SMART PARKING MANAGEMENT SYSTEM

SOFTWARE REQUIREMENTS DOCUMENT (SRS)

Project Name: Smart Parking Management System

Version: v1.0

Document No: ABYS-SRS-001

Prepared by: Mert Özkaya, Yusuf Talha Toğrul, Ceren Sivri, Sinem Kuru, Talha Berkay Eren, Umut Çobanoğlu, Nisa Of, Işıl Hocaoğlu, Ahmet Ege, Anıl Çetinkaya, Ece Öykü Ersavaş, Kerem Kuşcu, Eray Yılmaz, Emre Baliç, Yiğit Can Turan

Reviewers: Mert Özkaya, Alper Turunç

Approved by: Arda Burak Mamur

Institution / Company: Yeditepe University

Date: 05.08.2025

Privacy Level: Institutional

Document Status: Final

Number of Requirements: 226

REQUIREMENTS

Requirement [REQ-ID]

- **Requirement ID:** [Unique identifier, e.g. REQ-001]
- **Title:** [Short and clear title of the requirement]
- **Category:** [Functional, Non-Functional, Technical, etc.]
- **Description:** [Detailed description of the requirement, clearly stating what needs to be done]
- **Priority:** [High, Medium, Low]
- **Stakeholders:** [Stakeholders affected by or requesting this requirement]
- **Acceptance Criteria:**
    - [Criteria 1: Measurable criteria to verify that the requirement is met]
    - [Criterion 2: Another criterion that demonstrates successful completion of the requirement]
- **Dependencies:** [Other requirements or systems on which this requirement depends, e.g. REQ-002]
- **Constraints:** [Technical, time, or budget constraints affecting this requirement]
- **Assumptions:** [Assumptions made for the implementation of the requirement]
- **Risks:** [Potential risks associated with implementing this requirement]
- **Status:** [Draft, Approved, Implementing, Completed, etc.]
- **Responsible:** [Person or team who will implement this requirement]

Stakeholder List

Internal Stakeholders in the Software Development Role

These stakeholder groups are the core teams that do the direct technical development of the product, writing the code, creating the design, and setting up the infrastructure.

1. AI & Data Development Teams

They are the teams that develop the "smart" features of the application, make data-driven decisions, and personalize the user experience.

- **AI/Artificial Intelligence Teams:** They develop natural language processing (NLP) models that allow users to navigate the app with voice commands. They also create intelligent algorithms that analyze users' past parking habits to provide personalized parking suggestions.
- **Data Science Teams:** They analyze the big data collected by the system (parking times, density, etc.) to create predictive models, such as estimated wait times for occupied parking spaces. They provide historical predictions even when real-time data is unavailable.
- **Data Analytics Team:** Analyzes user behavior and system performance, provides reports on which features are used more, where problems occur, and guides product development decisions.

2. Frontend & Design Development Teams

These are the teams responsible for everything the user sees, touches, and interacts with on the app.

- **Frontend Team:** Develops web-based interfaces or the visual layer of mobile applications. They code what happens when a user clicks a button and how menus work.
- **Mobile Development Teams (UI/UX):** They ensure the smooth, fast, and stable operation of the app, particularly for iOS and Android devices. They integrate with maps, notifications, and other device features.
- **UI/UX Design Teams:** They design how the app will look and how easy and intuitive it will be to use. They simplify complex processes by planning colors, icons, menu layouts, and overall user flow.

3. Backend & Infrastructure Development Teams

They manage all the server, database, and logic operations that are invisible to the user but that make the application work.

- **Backend Teams:** They code the core business rules, such as how users create and log in, reserve parking spaces, make secure payments through the app, and implement penalties for overdue payments. They write the APIs that provide the data the frontend needs.
- **IoT Team / IoT Integration Team:** They collect data from physical sensors in parking lots (occupancy, EV charge status, etc.), process it, and convert it into a format the system can understand. This allows the app to display real-time availability.
- **Data Security Team:** Ensures that user data (personal information, payment cards, etc.) is stored securely, passwords are protected with the right methods, users can delete their data within the framework of the "right to be forgotten" and cyber attacks on the system are prevented.

Other Internal Stakeholders (Management, Operations, Support, etc.)

These stakeholder groups are critical players who guide technical teams, ensure the project achieves its goals, oversee quality, and support the end user.

- **Product Management:** Determines the "what" of the project. They identify and prioritize features to be developed by analyzing user needs and business goals.
- **Quality and Safety Management:**
    - **The QA (Quality Assurance) Team** systematically tests developed features to ensure they function correctly and for any errors. This ensures that the application reaches the user without any problems.
    - **Security Policy Makers:** Oversee compliance with legal regulations such as KVKK/GDPR and determine general data security strategies.
- **Operations and System Management:** Responsible for ensuring the application runs 24/7. They maintain servers, respond to system crashes, and monitor overall performance.
- **Business Units and Support:**
    - **Finance Team:** Checks the financial accuracy of payment and subscription systems and tracks income and expenses.
    - **Support Team:** Responds directly to problems experienced by users (e.g., "I cannot make payment", "My reservation is not visible") and provides solutions.

