ELECTRAMECH

Revolutionizing Vehicle Assistance: AI-Powered Solutions for Breakdown Challenges

Prepared By
TMP-2023-24-117

21'st May, 2024

PROJECT INTRODUCTION

- Currently, the world demand for electric vehicles is increasing, but in countries like Sri Lanka, there is a lack of people turning to electric vehicles.
- The reason for this is the lack of knowledge about electric vehicles, the difficulty of finding mechanics when the vehicle breaks down and the difficulty of finding necessary spare parts.
- The purpose of our grant is to provide a successful solution to avoid the hassle and inconvenience that occurs when an electric vehicle breaks down on the road.

OUR TEAM

PERERA B.N.H.
IT20096434

SIRIMANNA D.J.T.K
IT20038328

MADHUBHASHANA A.G.K
IT20038182

VIJERATHNA A.G.V.K.M.
IT20644826

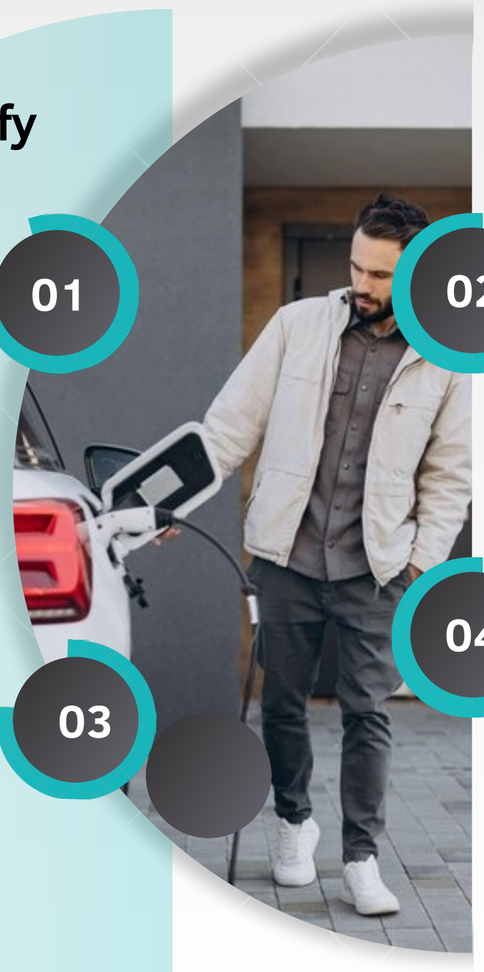
OVERALL PROJECT DESCRIPTION

- Individuals, especially those with limited vehicle knowledge, struggle to identify and address breakdown issues.
- Lack of easily accessible information, mechanics, spare part shops, and maintenance advice.
- Uncertainty about the appropriate actions when a breakdown occurs.
- Difficulty in finding reliable mechanics, particularly in unfamiliar areas.
- No effective mechanisms to inform vehicle owners about essential maintenance tasks, resulting in inconvenience and potential safety risks.

RESEARCH OBJECTIVES

01 Conversational AI chatbot to identify electric vehicle faults using text, dashboard warnings and provide solutions

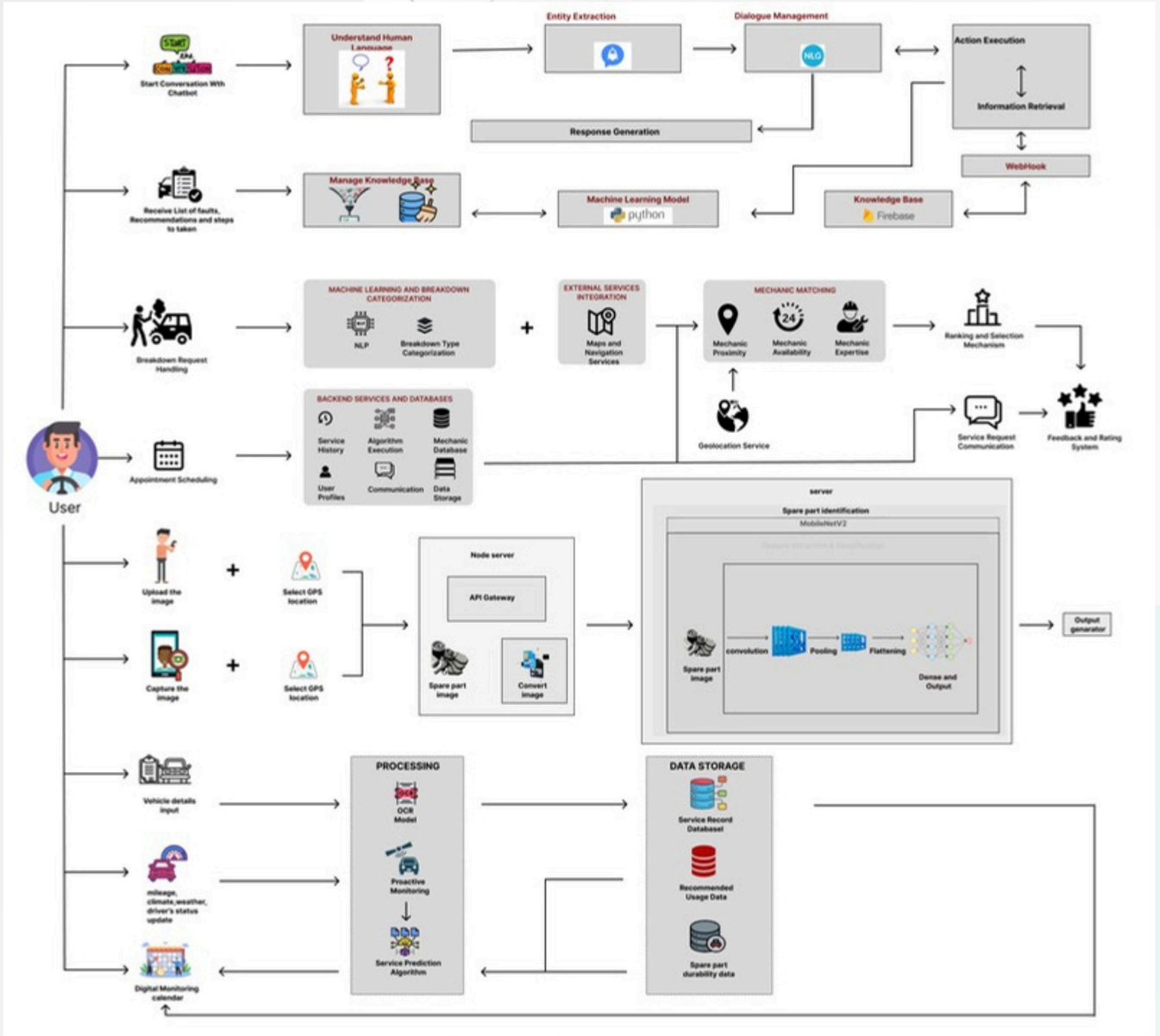
02 Intelligent Geolocation-based EV Breakdown Assistance with AI based Repair Service Cost Estimation

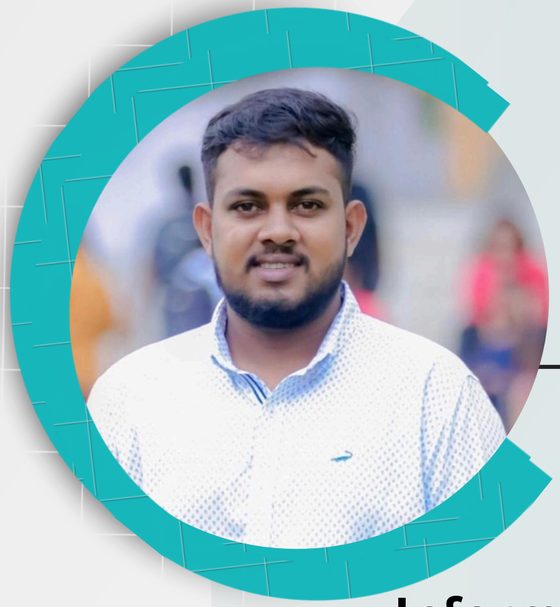


03 Automated Spare Parts Identification and Location System using Image Processing and Computer Vision Techniques

04 Enhancing User Experience in Vehicle Maintenance with a Smart Digital Monitoring Calendar and Reminder System

Overall System Architecture Diagram





IT20096434

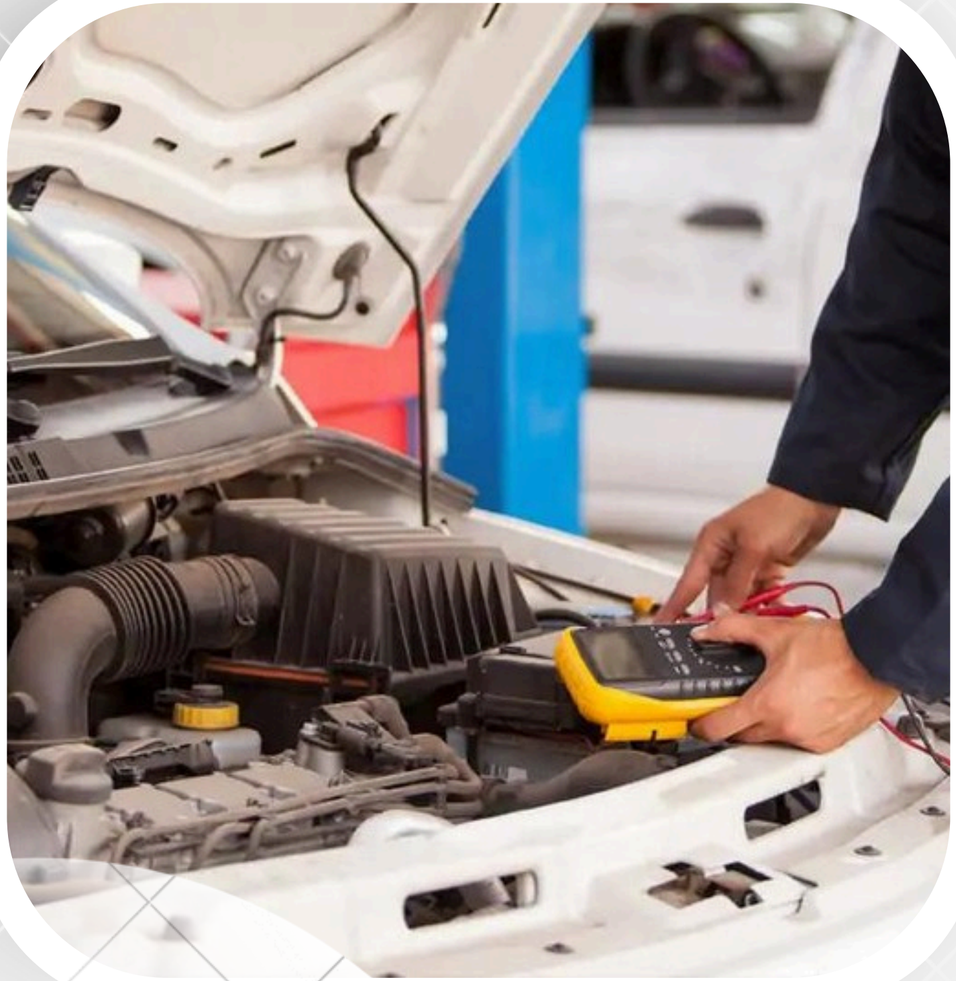
Perera B.N.H.

Bachelor of Science (Hons) in Information Technology Specializing in Information Technology

INTRODUCTION

Background

- EVs promise reduced emissions and reduced dependence on fossil fuels.
- However, unique challenges arise in managing breakdowns due to EVs' distinct technology.
- EV adoption is growing but still lags behind ICE vehicles.
- Proposed solution: AI-based chatbot for EV breakdown assistance.
- The chatbot would use AI and manufacturer data for real-time support.
- It offers accurate diagnostics, immediate assistance, and regular updates.
- Current breakdown assistance systems mainly focus on ICE vehicles, leaving a gap for specialized EV support.



Research Question

"In the event of an unexpected breakdown of an electric vehicle, How Can Conversational AI Chatbots Enhance Electric Vehicle Breakdown Support? "

This research addresses the need for efficient support during electric vehicle (EV) breakdowns on the road. By harnessing machine learning, AI, and natural language processing, the goal is to create a user-friendly chatbot that can analyze driver-provided text data and dashboard warning images to swiftly identify faults and offer recommendations, ultimately enhancing the assistance experience for EV drivers.

