter SQL code into the input field that will modify or delete data from the database.
  3. Submit the form or input and verify that the SQL code was not executed and data was not modified or deleted.
- Outcome:
  1. The system should detect and prevent SQL injection attacks.
- Severity: High

Test ID: ST-2
Title: Cross-site scripting (XSS)

- Procedure:

1. Identify a form or input field that accepts user input that will be displayed on a web page.
2. Enter HTML or JavaScript code into the input field that will be executed when the page is loaded.
3. Reload the page and verify that the code was not executed.

- Outcome:
    1. The system should detect and prevent XSS attacks.
- Severity: High

Test ID: ST-3
Title: Password security

- Procedure:
    1. Attempt to create a new account with a weak password (e.g. "password" or "123456").
    2. Verify that the password is rejected and the user is prompted to enter a stronger password.
    3. Attempt to log in with a correct username and a weak password.
    4. Verify that the login attempt is rejected and the user is prompted to enter a stronger password.
- Outcome:
    1. The system should enforce strong password requirements and prevent login attempts with weak passwords.
- Severity: Medium

Test ID: ST-4
Title: User authentication and authorization

- Procedure:
    1. Attempt to access a protected resource (e.g. a page or API endpoint) without logging in.
    2. Verify that the access is denied and the user is redirected to the login page.
    3. Attempt to access a protected resource with a valid username and password.
    4. Verify that the access is granted and the resource is displayed or returned.
    5. Attempt to access a protected resource with an invalid username and password.
    6. Verify that the access is denied and the user is prompted to enter valid credentials.
- Outcome:

1. The system should enforce authentication and authorization rules and prevent unauthorized access to protected resources.

- Severity: High

Test ID: ST-5
Title: Input validation

- Procedure:
    1. Attempt to enter invalid data into a form or input field (e.g. a string where a number is expected).
    2. Verify that the data is rejected and the user is prompted to enter valid data.
    3. Attempt to enter malicious data into a form or input field (e.g. a script or SQL code).
    4. Verify that the data is rejected and the user is prompted to enter valid data.
- Outcome:
    1. The system should validate all user input and prevent malicious or invalid data from being processed.
- Severity: Medium

Logging Tests

Test ID: LT-1
Title: Test if all significant events in the system are logged

- Procedure:
    1. Perform actions in the system that are expected to generate logs.
    2. Check the logs to ensure that all significant events have been logged.
- Outcome:
    1. All significant events in the system should be logged.
- Severity: High

Test ID: LT-2
Title: Test if the logs are properly formatted

- Procedure:
    1. Generate some logs in the system.
    2. Check the format of the logs.
    3. Ensure that the logs contain all necessary information, such as timestamps and severity levels.
- Outcome:

1. The logs should be properly formatted and contain all necessary information.

- Severity: Medium

Test ID: LT-3
Title: Test if the logs are secure

- Procedure:
    1. Generate some logs in the system.
    2. Check that the logs are not accessible by unauthorized users.
- Outcome:
    1. The logs should be secure and not accessible by unauthorized users.
- Severity: High

Test ID: LT-4
Title: Test if the logs are rotated and archived

- Procedure:
    1. Generate a large number of logs.
    2. Check that logs are rotated and archived at regular intervals.
    3. Check that archived logs are accessible when needed.
- Outcome:
    1. The logs should be rotated and archived at regular intervals and archived logs should be accessible when needed.
- Severity: Medium

Image Detection Tests

Test ID: ML -1
Title: Test if pedestrians are detected in the image

- Procedure:
    1. Run the trained yolo model on an image containing pedestrians walking on the street.
- Outcome:
    1. Check if the pedestrians are being detected by the model. Otherwise, we will train the model on more data.
- Severity: High

Test ID: ML -2
Title: Test if traffic lights are detected in the image

- Procedure:

1. Run the trained yolo model on an image containing traffic lights of different colors.

- Outcome:
    1. Check if the traffic lights are being detected by the model. Otherwise, we will train the model on more data.
- Severity: High

Test ID: ML - 3
Title: Test if traffic signs are detected in the image

- Procedure:
    1. Run the trained yolo model on an image containing traffic signs on the road.
- Outcome:
    1. If the traffic signs are being detected by the model, then we are ok. Otherwise, we will train the model on more data.
- Severity: High