    - **Content Team:** Creates and keeps updated all informative texts within the application, such as the "Help and Frequently Asked Questions" section where users can find answers to their questions.

External Stakeholders

These stakeholder groups are individuals and institutions that are outside the project but are directly affected by the project or provide data/services to the project.

1. User Groups

They are the reason for the project's existence. Their feedback determines the future development direction of the application.

- **Drivers:** The app's primary target audience. All features are designed to simplify the process of finding and booking parking.
- **Disabled Users / Electric Vehicle Owners:** These are important user groups who have special needs, such as accessible parking or EV charging stations, and are testing the inclusiveness of the app.

2. Corporate Partners and Service Providers

They are critical business partners that provide services that the application cannot provide alone.

- **Parking Providers:** These are businesses that integrate their physical parking spaces and parking garages into this system. They provide information such as pricing, capacity, and ceiling height.
- **Map and Traffic API Providers: These are** map and traffic data services received from companies such as Google Maps and Yandex, which allow the application to display maps, draw routes and calculate estimated time of arrival based on real-time traffic.
- **Other Service Providers:** These are third-party companies that provide specialized services such as identity verification (OTP), payment infrastructures, and real-time weather data.

3. Regulatory and Public Institutions

They are the institutions that ensure that the project operates within a legal framework and sometimes provide data.

Municipality / Government: These are institutions that require cooperation, especially on matters such as roadside parking, general parking regulations, legal permits and sometimes disabled parking data.

List of Requirement Categories:

1. Functional Requirements

- **Definition** : Requirements define what the system must do. They describe the functions that users can perform with the system.

- **Example** :
    - In the smart parking management system, drivers can reserve parking spaces via mobile application (REQ-003).
    - Display of real-time parking availability information on LED signs (REQ-001).
- **Feature** : User-focused and defines a specific functionality.

2. Non-Functional Requirements

- **Definition** : Requirements that define how the system should work, that is, quality characteristics such as performance, security and usability.
- **Example** :
    - The central database provides a 99.9% availability rate (REQ-004).
    - Payment transactions are completed in less than 5 seconds (REQ-005).
- **Feature** : Contains criteria that affect the quality of the system and user experience.

3. Technical Requirements

- **Definition** : The technical infrastructure of the system includes requirements related to hardware, software or integrations.
- **Example** :
    - Using IoT-based sensors in parking lots and transmitting data to the central database within 10 seconds (REQ-002).
    - the payment system (REQ-005).
- **Feature** : Related to system architecture and technical limitations.

4. Job Requirements

- **Definition** : Requirements that support the project's business objectives and the strategic needs of the stakeholders.
- **Example** :
    - The city government needs to share parking lot occupancy information in real time to reduce traffic congestion.
    - Integration of contactless payment to increase revenues for parking operators.
- **Feature** : Include higher-level, business-oriented goals.

5. Legal and Regulatory Requirements

- **Definition** : Legal, regulatory or standards-related requirements that the system must comply with.
- **Example** :
    - Storing data encrypted in GDPR compliance (REQ-004).
    - Payment system compliance with PCI DSS standards (REQ-005).
- **Special feature** : Focuses on legal compliance and regulatory requirements.

6. User Interface (UI) and User Experience (UX) Requirements

- **Definition** : Requirements related to the interface and experience that facilitate users' interaction with the system.
- **Example** :
    - Mobile application showing available parking spaces on the map (REQ-003).
    - Users can view their payment history via the application (REQ-005).
- **Feature** : Focused on user-friendly design and interaction.

A) Driver

REQ-001

- **Requirement ID:** REQ-001
- **Title:** Driver's Ability to Register and Login to the Mobile Application
- **Category:** This requirement belongs to the functional, legal and regulatory requirement categories.
- **Description:** The system must require drivers to create an account and authenticate before they can use the platform. This requirement includes OTP verification of email and phone number during registration, followed by the creation of a secure session with a password. Upon successful login, the system must provide the user with a valid session token.
- **Priority:** High.
- **Stakeholders:** This requirement was requested or is impacted by Drivers, Authentication Service Providers, Mobile UI/UX Team, QA Team, and Security Policy Makers.
- **Acceptance Criteria:**
    - The system should prevent user account creation before a valid OTP verification is completed.
    - When the user enters incorrect credentials, the system should display a warning message.

