Bilkent University
Department of Computer Engineering

Senior Design Project
T2511
E-Way

Detailed Design Report

22103511, Utku Yüksel, utku.yuksel@ug.bilkent.edu.tr
22103680, Furkan Özek, furkan.ozek@ug.bilkent.edu.tr
22102898, Halis Vefa Türkyılmaz, vefa.turkyilmaz@ug.bilkent.edu.tr
22102800, Aziz Üzümcü, aziz.uzumcu@ug.bilkent.edu.tr

Supervisor: Salih Özgür Öğüz
Course Instructors: Mert Bıçakçı

March 13, 2026

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfilment of the requirements of the Senior Design Project course CS491/2.

Contents

Detailed Design Report

1 Introduction

1.1 Purpose of the System

E-Way is an intelligent route planning system designed to address critical challenges faced by Electric Vehicle (EV) drivers, such as range anxiety, unpredictable charging station availability and long waiting times due to congestion. The primary objective of the system is to collect and analyze historical data from public sources to estimate station availability using artificial intelligence (AI) models. Unlike traditional navigation tools, E-Way incorporates these AI-based predictions into route generation allowing users to actively avoid congested stations and minimize total travel time. Furthermore, the system is designed for user comfort, providing detailed station information (e.g., socket types, power levels, pricing) and highlighting nearby amenities like cafés and markets based on user preferences.

1.2 Design Goals

The design of the E-Way system is driven by several critical non-functional requirements to ensure a seamless and robust user experience:

- **Usability:** The mobile interface is intuitively designed specifically for drivers, requiring minimal interaction to ensure public safety while driving. Visual elements are designed for maximum visual clarity at a glance.
- **Reliability:** The system is built to reliably present charging station data, employing graceful degradation; if external APIs (like the EPDK API) temporarily fail, the system will display the last known valid data rather than crashing or showing an empty state.
- **Performance:** The system architecture supports near real time updates for recalculating routes or refreshing station status while a trip is in progress. Database queries are optimized for efficiency using proper indexing strategies in MongoDB.
- **Scalability:** The backend, built with FastAPI and the MongoDB Atlas database are designed to scale horizontally to support an increasing number of concurrent users and rapidly growing historical occupancy datasets. The AI module is also capable of being retrained on larger datasets over time.
- **Maintainability and Modularity:** The system utilizes a service oriented, modular architecture (Client, Backend, Logic, Storage layers) to allow independent component updates. The backend adheres to clean RESTful API design principles to facilitate debugging and future integrations.

1.3 Definitions, Acronyms, and Abbreviations

- **AI:** Artificial Intelligence
- **API:** Application Programming Interface
- **DB:** Database
- **ETA:** Estimated Time of Arrival
- **EV:** Electric Vehicle
- **GIS:** Geographic Information System
- **KVKK:** Kişisel Verilerin Korunması Kanunu (Law on Protection of Personal Data)
- **LSTM:** Long Short-Term Memory (Neural Network Model)
- **ML:** Machine Learning
- **POI:** Point of Interest
- **SoC:** State of Charge
- **UI:** User Interface

1.4 Overview

This Detailed Design Report provides an in depth blueprint of the E-Way system's architecture and operational components. Section 2 reviews the current software architecture and alternatives in the market. Section 3 details the proposed software architecture including subsystem decomposition, hardware/software mapping and access control policies. Section 4 defines the specific services provided by each subsystem. Section 5 outlines the integration test cases designed to verify system functionality. Section 6 discusses the consideration of various engineering factors on the design. Finally, Section 7 elaborates on teamwork details and leadership distribution throughout the project lifecycle.

2 Current Software Architecture and Alternatives

Currently, Electric Vehicle (EV) drivers in Türkiye rely on a fragmented software ecosystem to plan their journeys and locate charging stations. There is no single unified architecture that seamlessly integrates route planning with comprehensive, predictive and real time station data across all network operators. The existing software solutions and competitors can be categorized into three main architectural approaches, each with significant limitations regarding the problem E-Way aims to solve:

- **General Navigation Platforms (e.g., Google Maps):** These platforms utilize globally scaled architectures for routing and basic POI (Point of Interest) retrieval. However, their data models lack detailed, region specific EV infrastructure attributes, such as real time power levels, socket types and historical occupancy analysis. Architecturally, they do not integrate predictive occupancy models for EV charging which means the system cannot forecast whether a station will be congested by the user's estimated time of arrival (ETA).
- **Operator-Specific Applications (e.g., Eşarj, ZES):** These systems employ closed loop architectures. Their backends provide highly accurate, real time data directly from their proprietary charging

hardware networks. However, their independent approach prevents cross network integration. Drivers planning a long distance trip are forced to manually switch between multiple client applications to find available stations which is inefficient and distracting. Furthermore, these systems lack cross operator route optimization algorithms and AI-based demand prediction logic.

- **Crowdsourced Platforms (e.g., PlugShare):** These platforms rely on architectures heavily dependent on user generated content for state updates. While effective for static data (station locations) and community reviews, the architectural reliance on manual user inputs often results in high latency for state changes which leads to outdated or inaccurate availability information. They generally lack the robust, integrated routing engines and automated predictive analytics required for reliable dynamic travel planning.
- **Official Occupancy Application (Şarj@TR):** This mobile application serves the momentary data of all stations from all brands registered in Türkiye. However, it does not hold a score for estimation of the near future occupancy.