Research Gap

	Research A	Research B	Research C	Proposed System
Specified for Electric Vehicles (EV chatbot)	✗	✗	✗	✓
Availability of Personalized Solutions	✓	✓	✗	✓
Ability to Ask Follow Back Questions	✗	✗	✗	✓
Availability of a Knowledge Base	✓	✗	✓	✓
Update Knowledgebase Easily	✗	✗	✗	✓
Using ML and NLP for Real-Time Breakdown Identification and Assistance	✗	✗	✗	✓

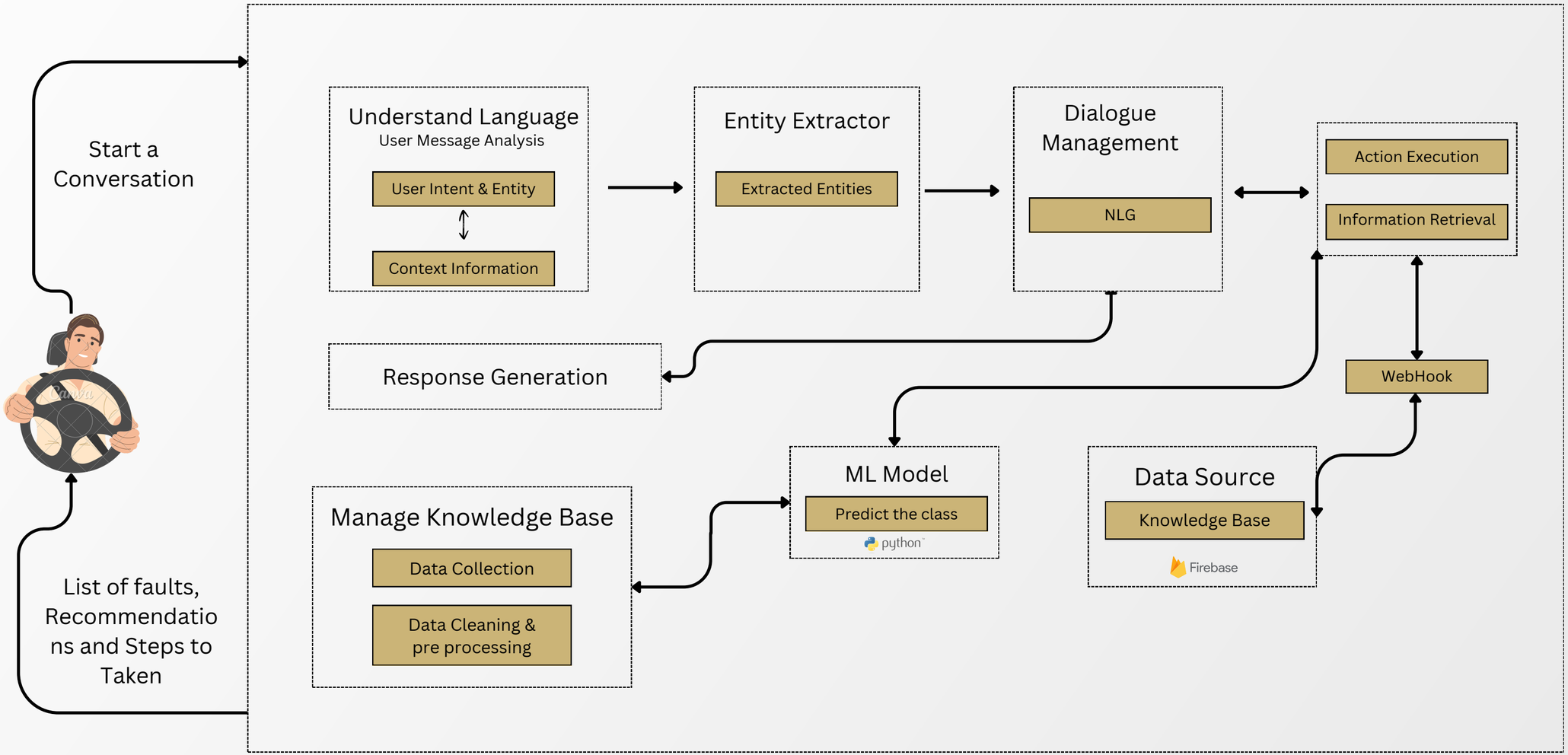
Specific Objective

IMPLEMENT A SMART AI CHATBOT TO RECEIVE TEXT AND DASHBOARD WARNING IMAGES, PROVIDING TIMELY SOLUTIONS FOR ELECTRIC VEHICLE BREAKDOWN SITUATIONS. THE KNOWLEDGE BASE WILL BE KEPT CURRENT AND ENHANCED TO ENSURE ACCURATE ASSISTANCE.

Sub Objectives

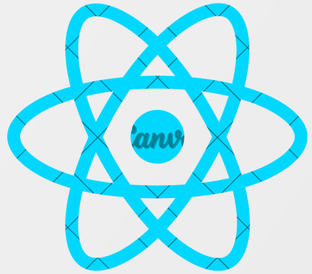
- Data Collection
- Data Preprocessing
- Data Cleaning
- AI Model Development
- Knowledge Base Creation
- Implement NLP techniques
- User Interface
- Testing and Validation

System Architecture Diagram



TECHNIQUES & TECHNOLOGIES

Technology	• React Native • Python • Flask • MongoDB • Node Server
Techniques	• Machine Learning model • Recurrent Neural Network (RNN) • Intent Recognition • Entity Extraction • Dialogue Management • BERT Framework
Algorithm	• Sec2Sec • LSTM



METHODOLOGY

DATA COLLECTION

DATA PRE-PROCESSING

CREATING ACTIONS BASED ON THE EV FAULTS

VALIDATE ANSWERS

CREATE EV FAULT-BASED KNOWLEDGE BASERE

DATA VISUALISATION

KNOWLEDGE BASE UPDATED

VERIFY RECOMENDATIONS WITH EXPERTS

COMPARISON

FRAMEWORK	EV BOT	MICROSOFT BOT FRAMEWORK	IBM WATSON ASSISTANCE	Dialogflow (formerly API.ai)
NLU	Highly customizable NLU component with support for complex intents and entities	Limited NLU capabilities with predefined intents and entities	Provides NLU capabilities with predefined intents and entities	Provides NLU capabilities with predefined intents and entities
DIALOGUE MANAGEMENT	Customizable dialogue management options	Rule-based dialogue management	Rule-based dialogue management options	Rule-based dialogue management options
CUSTOMIZABLITY	Highly customizable	Limited customizability due to closed-source nature	Limited Customizability	Limited Customizability
MACHINE LEARNING CAPABILITIES	Support integration of ML models	Limited ML capabilities	Provides machine learning capabilities	Provides ML capabilities.
OPEN-SOURCE	YES	NO	NO	NO

DATA COLLECTION AND PRE-PROCESSING

```
_=train_data_iterator.next()
print(f'Number of train batches: {len(train_data)}')
print(f'Number of training data: {len(train_data)*batch_size}')
print(f'Number of validation batches: {len(val_data)}')
print(f'Number of validation data: {len(val_data)*batch_size}')
print(f'Encoder Input shape (with batches): {_[0].shape}')
print(f'Decoder Input shape (with batches): {_[1].shape}')
print(f'Target Output shape (with batches): {_[2].shape}')
```

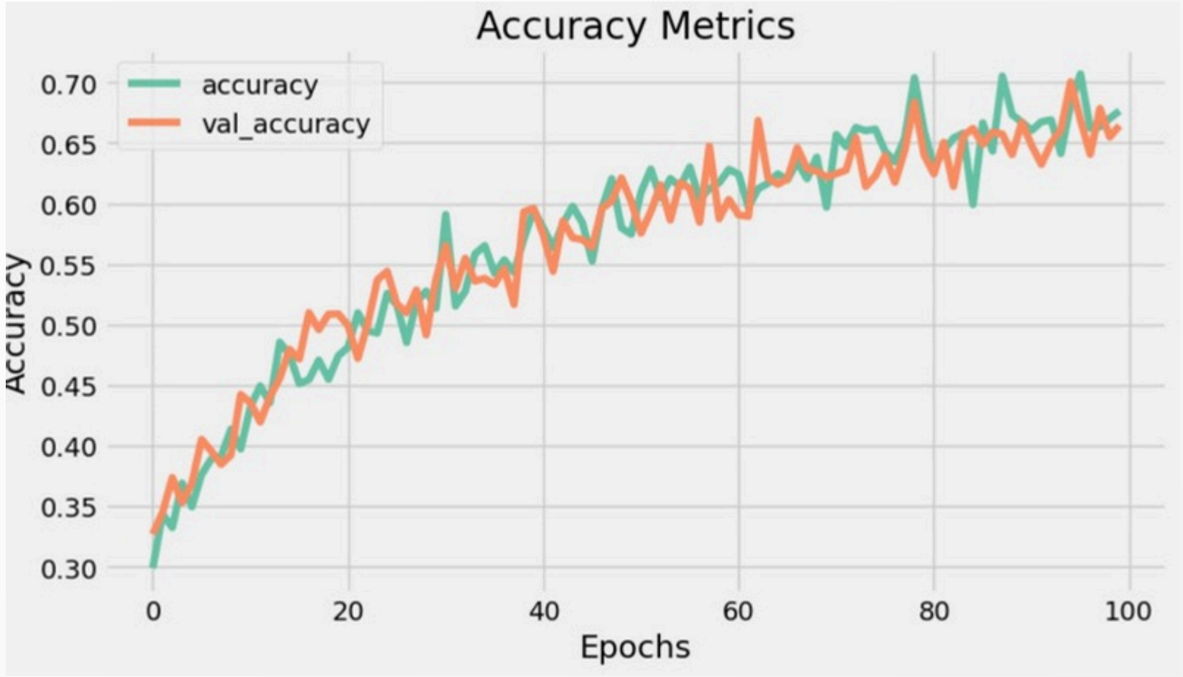
```
Number of train batches: 23
Number of training data: 3427
Number of validation batches: 3
Number of validation data: 447
Encoder Input shape (with batches): (149, 30)
Decoder Input shape (with batches): (149, 30)
```

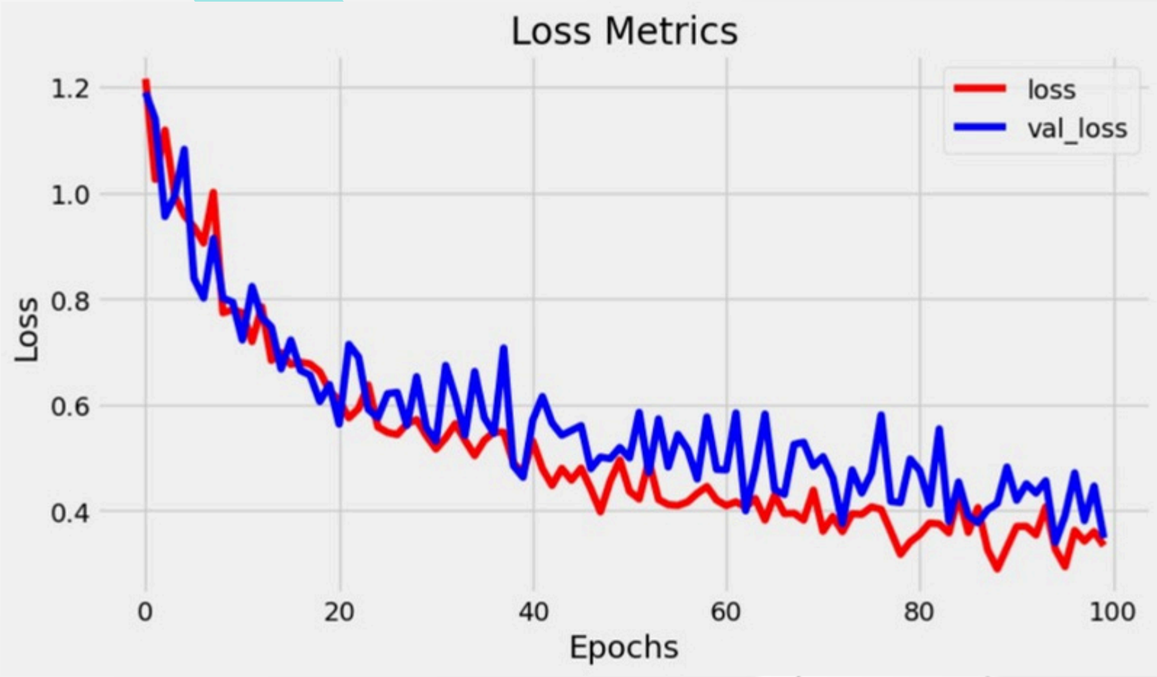
```
[ ]  history=model.fit(
        train_data,
        epochs=100,
        validation_data=val_data,
        callbacks=[
            tf.keras.callbacks.TensorBoard(log_dir='logs'),
            tf.keras.callbacks.ModelCheckpoint('ckpt',verbose=1,save_best_only=True)
        ]
    )
```

ACCURACY AND LOSS

```
Epoch 1/100
23/23 [==============================] - ETA: 0s - loss: 1.6494 - accuracy: 0.2188
Epoch 1: val_loss improved from inf to 1.19003, saving model to ckpt
WARNING:tensorflow:Model's `__init__()` arguments contain non-serializable objects. Please implement a `get_config()` method in the subclassed Model for proper saving and loading. Defaulting to empty config
WARNING:tensorflow:Model's `__init__()` arguments contain non-serializable objects. Please implement a `get_config()` method in the subclassed Model for proper saving and loading. Defaulting to empty config
WARNING:absl:Found untraced functions such as _update_step_xla, lstm_cell_layer_call_fn, lstm_cell_layer_call_and_return_conditional_losses, lstm_cell_1_layer_call_fn, lstm_cell_1_layer_call_and_return_cond
WARNING:tensorflow:Model's `__init__()` arguments contain non-serializable objects. Please implement a `get_config()` method in the subclassed Model for proper saving and loading. Defaulting to empty config
WARNING:tensorflow:Model's `__init__()` arguments contain non-serializable objects. Please implement a `get_config()` method in the subclassed Model for proper saving and loading. Defaulting to empty config
23/23 [==============================] - 108s 4s/step - loss: 1.6314 - accuracy: 0.2221 - val_loss: 1.1900 - val_accuracy: 0.3270
Epoch 2/100
23/23 [==============================] - ETA: 0s - loss: 1.2275 - accuracy: 0.3114
Epoch 2: val_loss improved from 1.19003 to 1.13934, saving model to ckpt
WARNING:tensorflow:Model's `__init__()` arguments contain non-serializable objects. Please implement a `get_config()` method in the subclassed Model for proper saving and loading. Defaulting to empty config
WARNING:tensorflow:Model's `__init__()` arguments contain non-serializable objects. Please implement a `get_config()` method in the subclassed Model for proper saving and loading. Defaulting to empty config
WARNING:absl:Found untraced functions such as _update_step_xla, lstm_cell_layer_call_fn, lstm_cell_layer_call_and_return_conditional_losses, lstm_cell_1_layer_call_fn, lstm_cell_1_layer_call_and_return_cond
WARNING:tensorflow:Model's `__init__()` arguments contain non-serializable objects. Please implement a `get_config()` method in the subclassed Model for proper saving and loading. Defaulting to empty config
WARNING:tensorflow:Model's `__init__()` arguments contain non-serializable objects. Please implement a `get_config()` method in the subclassed Model for proper saving and loading. Defaulting to empty config
23/23 [==============================] - 79s 4s/step - loss: 1.2191 - accuracy: 0.3128 - val_loss: 1.1393 - val_accuracy: 0.3669
```

Accuracy and Loss Metrics





METHODOLOGY

FUNCTIONAL REQUIREMENTS

- Driver able to ask questions and chatbot provide feedback based on their EV faults.
- Able to receive and understand questions related to EV faults from driver.
- Able to identify the fault based on driver provided data and provide recommendations for the identified fault.
- Provide information about the steps to be taken.
- Able to maintain a record of EV fault and provide more adviced as necessary.

