

SMART TAXI METER SYSTEM

Electrical and Electronic Engineering

HNDE – Labuduwa

Presented by: Group 05



GROUP MEMBERS

G.W.I. DEVINDI

G.T.N. SAMARAJEEWA

W.T.M.N.U. WIJERATHNA

L.N. WEERASINGHE

E.M.I.S. EGALLA

OVERVIEW

01

Intro

02

Scope

03

Aim

04

Problem
Statement

05

Objectives

06

Similar
Systems

07

Components

08

Arduino
Codes

09

Methodology

10

Time Plan

11

Cost Analyze

12

Future
Development

13

Resources &
References

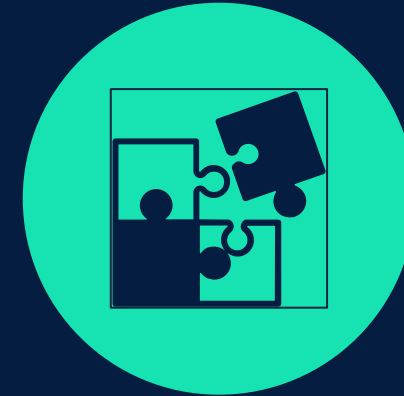
14

Q&A

01 INTRO



- Traditional taxi fare systems struggle with real
- Time fuel price changes and accurate distance measurement.
- Our solution uses a rotary encoder to measure distance and calculate fares dynamically.
- Data from up to 10 meters is collected in a central database and updated in real-time according to fuel price fluctuations.
- New meters automatically connect to the server to synchronize and update fare rates.



KEY BENEFITS

- Real-time fare updates
- Accurate distance
- Based fare calculation
- Reliable and fair system for drivers and passengers
- Efficient, scalable, and modernized solution

02

SCOPE



Develop a smart taxi meter system that provides accurate, transparent, and real-time fare calculations reflecting actual distance traveled and fluctuating fuel prices.

03

AIM



To develop a smart taxi meter system that accurately calculates fares in real-time by measuring the distance traveled and adjusting prices according to current fuel costs.

04

PROBLEM STATEMENT

01

Fare calculations are inaccurate as they don't reflect real-time fuel price changes.

02

Fare rate updates are manual, time-consuming, and error-prone.

03

Distance measurement is often inaccurate, causing fare errors.

04

Managing multiple taxi meters independently leads to inconsistent pricing and inefficient fleet management.

05

Updating new meters with current pricing is complicated without a centralized system

06

Lack of transparency leaves passengers unsure if fares are fair and accurate.



05 OBJECTIVES

1. To design a smart meter using a rotary encoder for accurate distance measurement.
2. To implement real-time fare calculation based on distance and current fuel prices.
3. To develop a central database for managing data from multiple taxi meters.
4. To enable automatic synchronization of fare rates when new meters are added.
5. To ensure transparency and fairness in fare charging for both drivers and passengers.
6. To build a scalable system that can support multiple taxis efficiently.



06

SIMILAR SYSTEMS



PickMe / Uber

These ride-hailing apps use GPS-based fare calculation, which adjusts with distance and time, and includes fuel cost indirectly.



Modern Taxi Meters

Some three-wheelers and taxis use electronic meters, but they mostly operate on static rates without real-time fuel price updates.



Fleet Management Systems

Used by logistics and transport companies to monitor distance, fuel usage, and performance — but not directly for dynamic fare calculation.