In summary, the current software landscape forces the user to act as the integration layer, manually synthesizing data from multiple isolated applications which leads to uncertain travel times. There is a clear lack of an integrated architectural solution that offers predictive, data driven route optimization tailored specifically to the EV charging infrastructure in Türkiye.

3 Proposed Software Architecture

3.1 Overview

The transition from the analysis model to the detailed system design model requires decomposing the E-Way platform into manageable, highly cohesive and loosely coupled subsystems. The proposed software architecture of E-Way follows a Service Oriented Architecture (SOA) pattern. This architectural style was selected because it naturally separates the client side presentation from the complex backend business logic, external API integrations and computationally heavy Artificial Intelligence (AI) operations.

The global control flow of the system is centralized around a RESTful API Gateway built with Python's FastAPI framework. When an EV driver makes a request via the Flutter based mobile client, the request is transmitted over HTTPS to the backend. The backend then orchestrates the necessary operations: verifying user authentication, fetching real time mapping data from external providers, retrieving station metadata from the official EPDK API and querying the MongoDB Atlas database for user preferences and vehicle parameters. Crucially, the backend interacts with the internal Logic Layer where the AI prediction model resides, injecting historical occupancy data to forecast station availability. Finally, the optimized routing payload is aggregated and returned to the client.

By decoupling these layers, the architecture ensures that the AI prediction algorithms can be retrained or updated and external APIs can be

swapped or modified without requiring updates to the mobile application or causing system wide downtime.

3.2 Subsystem Decomposition

To achieve the design goals of scalability, maintainability and modularity, the E-Way system is decomposed into four primary subsystems. Each subsystem encapsulates specific responsibilities and communicates with others through well defined interfaces.

3.2.1 Mobile Client Subsystem (Frontend)

The Mobile Client Subsystem is the sole interface for the EV drivers, developed using the Flutter framework to provide a native like experience on both iOS and Android from a single codebase. It is responsible for user interaction, data presentation and capturing hardware level data such as GPS location.

This subsystem is further divided into:

- **UI Components Layer:** Manages the rendering of pages (Welcome, Login, Map, Route Input, Station Details) as defined in the UI mockups. It ensures visual clarity and accessibility.
- **State Management Module:** Handles the dynamic state of the application (e.g., current battery level, selected route, active user session) using modern state management solutions to ensure the UI reflects real time data without unnecessary rerenders.
- **API Client & Networking Service:** Acts as the communication bridge. It formats user inputs into JSON payloads, handles asynchronous HTTP/HTTPS requests to the backend, manages authentication tokens (JWT) and gracefully catches network errors to display appropriate fallback UI to the user.

3.2.2 Backend & API Gateway Subsystem

This subsystem is the core coordinator of the E-Way platform, implemented in Python using FastAPI. It serves as the single entry point for all client requests and manages the business logic that does not involve heavy machine learning computations.

Its internal modules include:

- **Authentication and Authorization Module:** Secures the system by verifying user identities issuing JSON Web Tokens (JWT) and protecting sensitive endpoints. It ensures that users can only access and modify their own profiles and vehicle configurations.
- **External Integration Service:** A dedicated layer for managing outbound communications to third party services. It handles rate limiting, error handling and data parsing for Google Maps Platform (fetching base routes, traffic data and nearby amenities).
- **Route Coordination Engine:** Receives the raw route data from Google Maps and the charging station coordinates. It filters the stations

based on user preferences (e.g., AC/DC, specific operators) and prepares the data matrix required by the AI Logic Subsystem to evaluate the routes.

3.2.3 AI & Predictive Logic Subsystem

This subsystem isolates the intelligent decision making and machine learning capabilities of E-Way. Separating this from the general backend logic allows for independent scaling as ML operations are computationally intensive.

- **Data Collection Pipeline:** It fetches momentary occupancy data of stations and saves it to corresponding clusters in the database.
- **Data Preprocessing Pipeline:** Formats and normalizes historical occupancy data retrieved from the database.
- **Occupancy Prediction Engine:** Utilizes trained Machine Learning models (e.g., Time-Series Forecasting or Regression algorithms) to process the preprocessed data. Based on the user's ETA at a specific station, it outputs a predicted occupancy percentage which is categorized into "Available," "Moderate," or "Busy."
- **Smart Route Optimizer:** Integrates the occupancy predictions with the geographic route data. It applies a penalty scoring system to routes that include congested stations, ultimately sorting and selecting the optimal route alternatives (Fastest, Least Busy, Comfort) to be sent back to the user.

3.2.4 Data Storage Subsystem

The persistent data management is handled by MongoDB Atlas, a NoSQL cloud database. The schemaless nature of document based databases provides the flexibility needed to store varying structures of external API responses and rapidly growing historical datasets.

- **User & Profile Collection:** Stores securely hashed user credentials, saved vehicle profiles (battery capacity, consumption rates) and personalized routing preferences.
- **Station Metadata Collection:** Acts as a cache and persistent store for the EPDK station data including static properties like location, socket types, power capacity and operator details.
- **Historical Occupancy Collection:** The largest dataset in the system, storing time stamped logs of station usage. This collection is continuously updated and serves as the foundational training data for the AI Prediction Engine.