NON-FUNCTIONAL REQUIREMENTS

- Performance
- Security
- Reliability
- Scalability
- Usability
- Accessibility
- Compatibility
- Maintainability
- Compliance
- Performance metrics

SYSTEM REQUIREMENTS

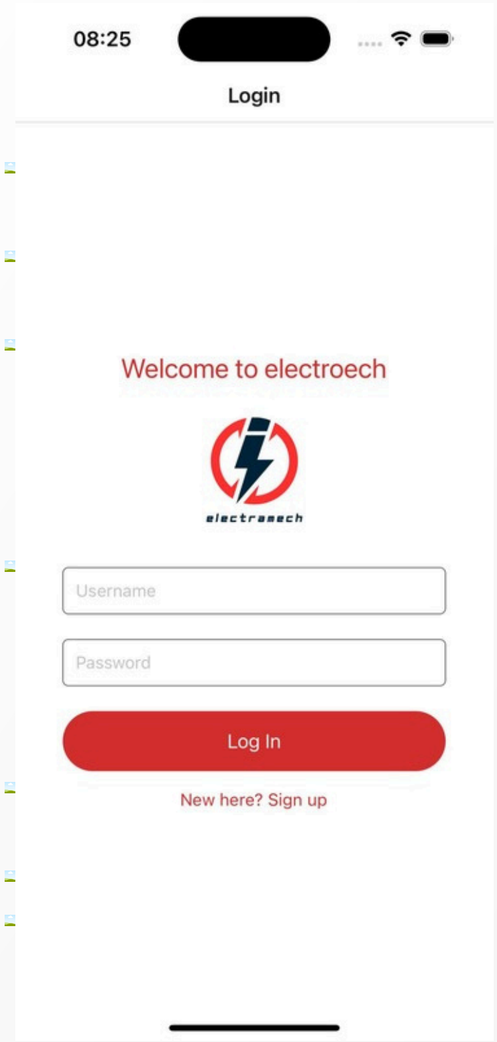
- Server infrastructure
- Software development platform
- Natural language processing (NLP) tools
- Machine learning models
- Database management system
- Integration with third-party systems
- Communication channels
- Security measures
- Monitoring and analytics tools

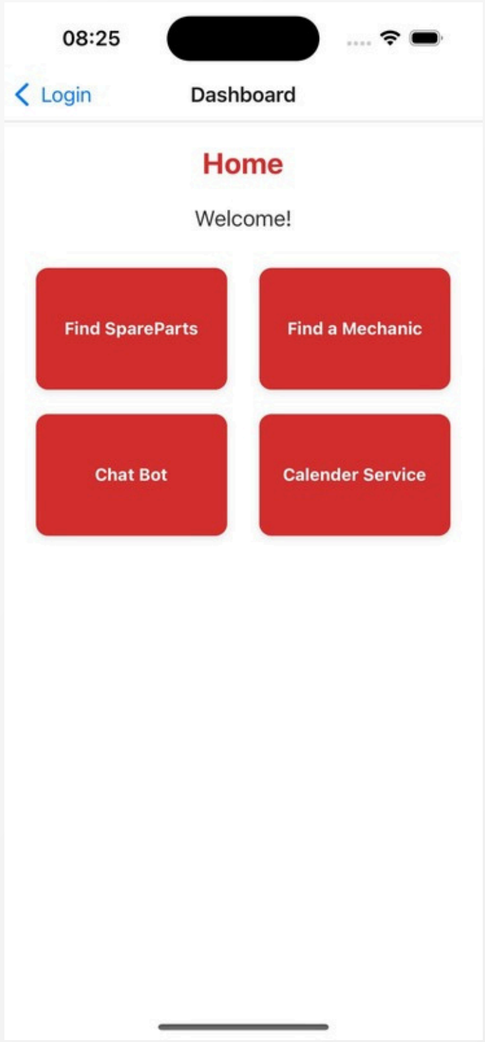
COMPLETION WORKS

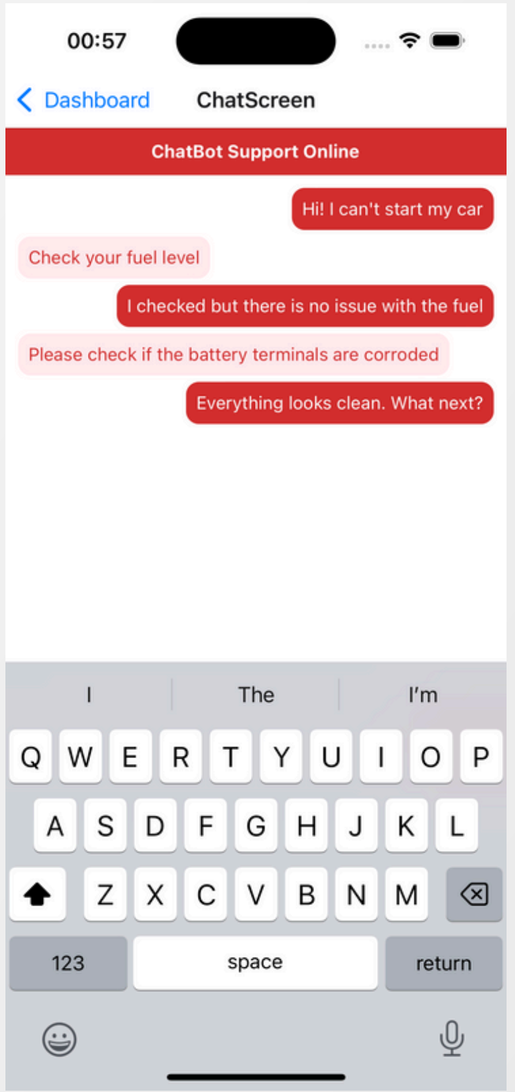
COMPLETION OF COMPONENTS

- DATA GATHERING AND PREPROCESSING
- CREATE FAULT-BASED KNOWLEDGE BASE
- CREATING THE ACTIONS BASED ON THE FAULT
- PROVIDING THE RECOMENDATION BASED ON THE FAULT GIVEN BY THE DRIVER
- VERIFY THE PROVIDED ANSWERS USING ML M
- HOSTING
- IMPLEMENTATION OF USER INTERFACES
- VERIFY SUGGESTIONS WITH EV EXPERTS

DISPLAY RESULTS IN MOBILE APPLICATION







IT20038328

Sirimanna D.J.T.K.

Bachelor of Science (Hons) in Information Technology Specializing in Information Technology

Background

- System for Electric Vehicle breakdown solutions using machine learning and geolocation.
- Connects vehicle owners with registered mechanics for emergency breakdown assistance.
- Skilled mechanics are matched to breakdown types and user locations from a database for efficient service.
- Allows users to schedule appointments for non-emergency breakdowns or inspections.
- Mechanic rating and feedback system to help users make informed decisions.

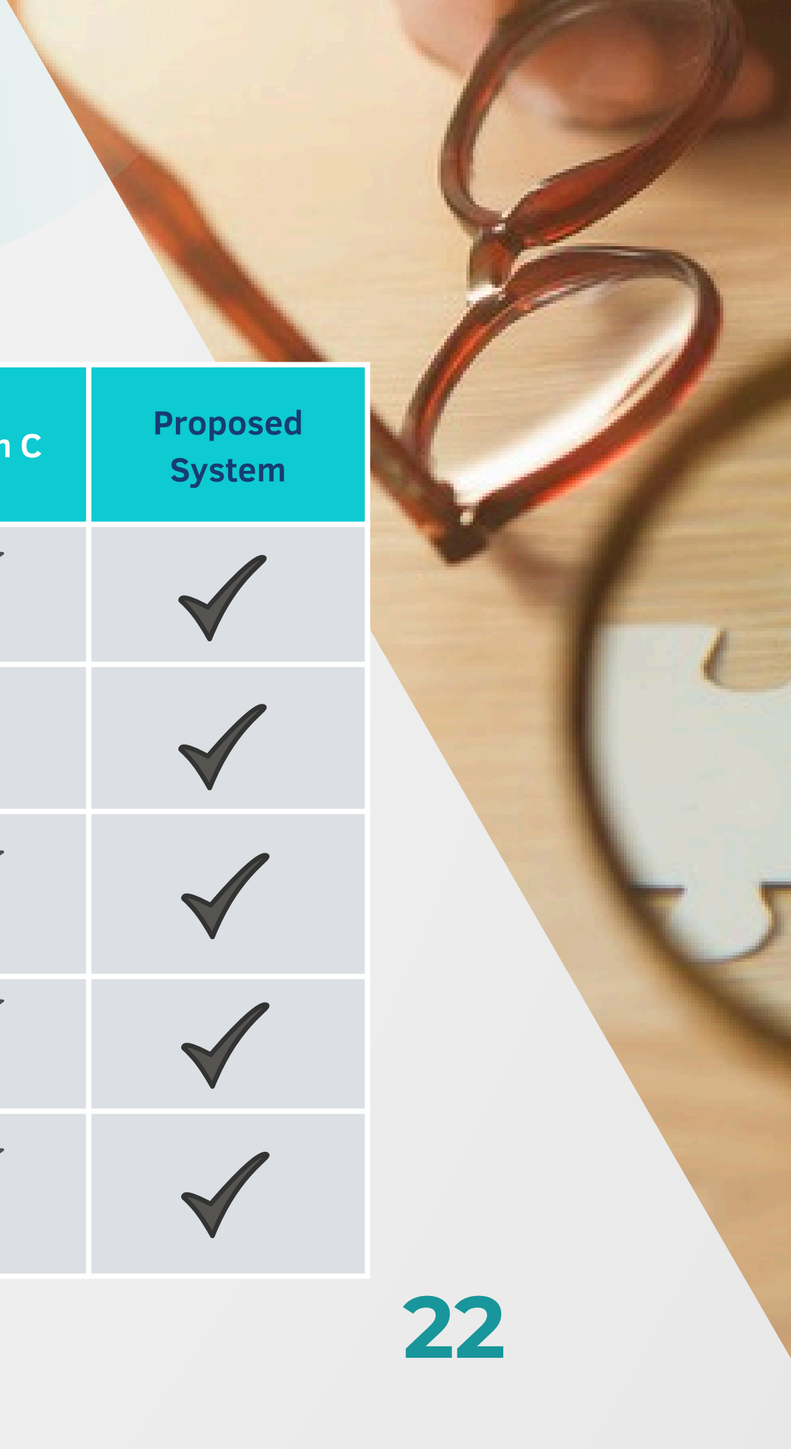
Research Question

"In the event of an unexpected breakdown of an electric vehicle in an unfamiliar location, how can the driver effectively diagnose the problem and locate a qualified mechanic for timely repairs?"

The research identifies a lack of efficient support for electric vehicle (EV) breakdowns in unfamiliar areas. By integrating machine learning, geolocation, and natural language processing, the aim is to develop a user-friendly system that can rapidly diagnose breakdowns, connect drivers with suitable mechanics, and improve the overall assistance experience for EV drivers.

Research Gap

	Research A	Research B	Research C	Proposed System
Specified for Electric Vehicles	✗	✗	✗	✓
Using Geolocation and GPS for locating	✓	✓	✓	✓
Using ML and NLP for Real-Time Breakdown Identification and Assistance	✗	✗	✗	✓
Proactive Repair Service scheduling	✓	✗	✗	✓
Emergency or Priority Service Request Broadcast	✗	✗	✗	✓



Specific Objective

Implement a Machine Learning model and algorithms that analyzes the breakdown description and suggests the most suitable mechanic based on specialization of mechanic and proximity to the driver’s location