07 COMPONENTS

Rotary
Encoder

Microcontroller

Power Supply
&
Wi-Fi Module

Humidity
Sensor

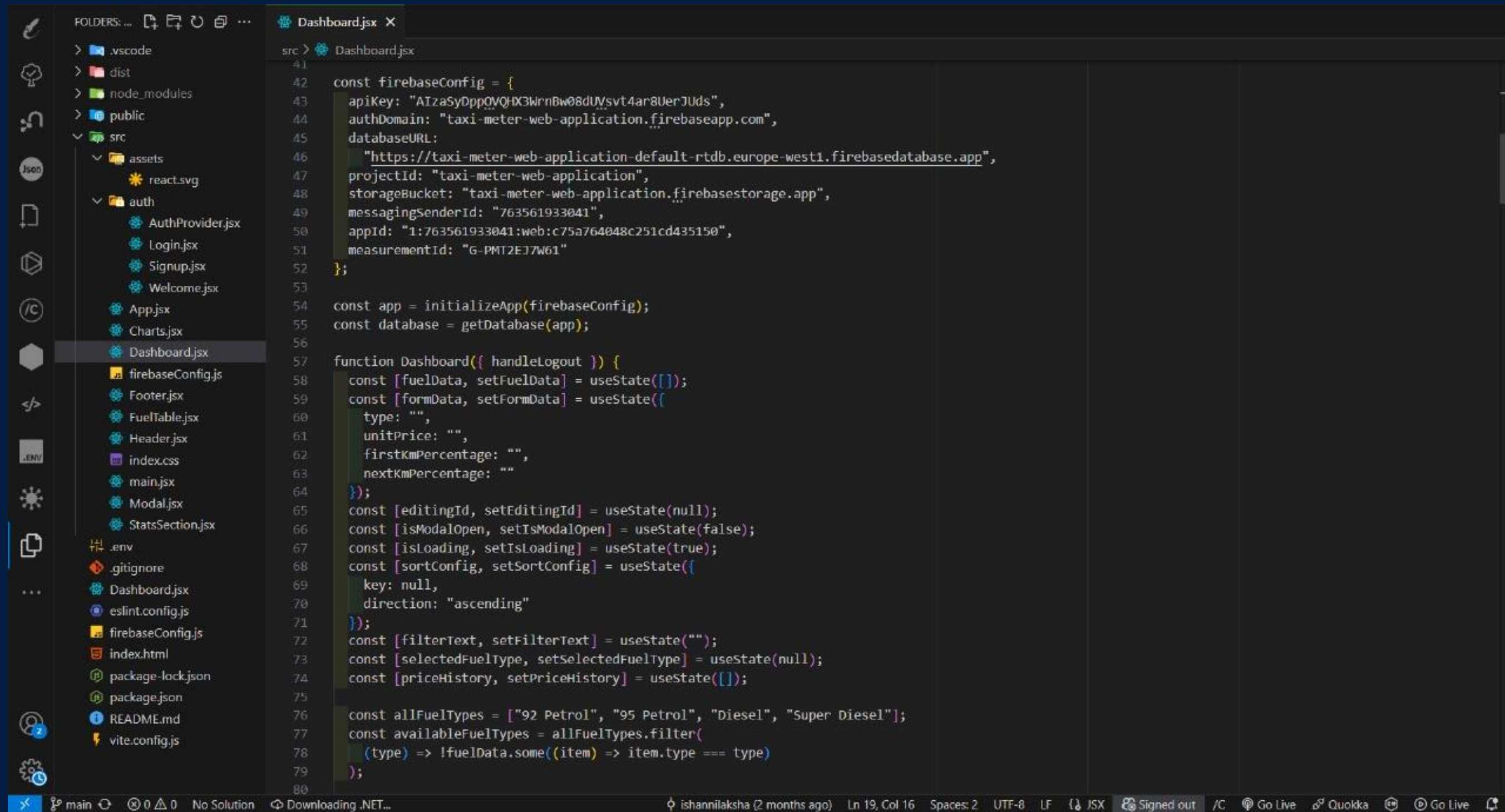
Real Time

Ultrasonic
Sensor

Motor
Driver

LCD
Display

08 ARDUINO CODES



```
41
42 const firebaseConfig = {
43   apiKey: "AIzaSyDppQVQH3WmBw08dUySvt4ar8UerJUds",
44   authDomain: "taxi-meter-web-application.firebaseio.com",
45   databaseURL:
46     "https://taxi-meter-web-application-default-rtdb.europe-west1.firebaseio.com",
47   projectId: "taxi-meter-web-application",
48   storageBucket: "taxi-meter-web-application.firebaseio.com",
49   messagingSenderId: "763561933041",
50   appId: "1:763561933041:web:c75a764048c251cd435150",
51   measurementId: "G-PMT2EJ7W61"
52 };
53
54 const app = initializeApp(firebaseConfig);
55 const database = getDatabase(app);
56
57 function Dashboard({ handleLogout }) {
58   const [fuelData, setFuelData] = useState([]);
59   const [formData, setFormData] = useState({
60     type: "",
61     unitPrice: "",
62     firstKmPercentage: "",
63     nextKmPercentage: ""
64   });
65   const [editingId, setEditingId] = useState(null);
66   const [isModalOpen, setIsModalOpen] = useState(false);
67   const [isLoading, setIsLoading] = useState(true);
68   const [sortConfig, setSortConfig] = useState({
69     key: null,
70     direction: "ascending"
71   });
72   const [filterText, setFilterText] = useState("");
73   const [selectedFuelType, setSelectedFuelType] = useState(null);
74   const [priceHistory, setPriceHistory] = useState([]);
75
76   const allFuelTypes = ["92 Petrol", "95 Petrol", "Diesel", "Super Diesel"];
77   const availableFuelTypes = allFuelTypes.filter(
78     (type) => !fuelData.some((item) => item.type === type)
79   );
80
```