3.3 Hardware/Software Mapping

As the E-Way system is a software intensive mobile application, it does not involve the development of custom physical hardware components. The hardware/software mapping strictly follows a standard cloud deployment model:

- **Client Environment:** The mobile application (Client Layer) maps directly to the end users' personal smartphones, utilizing the device's native CPU, memory and GPS hardware for UI rendering and location tracking.
- **Server Environment:** The backend API, AI logic and database are containerized (via Docker) and mapped to a cloud based host server (Virtual Private Server). The host machine provides the computational resources (CPU/RAM) required for executing the machine learning models and routing algorithms.

3.4 Persistent Data Management

The persistent data management strategy for the E-Way platform is built around MongoDB, a highly scalable NoSQL document database. A nonrelational database architecture was explicitly chosen for this project due to the semistructured and rapidly changing nature of the data retrieved from external APIs (such as the EPDK service). Unlike traditional SQL databases that require rigid schemas, MongoDB's flexible BSON (Binary JSON) document model allows the system to seamlessly ingest, store and query complex nested data structures like charging station metadata and historical occupancy logs without requiring constant schema migrations. The data layer is conceptually divided into two primary operations: the Data Acquisition Pipeline and the Core Collections.

3.4.1 Data Acquisition Pipeline

To power the AI prediction models with accurate historical data, E-Way employs automated background workers that continuously poll the national EPDK API for station statuses.

- **Regular Polling Mechanism:** The system utilizes a dedicated fetcher script that queries specific target station IDs at defined polling intervals. The raw JSON responses from endpoints are captured as temporal snapshots, cataloged by precise date and time labels and saved into a lifecycle logs directory.
- **Targeted State Fetching:** To enrich the dataset with congested state scenarios, a specialized worker iterates through a master list of stations, specifically hunting for sockets that return an "IN_USE" status rather than "FREE". This dual pipeline approach ensures a balanced dataset of both idle and active states for accurate machine learning training.

3.4.2 Core Database Collections and Document Schemas

The MongoDB database is composed of three primary collections, each serving a distinct domain of the application:

- **Users Collection (users):** This collection manages all registered drivers on the platform. When a user registers, a document is inserted containing fundamental properties: username, email, a securely hashed version of their password (hashed_password), a UTC timestamp of their registration (created_at) and an active status flag (is_active).

- **Stations Metadata Collection (stations):** This collection acts as the persistent cache for the physical attributes of charging stations. The schema accommodates nested arrays for sockets capturing vital EV parameters. A typical document includes the station ID, detailed address, network operator information and whether it supplies green energy. Crucially, it contains a sockets array detailing the power output (e.g., 180.0 kW), socket type (e.g., DC) and subtype (e.g., DC_CCS).
- **Occupancy Logs Collection (raw_in_use):** This is a high volume, time series collection fundamental to the AI prediction engine. It records the dynamic availability of the sockets. Each document logs the time intervals that the station was “IN_USE”.
- **Processed Occupancy Data Collection (socket_stats):** This collection holds the total occupancy observed for intervals of a day and week for each station. This total occupancy gives an occupancy ratio when it is divided to total time of observation.

3.5 Access control and security

Ensuring the privacy of user data and the integrity of the system's endpoints is paramount. The E-Way system implements a robust access control and security policy within its FastAPI backend, strictly adhering to modern authentication standards.

- **Authentication Protocol:** The system utilizes JSON Web Tokens (JWT) for stateless authentication. Upon a successful login request, the system verifies the user's identity and generates an access token. This token must be included in the Authorization header of subsequent requests, ensuring that users can only access endpoints and data (such as their personal route history or saved vehicles) for which they are authorized.
- **Password Cryptography:** Plain text passwords are never stored in the database. During the registration process, the user's password undergoes a secure hashing algorithm before being persisted in the MongoDB users collection. When an authentication attempt occurs, the provided password is cryptographically compared against the stored hash rather than being decrypted.
- **Input Validation and Error Handling:** The authentication router strictly validates incoming data models (e.g. UserRegister, UserLogin). If a user attempts to register with an email that already exists in the system or provides incorrect login credentials, the API immediately intercepts the request and raises a standard HTTP 400 Bad Request exception with generic error messages ("Email already registered" or "Incorrect email or password") to prevent malicious actors from enumerating valid accounts.

4 Subsystem Services

The E-Way system relies on several core services distributed across its backend and logic layers. Each service encapsulates a specific domain of business logic, receiving structured inputs via API requests or background

cron jobs, processing the data and returning standardized outputs to be utilized by the mobile client or other internal subsystems.

4.1 Authentication and User Management Service

This service is responsible for securely managing user identities, application sessions and profile preferences. It strictly enforces security protocols to protect user data from unauthorized access.

- **Description:** Handles user registration and login authentication using the FastAPI routing framework. It interacts directly with the users collection to validate incoming requests and apply password cryptography. Passwords are securely hashed before insertion and login attempts are verified against these hashes.
- **Inputs:**
  - **Registration:** UserRegister data model including username, email and raw password.
  - **Login:** UserLogin data model containing email and raw password.
- **Outputs:**
  - **On success:** Generates and returns a JSON Web Token (JWT) encapsulated in a Token model, enabling stateless authorization for subsequent requests.
  - **On failure:** Raises HTTP 400 Bad Request exceptions with specific details (e.g., "Email already registered" or "Incorrect email or password").

4.2 Data Acquisition and Synchronization Service

This automated background service acts as the data ingestion engine which ensures the E-Way platform is constantly synchronized with the national EV charging infrastructure.