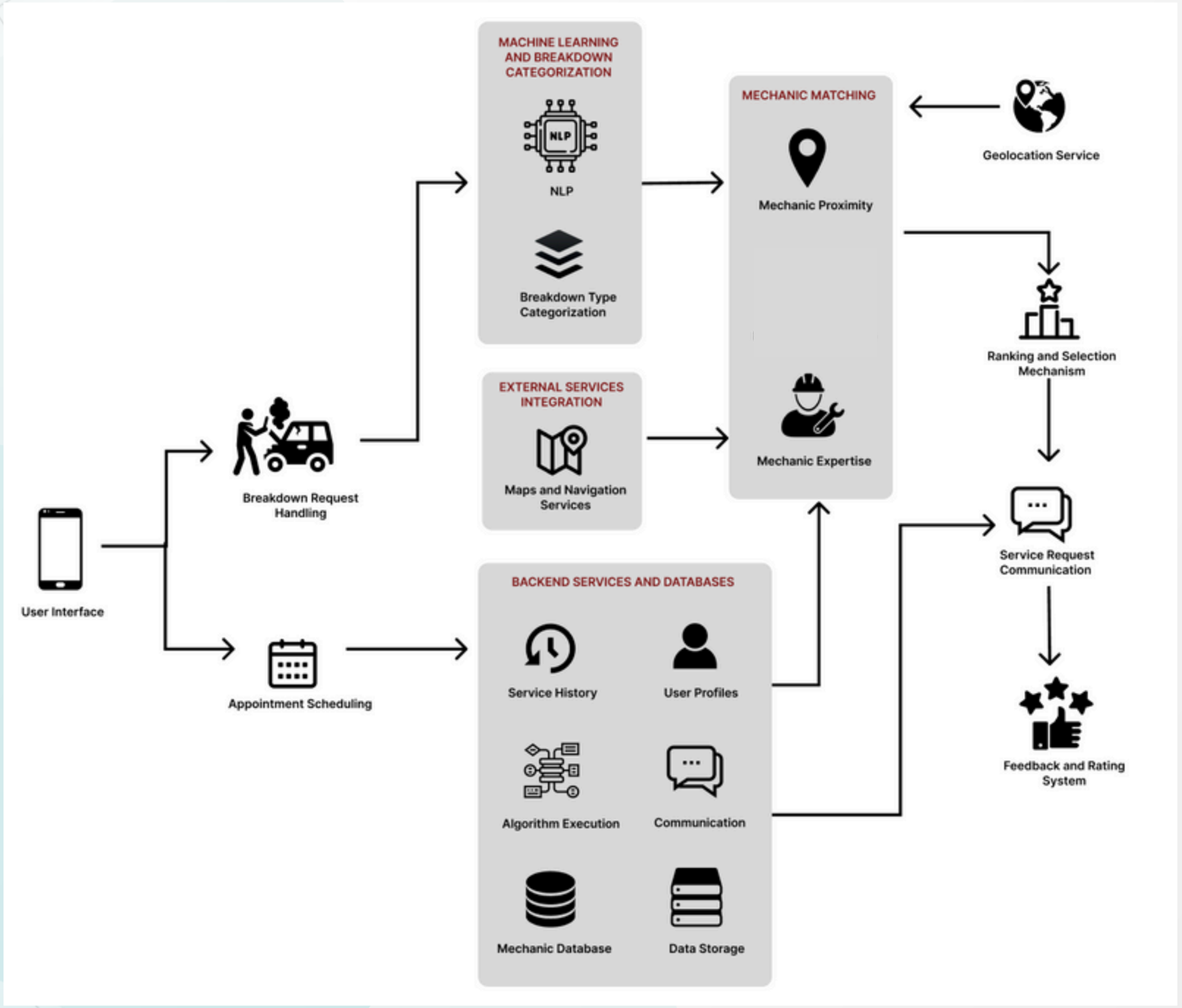
Sub Objectives

- Data Collection
- Data Preprocessing
- ML Model Development
- Implement NLP techniques
- Specialization Matching
- Geolocation Integration
- User Interface
- Testing and Validation
- Scalability and Performance:

Other Objectives

- Implement a private chat feature for Drivers and Mechanics
- Integrating a feedback and rating system for rate mechanics
- Implement a system for repair scheduling feature
- Test the final model on the testing data and implement it in a software application for practical use.

System Architecture Diagram



TECHNIQUES & TECHNOLOGIES

METHODOLOGY

Firebase

Technology	• React Native • Python • Firebase • Node Server • Geo Location • Flask
Techniques	• Auto Machine Learning (AML) • Sentiment analysis
Model	• Logistic regression

METHODOLOGY

- DATA COLLECTION
- DATA PRE-PROCESSING
- USE TRANSFER LEARNING BASED CNN ARCHITECTURES FOR TRAINING THE MODELS
- DATA AUGMENTATION AND SELECTED THE BEST TECHNIQUE TO RETRAIN MODELS TO ACHIEVE GOOD FITTING
- TUNING HYPERPARAMETERS AND RETRAINED MODELS TO SELECT THE BEST ARCHITECTURE
- DATA VISUALISATION
- HOST THE FINAL MODEL
- DISPLAYED THE RESULTS IN MOBILE APPLICATION

METHODOLOGY-
EVIDENCE OF COMPLETION

Classification report

```
# Model evaluation
report = classification_report(y_test, y_pred)
print(report)
```

```
                                precision    recall  f1-score   support

       Electrical and Software Issues       0.58      1.00      0.74        25
Infrastructure and Operational Issues       1.00      0.22      0.36         9
                  Mechanical Issues       0.95      0.72      0.82        25
          Safety and Hardware Issues       1.00      0.17      0.29         6

                            accuracy                           0.71        65
                           macro avg       0.88      0.53      0.55        65
                        weighted avg       0.82      0.71      0.67        65
```

+ Code + Markdown

METHODOLOGY-
EVIDENCE OF COMPLETION

Accuracy of the current model

```python
from sklearn.metrics import accuracy_score

model.eval()
predictions, true_labels = [], []
for batch in loader:
    batch = {k: v.to(device) for k, v in batch.items()}
    with torch.no_grad():
        outputs = model(**batch)

    logits = outputs.logits
    predictions.extend(torch.argmax(logits, dim=-1).tolist())
    true_labels.extend(batch['labels'].tolist())

print("Accuracy:", accuracy_score(true_labels, predictions))
```

Python

```
Accuracy: 0.8307692307692308
```

METHODOLOGY- EVIDENCE OF COMPLETION

Model training

```
# Model training
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