    - When the user successfully logs in, the system must generate a session token.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-402 «deriveReqt» (Optional registration and login with third-party authentication)
    - REQ-448 «deriveReqt» (Registration and autofill of fields with third-party OAuth)
    - REQ-449 «deriveReqt» (Phone number uniqueness enforcement and authentication security)
    - REQ-710 «deriveReqt» (Acceptance of privacy policy and terms of service during registration process)
    - «satisfy» Authentication Module
    - «satisfy» User Database
    - «verify» Session management test case
- **Restrictions:** Due to security restrictions, this requirement must be implemented such that the user password is at least 8 characters long.
- **Assumptions:** For this requirement to apply, it is assumed that the user has entered valid contact information and has an active internet connection.
- **Risks:** Implementing this requirement could lead to malicious actors taking over user accounts via brute-force attacks due to the risk of weak password policies.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-002

- **Requirement ID:** REQ-002
- **Title:** Driver's Ability to Manage Profile and Vehicle Information
- **Category:** This requirement belongs to the functional, legal and regulatory requirement categories.
- **Description:** The system must enable drivers to manage their personal and vehicle information so they can personalize services and make reservations. This requirement includes editing personal information (name, surname, etc.) and vehicle details (license plate, model, EV support, etc.), as well as adding and deleting new vehicles. The system must ensure that changes are instantly synchronized.
- **Priority:** This requirement has been set to high priority.

- **Stakeholders:** This requirement was requested or is impacted by Drivers, Profile Management Product Owners, QA Team, Mobile UI/UX Team, and System Operations Team.
- **Acceptance Criteria:**
    - The system must ensure that the driver can save at least one vehicle to his profile.
    - When the user updates their profile information, the system should instantly synchronize the data.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «deriveReqt» REQ-001 (Driver Registration and Login to Mobile Application)
    - «deriveReqt» REQ-700 (Vehicle Photo Upload)
    - «deriveReqt» REQ-719 (Reservation Blocked with Incomplete Profile Information)
    - «satisfy» User Profile Module
    - «verify» Profile Update Test Scenario
- **Restrictions:** This requirement must be implemented in such a way that a user can add a maximum of 5 vehicles due to system capacity constraints.
- **Assumptions:** For this requirement to apply, it is assumed that the user has entered the vehicle data correctly.
- **Risks:** Implementing this requirement may lead to data synchronization errors and outdated information during profile update due to the risk of potential network issues.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-003

- **Requirement ID:** REQ-003
- **Title:** Real-Time Parking Availability Display
- **Category:** This requirement belongs to the functional and user interface (UI)/user experience (UX) requirement categories.
- **Description:** To enable drivers to instantly and accurately find parking spaces, the system must display available parking spaces on the map using data from IoT devices. This requirement includes displaying parking occupancy status with a delay of less than 30 seconds and using colored icons. This system should reduce users' parking search time.
- **Priority:** High

- **Stakeholders:** This requirement is requested or impacted by Drivers, IoT Data Providers, Map API Providers, Data Analytics Team, and QA Team.
- **Acceptance Criteria:**
    - The system should update available parking data on the map with a delay of less than 30 seconds.
    - Parking status (available/occupied) should be displayed on the map with color-coded icons.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «deriveReqt» REQ-400 (Access to Real-Time Location Data (GPS))
    - «deriveReqt» REQ-403 (Collecting and Managing Real-Time Occupancy Data from IoT Devices)
    - «deriveReqt» REQ-424 (Parking Space Status Control with IoT Integration)
    - «deriveReqt» REQ-501 (Receiving and Processing Real-time Parking Status Data from External IoT Sensors)
    - «satisfy» IoT Sensor Module
    - «satisfy» Map Module
    - «verify» Parking Area Imaging Tests
- **Restrictions:** Due to technical limitations, this requirement must be implemented in a way that it will only work if the user has an active internet connection and location permission.
- **Assumptions:** For the implementation of this requirement, it is assumed that IoT sensor data arrives seamlessly and accurately via the API.
- **Risks:** Implementing this requirement may lead to misleading information being presented to users (such as showing occupied space as empty) due to the risk of malfunctioning IoT sensors.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-004

- **Requirement ID:** REQ-004
- **Title:** Accessing Application Operations with Voice Command
- **Category:** This requirement belongs to the non-functional and UI/UX requirement categories.
- **Description:** The system must enable the driver to issue voice commands, particularly for hands-free operation while driving. This requirement includes using natural language processing (NLP) technology to perform actions such as

initiating a reservation or obtaining directions. The system must understand commands and generate audio/visual responses with 90% accuracy.

- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested or is impacted by the Drivers, NLP/Voice Assistant Team, UX Design Team, and QA Team.
- **Acceptance Criteria:**
    - The system must process defined voice commands with 90% accuracy.
    - When a voice command is given, the system should produce both audible and visual feedback.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - This requirement cannot be implemented without completing REQ-409 (Translation of Voice Commands into Structured Commands with NLP and LLM) with the «derivedReqt» relationship.
    - This requirement cannot be implemented without completing REQ-410 (Detecting Driver Intent from Voice Commands) with the «derivedReqt» relationship.
    - This requirement cannot be implemented without completing REQ-603 (Voice Command Recognition in Noisy Environments) with the «derivedReqt» relationship.
    - This requirement cannot be implemented without completing the NLP Module with the «satisfy» relationship.
    - This requirement cannot be implemented without completing the Voice Command Test Case with the «verify» relationship.
    - This requirement cannot be implemented without completing the Voice Command Interface Module with the «trace» relationship.
- **Restrictions:** Due to technical constraints, this requirement should be implemented in a way that only works for predefined instruction sets in the system.
- **Assumptions:** For this requirement to apply, it is assumed that the user has granted microphone permission on their device.
- **Risks:** Implementing this requirement may lead to unintended actions being triggered due to the risk of incorrect command recognition in noisy environments.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** AI and Frontend Teams are responsible for implementing this requirement.