- **Description:** Periodically polls the EPDK API to capture the real time status of charging stations. It runs dedicated scripts to take hourly snapshots of target stations and a specialized worker that actively hunts for sockets currently returning an "IN_USE" status rather than "FREE".
- **Inputs:** Target Station IDs, current temporal parameters (date and time) and specific API request headers designed to handle gzip encoding and keep alive connections.
- **Outputs:** Structured JSON snapshots detailing station metadata, socket types, power capacities and active pricing configurations which are logged and prepared for database insertion.

4.3 Occupancy Prediction and Statistical Service (AI Module)

This service is the intelligent core of the E-Way system, designed to forecast future station availability by analyzing historical usage patterns.

- **Description:** Running continuously as a background process, this service analyzes raw, unprocessed charging sessions. It calculates the

exact duration a socket was occupied and categorizes this data into temporal clusters using custom calendar utilities (season, day cluster, time interval). It aggregates these statistics and uses bulk write operations to continuously update the predictive models.

- **Inputs:** Raw availability data fetched from the EPDK API and unprocessed session documents queried from the MongoDB database.
- **Outputs:** It performs optimized bulk_write operations to upsert the aggregated data into the socket_stats and occupancy_predictions collections. After processing, it updates the original raw documents by setting a processed: True flag to prevent redundant calculations.

4.4 Smart Routing and Station Filtering Service

This service bridges the gap between raw geographic navigation and EV specific constraints, providing the user with actionable travel plans and filtered station data.

- **Description:** It utilizes an asynchronous MongoDB client to efficiently query and filter the charging stations database without blocking the main event loop. Additionally, it exposes RESTful endpoints to receive and record user travel runs or routes.
- **Inputs:** Payload models such as RunUpload containing user route data, alongside filtering criteria (e.g., required socket types or minimum power levels).
- **Outputs:** It inserts the structured route data into the runs collection, automatically appending a UTC timestamp to ensure chronological accuracy and returns a confirmation payload. It also provides filtered station lists to the mobile application for map rendering.

5 Test Cases

This section outlines the functional and non-functional integration test cases that will be executed during and after the implementation phase of the E-Way project. These procedures are designed for software test engineers to verify and validate the system's functionality without requiring access to the underlying source code. The results of these tests will be documented in the Final Report.

5.1 Authentication and User Management Tests

Table 1. Test 1

Test ID	TC-01	Category	Functional	Severity	Critical
Objective	User Registration - Successful Account Creation				
Steps	1. Navigate to the Sign-Up page. 2. Enter a valid unique email, username and password. 3. Click the "Register" button.				
Expected	System creates the account, hashes the password in the database and returns a "User created successfully" message.				
Date-Result					

Table 2. Test 2

Test ID	TC-02	Category	Functional	Severity	Major
Objective	User Registration - Duplicate Email Handling				
Steps	1. Navigate to the Sign-Up page. 2. Enter an email address that is already registered in the system. 3. Click the "Register" button.				
Expected	Registration fails. The API returns an HTTP 400 error with the detail "Email already registered".				
Date-Result					

Table 3. Test 3

Test ID	TC-03	Category	Functional	Severity	Critical
Objective	User Login - Successful Authentication				
Steps	1. Navigate to the Login page. 2. Enter valid registered email and correct password. 3. Click "Login".				
Expected	Login is successful. The API returns a valid JWT (JSON Web Token) access token for the session.				
Date-Result					

Table 4. Test 4

Test ID	TC-04	Category	Security	Severity	Critical
Objective	User Login - Incorrect Credentials				
Steps	1. Navigate to the Login page. 2. Enter a valid email but an incorrect password. 3. Click "Login".				
Expected	Login fails. The API returns an HTTP 400 error with the detail "Incorrect email or password".				
Date-Result					

5.2 Backend and Database Operations Tests

Table 5. Test 5

Test ID	TC-07	Category	Integration	Severity	Major
Objective	Route Upload - Save User Run				
Steps	1. Authenticate user and obtain JWT token. 2. Send a POST request to /run/upload with a valid RunUpload JSON payload. 3. Check the database runs collection.				

Expected	The API returns {"inserted": True} and the document is saved with a precise created_at UTC timestamp.
Date-Result	

Table 6. Test 6

Test ID	TC-08	Category	Functional	Severity	Minor
Objective	Database Health Check				
Steps	1. Send a GET request to the /health endpoint. 2. Inspect the JSON response.				
Expected	The API returns {"status": "ok"} and lists the available MongoDB collections (e.g., users, runs, stations).				
Date-Result					

Table 7. Test 7

Test ID	TC-09	Category	Logic / AI	Severity	Critical
Objective	Prediction Updater - Calculate Occupied Mins				
Steps	1. Insert a mock raw session with status: "FREE" and status: "IN_USE". 2. Trigger the prediction_updater.py process. 3. Check the socket_stats collection.				
Expected	The system correctly calculates the occupied_mins based on start and end times, applies calendar clustering and inserts the data.				
Date-Result					

Table 8. Test 8

Test ID	TC-10	Category	Integration	Severity	Major
Objective	Prediction Updater - Flag Processed Data				
Steps	1. Verify a raw document in raw_in_use has processed: False. 2. Run the prediction updater script. 3. Re-query the same raw document.				
Expected	After bulk writing stats, the original raw document is updated via update_many to have {"processed": True} to avoid double counting.				
Date-Result					