```
# Model prediction
y_pred = model.predict(X_test)
```

SYSTEM REQUIREMNTS

FUNCTIONAL REQUIREMENTS

- Users initiate breakdown assistance requests, providing descriptions and attaching media.
- The system identifies breakdown types using natural language processing and machine learning.
- Mechanics matching is listed by proximity and expertise.
- Real-time communication, mechanic selection, scheduling, feedback, and location tracking enhance user-mechanic interactions.

NON-FUNCTIONAL REQUIREMENTS

- Performance
- Usability
- Reliability
- Security
- Compatibility
- Maintainability
- Scalability
- Accessibility

SYSTEM REQUIREMENTS

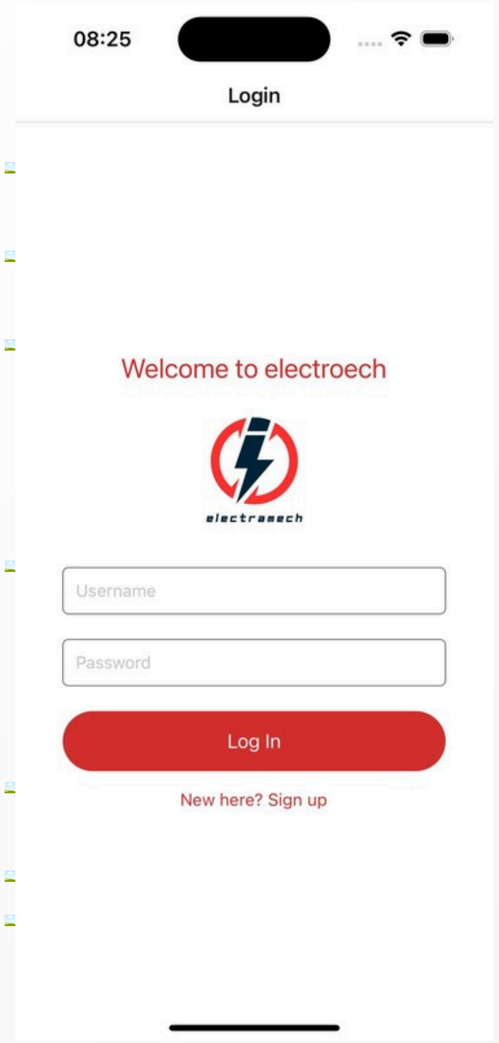
- Machine learning and NLP identify breakdown types and match mechanics by expertise and proximity.
- Real-time communication between users and mechanics is facilitated.
- Mechanic selection options and appointment scheduling are available.
- Feedback and ratings for mechanics' services are supported.
- Accurate location tracking via Google Geolocation API and real-time mapping.

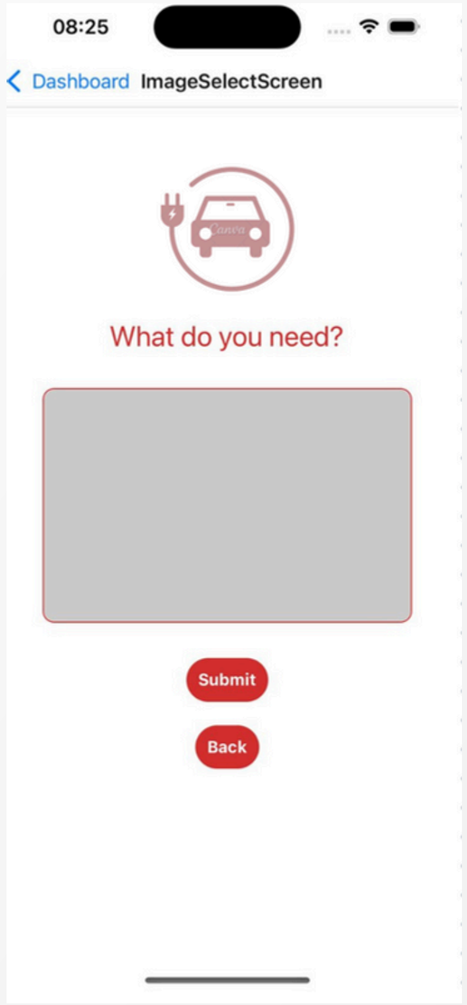
COMPLETION AND

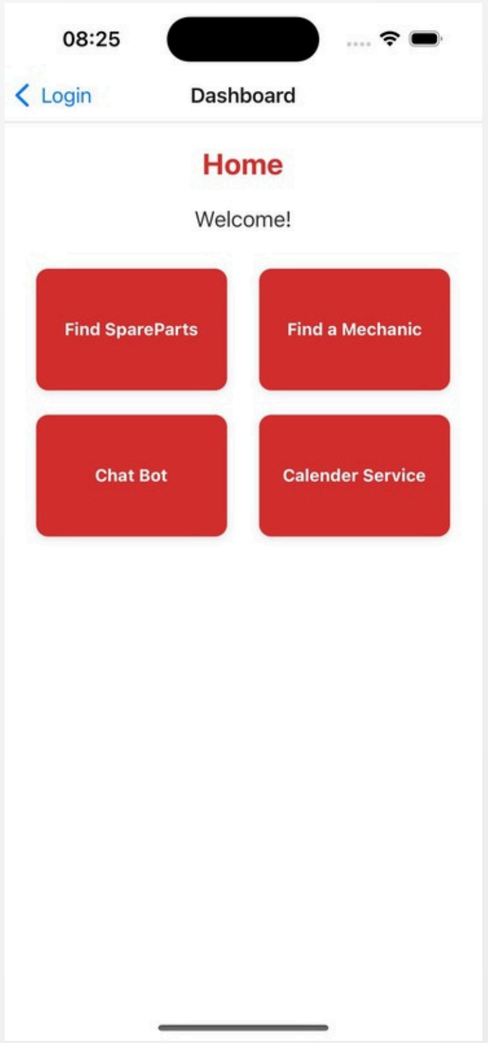
COMPLETION OF COMPONENTS

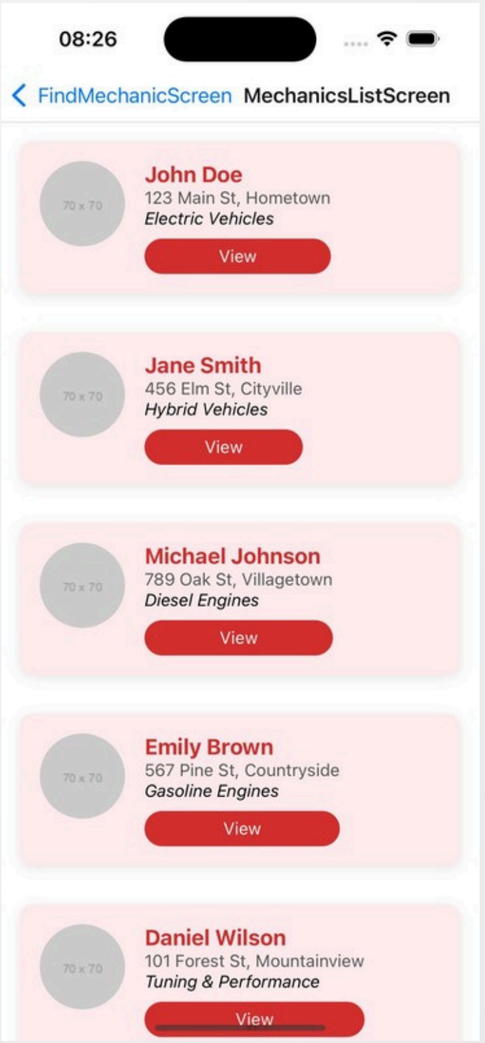
- DATA GATHERING AND PREPROCESSING
- DATASET APPROVAL
- CREATING THE ACTIONS BASED ON BREAKDOWN DESCRIPTION
- PROVIDING THE BREAKDOWN TYPES
- VERIFY THE PROVIDED ANSWERS USING ML MODEL
- IMPLEMENTATION OF USER INTERFACES
- HOSTING
- OTHER OBJECTIVES

DISPLAY RESULTS IN MOBILE APPLICATION

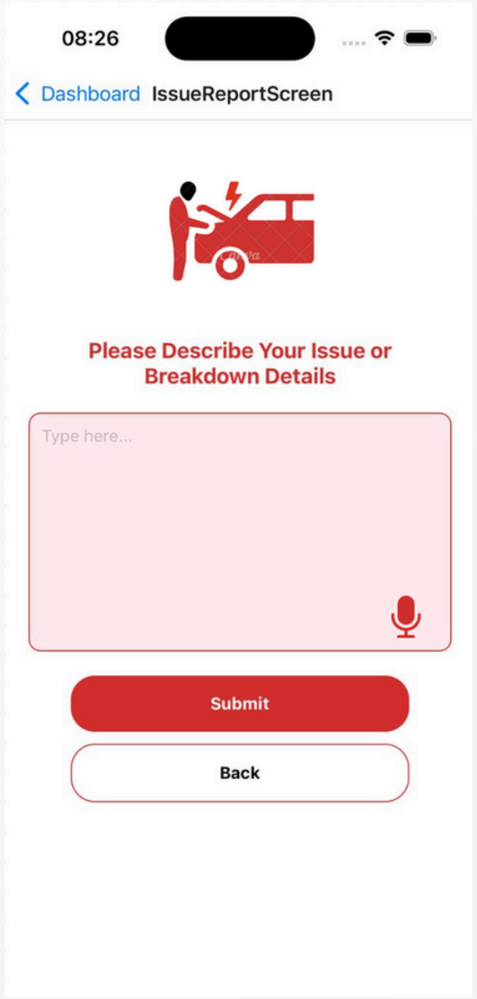


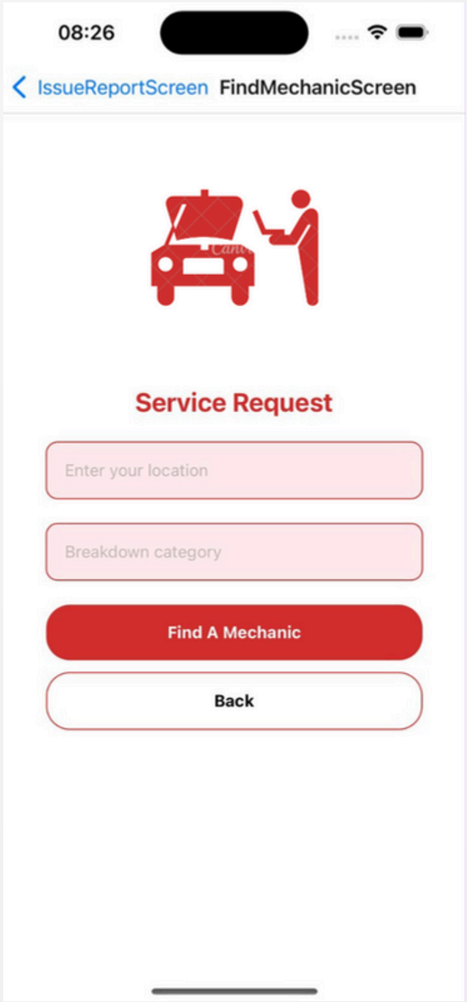






DISPLAY RESULTS IN MOBILE APPLICATION





IT20038182

Madhubhashana A.G.K.

Bachelor of Science (Hons) in Information Technology Specializing in Information Technology

Background

- In today's fast-paced world, vehicle ownership is essential, but breakdowns pose challenges in identifying parts and finding reliable repair shops, especially for those with limited automotive knowledge.
- Traditional methods often lead to inefficiencies.
- To tackle this, we propose an innovative solution using AI, image processing, and real-time data.
- Our image-based system accurately recognizes parts, provides real-time compatibility information, and lists nearby shops with the required parts.
- Privacy measures protect user data, and our user-friendly interface empowers vehicle owners to confidently address breakdowns.

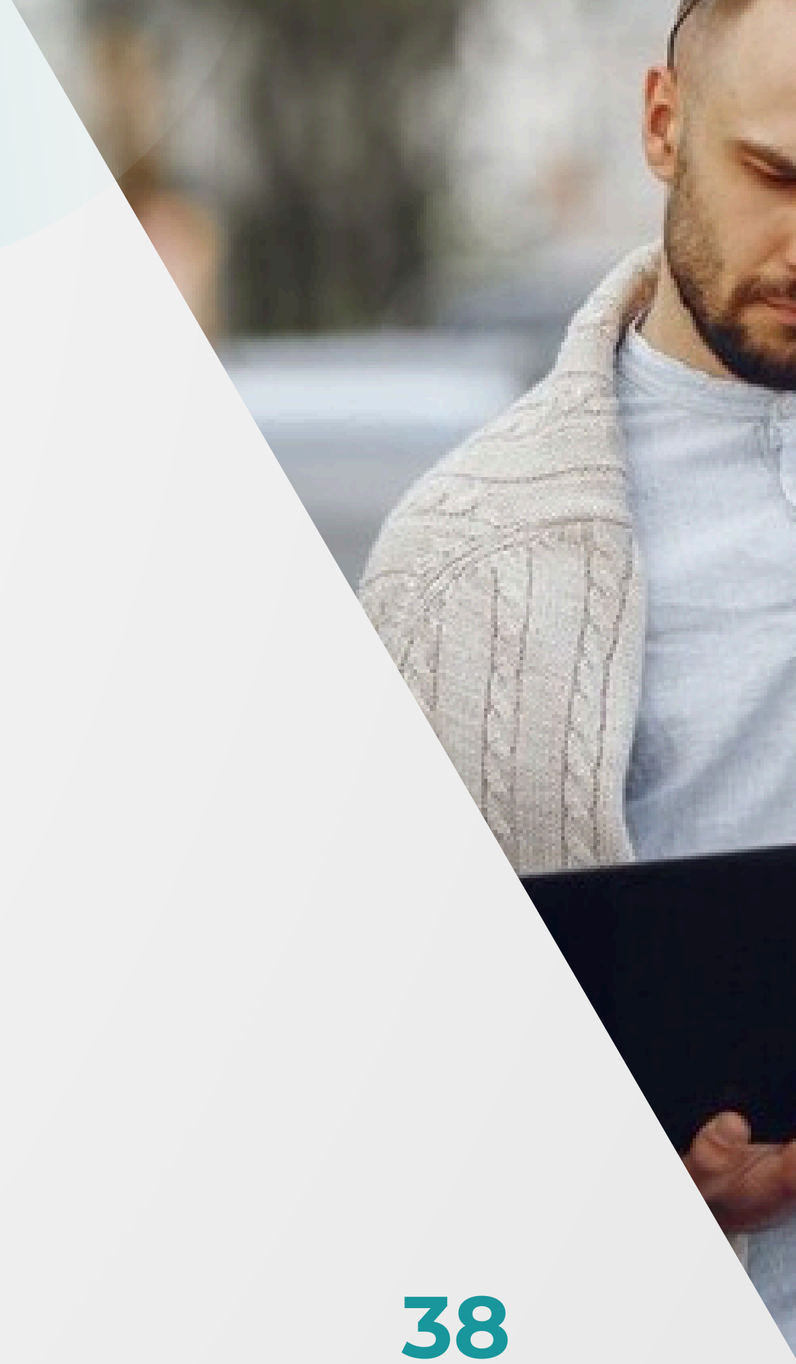
Research Question

“The current issue involves the absence of an efficient and accurate solution for electric vehicle owners to identify spare parts in the event of an unexpected breakdown of an electric vehicle in an unfamiliar location”

There's a need for an AI-based system to identify spare parts and sellers' locations accurately using image processing and machine learning. Continuous learning, transfer learning, and Auto Machine Learning will enhance accessibility and accuracy. Developing such a solution is essential

Research Gap

	Research A	Research B	Research C	Proposed System
Spare Parts Identification	✗	✗	✗	✓
Usage of image processing	✓	✓	✓	✓
Spare parts Identification	✗	✗	✗	✓
Representative datasets	✓	✗	✗	✓
Noise sensitivity	✗	✗	✗	✓



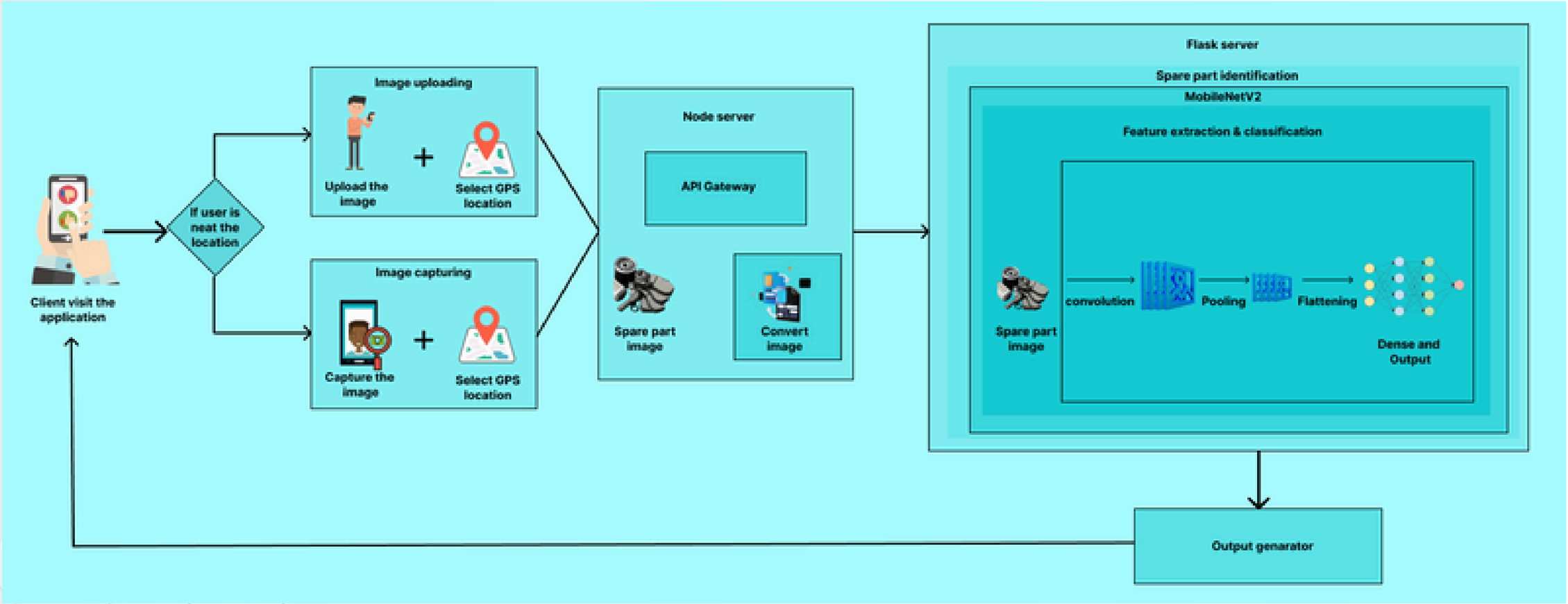
Specific Objective

The research aims to create a real-time image processing system using machine learning to help vehicle owners accurately identify and locate compatible spare parts. It offers convenience through image capture and analysis, enhancing spare parts sourcing

Sub Objectives

- Data Collection
- Data Preprocessing
- ML Model Development
- Implement CNN techniques
- Specialization Matching
- Geolocation Integration
- User Interface
- Testing and Validation
- Scalability and Performance

HIGH LEVEL SYSTEM ARCHITECTURE DIAGRAM



TECHNIQUES & TECHNOLOGIES

METHODOLOGY

Technology	• Python • React Native • Node js • Flask • Tensorflow
Techniques	• Image processing • ReLU (rectified linear unit)
Model	• CNN (Conclusion Neural Networks)

Firebase

METHODOLOGY

- DATA COLLECTION
- DATA PRE-PROCESSING
- USE TRANSFER LEARNING BASED CNN ARCHITECTURES FOR TRAINING THE MODELS
- DATA AUGMENTATION AND SELECTED THE BEST TECHNIQUE TO RETRAIN MODELS TO ACHIEVE GOOD FITTING
- TUNING HYPERPARAMETERS AND RETRAINED MODELS TO SELECT THE BEST ARCHITECTURE
- DATA VISUALISATION
- HOST THE FINAL MODEL
- DISPLAYED THE RESULTS IN MOBILE APPLICATION

DATA COLLECTION AND PRE-PROCESSING

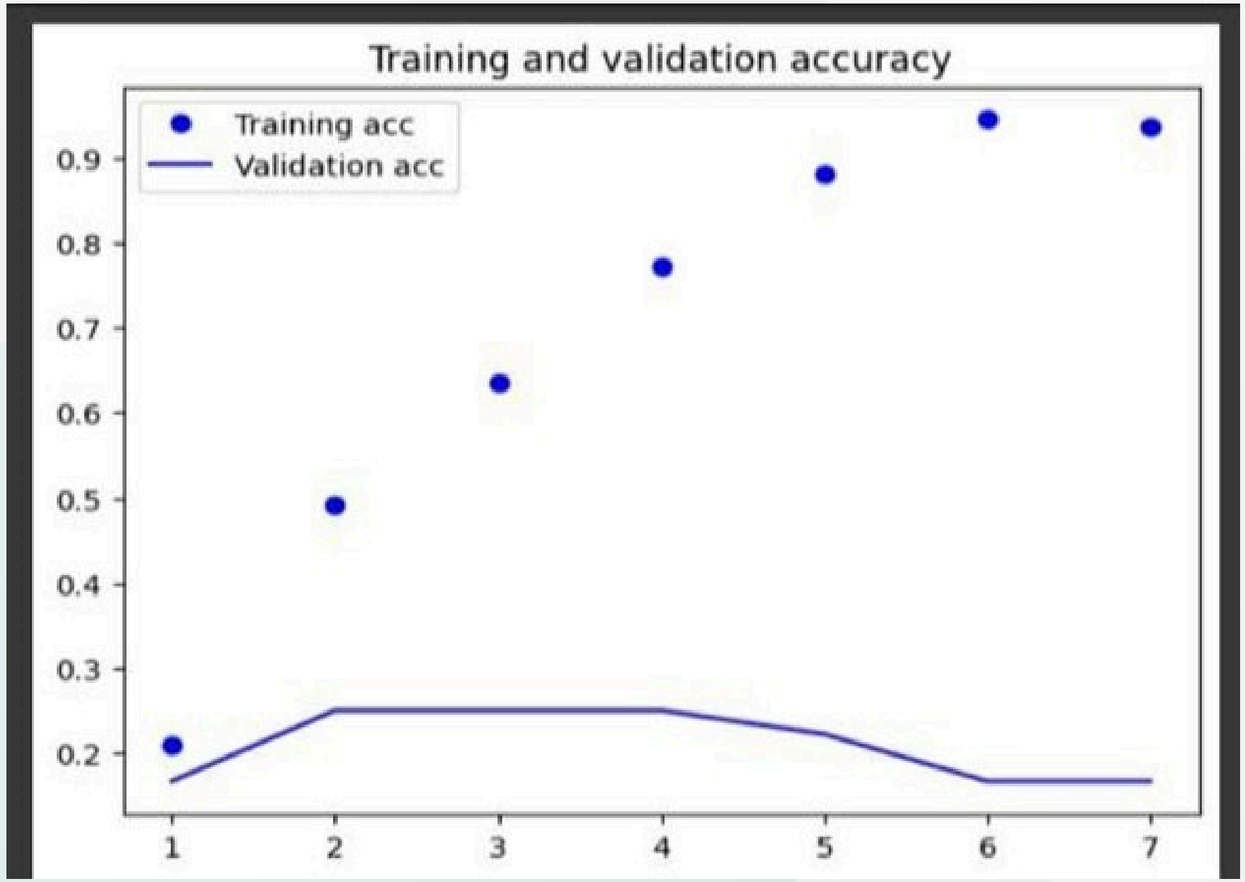
```
Found 194 images belonging to 10 classes.
Found 194 images belonging to 10 classes.
```

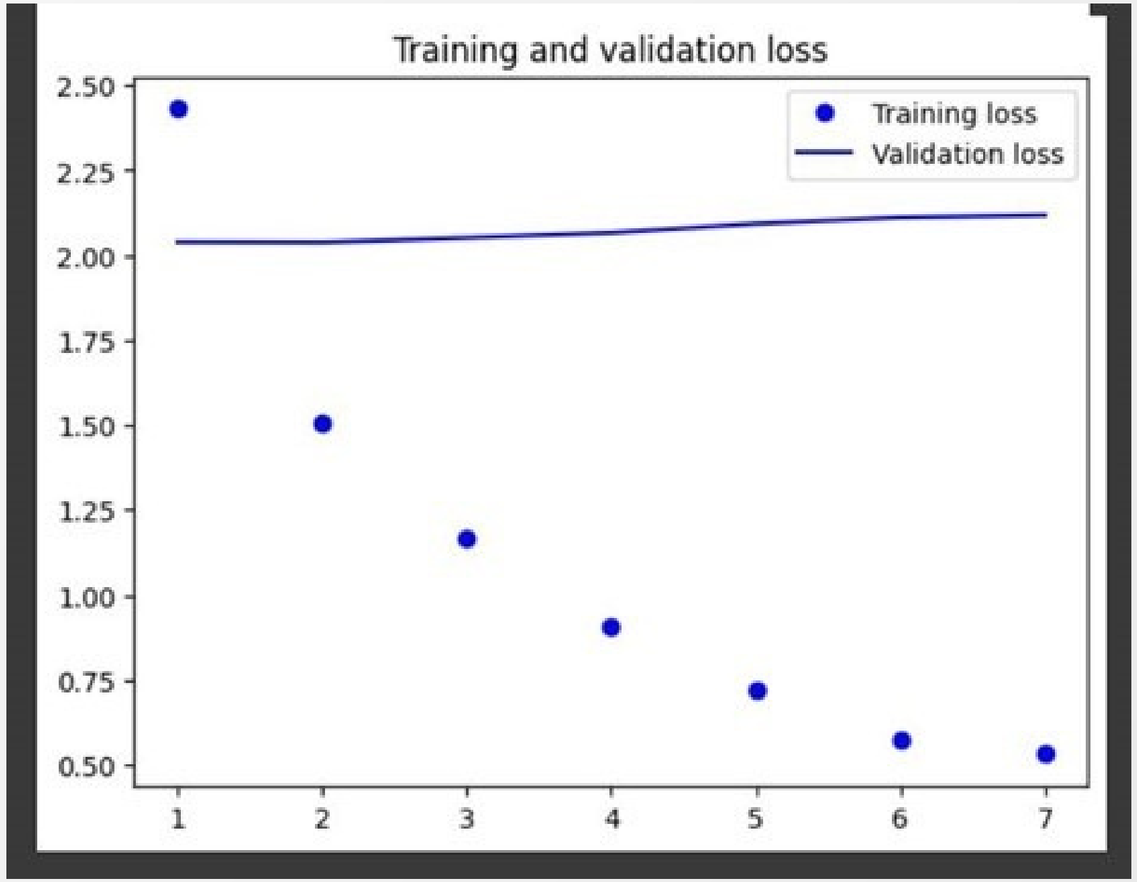
```
Controller
Inventor
BrakingSystem
VehicleInterior
Battery
Transmission
onboardchager
ExteriorComponents
Elecricmotor
CoolingSystem
```

MODEL OUTPUT