Test ID: ML - 4
Title: Test if the model can be run on an android device
Procedure:

1. Run yolo image detector on an android device.
2. Check if it can be run on more simpler devices.

- Outcome:
    1. If the model can be run on an Android device, then we will locally run the detector in the application. Otherwise, we will find alternative mechanisms.
- Severity: High

Test ID: ML - 5
Title: Find the time take for processing each image on the device
Procedure:

1. Run the yolo image detector on a selected device.

- Outcome:
    1. If the model can be run on the device in feasible time, then we will locally run the detector in the application. Otherwise, we will find alternative mechanisms to decrease run time of the model.
- Severity: High

Augmented Reality Tests

Test ID: AR - 1
Title: Check if the turn-by-turn navigation data is properly acquired from the Mapbox API.
Procedure:

1. Login to the application.
2. Choose the destination path.
3. Click “Navigate” button.
4. Check if turns are properly marked in the Google Maps view, which is below the real-life view.

- Outcome:
    1. If the turns are marked properly, then we are done. Otherwise, there is a bug in marking algorithm. It should be fixed. Additionally, check if Mapbox API provides the correct latitude and longitude values.
- Severity: High

Test ID: AR - 2
Title: Check if the deployed augmented reality markers are correctly put.
Procedure:

1. Login to the application.
2. Choose the destination path.
3. Click “Navigate” button.
4. Check the actual real-life locations of the AR markers.

- Outcome:
    5. Altitude can cause problem. If this is the case, deploy Terrain Anchors rather than Geospatial Anchors (in the same Google Geospatial library). If the accuracy is low, wait for a few seconds and test it again.
- Severity: High

Test ID: AR - 3
Title: Check if the deployed augmented reality markers are pointed to the correct turn direction (left or right).
Procedure:

1. Login to the application.
2. Choose the destination path.
3. Click “Navigate” button.
4. Check the actual real-life locations of the AR markers.

- Outcome:
    1. If they are not aligned properly to show the turn directions, then the rotational pose values are not entered correctly. Check the algorithm that calculates those values and fix it. Otherwise, it passes the test.
- Severity: High

6. Consideration of Various Factors in Engineering Design

When designing an engineering project, various factors are considered. These factors can vary from economic to socio-cultural aspects. This situation is also valid in our senior project. Here are all the factors that we think are important in our progress:

- **Public Health**: RoadVisor does not necessarily have an impact on public health. Usually, projects in the medical domain focus on this factor. Therefore public health will not be discussed further.
- **Safety**: Both the drivers' and pedestrians' safety is one of the most important factors for RoadVisor. RoadVisor is a mobile application that requires drivers to look at the screen of their phones. We recognize that this can be risky if the driver does not pay enough attention to the road. That is why our machine learning-backed features, such as traffic sign/light detection and pedestrian detection, play a massive role in this matter. We highly care about that the driver's attention must be primarily on the road. Therefore, if the driver losses their focus on the road, these features will help the driver to get back to it. Additionally, independent from our application, we must consider the human factor. This means that drivers can get distracted because of other factors. They can be tired, or some other elements might catch their attention. Hence, RoadVisor has the potential to increase the safety of the drivers.
- **Cultural:** RoadVisor does not necessarily have an impact on cultural factors either. Therefore it will not be discussed further.
- **Environmental:** RoadVisor is primarily a navigation application. And like all navigation applications, it aims to provide the shortest and fastest path to the destination. Because the drivers know how to go to their destination beforehand, they will probably spend less time trying to find the proper direction. This will also mean that their car will exhaust less carbon dioxide. Therefore RoadVisor can have a positive effect on the environment.
- **Economic:** The inspiration that led us to develop RoadVisor is the augmented reality feature in high-tier Mercedes-Benz cars. As mentioned, this feature is only available in expensive Mercedes cars. The aim of RoadVisor is to bring this feature to mobile phones; therefore, everyone with a suitable phone can access this feature. From an economic perspective, this will save drivers money if they want this feature. Also, they do not require to purchase any external pieces as they already have their phones. That is why the economic factor is the most important one among all factors for RoadVisor.
- **Welfare**: This factor can be addressed with the economic factor. It is mentioned that RoadVisor can help drivers to save money. If we think about a country or globally, people can spend these resources on the things they need. With these needs being satisfied, welfare can increase.