5.3 Station Filtering and Map Display Tests

Table 9. Test 9

Test ID	TC-11	Category	Functional	Severity	Major
Objective	Station Filter - By Socket Type				
Steps	1. Open the Map page. 2. Open the filter menu and select "DC" socket type. 3. Click "Apply".				
Expected	The map updates to display only charging stations that have at least one DC socket.				
Date-Result					

Table 10. Test 10

Test ID	TC-12	Category	Functional	Severity	Major
Objective	Station Filter - By Minimum Power				
Steps	1. Open the filter menu. 2. Set minimum power slider to "120 kW". 3. Click "Apply".				
Expected	The map only shows stations where socketPower is >= 120.0.				
Date-Result					

Table 11. Test 11

Test ID	TC-13	Category	Functional	Severity	Critical
Objective	Station Details - Bottom Sheet				
Steps	1. Tap on a specific charging station marker on the map. 2. Observe the popped-up bottom sheet.				
Expected	The UI displays the station title, network operator, address and a list of available sockets.				
Date-Result					

Table 12. Test 12

Test ID	TC-14	Category	Usability	Severity	Minor
Objective	Map Navigation - Zoom and Pan				
Steps	1. Perform pinch-to-zoom in and out on the map. 2. Pan across different cities.				
Expected	The map renders smoothly without crashing and station markers dynamically cluster or uncluster based on zoom level.				
Date-Result					

5.4 Smart Routing and Navigation Tests

Table 13. Test 13

Test ID	TC-15	Category	Integration	Severity	Critical
Objective	Route Generation - Sufficient Battery				
Steps	1. Enter a destination that is 50 km away. 2. Set current vehicle SoC (State of Charge) to 90%. 3. Request route.				
Expected	The system generates a direct route to the destination without adding any intermediate charging stops.				
Date-Result					

Table 14. Test 14

Test ID	TC-16	Category	Integration	Severity	Critical
Objective	Route Generation - Low Battery				
Steps	1. Enter a destination that is 400 km away. 2. Set current vehicle SoC to 20%. 3. Request route.				
Expected	The routing engine calculates the range and automatically inserts necessary charging station waypoints along the route.				
Date-Result					

Table 15. Test 15

Test ID	TC-17	Category	Performance	Severity	Major
Objective	Route Recalculation				
Steps	1. Start navigation for a generated route. 2. Simulate GPS location deviating from the planned path by 1 km.				
Expected	The system detects the deviation and recalculates the route and ETA within 3 seconds.				
Date-Result					

Table 16. Test 16

Test ID	TC-18	Category	Functional	Severity	Minor
Objective	Invalid Destination Handling				
Steps	1. Enter an unreachable destination (e.g., across an ocean with no ferry). 2. Request route.				
Expected	The system gracefully fails and displays an error message: "Unable to find a valid route to this destination."				
Date-Result					

5.5 Occupancy Prediction (AI) UI Tests

Table 17. Test 17

Test ID	TC-19	Category	Integration	Severity	Critical
Objective	Display Predicted Occupancy				
Steps	1. Generate a route with a charging stop. 2. Observe the charging station waypoint details in the route list.				
Expected	The UI shows the AI-predicted occupancy status (e.g., "Available", "Busy") based on the specific Estimated Time of Arrival (ETA) at that station.				
Date-Result					

Table 18. Test 18

Test ID	TC-20	Category	Reliability	Severity	Major
Objective	AI Service Degradation Fallback				
Steps	1. Temporarily disable the AI Prediction microservice. 2. Request a route with a charging stop.				
Expected	The system does not crash. It falls back to displaying the "last known real-time status" instead of a future prediction, showing a warning icon.				
Date-Result					

5.6 User Profile and Vehicle Management Tests

Table 19. Test 19

Test ID	TC-21	Category	Functional	Severity	Critical
Objective	Add Vehicle Profile				
Steps	1. Navigate to Profile -> My Vehicles. 2. Enter vehicle details (Battery Capacity, Consumption rate, Connector Type). 3. Save.				
Expected	The vehicle profile is successfully saved to the database and set as the active vehicle for future routing calculations.				
Date-Result					

Table 20. Test 20

Test ID	TC-22	Category	Functional	Severity	Minor
Objective	Edit User Profile				
Steps	1. Navigate to Profile settings. 2. Update the username and click "Save".				
Expected	The profile is updated and the new username is immediately reflected in the application drawer/header.				
Date-Result					

Table 21. Test 21

Test ID	TC-23	Category	Functional	Severity	Minor
Objective	Save Station to Favorites				
Steps	1. Open a station's detail bottom sheet. 2. Tap the "Heart/Favorite" icon. 3. Check the "Favorite Stations" list.				
Expected	The icon toggles its state and the station is successfully listed in the user's Favorite Stations menu.				
Date-Result					

Table 22. Test 22

Test ID	TC-24	Category	Functional	Severity	Major
Objective	View Trip History				
Steps	1. Navigate to the "My Trips" or "Runs" page. 2. Verify the list of past routes.				
Expected	The application successfully fetches and displays the list of previously completed routes, sorted by created_at timestamp.				
Date-Result					

Table 23. Test 23

Test ID	TC-25	Category	Security	Severity	Critical
Objective	Account Deletion				
Steps	1. Navigate to Profile -> Security. 2. Click "Delete Account" and confirm the prompt.				
Expected	The user session is terminated, the user is redirected to the login screen and their data is flagged as deleted/removed from the database.				
Date-Result					