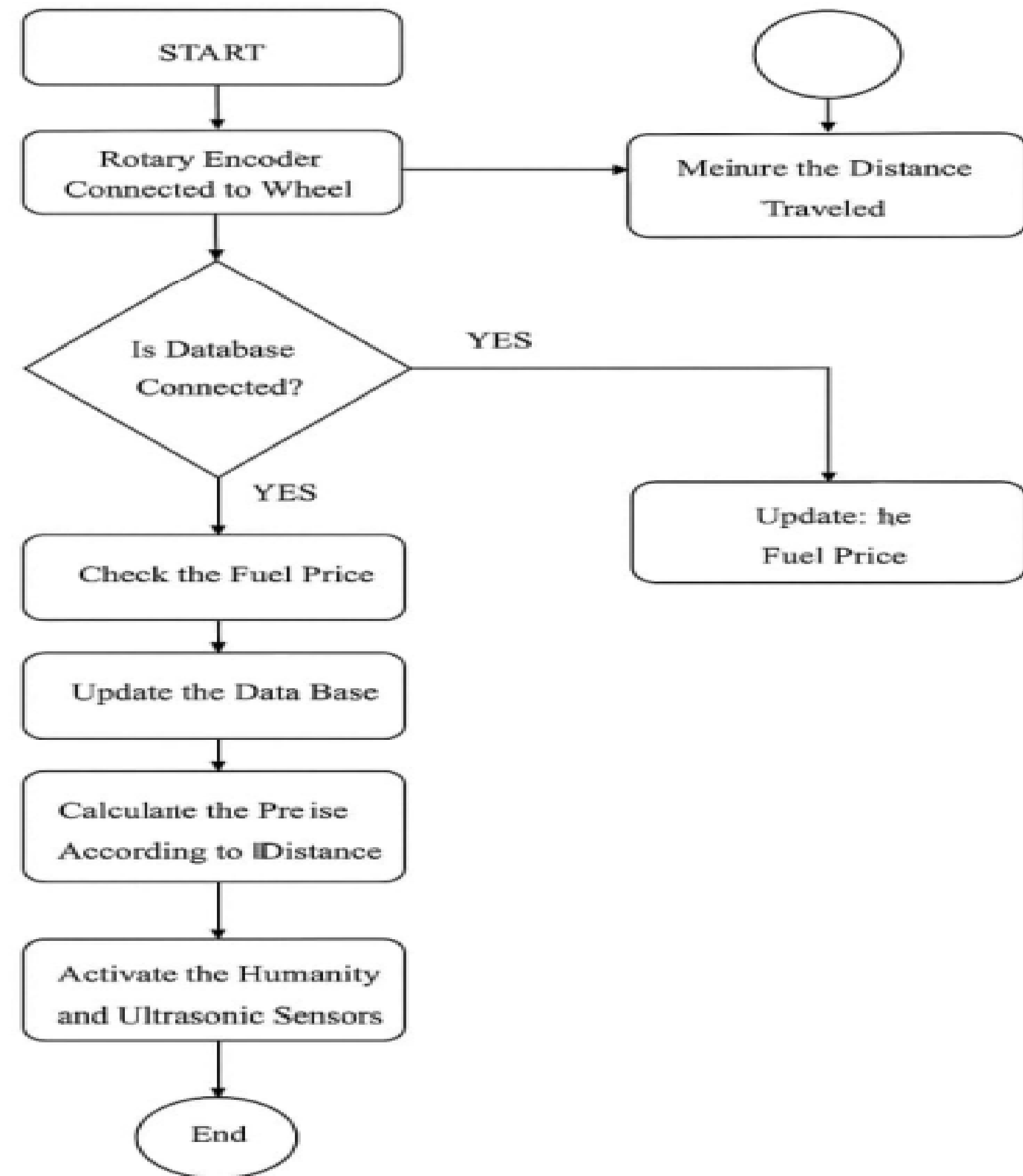
08 ARDUINO CODES

```
Dashboard.jsx X
src > Dashboard.jsx
41
42 const firebaseConfig = {
43   apiKey: "AIzaSyDppQVQhX3WnBw08dUysvt4ar8UerJUds",
44   authDomain: "taxi-meter-web-application.firebaseio.com",
45   databaseURL:
46     "https://taxi-meter-web-application-default-rtdb.europe-west1.firebaseio.com",
47   projectId: "taxi-meter-web-application",
48   storageBucket: "taxi-meter-web-application.firebaseio.com",
49   messagingSenderId: "763561933041",
50   appId: "1:763561933041:web:c75a764048c251cd435150",
51   measurementId: "G-PMT2EJ7W61"
52 };
53
54 const app = initializeApp(firebaseConfig);
55 const database = getDatabase(app);
56
57 function Dashboard({ handleLogout }) {
58   const [fuelData, setFuelData] = useState([]);
59   const [formData, setFormData] = useState({
60     type: "",
61     unitPrice: "",
62     firstKmPercentage: "",
63     nextKmPercentage: ""
64   });
65   const [editingId, setEditingId] = useState(null);
66   const [isModalOpen, setIsModalOpen] = useState(false);
67   const [isLoading, setIsLoading] = useState(true);
68   const [sortConfig, setSortConfig] = useState({
69     key: null,
70     direction: "ascending"
71   });
72   const [filterText, setFilterText] = useState("");
73   const [selectedFuelType, setSelectedFuelType] = useState(null);
74   const [priceHistory, setPriceHistory] = useState([]);
75
76   const allFuelTypes = ["92 Petrol", "95 Petrol", "Diesel", "Super Diesel"];
77   const availableFuelTypes = allFuelTypes.filter(
78     (type) => !fuelData.some((item) => item.type === type)
79   );
80 }
```