```
[ ]  predicted_part_index = np.argmax(predictions[0])
     predicted_part_name = classes[predicted_part_index]

     print("Predicted car part:", predicted_part_name)

     Predicted car part: shocker
```

SPARE PARTS IDENTIFICATION ACCURACIES AND LOSS





SYSTEM REQUIREMNTS

FUNCTIONAL REQUIREMENTS

Image-Based Spare Parts Identification enables real-time camera use for capturing spare parts, accurately identifying their type and condition, and presenting comprehensive details, including manufacturing specifics and pricing.

Spare Parts Shop Registration offers a mechanism for shops to join, collect and store their details, while the Spare Parts Shop Database maintains current spare part data and allows real-time inventory updates through seamless integration.

NON-FUNCTIONAL REQUIREMENTS

- Performance
- Usability
- Reliability
- Security
- Compatibility
- Maintainability
- Scalability
- Accessibility

SYSTEM REQUIREMENTS

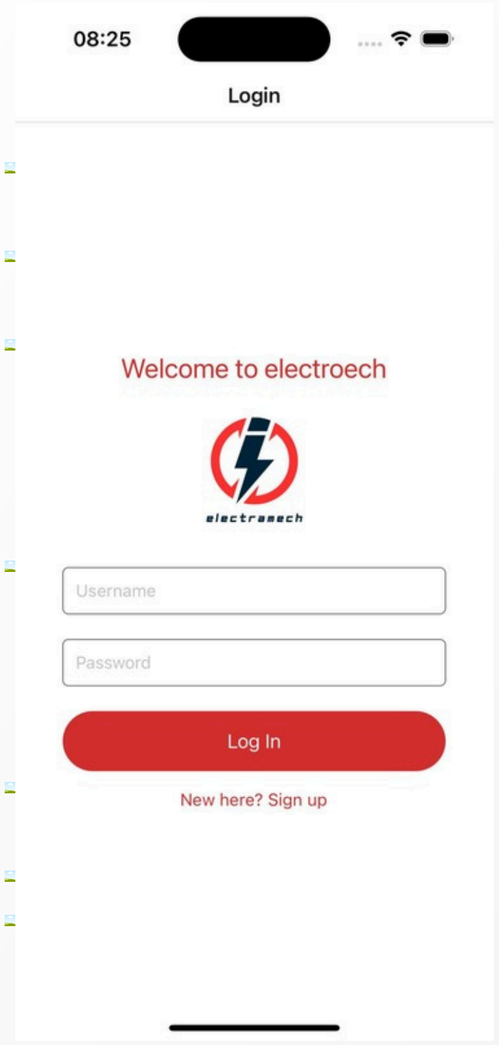
- Image processing capabilities Machine learning algorithms
- Database integration
- Mobile compatibility
- Scalability
- Performance
- Usability
- Security
- Reliability
- Integration with third-party APIs
- System backups and data recovery

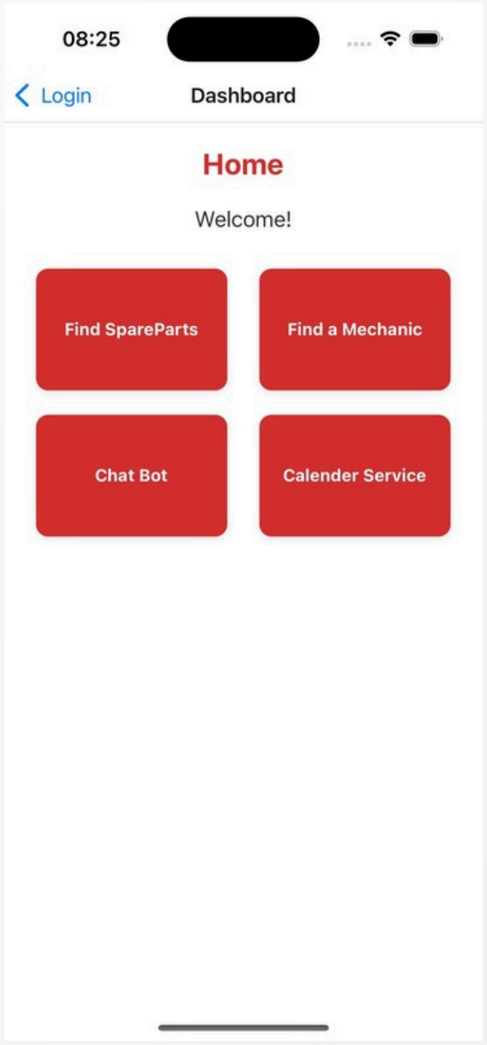
COMPLETION AND

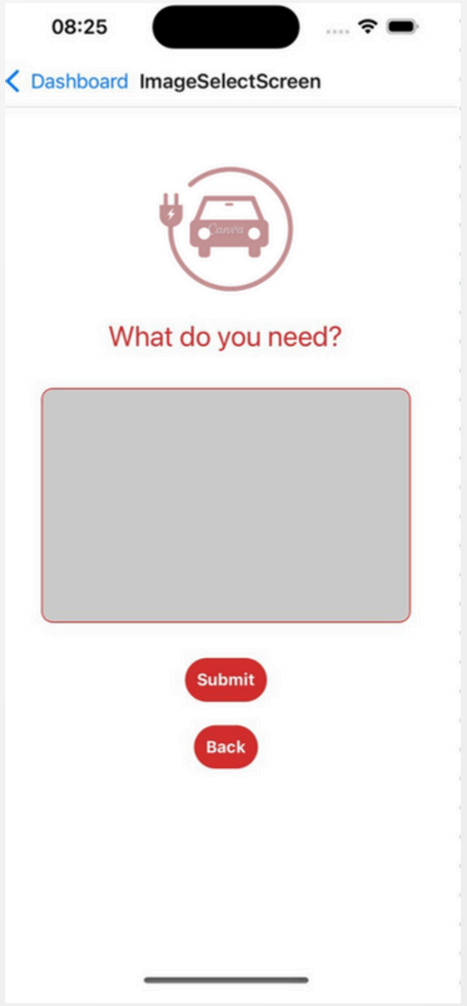
COMPLETION OF COMPONENTS

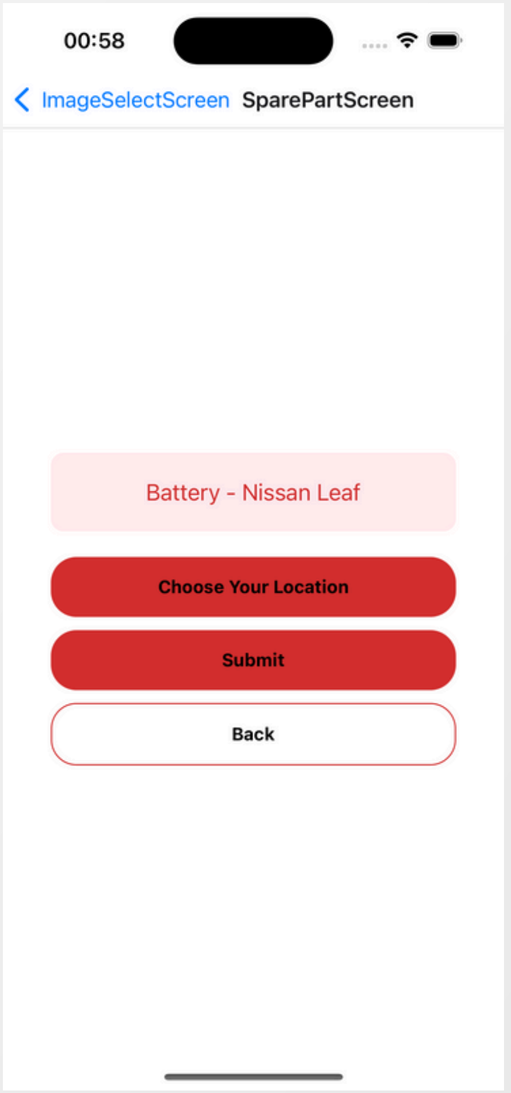
- DATA GATHERING AND PREPROCESSING
- DATASET APPROVAL
- TRAINING THE MACHINE LEARNING MODEL
- INITIAL SPARE PARTS RECOGNITION
- EVALUATING AND FINE TUNING OF THE MODEL
- LOCATION MAPPING
- AUTO MACHINE LEARNING IMPLEMENTATION TO ADD NEW PARTS EASILY
- IMPROVING SYSTEM PERFORMENCE

DISPLAY RESULTS IN MOBILE APPLICATION

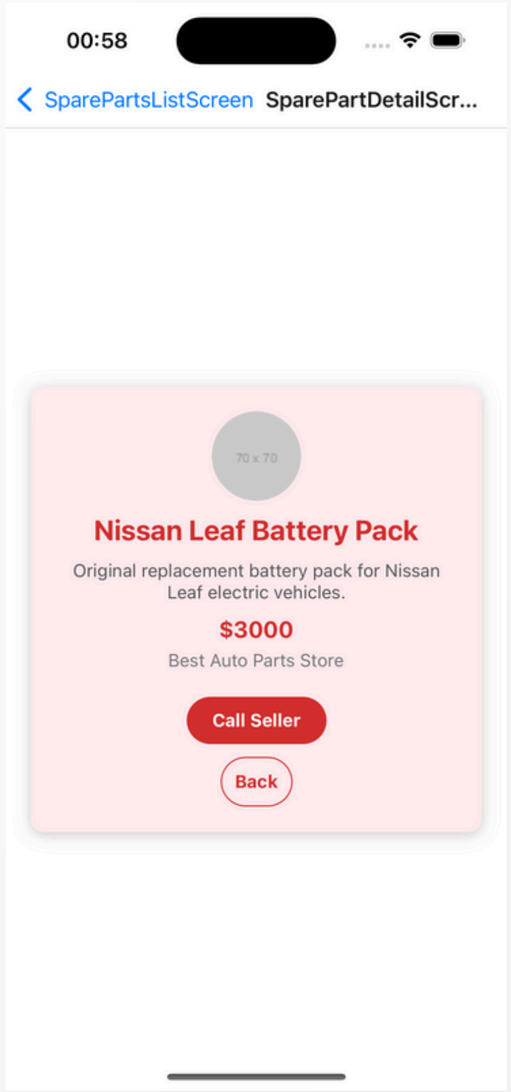


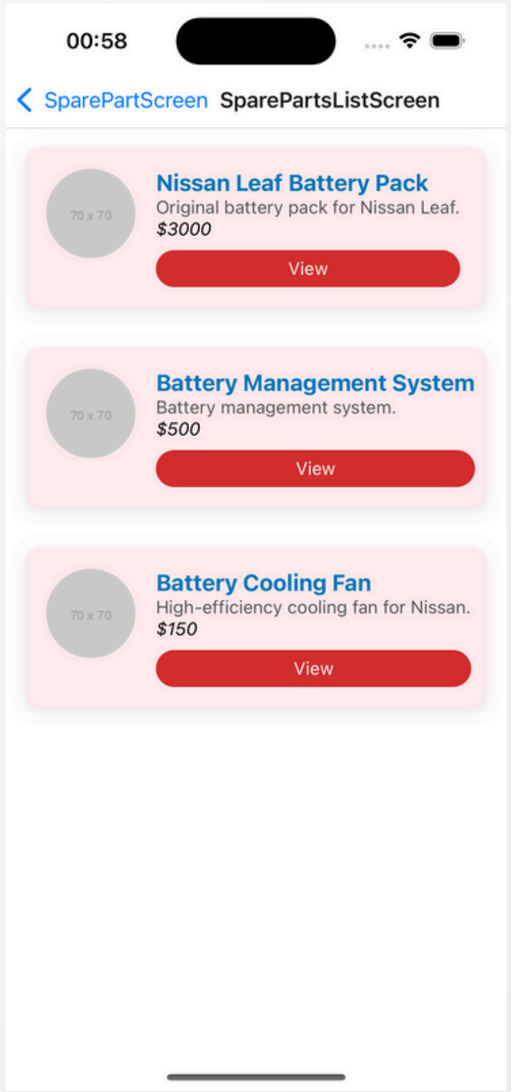






DISPLAY RESULTS IN MOBILE APPLICATION





IT200644826

Vijerathna A.G.V.K.M

Enhancing User Experience in Vehicle Maintenance with a Smart Digital Monitoring Calendar and Reminder System

Bachelor of Science (Hons) in
Information Technology Specializing in Information Technology

INTRODUCTION

Background

- system for electric vehicle maintenance using Machine learning and image OCR.
- **Efficient record management** is important. It organizes vast maintenance data, including service records, parts, and user history, ensuring easy access.
- Streamlining maintenance data, the research integrates Optical Character Recognition **(OCR)** tech.
- Central to this research is the development and deployment of **predictive algorithms**. These algorithms analyze historical service data, usage patterns, and driving conditions to predict ideal spare part replacement times.
- The research proposes an innovative **digital monitoring calendar** for users to input, track, and receive service reminders, promoting proactive maintenance.

Research Question

How can a systematic approach, combining proactive monitoring, streamlined record-keeping, predictive algorithms, and a digital monitoring calendar, revolutionize vehicle maintenance to prevent breakdowns efficiently?

"How to avoid breakdowns of Vehicle through proper maintenance and smart digital monitoring Calendar system ?"

“How to calculate the time of next service after doing a repair to certain part of a vehicle? ”

“How to track average maintenance cost based on vehicle type for a period?”

Research Gap

Features	Research A	Research B	Research C	Proposed System
Smart Digital Monitoring calendar	✗	✗	✗	✓
Predictive algorithms to calculate future repairs	✗	✓	✗	✓
Availability of personalized solutions	✓	✗	✗	✓
Using OCR technology	✗	✗	✗	✓
User Friendly Platform	✓	✗	✓	✓

Specific Objective

Create a system for improved vehicle maintenance with proactive monitoring, OCR-based bill capture, and digital calendars. Empower users with informed decisions for optimal performance and Durability.

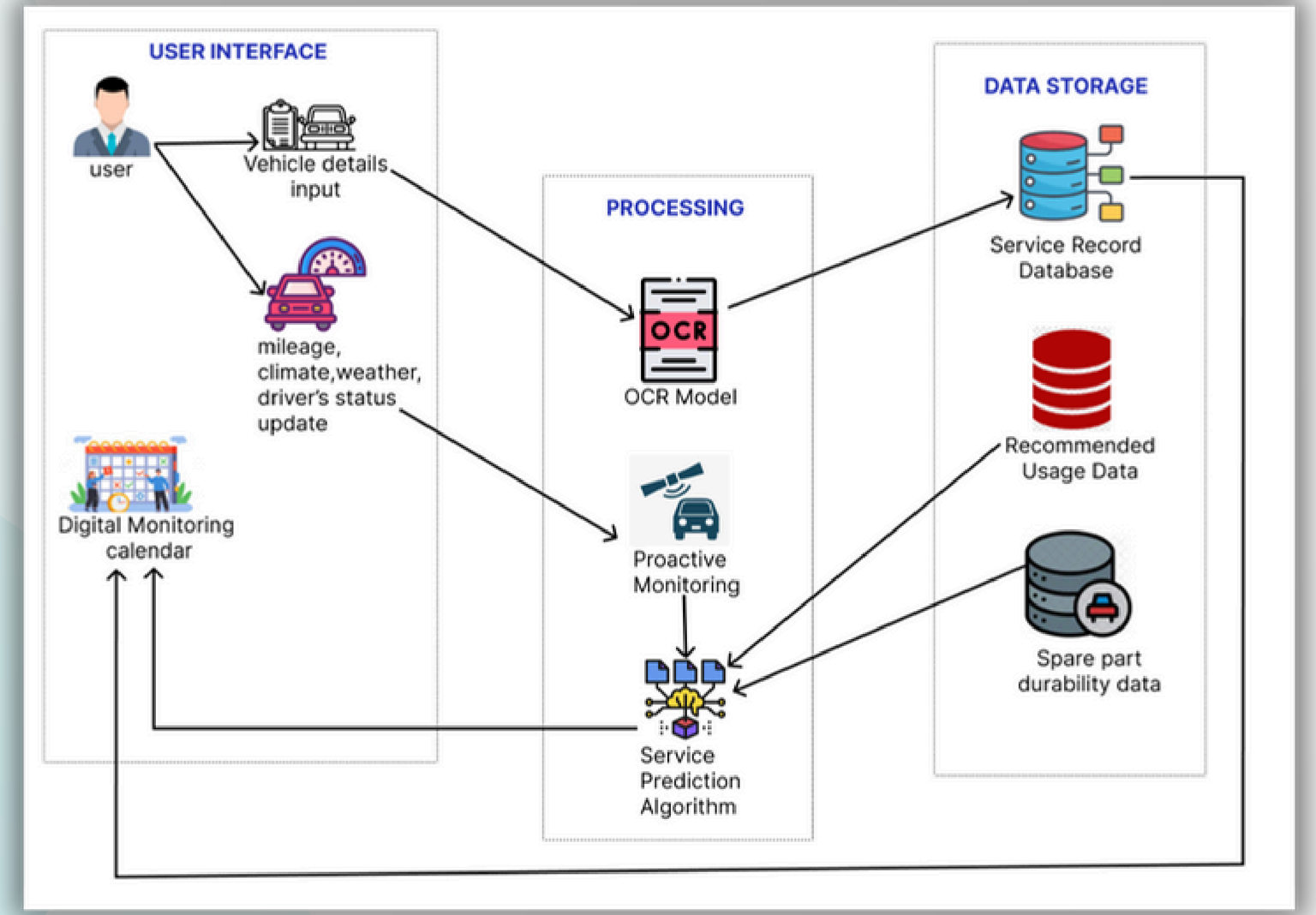
Sub Objectives

- Research and Data Gathering
- Data Preprocessing
- OCR Model Implementation
- Predictive Algorithm Development
- Digital Monitoring Calendar System Design
- Database Creation
- User Interface Development
- Testing and Validation
- Scalability and Performance

METHODOLOGY

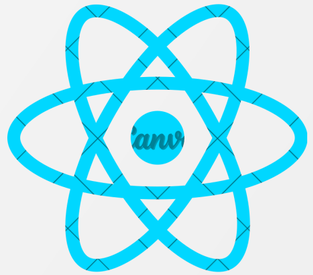
The solution I have suggested is very important for anyone who owns an electric vehicle. With this remedy, the owner of the electric vehicle can do his repairs without missing anything. The system collects the details of the spare parts, information about the vehicle, information about the road the vehicle is driven on, the weather in the area, the nature of the person driving the vehicle, etc. and informs the user about future repairs. System present a user friendly tool through a digital monitoring calendar. By means of an algorithm, system inform the user about possible repairs in the future and stop sudden breakdowns that can happen to the vehicle.

System Architecture Diagram



TECHNIQUES & TECHNOLOGIES

Technology	• React Native • Python • Flask • Node Server • open-cv • MongoDB
Techniques	• Pytesseract and genism • Machine Learning (ML)
Algorithm	• NLP Algorithms • Random Forest • Catboost • XGboost



METHODOLOGY

- Research and Data Gathering
- DATA PRE-PROCESSING
- Implementation of OCR Model
- Algorithm Development
- Digital Monitoring Calendar System Design
- Integration of Proactive Monitoring and Record-Keeping Implementation
- HOST THE FINAL MODEL
- DISPLAYED THE RESULTS IN MOBILE APPLICATION

METHODOLOGY

EVIDENCE OF COMPLETION