REQ-005

- **Requirement ID:** REQ-005
- **Title:** Evaluating the Park Experience and Providing Feedback

- **Category:** This requirement belongs to the functional and UI/UX requirement categories.
- **Description:** To improve service quality, the system must enable drivers to leave feedback after completing parking sessions. This requirement includes assigning a 1-5 star rating to the parking area and allowing comments of up to 500 characters. The system must collect this data and provide it to parking area owners.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested or is affected by Drivers, Parking Lot Operators, Feedback Analysis Team, and QA Team.
- **Acceptance Criteria:**
    - Once the user completes the parking session, the system should automatically display the evaluation screen.
    - The system must allow the user to leave a comment with a maximum length of 500 characters.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «deriveReqt» REQ-048 (Parking Session Completion and Evaluation)
    - «deriveReqt» REQ-108 (Feedback and Complaints Display)
    - «satisfy» Feedback Module
    - «verify» Evaluation Function Test
- **Restrictions:** This requirement must be implemented to only work for successfully completed parking sessions due to business rules restrictions.
- **Assumptions:** For this requirement to apply, it is assumed that the user is using an up-to-date version of the application.
- **Risks:** Implementing this requirement may lead to misleading other users due to the risk of malicious users posting fake comments (spam).
- **Status:** The current status of this requirement is Draft.
- **Responsible:** The Frontend Team is responsible for the implementation of this requirement.

REQ-006

- **Requirement ID:** REQ-006
- **Title:** Displaying Accessible Parking Areas
- **Category:** This requirement belongs to the functional and user interface (UI)/user experience (UX) requirement categories.
- **Description:** To facilitate the process of finding parking for individuals with disabilities, the system must provide real-time information on the availability of designated parking spaces for these users. This requirement includes displaying

accessible spaces on the map with dedicated icons and filtering capabilities. This ensures an inclusive parking experience.

- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is impacted by Drivers, Disabled Users, Municipality, IoT Integration Team, and UI/UX Team.
- **Acceptance Criteria:**
    - The system should display disabled parking spaces on the map with special and distinctive symbols.
    - When the user filters the search results to show only accessible areas, the system should list only those areas on the map.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «deriveReqt» REQ-003 (Real-Time Parking Availability Display)
    - «deriveReqt» REQ-023 (Filtering Based on Driver Preferences)
    - «verify» Accessibility Filtering Test Case
    - «satisfy» IoT Sensor Module
- **Restrictions:** Due to technical limitations, this requirement must be implemented so that it only works for areas labeled as "accessible" by the parking area management.
- **Assumptions:** For the implementation of this requirement, it is assumed that the IoT infrastructure covers all parking areas.
- **Risks:** Implementing this requirement may lead to inconvenience to vulnerable users due to the risk of mislabeling parking area data.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-007

- **Requirement ID:** REQ-007
- **Title:** Ability to Record and Remind the Location of a Parked Vehicle
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system should provide the location where the vehicle was last parked so that drivers can easily find their vehicles, especially in large or unfamiliar parking spaces. This requirement involves automatically recording the location via GPS or manually by the user and displaying it again through a feature such as "Find My Car." This should prevent users from wasting time.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested or is impacted by Drivers, Mobile UX Team, and Location Service Providers.
- **Acceptance Criteria:**

    - The system must accurately record the parked location based on GPS data or manual input by the user.
    - When the user uses the "Find My Car" function, the system should display the saved location on the map.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «deriveReqt» REQ-400 (Access to Real-Time Location Data (GPS))
    - «satisfy» Location Services Module (GPS)
    - «verify» Location Record Test Scenario
    - «trace» Map Module
- **Restrictions:** This requirement must be implemented in enclosed areas where GPS accuracy may be reduced due to technical limitations.
- **Assumptions:** For this requirement to apply, it is assumed that the location permission is enabled on the user's device.
- **Risks:** Implementing this requirement may lead to incorrect location recording and misleading the user due to the risk of a weak GPS signal.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** The Frontend Team is responsible for the implementation of this requirement.