08 ARDUINO CODES

```
firebaseConfig.js X
src > firebaseConfig.js > ...

1 // src/firebaseConfig.js
2 import { getAnalytics } from "firebase/analytics";
3 import { initializeApp } from "firebase/app";
4 import { getAuth, GoogleAuthProvider } from "firebase/auth";
5 import { getDatabase } from "firebase/database";
6
7 const firebaseConfig = {
8   apiKey: "AIzaSyDppOVQHx3WrrnBw08dUvsvt4ar8UerJUds",
9   authDomain: "taxi-meter-web-application.firebaseio.com",
10  databaseURL: "https://taxi-meter-web-application-default-rtdb.europe-west1.firebaseio.com",
11  projectId: "taxi-meter-web-application",
12  storageBucket: "taxi-meter-web-application.firebaseio.com",
13  messagingSenderId: "763561933041",
14  appId: "1:763561933041:web:c75a764048c251cd435150",
15  measurementId: "G-PMT2EJ7W61"
16 };
17
18 // Initialize Firebase
19 const app = initializeApp(firebaseConfig);
20 const auth = getAuth(app);
21 const database = getDatabase(app);
22 const googleProvider = new GoogleAuthProvider();
23 const analytics = getAnalytics(app);
24
25 export { analytics, app, auth, database, googleProvider };
26
27
```

Start

The system initializes when the taxi begins operation.

Rotary Encoder

A rotary encoder connected to the taxi's wheel measures the actual distance traveled during the ride

Oil Price Data

The system fetches current fuel price data from a central database or external source.