5.7 Performance and Load Tests

Table 24. Test 24

Test ID	TC-26	Category	Performance	Severity	Major
Objective	App Launch Time				
Steps	1. Completely close the mobile application. 2. Tap the app icon to launch. 3. Measure time until the home screen is fully rendered.				
Expected	The application must load and become interactive in under 3.0 seconds under normal network conditions.				
Date-Result					

Table 25. Test 25

Test ID	TC-27	Category	Performance	Severity	Major
Objective	Map Rendering with Heavy Data				
Steps	1. Zoom out the map to view the entire country. 2. Wait for all station markers to load.				
Expected	The map clusters the markers efficiently, maintaining a minimum of 30 FPS without freezing the UI.				
Date-Result					

Table 26. Test 26

Test ID	TC-28	Category	Performance	Severity	Critical
Objective	Route Calculation Speed				
Steps	1. Enter a complex cross-country destination. 2. Request a route.				
Expected	The backend processes the Google Maps data, filters stations and returns the optimized payload in under 5 seconds.				
Date-Result					

Table 27. Test 27

Test ID	TC-29	Category	Performance	Severity	Major
Objective	AI Prediction Response Time				
Steps	1. Open a station detail view that requires an AI prediction. 2. Measure the latency of the prediction result.				
Expected	The AI service returns the calculated occupancy prediction within 1 second.				
Date-Result					

Table 28. Test 28

Test ID	TC-30	Category	Performance	Severity	Minor
Objective	Database Query Efficiency				
Steps	1. Apply multiple filters (e.g., AC, >50kW, Free only). 2. Execute search.				
Expected	The MongoDB query utilizes indexes to return the filtered list in under 500 milliseconds.				
Date-Result					

5.8 Security and Authorization Tests

Table 29. Test 29

Test ID	TC-31	Category	Security	Severity	Critical
Objective	Unauthorized API Access				
Steps	1. Intercept API requests using a tool like Postman. 2. Attempt to call the /run/upload endpoint without a JWT token.				
Expected	The API blocks the request and returns an HTTP 401 Unauthorized error.				
Date-Result					

Table 30. Test 30

Test ID	TC-32	Category	Security	Severity	Critical
Objective	Expired Token Handling				
Steps	1. Authenticate and obtain a JWT. 2. Wait for the token's expiration time to pass. 3. Attempt to fetch user profile data.				
Expected	The API rejects the token. The mobile client intercepts the 401 error and gracefully redirects the user to the login screen.				
Date-Result					

Table 31. Test 31

Test ID	TC-33	Category	Security	Severity	Critical
Objective	Cross-User Data Isolation				
Steps	1. Login as User A and obtain User A's ID. 2. Login as User B. 3. Attempt to fetch trip history using User A's ID.				
Expected	The system returns an HTTP 403 Forbidden error, preventing users from accessing others' data.				
Date-Result					

Table 32. Test 32

Test ID	TC-34	Category	Security	Severity	Critical

Objective	NoSQL Injection Prevention
Steps	1. Attempt to login using a MongoDB query operator (e.g., {"$gt": ""}) in the email field.
Expected	The auth.py router strictly validates the UserLogin model and rejects the malformed input, preventing injection attacks.
Date-Result	

Table 33. Test 33

Test ID	TC-35	Category	Security	Severity	Critical
Objective	Secure Password Verification				
Steps	1. Directly query the MongoDB users collection. 2. Inspect the stored user records.				
Expected	The hashed_password field contains only cryptographic hashes, with no plain-text passwords visible.				
Date-Result					

5.9 Reliability and Error Handling Tests

Table 34. Test 34

Test ID	TC-36	Category	Non-functional	Severity	Major
Objective	GPS Signal Loss				
Steps	1. Start active route navigation. 2. Enter a tunnel to simulate total GPS signal loss.				
Expected	The app displays a "Searching for GPS" banner and relies on last-known heading until the signal is restored.				
Date-Result					

Table 35. Test 35

Test ID	TC-37	Category	Non-functional	Severity	Major
Objective	Network Disconnection				
Steps	1. Disconnect Wi-Fi and Cellular data. 2. Attempt to search for a new station.				
Expected	The app displays a friendly "No Internet Connection" alert instead of crashing or freezing.				
Date-Result					

Table 36. Test 36

Test ID	TC-38	Category	Integration	Severity	Critical
Objective	External API Downtime				
Steps	1. Simulate the EPDK API being offline or returning 500 errors. 2. Trigger regular_check.py.				
Expected	The script catches the exception, logs the error gracefully and does not overwrite existing valid database records.				
Date-Result					

Table 37. Test 37

Test ID	TC-39	Category	Functional	Severity	Minor
Objective	Invalid Input Handling				
Steps	1. Open the vehicle profile creation page. 2. Enter alphabetical characters into the "Battery Capacity" numeric field.				
Expected	The frontend form validation prevents submission and highlights the field with an error message.				
Date-Result					

Table 38. Test 38

Test ID	TC-40	Category	Reliability	Severity	Major
Objective	Corrupted JSON Response				
Steps	1. Simulate the backend returning a malformed JSON payload. 2. Open the mobile app to receive the data.				
Expected	The mobile app's serialization layer catches the parsing error and displays a general fallback UI rather than a blank screen.				
Date-Result					

Table 39. Test 39

Test ID	TC-41	Category	Functional	Severity	Critical
Objective	Database Connection Loss				
Steps	1. Stop the MongoDB Docker container. 2. Attempt to log in or fetch a route.				
Expected	The FastAPI backend returns a clear HTTP 503 Service Unavailable error instead of hanging indefinitely.				
Date-Result					