REQ-008

- **Requirement ID:** REQ-008
- **Title:** Receiving Navigation Directions to the Selected Parking Area
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system should trigger external navigation apps (such as Google Maps, Yandex, etc.) installed on drivers' devices to easily navigate to their chosen parking spot. This requirement involves opening the external app with the destination location automatically filled in when the user clicks the "Get Directions" button. This allows the system to maintain the user's familiar navigation experience.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is impacted by Drivers, Map API Providers, and the Mobile Application Team.
- **Acceptance Criteria:**
    - When the user clicks the "Get Directions" button, the system should launch the external navigation application, automatically filling in the destination location.
    - The navigation process should start with the map application set as default on the user's device.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «deriveReqt» REQ-003 (Real-Time Parking Availability Display)
    - «deriveReqt» REQ-406 (Real-Time Map-Based Navigation)
    - «deriveReqt» REQ-407 (Redirect to Third Party Navigation Applications)
    - «satisfy» External Map API
    - «verify» Navigation Guidance Test
- **Restrictions:** Due to technical limitations, this requirement must be implemented in a way that only works if the user has a compatible map application installed on their device.
- **Assumptions:** For this requirement to be implemented, it is assumed that the user has an active internet connection.
- **Risks:** Implementing this requirement may lead to redirect failure or crash of the external application due to the risk of an error occurring during the inter-application transition.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-009

- **Requirement ID:** REQ-009
- **Title:** Calculating Estimated Time of Arrival (ETA)
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must calculate the estimated time of arrival (ETA) at the selected parking area to help drivers better plan their trips. This requirement involves constantly updating the ETA using real-time traffic data. This means the system must provide the user with the most accurate and up-to-date route times.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is impacted by Drivers, Map and Traffic API Providers, and Backend Traffic Analysis Team.
- **Acceptance Criteria:**
    - The system must constantly keep the ETA data updated according to the current traffic density.
    - The estimated time of arrival must be presented to the user in a clear and understandable manner in minutes.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «deriveReqt» REQ-008 (Receiving Navigation Directions to the Selected Parking Area)
    - «deriveReqt» REQ-504 (Google Maps and Google Places API Integration)

    - «satisfy» External Traffic Data API Module
    - «verify» ETA Calculation Test
- **Restrictions:** Due to technical limitations, this requirement must be implemented in such a way that it only works if the user has an active internet connection.
- **Assumptions:** For this requirement to be implemented, it is assumed that the external API provides up-to-date and accurate traffic data.
- **Risks:** Implementing this requirement may lead to an inaccurate calculated arrival time and misdirection of the user due to the risk of incomplete data from the external traffic API service.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-010

- **Requirement ID:** REQ-010
- **Title:** Ability to Display Estimated Waiting Time for Occupied Parking Spaces
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** To help drivers make more informed decisions about waiting in occupied parking spaces, the system should provide an estimate of how long it will take for a space to become available. This requirement involves a calculation based on historical usage data. This should help drivers manage their waiting times.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested or is influenced by the Drivers, Data Science Team, and Backend Scheduling Module Team.
- **Acceptance Criteria:**
    - The system should calculate and visually display the estimated waiting time in minutes.
    - Estimated waiting time should only be displayed when sufficient historical usage data is available in the system.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-404 (Future Demand Forecast Based on Parking Occupancy Analysis)
    - «satisfy» Forecast Module
    - «satisfy» Park History Database
    - «verify» Waiting Time Estimation Test
- **Restrictions:** This requirement should be implemented in a way that will not work in new parking areas without sufficient historical data due to technical limitations.

- **Assumptions:** For this requirement to apply, it is assumed that the system stores historical parking data accurately.
- **Risks:** Implementing this requirement may lead to lower forecast accuracy and reduced user satisfaction due to the risk of misleading historical data.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-011

- **Requirement ID:** REQ-011
- **Title:** Receiving Notification Before Parking Time Ends
- **Category:** This requirement belongs to the functional and user interface (UI)/user experience (UX) requirement categories.
- **Description:** The system should notify drivers near the end of a parking period to prevent them from exceeding their parking time and incurring fines. This requirement includes sending a notification with action buttons to extend or terminate the session. This allows users to proactively manage their parking sessions.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by Drivers, Software Developer, System Administrator and Parking Space Providers.
- **Acceptance Criteria:**
    - The system should send a notification to the user 5 minutes before the end of the parking time.
    - The notification sent should include actionable options such as “Extend Session” or “Sign Out.”
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-020 (Active Park Session Extension)
    - «derivedReqt» REQ-210 (Notification to Parking Attendant in Case of Reservation Excess and Violations)
    - «derivedReqt» REQ-415 (Multi-Channel Notification Support)
    - «satisfy» Notification Module
    - «verify» Notification Sending Test
- **Restrictions:** Due to technical limitations, this requirement must be implemented in a way that only works if the user has notification services enabled on their device.
- **Assumptions:** For this requirement to apply, it is assumed that the user has consented to application notifications.

- **Risks:** Implementing this requirement may lead to the notification not being delivered due to the risk of network issues and the user being penalized for exceeding the deadline.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** The Frontend Team is responsible for the implementation of this requirement.