09

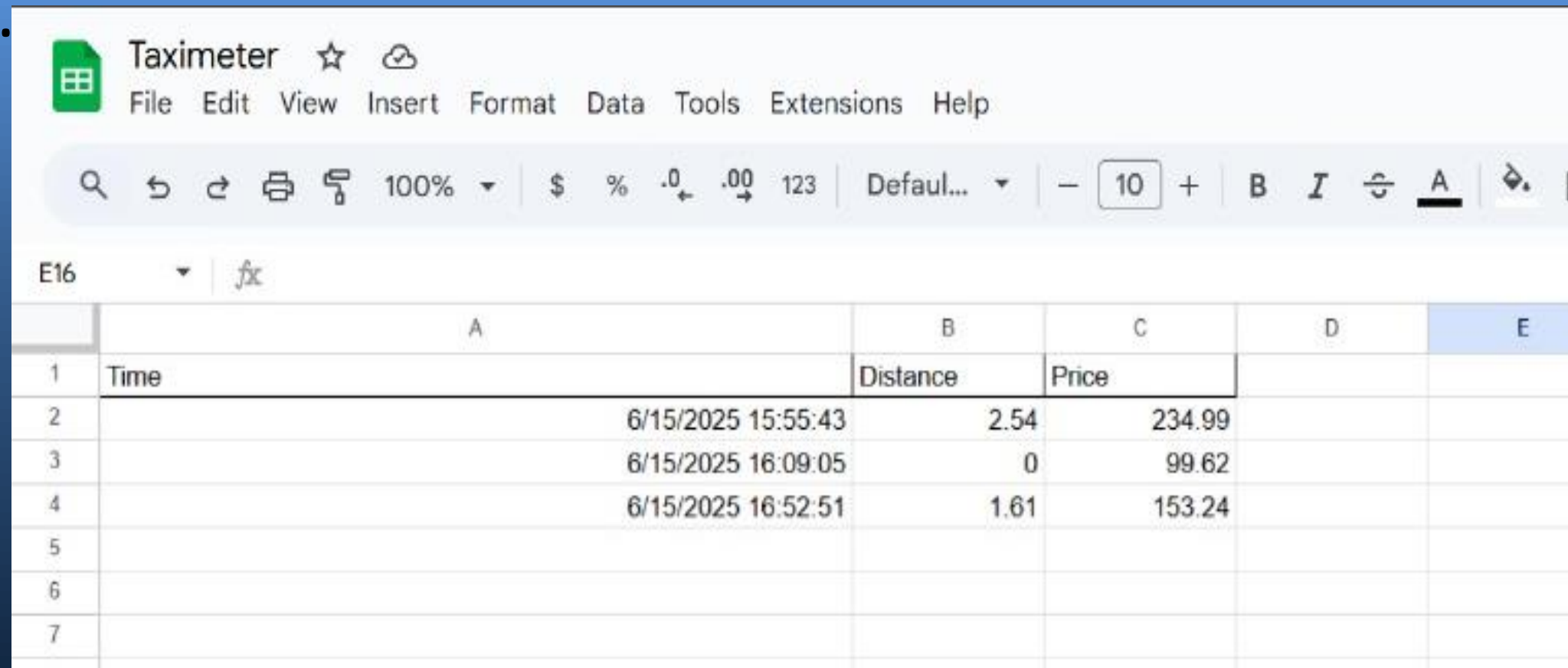
METHODOLOGY

Microcontroller

The microcontroller receives input from the rotary encoder and calculates the total distance covered.

Update Central Database

The calculated distance and current fuel price are sent to a central database in real time.



The screenshot shows a Google Sheet titled 'Taximeter' with a menu bar (File, Edit, View, Insert, Format, Data, Tools, Extensions, Help) and a toolbar with various icons. The spreadsheet contains a table with the following data:

	A	B	C	D	E
1	Time	Distance	Price		
2	6/15/2025 15:55:43	2.54	234.99		
3	6/15/2025 16:09:05	0	99.62		
4	6/15/2025 16:52:51	1.61	153.24		
5					
6					
7					

09

METHODOLOGY

Update Fare
from Database

The fare is calculated dynamically using the latest distance and fuel price data, then updated on the system.

Display Fare to
Customer

The final fare is displayed on the taxi meter screen for the passenger to see.