Table 40. Test 40

Test ID	TC-42	Category	Logic	Severity	Major
Objective	Overlapping AI Sessions				
Steps	1. Run multiple instances of prediction_updater.py simultaneously.				
Expected	The script handles concurrency correctly and the processed: True flag prevents the same raw data from being calculated twice.				
Date-Result					

5.10 Compatibility and Usability Tests

Table 41. Test 41

Test ID	TC-43	Category	Compatibility	Severity	Major
Objective	Cross-Platform UI				
Steps	1. Install the Flutter app on an Android device. 2. Install the app on an iOS device. 3. Compare the UI elements.				
Expected	Both versions display consistent layouts, fonts and colors, respecting native platform safe areas (notches/status bars).				
Date-Result					

Table 42. Test 42

Test ID	TC-44	Category	Compatibility	Severity	Minor
Objective	Screen Size Responsiveness				
Steps	1. Open the app on a standard smartphone. 2. Open the app on a larger tablet device.				
Expected	The layout adapts responsively; grids expand and text does not overlap or stretch unpleasantly.				
Date-Result					

Table 43. Test 43

Test ID	TC-45	Category	Usability	Severity	Minor
Objective	Dark/Light Mode Switch				
Steps	1. Go to device system settings. 2. Toggle between Light Mode and Dark Mode. 3. Return to the app.				
Expected	The application's theme dynamically updates to match the system preference without requiring an app restart.				
Date-Result					

Table 44. Test 44

Test ID	TC-46	Category	Non-functional	Severity	Critical
Objective	Background Execution				
Steps	1. Start a route navigation. 2. Send the app to the background for 5 minutes. 3. Reopen the app.				
Expected	The app restores its state immediately and the route navigation continues tracking accurately.				
Date-Result					

Table 45. Test 45

Test ID	TC-47	Category	Non-functional	Severity	Major
Objective	Battery Consumption				
Steps	1. Leave the app open on the map screen for 30 minutes with GPS active.				
Expected	The application should not cause excessive battery drain or device overheating (monitored via native profilers).				
Date-Result					

Table 46. Test 46

Test ID	TC-48	Category	Usability	Severity	Minor
Objective	Visual Accessibility				
Steps	1. Change the device's default font size to maximum. 2. Open the application.				
Expected	The text scales appropriately and the UI remains readable without text getting cut off or hidden.				
Date-Result					

Table 47. Test 47

Test ID	TC-49	Category	Usability	Severity	Minor
Objective	Interactive Element Size				
Steps	1. Navigate through the app using standard touch gestures.				
Expected	All tap targets (buttons, map markers, list items) have a minimum touch area (e.g., 48x48 dp) for easy interaction while driving.				
Date-Result					

Table 48. Test 48

Test ID	TC-50	Category	Installation	Severity	Critical
Objective	App Installation & Permissions				
Steps	1. Install the app from a clean state. 2. Launch the app for the first time.				
Expected	The app successfully installs and correctly prompts the user for Location and Notification permissions with clear explanations.				
Date-Result					

6 Consideration of Various Factors in Engineering Design

In this section, the impact of various engineering, environmental and social factors on the design of the E-Way system is evaluated. Each factor is scored on a scale of 0 (no effect) to 10 (maximum effect) to indicate its influence on our architectural and functional decisions.

6.1 Public Health

While E-Way is a software application and does not directly impact physical public health, it actively addresses the psychological stress known as "range anxiety" commonly experienced by EV drivers. By providing reliable route optimization and AI-predicted charging station availability, the system reduces driver uncertainty and cognitive load, indirectly contributing to mental wellbeing and a calmer driving experience.

6.2 Public Safety and Security

Safety and security deeply influenced the E-Way design. From a public safety perspective, the mobile interface is heavily optimized for drivers. It features large touch targets and color coded availability indicators (Green, Orange, Red) to minimize screen time and prevent distracted driving. From a security standpoint, the system architecture strictly enforces data protection via JWT authentication and password hashing. Furthermore, the data acquisition service is specifically designed to respect the legal feasibility and terms of service of public APIs (such as the EPDK infrastructure) which ensures that continuous data scraping and aggregation are conducted within legal boundaries without overwhelming public servers.

6.3 Public Welfare

The system promotes public welfare by making electric vehicle ownership more accessible, reliable and practical. By solving the charging infrastructure fragmentation problem, E-Way encourages the broader public adoption of EVs which aligns with national goals for modernized and sustainable transportation networks.

6.4 Global and Cultural Factors

The transition to electric mobility is a global movement. E-Way incorporates global standards (handling international socket types like CCS, CHAdeMO and Type-2) ensuring the platform is globally compatible and can be seamlessly used by international tourists driving EVs in Türkiye. The

cultural impact is relatively low but involves adapting to local traveling habits and long distance holiday routes.

6.5 Social Factors

The platform fosters a more collaborative EV community by democratizing access to station occupancy data. It prevents the social friction and disputes often caused by long, unpredictable queues at charging stations which lead to a smoother shared infrastructure experience among citizens.

6.6 Environmental Factors

Environmental sustainability is the core driver of the E-Way project. The entire application is built to optimize the usage of zero emission vehicles. By integrating features that highlight "Green Energy" certified charging stations, the application directly supports the reduction of greenhouse gas emissions. Efficient routing also minimizes unnecessary energy consumption that would otherwise be spent searching for available chargers.