REQ-012

- **Requirement ID:** REQ-012
- **Title:** Viewing Park History
- **Category:** This requirement belongs to the functional and UI/UX requirement categories.
- **Description:** The system must list previous parking sessions so drivers can track their past spending and access previous parking details. This requirement requires each historical record to include details such as location, date, duration, and cost. This allows the system to provide users with a transparent breakdown of their spending.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by Drivers, Software Developer, System Administrator, and Data Scientist.
- **Acceptance Criteria:**
    - The system should enable the user to view all previous parking history in chronological order.
    - Each history record displayed must include complete location, date, duration, and fee paid.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-016 (Parking Space Reservation in Advance)
    - «derivedReqt» REQ-033 (Access to Weekly Usage and Expenditure Reports)
    - «satisfy» User Database
    - «verify» Parking History View Test
- **Restrictions:** This requirement must be implemented such that parking history data is retained for a maximum of 1 year due to regulatory restrictions.
- **Assumptions:** For this requirement to be implemented, it is assumed that the system stores all session records regularly and completely.
- **Risks:** Implementing this requirement may result in data loss or incomplete records in historical parking records due to the risk of database errors.
- **Status:** The current status of this requirement is Draft.

- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-013

- **Requirement ID:** REQ-013
- **Title:** Payments Made Through the Application
- **Category:** This requirement belongs to the functional and business requirement categories.
- **Description:** The system must accept payments through the mobile app so drivers can easily pay for parking without using cash. This requirement includes supporting multiple payment methods, such as credit cards or mobile wallets. The system must automatically terminate the session after a successful payment.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by Drivers, Parking Space Providers, Software Developer, System Administrator, Government and External System Integration.
- **Acceptance Criteria:**
    - The system must ensure that the user can make payments by adding at least one credit card or mobile wallet.
    - Once the payment is successful, the system should automatically terminate the relevant parking session.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «satisfy» REQ-500 (Third Party Payment Systems Integration)
    - «derivedReqt» REQ-701 (Defining Default Payment Method)
    - «derivedReqt» REQ-018 (Obtaining Pre-Reservation Total Fee and Payment Method Approval)
    - «verify» Payment Flow Test
- **Restrictions:** This requirement must be implemented as due to regional variations restriction, supported payment methods will vary depending on the country where the user is located.
- **Assumptions:** For this requirement to apply, it is assumed that the user has a valid payment method.
- **Risks:** Implementing this requirement may lead to failed transactions or financial losses due to the risk of an outage in the external payment API.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-014

- **Requirement ID:** REQ-014
- **Title:** Providing Discounted Parking Usage with the Subscription System
- **Category:** This requirement is functional and business belongs to the requirement categories.
- **Description:** The system must offer weekly, monthly, or annual subscription packages to provide cost savings to frequent drivers. This requirement requires users to select a subscription plan and pay for it to receive discounted parking. The system must track the duration of subscriptions and provide automatic renewal.
- **Priority:** The priority of this requirement is MEDIUM.
- **Stakeholders:** This requirement is requested or affected by Drivers, Parking Space Providers, Software Developer and System Administrator.
- **Acceptance Criteria:**
    - The system should enable the user to activate their subscription by selecting a subscription plan and making payment.
    - The system should track the duration of active subscriptions and automatically renew them on the expiration date.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-013 (Ability to Make Payments via Application)
    - «satisfy» Subscription Module
    - «verify» Subscription Renewal Test
- **Restrictions:** This requirement should only be applied to contracted parking providers due to business agreement restrictions.
- **Assumptions:** For the implementation of this requirement, it is assumed that parking space providers support the subscription model.
- **Risks:** Implementing this requirement may lead to the user being incorrectly charged or service being interrupted due to the risk of an error in the auto-renewal mechanism.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-015

- **Requirement ID:** REQ-015

- **Title:** Viewing Parking Fee and Availability Information
- **Category:** This requirement belongs to the functional and user interface (UI)/user experience (UX) requirement categories.
- **Description:** The system should display parking fees and availability to ensure drivers are informed about costs and occupancy before selecting a parking space. This requirement includes displaying hourly/daily rates and current occupancy for each space using colored icons. This should enable users to make informed decisions.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by Drivers, Parking Providers, Software Developer, Municipality, System Administrator, Sensor Provider, and Camera Provider.
- **Acceptance Criteria:**
    - The system should display hourly and daily fee information for each parking space.
    - The user should be able to see the availability of a parking space through color-coded icons or clear text descriptions.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-107 (Dynamic Pricing Request Generation)
    - «derivedReqt» REQ-720 (Display of Parking Fee Information, Estimated Costs and Discounts)
    - «satisfy» Pricing Module
    - «satisfy» IoT Sensor Module
    - «trace» Map Module
- **Restrictions:** Due to data limitations, this requirement must be implemented to only display pricing information for parking providers that have entered this data into the system.
- **Assumptions:** For this requirement to be implemented, it is assumed that parking providers enter accurate and up-to-date pricing information.
- **Risks:** Implementing this requirement may lead to misinformation for the user due to the risk of the parking provider entering outdated or incomplete pricing information.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-016