- **Global:** Considering the economic, safety and environmental effects of RoadVisor, if these aspects are scaled to have a global role, it is safe to say that RoadVisor can have positive global effects.
- **Social:** It is worth mentioning that with the safety provided for drivers and pedestrians, roads can be safer. By having safe roads and safety measures, this can create a sense of security among people. Therefore RoadVisor can positively affect the safety of society and each person in this society.

Factors	**Level of Effect** (on a scale of 0 to 10)
Public Health	1
Safety	9
Cultural	0
Environmental	6
Economic	10
Welfare	8
Global	7
Social	7

Figure 6.1: Various Factors Affecting RoadVisor and Their Importance

7. Teamwork Details

In this section, the teamwork details will be explained in detail.

7.1 Contributing and Functioning Effectively on the Team

Contribution of each team member is essential for the future and robustness of the project. Each team member is required to contribute to the project and if the percentage of contribution of each team member is similar to each other, a fair and efficient working environment can be achieved. Functioning of each team member is also crucial because software development is a complex and collaborative process that requires a team effort to achieve successful outcomes. Each stage of project planning and development requires specific skills and expertise that are often spread among team members. Effective functioning of team members is essential for and improves the

completeness and quality of work, meeting deadlines, collaboration, communication and cost effectiveness. Here is the contribution of each team member to the project.

It should be noted that all the decision making and project planning involved all of the team members' thoughts and judgment. Similarly, all of the team members contributed equally to the documentation of the project.

Emin Berke Ay: He is working on the frontend portion of the application. He has regularly contributed to the project reports. He regularly attends meetings. He has made significant contributions to the Design report. He is working on the sign up part of the frontend.

Nurettin Onur Vural: He has been working on the frontend. He has made contributions to all reports. He also regularly attends team discussions. He has also made contributions to the Design report.

Arda İçöz: He is responsible for working on the frontend. Arda has contributed extensively to the reports. He also worked on formatting documents. He served as the team lead during the previous semester. He regularly participates in meetings. During CS491, he has made significant progress in the area of the frontend. The progress involves the AR navigation section of the application. He has also made substantial contributions to the Design report.

Ammaar Iftikhar: He worked on Machine learning models. Completed and tested the lane detection model during CS491. He contributed to all the reports. He regularly participates in meetings. He is also responsible for creating and managing the group website. During CS492, he completed the yolo model for the detection of traffic lights, traffic signs, and pedestrians. The output of the model was analyzed, and they seem to be promising. He also contributed to the Design report. He is serving as the team leader this semester. In this role, he is responsible for arranging meetings and setting deadlines and tasks. He is currently working on making the yolo model work on Android devices.

Ahmet Faruk Ulutaş: He is responsible for working on the backend portion of the application. He completed implementing the backend and setting up the server. He has also contributed significantly to the Design report. He also made a decent amount of contributions in the frontend area. He is also regularly attending the meetings. During CS492, he has been completing tasks assigned.

7.2 Helping creating a collaborative and inclusive environment

Helping create a collaborative and inclusive environment is essential for enhancing productivity, improving problem-solving, increasing innovation, delivering high-quality products, and improving team dynamics. We, as a team, realized the importance of these and each team member did their best to help achieve a collaborative and inclusive environment. We did our best to not leave anyone behind or out. For instance, we tried to schedule our meetings so that everyone on the team could join it. It is quite difficult to find a time slot suitable to 5 people every single time, but inclusiveness is very essential for a team to function. It also helps boost morale and keeps the momentum and the dynamics of the team going. We also tried to find a time slot suitable for every one of us when meeting with our supervisors and innovation experts since those are also quite essential meetings and we frankly did not want anyone to be left behind. However, we do not live in a perfect world and setbacks do happen. Nevertheless, we were prepared for them as well. When a team member, for any reason, could not join a meeting we always kept them up to date by giving a brief about the meeting and the key points discussed in that meeting. This not only kept them up to date, but also felt them included as well.