6.7 Economic Factors

E-Way offers significant economic benefits and efficiencies. For the user, AI-optimized routing avoids congested stations, saving valuable travel time and minimizing energy consumption. Furthermore, by parsing and displaying dynamic charging prices and tariffs directly in the app, users can make cost effective charging decisions, promoting fair market competition among network operators.

Table 51. Effects of Various Factors

Factor	Effect Level (0-10)	Effect Summary
Public Health	2	Indirect benefit via reduced pollution; low direct impact.
Public Safety	9	High critical impact; driver distraction must be minimized to prevent accidents.
Public Welfare	7	Improves transport efficiency and reduces time waste for citizens.
Global Factors	5	Scalable architecture, but currently localized data dependencies.
Social Factors	6	Encourages sustainable living and equitable access to information.

Environmental	8	Strong alignment with sustainability goals and energy optimization.
Economic	7	Direct impact on development budget (API costs) and user savings.

7 Teamwork Details

The success of the E-Way project relies heavily on the effective division of labor, continuous communication and shared responsibilities among the team members: Utku Yüksel, Furkan Özek, Halis Vefa Türkyılmaz and Aziz Üzümcü. To ensure a high quality product, the workload was distributed according to the core competencies and interests of each member.

7.1 Contributing and Functioning Effectively on the Team

- **Furkan Özek** focused primarily on the backend development of the system. He built the RESTful API using FastAPI. He implemented the JWT based authentication mechanisms and established the core routing endpoints that the mobile client consumes.
- **Aziz Üzümcü** contributed heavily to the frontend layer. He developed the cross platform mobile application using Flutter. He implemented the user interface, interactive map views, bottom sheets for station details and the overall user experience (UX) design.
- **Utku Yüksel** designed and developed the Occupancy Prediction and AI modules. He was responsible for processing the historical data, training the machine learning models to forecast station availability and writing the scripts that calculate temporal statistics and update predictions in the database.
- **Halis Vefa Türkyılmaz** handled the remaining critical infrastructure, primarily focusing on Data Engineering and Database Management. He designed the MongoDB architecture. He wrote the automated data acquisition scripts (cron jobs fetching EPDK data) and ensured the seamless integration of all subcomponents.

7.2 Contributing and Functioning Effectively on the Team

- **Furkan Özek** helped create a collaborative environment by containerizing the backend via Docker which allowed all team members to effortlessly run and test the server environment on their local machines without dealing with dependency conflicts.
- **Aziz Üzümcü** actively organized UI/UX feedback sessions which ensures that the design of the mobile app reflected the collective ideas of the team. He built an inclusive environment by considering everyone's input regarding the application's usability.
- **Utku Yüksel** maintained clear and transparent communication regarding the AI models. By documenting the inputs and outputs of the prediction algorithms, he made it extremely easy for the backend and

frontend developers to integrate his work into the main pipeline without confusion.

- **Halis Vefa Türkyılmaz** fostered a collaborative technical environment. By proactively managing the MongoDB infrastructure and resolving data flow bottlenecks between the external APIs, backend and AI modules, he ensured that all team members had constant access to reliable, real time data.

7.3 Contributing and Functioning Effectively on the Team

As a modern agile team, the leadership role was decentralized and shared among members based on the specific domain of the project:

- **Furkan Özek** assumed the leadership role in backend architecture and API security making the final decisions on how the server should handle requests and protect user data.
- **Aziz Üzümcü** took the lead role in mobile client development, guiding the technical decisions related to the Flutter framework, state management and visual design patterns.
- **Utku Yüksel** led the data science and predictive logic aspects of the project. He took the initiative to determine which machine learning approaches and statistical clustering methods would yield the most accurate predictions for EV drivers.
- **Halis Vefa Türkyılmaz** shared leadership by taking charge of the data infrastructure. He led the design of the data acquisition pipelines and the persistent data management strategy, ensuring the system was robust and scalable and also documentation.

8 Glossary

EV: Electric Vehicle

SoC: State of Charge

API: Application Programming Interface

UI: User Interface

ML: Machine Learning

AI: Artificial Intelligence

DB: Database

ETA: Estimated Time of Arrival

POI: Point of Interest

GIS: Geographic Information System

LSTM: Long Short-Term Memory (Neural Network Model)

9 References

[1] Enerji Piyasası Düzenleme Kurumu (EPDK), "Şarj Hizmeti Yönetmeliği," Resmi Gazete, Sayı: 31797, Apr. 2022.

[2] Tiangolo, "FastAPI Framework," [Online]. Available: https://fastapi.tiangolo.com/

[3] IEEE Computer Society, "IEEE Recommended Practice for Software Requirements Specifications," IEEE Std 830-1998, Oct. 1998.

[4] Object Management Group (OMG), "OMG Unified Modeling Language (OMG UML)," Version 2.5.1, Dec. 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1

[5] T.C. Resmi Gazete, "Kişisel Verilerin Korunması Kanunu (Kanun No. 6698)," Apr. 2016.

[6] Google Developers, "Flutter Documentation - Build apps for any screen," [Online]. Available: https://flutter.dev/docs

[7] M. Schneider, A. Stenger, and D. Goeke, "The electric vehicle routing problem with time windows and recharging stations," Transportation Science, vol. 48, no. 4, pp. 500–520, 2014.