```
#df.describe()
df.head(10)
```

	Vehicle Component	Exposure	Age of the Vehicle (months)	Milage (km)	Vehicle Type	Component Condition	Climatic Zone	Road Condition	Lifetime	Unit
0	Parking Brake	internal	77	130781	FCEV	new	Intermediate Zone	Traffic Congestion	40888	km
1	Electronic Parking Brake	internal	33	57573	NEV	new	Upland Rainforest Zone	Urban Roads	39552	km
2	Speakers	internal	70	137433	HEV	new	Dry Zone	Urban Roads	41329	km
3	Rearview Camera	internal	195	342253	HEV	new	Central Highlands	Coastal Roads	37065	km
4	Interior lights	internal	125	243207	EV	new	Dry Zone	Highways and Expressways	38155	km
5	Center Console	internal	95	175670	EV	new	Intermediate Zone	Road Hazards	47	months
6	Charging Cable and Accessories	external	120	212875	NEV	new	Intermediate Zone	Mountain Roads	33	months
7	Wheel speed sensors	internal	168	309780	Electric Bicycle	new	Dry Zone	Highways and Expressways	37333	km
8	Center Console	internal	152	248620	FCEV	new	Dry Zone	Urban Roads	46	months
9	CV Joints	internal	33	63359	BEV	new	Central Highlands	Coastal Roads	80	months

Displaying the Dataframe

Catboost

```
# Create an XGBoost regression model
catmodel = CatBoostRegressor(random_state=42)

# Train the model using the training data
catmodel.fit(X_train, y_train)

# Predict the target for the testing set
y_pred = catmodel.predict(X_test)

# Evaluate the model's performance using mean squared error (MSE)
mse = mean_squared_error(y_test, y_pred)
print(f"Root Mean Squared Error (MSE): {mse **0.5}")

# Optionally, you can also calculate other evaluation metrics such as R2 score, MAE, etc.
```

```
131:    learn: 7.2099090        total: 2.5s     remaining: 16.5s
132:    learn: 7.1581109        total: 2.53s    remaining: 16.5s
133:    learn: 7.1061216        total: 2.55s    remaining: 16.5s
134:    learn: 7.0551934        total: 2.57s    remaining: 16.5s
```

Catboost

METHODOLOGY

EVIDENCE OF COMPLETION

```
[ ] print(f"Root Mean Squared Error (MSE): {mse **0.5}")

    Root Mean Squared Error (MSE): 2.6866648598170 33
```

Rootmean squared error of catboost

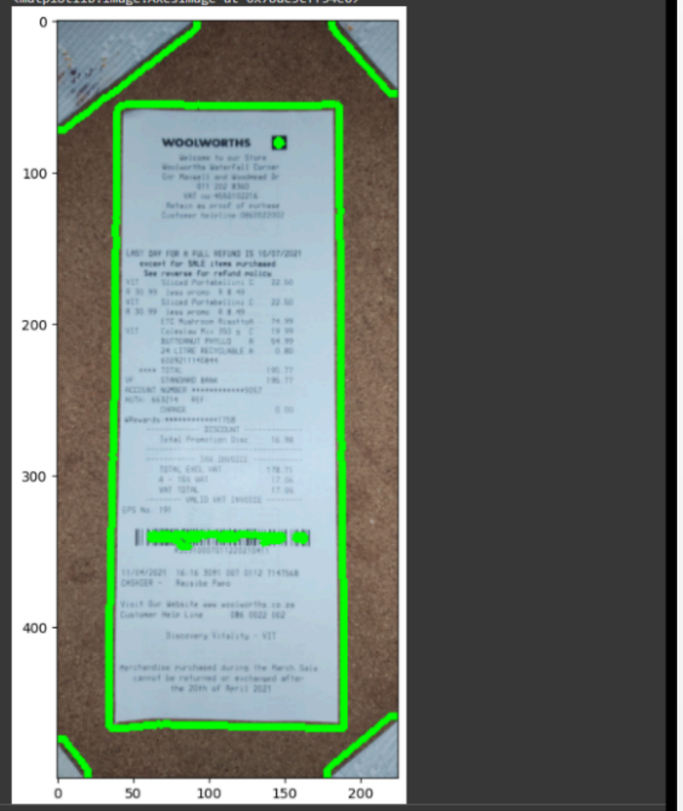
```
with open('/content/forest_model.pkl', 'rb') as file:
    model = pickle.load(file)