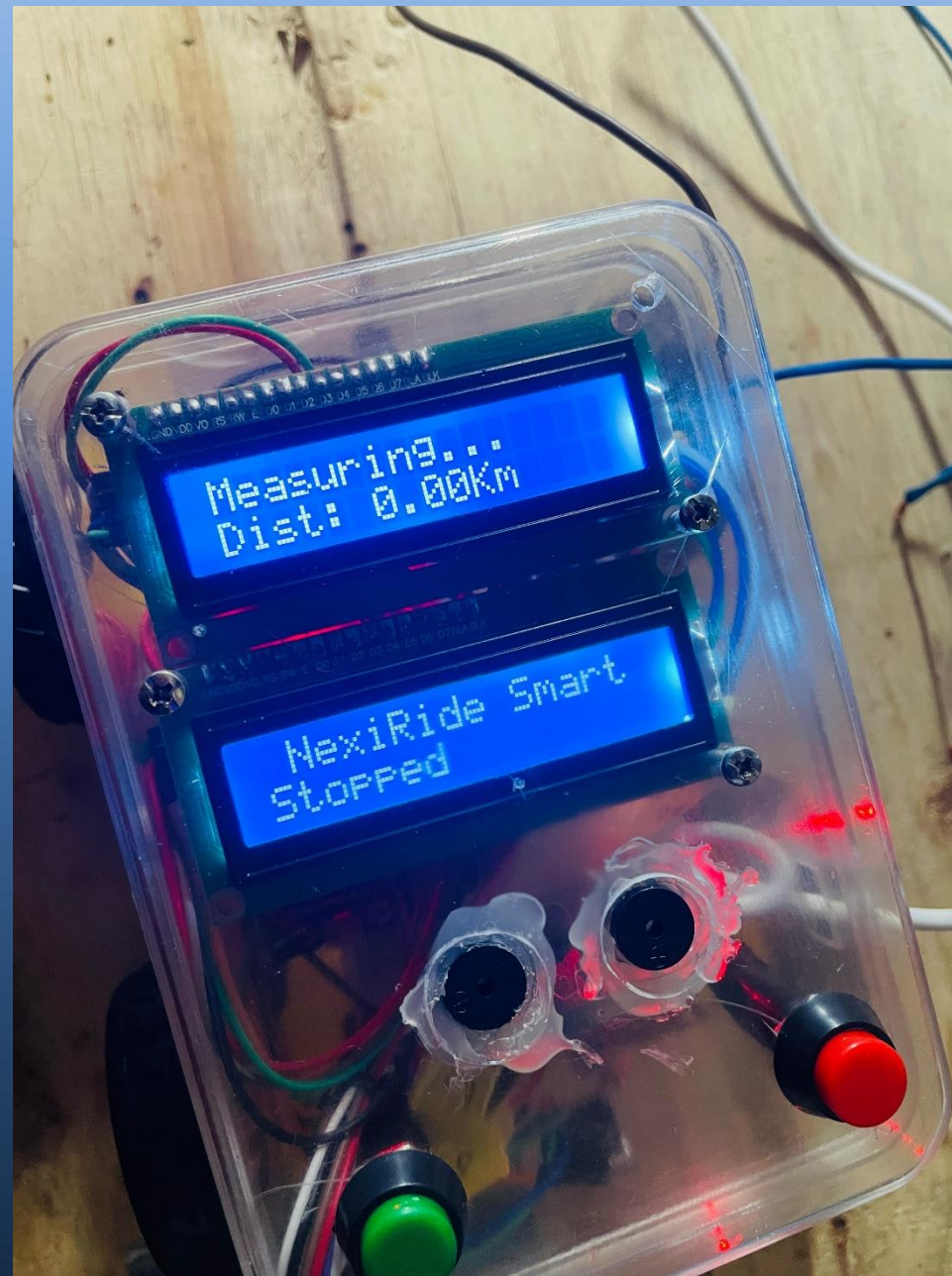


09

METHODOLOGY

End

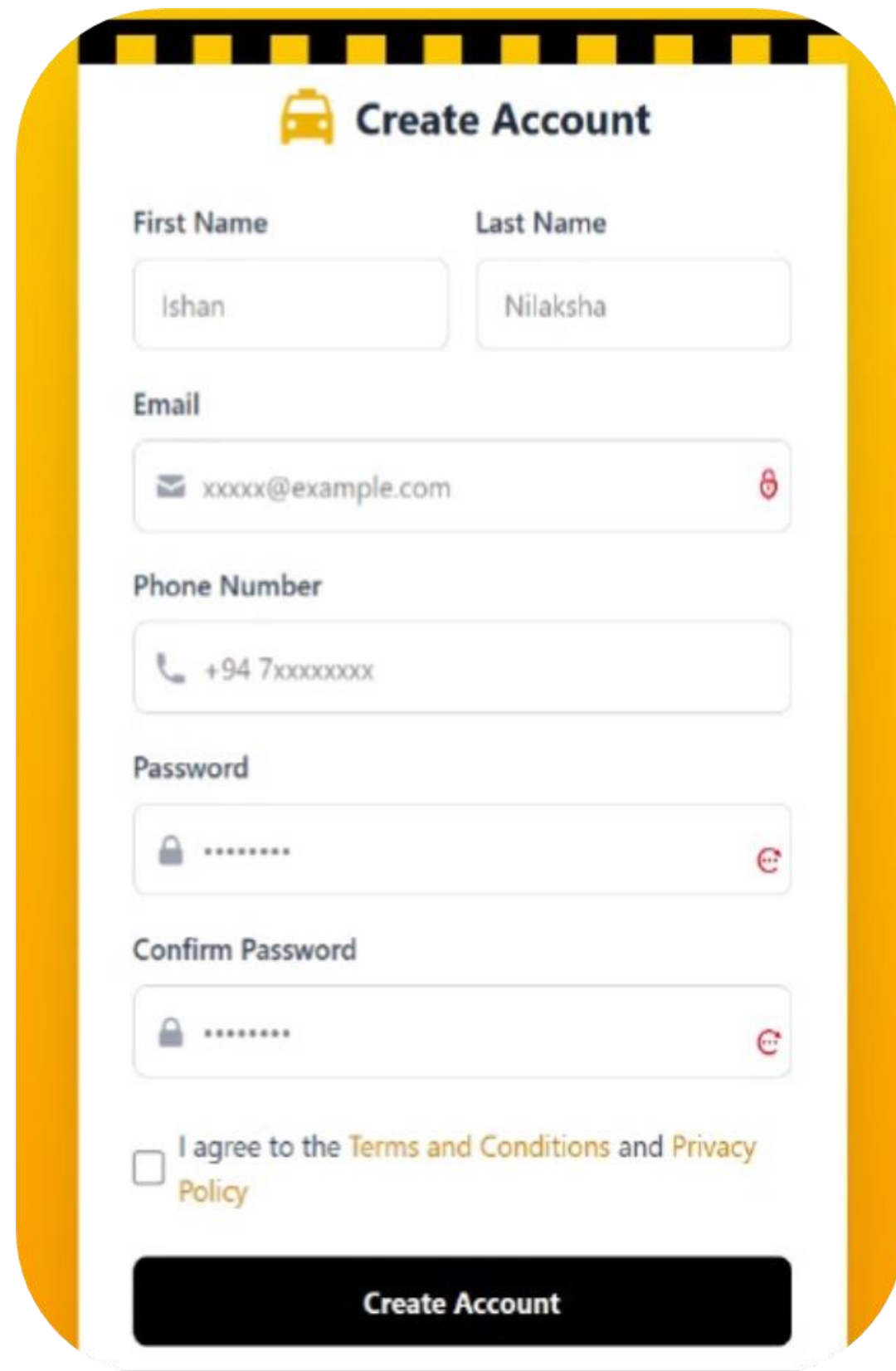
The process ends after the ride is completed and the fare is shown.