7.3 Taking lead role and sharing leadership on the team

Taking a lead role and sharing leadership on a team is essential for achieving success in any project. When one member of a team takes on a lead role, it can help the team stay organized, focused, and productive. Without a conductor, the orchestra would not perform correctly. This is also the case for a team without a leader. Sharing leadership fosters collaboration, increases accountability, improves delegation, enhances team dynamics, builds leadership skills, and provides greater flexibility. By working together and sharing leadership responsibilities, teams can achieve their goals more effectively and efficiently. We approached sharing leadership in a different manner. We have a team leader. Then, we have sub-team leaders. For instance, the frontend team has a separate leader then the main leader, but the sub-team leaders also report to the main leader. Sub-team leaders do not change, but we rotate the main leadership role among team members. This way, leadership roles are not burdened on a single person and the workload that being a leader brings is also shared among team members.

References

[1] E. Zvornicanin, “What is Yolo Algorithm?,” *Baeldung on Computer Science*, 04-Nov-2022. [Online]. Available: https://www.baeldung.com/cs/yolo-algorithm#:~:text=3.-,You%20Only%20Look%20Once%20(YOLO),main%20reason%20for%20its%20popularity. [Accessed: 13-Mar-2023].

[2] A. Pavlov, “Bosch Small Traffic Lights Dataset,” *Kaggle*, 15-Oct-2020. [Online]. Available: https://www.kaggle.com/datasets/researcherno1/small-traffic-lights. [Accessed: 13-Mar-2023].

[3] V. Sichkar, “Traffic signs dataset in Yolo Format,” *Kaggle*, 03-Apr-2020. [Online]. Available: https://www.kaggle.com/datasets/valentynsichkar/traffic-signs-dataset-in-yolo-format. [Accessed: 13-Mar-2023].

[4] “Papers with code - caltech pedestrian dataset dataset,” *Dataset | Papers With Code*. [Online]. Available: https://paperswithcode.com/dataset/caltech-pedestrian-dataset. [Accessed: 13-Mar-2023].

[5] VladYatsenko, “Vladyatsenko/car-assistant-android: Application which detects traffic signs using camera,” *GitHub*. [Online]. Available: https://github.com/VladYatsenko/car-assistant-android. [Accessed: 13-Mar-2023].

[6] Tensorflow, “Models/research/object_detection at master · Tensorflow/models,” *GitHub*. [Online]. Available: https://github.com/tensorflow/models/tree/master/research/object_detection. [Accessed: 13-Mar-2023].

[7] Kmshack, “Kmshack/trafficlightsdetector-android: TrafficLights detector for Android,” *GitHub*. [Online]. Available: https://github.com/kmshack/TrafficLightsDetector-Android. [Accessed: 13-Mar-2023].

[8] *Caffe*. [Online]. Available: http://caffe.berkeleyvision.org/. [Accessed: 13-Mar-2023].

[9] Davidbrai, “Davidbrai/deep-learning-traffic-lights: Code and files of the deep learning model used to win the NEXAR traffic light recognition challenge,” *GitHub*. [Online]. Available: https://github.com/davidbrai/deep-learning-traffic-lights. [Accessed: 13-Mar-2023].

[10] “Car and pedestrian detection using an Android smartphone,” *YouTube*, 11-Oct-2016. [Online]. Available: https://www.youtube.com/watch?v=Z1WfI7v7ibM. [Accessed: 13-Mar-2023].

[11] Gebort, “Gebort/FESTU.Navigator: Kotlin AR app for Indoor Navigation,” *GitHub*. [Online]. Available: https://github.com/Gebort/FESTU.Navigator. [Accessed: 13-Mar-2023].

[12] “How to: Augmented reality,” *YouTube*, 16-Jun-2021. [Online]. Available: https://www.youtube.com/watch?v=AHaSnyikRpU. [Accessed: 13-Mar-2023].

[13] “Phiar - live Ar Navigation,” *YouTube*, 16-Jan-2020. [Online]. Available: https://www.youtube.com/watch?v=iD1mv6qqg54. [Accessed: 13-Mar-2023].