# Create a DataFrame for the single row you want to predict
#call input mapper to use encoded data
component="Tires"
exposure = "internal"
age=137
milage=258914
vehi_type="BEV"
condition="new"
climatic_zone= "Central Highlands"
road_type= "Mountain Roads"

single_row_data = input_mapper(component, exposure, age, milage, vehi_type, condition, climatic_zone, road_type)

# Create a DataFrame from the single row data
single_row_df = pd.DataFrame([single_row_data])

# If there are any categorical columns, convert them to integer codes using the model's training method (e.g., LabelEncoder)
# You might have to encode the columns similar to how you did during training

# Make a prediction for the single row
prediction = model.predict(single_row_df)

# Print the prediction
print(f"Prediction for the single row: {prediction}")

Prediction for the single row: [16.57]
```

Output of prediction algorithm

METHODOLOGY

EVIDENCE OF COMPLETION

```
[ ] contours, hierarchy = cv2.findContours(edged, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    image_with_contours = cv2.drawContours(image.copy(), contours, -1, (0,255,0), 3)
    plot_rgb(image_with_contours)
```



Identify the edges of coloured Bill

```
scanned = wrap_perspective(original.copy(), contour_to_rect(receipt_contour))
plt.figure(figsize=(16,10))
plt.imshow(scanned)
```

Bill image after removing the background areas of the bill

METHODOLOGY

EVIDENCE OF COMPLETION

```python
import nltk
from gensim.models import FastText
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import string

tokenized_data = []
for sentence in items:
    words = word_tokenize(sentence.lower())  # Tokenize and convert to lowercase
    filtered_words = [word for word in words]
    tokenized_data.append(filtered_words)

# Load or train FastText embeddings with adjusted parameters
model_2 = FastText(sentences=tokenized_data, min_count=1, vector_size=100)
```

import Fast Text

```python
# prompt: iterate response through def correct_car_word(noise)

corrected_responses = []
for word in response:

  corrected_word = correct_car_word(word)
  if corrected_word:
    corrected_responses.append(corrected_word)


print("Corrected Responses:")
print(corrected_responses)
```

```
0.34265846 Windshield Wiper Motor
0.20801729 Headlights
0.3682567 Heating System
0.34731594 Steering
0.35406667 Spare Tire or Tire Repair Kit
0.29560694 ABS sensors
0.31997555 Rotors
0.2670804 Electric Motor
1.0000002 Front Buffer
0.8305947 L/H Front Buffer Retainer
0.7803198 L/H Head Lamp
0.8764824 L/H Head Lamp Gloss Strip
0.9137659 L/H Head Lamp Gloss Strip Mounting Brackets
1.0000001 Shell
1.0000001 Bonnet
```

Output of ocr algorithm

SYSTEM REQUIREMNTS

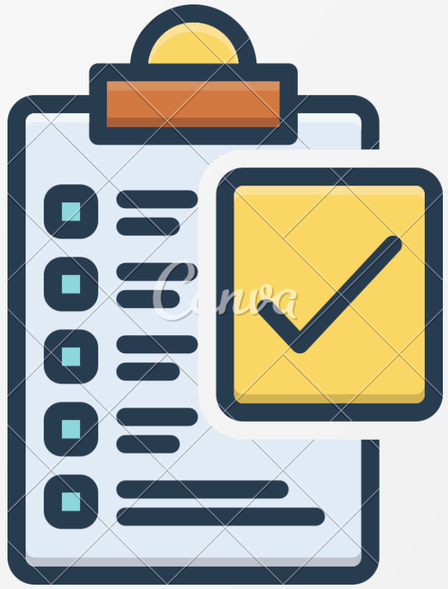
FUNCTIONAL REQUIREMENTS

- User Registration and Vehicle Details Input.
- Users should be able to periodically update their vehicle's mileage.
- Users should be able to upload images of service bills and invoices.
- The system should predict the optimal replacement time for each spare part based on vehicle mileage and historical service records.
- Users should be able to input and track maintenance activities, spare part replacements, and relevant data through a digital monitoring calendar.
- Alert system should notify users about upcoming service requirements.
- System should be able to accurately extract of relevant maintenance information from bill photos.

SYSTEM REQUIREMNTS

NON-FUNCTIONAL REQUIREMENTS

- Performance
- Security
- Reliability
- Scalability
- Compatibility
- Maintainability
- Usability



SYSTEM REQUIREMENTS

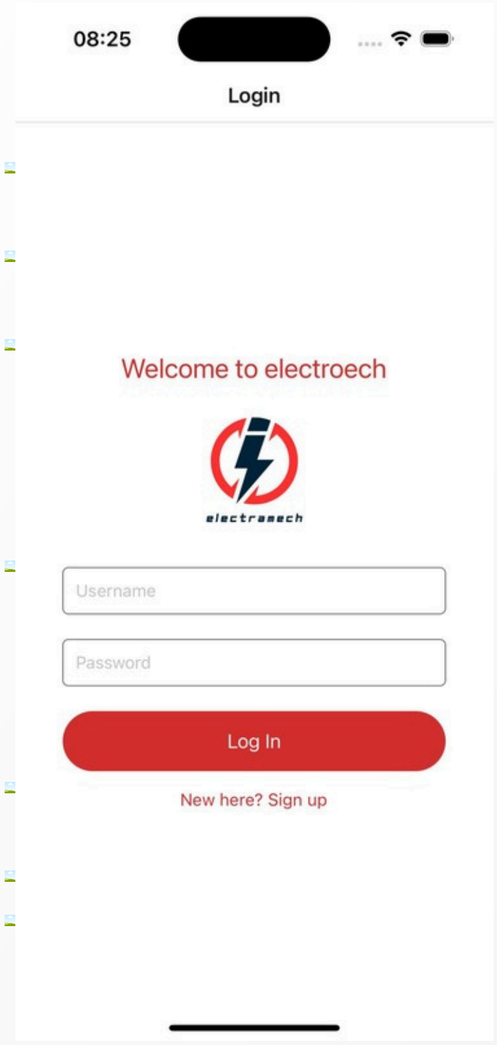
- OCR-Based Bill Capture
- Machine learning algorithms
- Database integration
- Mobile compatibility
- Performance
- Integration with predictive algorithms for generating service reminders. Reliability
- System backups and data recovery
- Data exchange protocols and APIs for component integration.

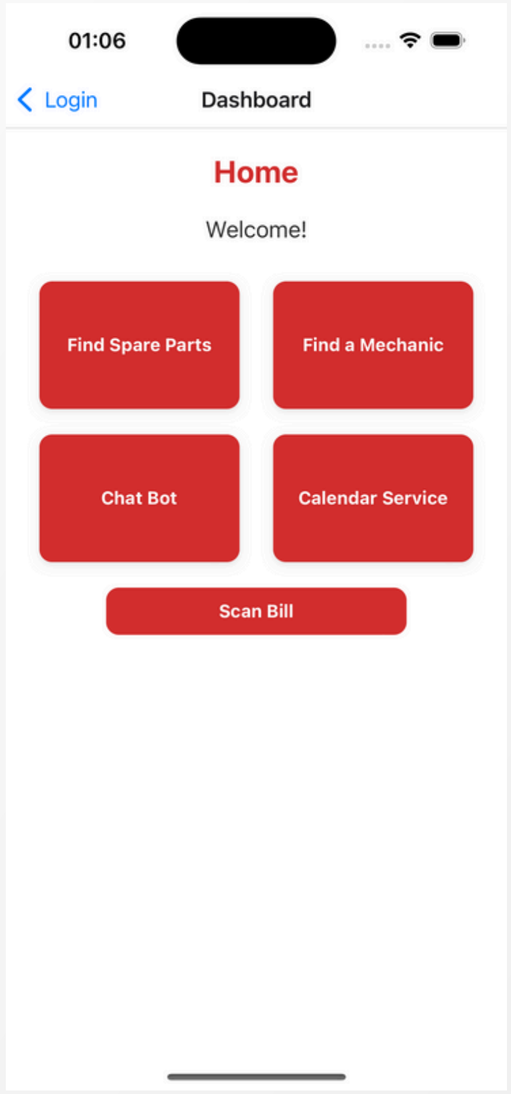
COMPLETION AND FUTURE WORKS

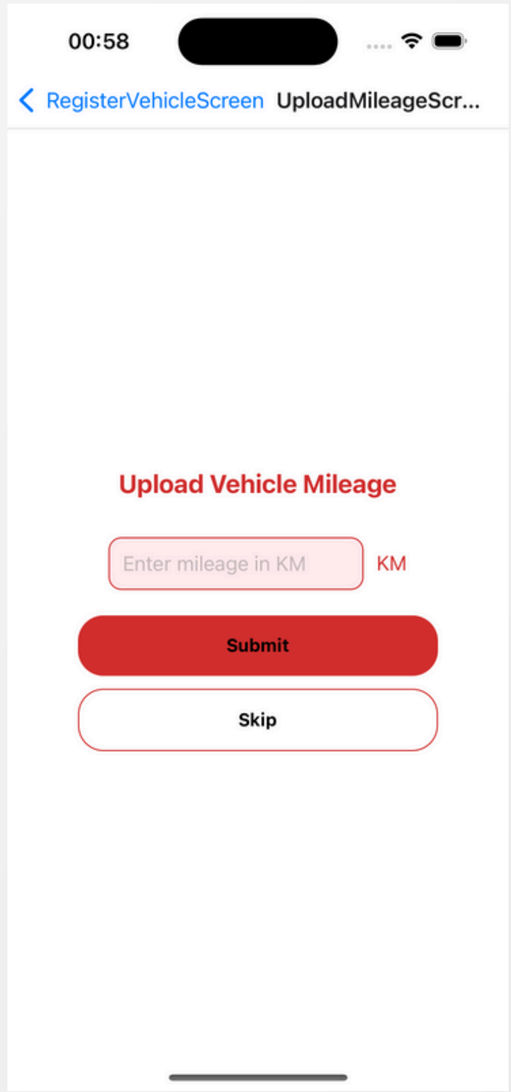
COMPLETION OF COMPONENTS

- DATA GATHERING AND PREPROCESSING
- TRAINING THE MACHINE LEARNING MODEL
- OPTICAL CHARACTOR RECOGNITION
- PREDICTIVE ALGORITHM
- IMPLEMENTATION OF USER INTERFACES
- DIGITAL MONITORING CALENDAR
- HOSTING
- IMPROVING SYSTEM PERFORMENCE

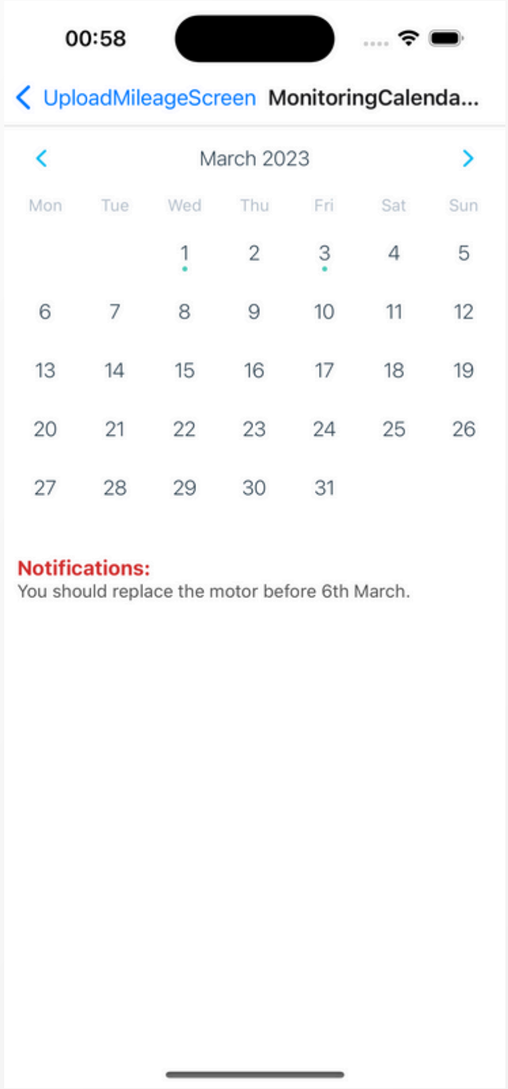
DISPLAY RESULTS IN MOBILE APPLICATION

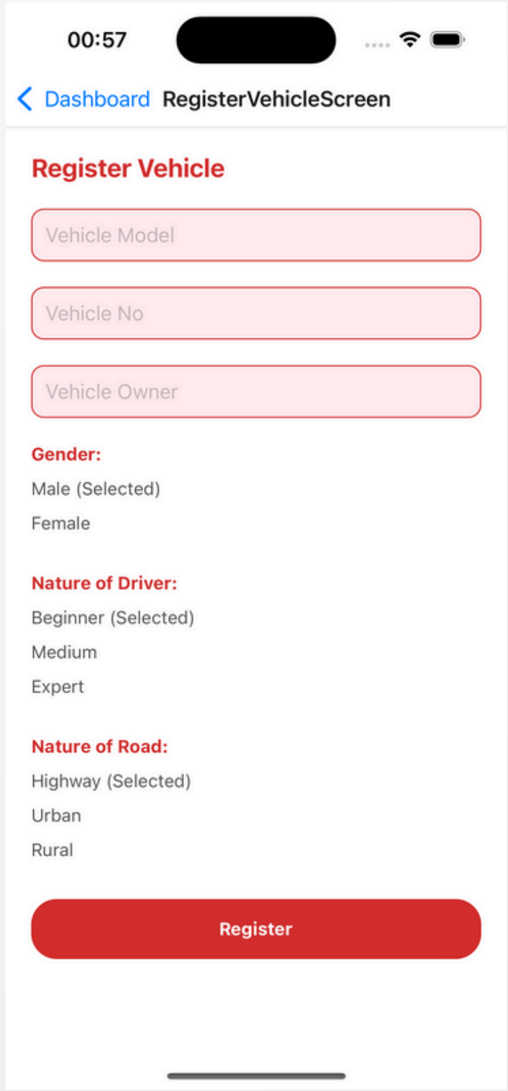






DISPLAY RESULTS IN MOBILE APPLICATION





THANK YOU