The background features a dark blue gradient with four concentric circles in lighter shades of blue. A small, solid blue circle is positioned behind the letter 'E' in the word 'METER'.

TAXI METER APP

01 Create an Account



The 'Create Account' form is displayed on a white background with a yellow taxi icon at the top left. It features a black and yellow checkered header bar. The form includes input fields for 'First Name' (Ishan), 'Last Name' (Nilaksha), 'Email' (xxxxx@example.com), 'Phone Number' (+94 7xxxxxxx), 'Password', and 'Confirm Password'. Each input field has a red lock icon on the right. At the bottom, there is a checkbox for 'I agree to the Terms and Conditions and Privacy Policy' and a black 'Create Account' button.

Create Account

First Name Last Name

Ishan Nilaksha

Email

xxxxx@example.com

Phone Number

+94 7xxxxxxx

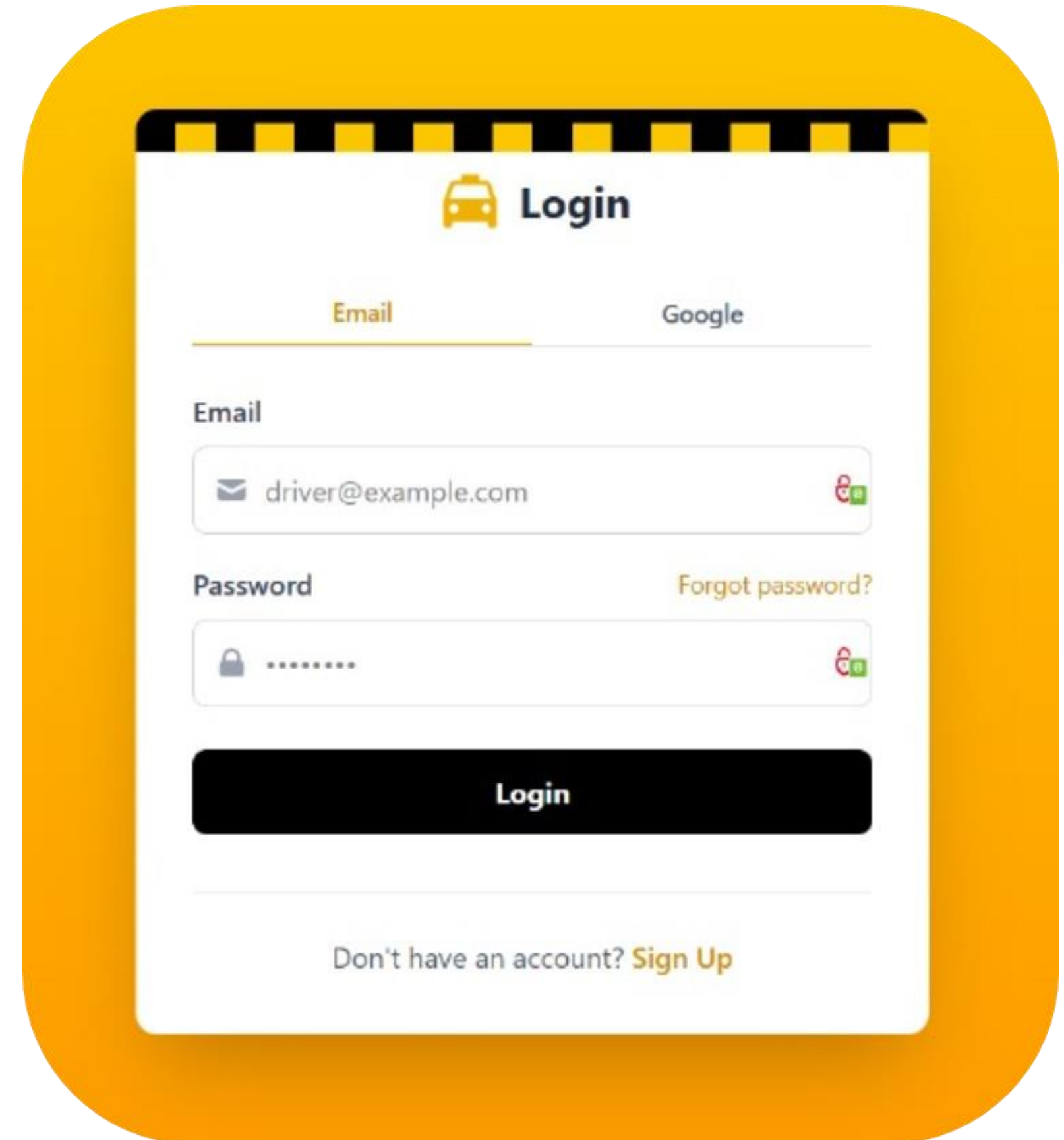
Password

Confirm Password

☐ I agree to the [Terms and Conditions](#) and [Privacy Policy](#)

Create Account

02 Login to the app



The 'Login' form is displayed on a white background with a yellow taxi icon at the top left. It features a black and yellow checkered header bar. The form includes a tabbed interface with 'Email' and 'Google' tabs. The 'Email' tab is active, showing an input field for 'Email' (driver@example.com) and a 'Password' field. A 'Forgot password?' link is next to the password field. Below the fields is a black 'Login' button. At the bottom, there is a link for 'Don't have an account? Sign Up'.

Login

Email Google

Email

driver@example.com

Password

[Forgot password?](#)

Login

Don't have an account? [Sign Up](#)

03

Updated Fuel Details

Fuel Prices Registry





Track and compare fuel prices over time

+ Add Fuel Type

Logout

Fuel Prices Registry

Search fuel types...

Fuel Type	Unit Price	1st KM Price
 92 Petrol	Rs. 293.00 /L	Rs. 99.62
 95 Petrol	Rs. 341.00 /L	Rs. 112.53
 Diesel	Rs. 274.00 /L	Rs. 90.42
 Super Diesel	Rs. 325.00 /L	Rs. 107.25

04

Edit Fuel Details

Edit Fuel Price

Fuel Type

92 Petrol

Unit Price (per liter)

Rs. 293.00 /L

First KM Percentage

34.00 %

Next KM Percentage

30.00 %

Calculated: Rs. 99.62

Calculated: Rs. 87.90

Update Prices

Cancel

Add New Fuel Type

Fuel Type

Select fuel type

Unit Price (per liter)

Rs. 0.00 /L

First KM Percentage

0.00 %

Next KM Percentage

0.00 %

Calculated: Rs. 0.00

Calculated: Rs. 0.00

Add Fuel Type

Cancel

05

Price Comparison

Fuel Prices Registry

Track and compare fuel prices over time

+ Add Fuel Type

Logout

Fuel Prices Comparison

Current Fuel Prices Comparison

Unit Price (Rs/L)

First KM Price (Rs)

92 Petrol

95 Petrol

Diesel

Super Diesel

Select a fuel type to view price history

Fuel Prices Registry

Track and compare fuel prices over time

+ Add Fuel Type

Logout

Select a fuel type to view price history

Please select a fuel type from the table

Avg. Unit Price

\$

Rs. 308.25 /L

Across 4 fuel types

Fuel Prices Registry

Track and compare fuel prices over time

+ Add Fuel Type

Logout

Avg. Unit Price

\$

Rs. 308.25 /L

Across 4 fuel types

Price Range

Rs. 274.00 - 341.00 /L

Difference: Rs. 67.00

Last Updated

Jun 15

95 Petrol at 02:44 PM

30-Day Trend

↑ +465.5%

Increase in last 30 days

Most Expensive

Fuel Prices Registry

Track and compare fuel prices over time

+ Add Fuel Type

Logout

Most Expensive

95 Petrol

Rs. 341.00 /L

Least Expensive

Diesel

Rs. 274.00 /L

TaxiMeter

Fuel Price Manager

© 2025 Fuel Price Manager

All rights reserved Ishan Hatharasinghe

T
I
M
E

P
L
A
N

- Planning
- Hardware Development
- software Development
- Testing & Calibration
- Finalization



11

COST ANALYZE

COMPONENT	QUANTITY	ESTIMATED COST (LKR)
Arduino Uno Board	1	100
Arduino Mega Board		
L298N Motor Driver	1	330
NEO-6M GPS Module	1	780
ULTRASONIC Sensor	1	195
DHT11 Module	1	175
5V Active Buzzer	1	32
DS3231 RTC Module	1	320
1602 LCD Display (Blue)	2	550
I2C	2	260
Mini Rocker Switch	1	10
Gear Motor (Yellow)	4	460
Wheel Normal	4	280
L293D Motor Shield	1	390
18650 Green 1800MAH-Button Top	2	590
18650*2 Battery Holder	1	50
Other		4000
Total		8000/=

12

FUTURE DEVELOPMENT



- ▶ **GPS Module:** Enable real-time location tracking and route-based fare.
- ▶ **Load Cell:** Detect passenger presence or package weight for fare adjustment.
- ▶ **Wheel Encoder:** Improve distance accuracy and vehicle diagnostics.
- ▶ **RFID Reader:** Add driver/passenger authentication and digital payments.
- ▶ **Ultrasonic Sensor:** Assist with obstacle detection and passenger presence

RESOURCES & REFERENCES



RESOURCES

- Arduino
- MS office
- Java script
- React



REFERENCES

- Arduino project hub
- Instructables
- youtube

14



Q & A

THANK YOU

CONNECT WITH US.

 077-843 2168

 test01@gmail.com