- **Requirement ID:** REQ-016

- **Title:** Parking Space Reservation in Advance
- **Category:** This requirement belongs to the functional and business requirement category.
- **Description:** The system must enable drivers to reserve a suitable parking space before arriving at the parking area. This requirement includes the ability to reserve a reservation by date, time, and duration, and to guarantee parking availability upon confirmation.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by Drivers, Parking Space Providers and Backend Team.
- **Acceptance Criteria:**
    - The system should display parking spaces available for the selected date and time range.
    - The user should receive a confirmation message when they successfully create a reservation.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-308 (Completion of Reservation Verification Service)
    - «derivedReqt» REQ-015 (Provision of Parking Area Information and Features)
    - «satisfy» Reservation Module
    - «verify» REQ-206 (Parking Attendant's Ability to View Made Reservations)
    - «derivedReqt» REQ-013 (Ability to Make Payments via Application)
    - «derivedReqt» REQ-104 (Management of Parking Reservations)
    - «derivedReqt» REQ-028 (Avoiding Conflicting Reservations)
    - «derivedReqt» REQ-719 (Reservation Blocked with Incomplete Profile Information)
- **Restrictions:** Parking spaces that can be reserved must be predefined in the system.
- **Assumptions:** The system checks real-time occupancy and conflicts at the time of booking.
- **Risks:** An error in the reservation system may lead to double bookings or user dissatisfaction as parking space cannot be guaranteed.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-017

- **Requirement ID:** REQ-017

- **Title:** Ability to Enter Start and End Information for Reservations
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system should allow drivers to select start and end times during their reservation so they can clearly specify their parking needs. This requirement involves the user selecting a time and completing the process by selecting one of the vehicles registered to their profile. This ensures the system is booking the correct time and vehicle.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by Drivers, Software Developer and System Administrator.
- **Acceptance Criteria:**
    - The system should allow the user to select start and end times on an hourly basis.
    - The booking screen should show a list of vehicles registered to the user's profile and the user should make a selection from that list.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-016 (Parking Space Reservation in Advance)
    - «derivedReqt» REQ-117 (Definition of Reservation Period, Discount and Cancellation Rules)
    - «satisfy» User Profile Module
    - «verify» Timing and Vehicle Selection Test
- **Restrictions:** This requirement must be implemented in such a way that a reservation cannot be made for a past date or time due to business rules restrictions.
- **Assumptions:** For this requirement to apply, it is assumed that the user has at least one registered agent in their profile.
- **Risks:** Implementing this requirement may lead to the user booking an undesirable time and falling victim to the risk of selecting the wrong time zone (AM/PM).
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-018

- **Requirement ID:** REQ-018
- **Title:** Obtaining Pre-Reservation Total Fee and Payment Method Approval
- **Category:** This requirement belongs to the functional requirement category.

- **Explanation:** Before a user can complete their reservation, the system must clearly display the total price and selected payment method and obtain the user's approval. The reservation cannot be completed without confirmation.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is influenced by the Drivers, Backend Team, UI/UX Design Team, and Finance Team.
- **Acceptance Criteria:**
    - The system should clearly display the total cost and the selected payment method to the user during the booking process.
    - Explicit consent must be obtained from the user with the option to “confirm” or “cancel”.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-013 (Ability to Make Payments via Application)
    - «derivedReqt» REQ-016 (Parking Space Reservation in Advance)
    - «derivedReqt» REQ-720 (Display of Parking Fee Information, Estimated Costs and Discounts)
    - «satisfy» Pricing Module
    - «satisfy» External Payment API Module
    - «verify» Fee Calculation and Approval Test
- **Restrictions:** Transactions can only be made with payment methods defined in the system.
- **Assumptions:** All fee and payment data is current.
- **Risks:** The user may raise an objection after the transaction due to incorrectly displayed fee or payment method.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** The Frontend Team is responsible for the implementation of this requirement.

REQ-019

- **Requirement ID:** REQ-019
- **Title:** Reservation Cancellation
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system should allow drivers to cancel their parking reservations before their parking session begins, providing flexibility in case their plans change. This requirement includes tracking cancellations and applying a fee deduction based on specific rules (e.g., for cancellations near the last

minute). This allows the system to offer both flexibility for the user and manage potential revenue loss for the parking provider.

- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by Drivers, Software Developer, System Administrator and Parking Space Providers.
- **Acceptance Criteria:**
    - The system must allow the user to cancel at least 30 minutes before the reservation start time.
    - When a user cancels a booking, it should be marked as "cancelled" in the user's list of past transactions.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-716 (Reservation Cancellation Restrictions and Justification Collection Process)
    - «derivedReqt» REQ-215 (Parking Attendant Override of Reservation)
    - «satisfy» Reservation Module
    - «verify» Cancellation Flow Test
- **Restrictions:** This requirement must be implemented in such a way that transactions cannot be processed outside of the parking provider's designated cancellation deadline due to business rules restrictions.
- **Assumptions:** For this requirement to apply, it is assumed that the user has a valid reason to cancel their reservation.
- **Risks:** Implementing this requirement may lead to some users clogging up the system with constant last-minute cancellations due to the risk of no anti-abuse mechanism.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-020

- **Requirement ID:** REQ-020
- **Title:** Extension of Active Parking Session
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system should allow drivers to extend their existing parking session by adding additional time without ending the active parking session. The new total charge and expiration time should be displayed during the extension process.
- **Priority:** This requirement has been set to high priority.

- **Stakeholders:** This requirement was requested or is influenced by the Drivers, Backend Team, UI/UX Design Team, and Finance Team.
- **Acceptance Criteria:**
    - The user must be able to extend the active parking session via the app in accordance with the current rules.
    - After the extension, the new period and total fee must be notified to the user.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-013 (Ability to Make Payments via Application)
    - «derivedReqt» REQ-011 (Notification Receiving Before Parking Time Expires)
    - «satisfy» Reservation Module
    - «verify» Extension Flow Test
- **Restrictions:** Applicable only in areas where there is an active parking session and this is allowed in the system.
- **Assumptions:** Active session data is current in the system.
- **Risks:** Problems with payment systems during the extension process may result in the session being extended incorrectly.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Backend and Frontend Teams are responsible for the implementation of this requirement.

REQ-021

- **Requirement ID:** REQ-021
- **Title:** Viewing Reservation Details
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must allow drivers to view reservation details so they can easily track upcoming or past parking activities. This requirement includes making basic information such as location, date, vehicle, and fee accessible through the app for each reservation. This allows the system to provide users with complete transparency and control.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by Drivers, Software Developer and System Administrator.
- **Acceptance Criteria:**
    - The system should enable the user to view all their reservations in separate tabs as "future" and "past".

    - The detail screen of each reservation must include complete parking area, date, time, vehicle and fee information.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-012 (Viewing Park History)
    - «derivedReqt» REQ-711 (Displaying Selected Parking Details)
    - «satisfy» Reservation Database
    - «verify» Reservation Details Test
- **Restrictions:** This requirement must be implemented in a way that does not reveal data from a deleted user account due to data privacy restrictions.
- **Assumptions:** For this requirement to be implemented, it is assumed that all reservations are successfully stored in the system.
- **Risks:** Implementing this requirement can lead to data sync issues and old reservations appearing incomplete due to the risk of the device being offline.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Software Developer and Mobile UI Team are responsible for the implementation of this requirement.

REQ-022

- **Requirement ID:** REQ-022
- **Title:** Viewing Available Parking Spaces on the Map
- **Category:** This requirement belongs to the functional and user interface (UI)/user experience (UX) requirement categories.
- **Description:** The system must instantly display available parking spaces nearby on the mobile app map so drivers can quickly assess parking options in their area. This requirement includes automatically opening the map based on the user's location and displaying availability with high accuracy. This allows users to efficiently find parking.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by Drivers, Software Developer, IoT Data Providers, Parking Space Providers, and System Administrator.
- **Acceptance Criteria:**
    - The system should automatically open the map screen centered on the user's current location.
    - The availability of parking spaces should be displayed in real time with 80% accuracy.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:

    - «deriveReqt» REQ-003 (Real-Time Parking Availability Display)
    - «satisfy» Map API Module
    - «satisfy» IoT Sensor Module
    - «verify» Map and Sensor Integration Test
- **Restrictions:** This requirement must be implemented in such a way that, due to technical limitations, data will not be updated in cases of insufficient internet connectivity.
- **Assumptions:** For this requirement to apply, it is assumed that the user has granted location permission on their device.
- **Risks:** Implementing this requirement could lead to an available spot being falsely shown as occupied due to the risk of a malfunction in IoT sensors.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-023

- **Requirement ID:** REQ-023
- **Title:** Filtering Based on Driver Preferences
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must allow drivers to filter parking results based on specific preferences so they can find parking spaces that best suit their needs. This requirement includes the ability to narrow search results based on various criteria, such as indoor parking, EV charging station availability, and valet service. This allows the system to provide a personalized search experience.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by Drivers, Software Developer, Parking Space Providers and System Administrator.
- **Acceptance Criteria:**
    - The system should provide a filtering screen where users can select and save parking preferences.
    - When filtering is applied, the system should display only parking areas on the map that match the selected criteria.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-002 (Driver Ability to Manage Profile and Vehicle Information)
    - «derivedReqt» REQ-102 (Defining and Adding Parking Areas and Parking Spots)
    - «satisfy» Parking Database

    - «verify» Filtering Function Test
- **Restrictions:** This requirement must be implemented in such a way that, due to data limitations, not every parking area will support all filtering criteria.
- **Assumptions:** For this requirement to be implemented, it is assumed that parking providers have correctly defined all the features of their areas into the system.
- **Risks:** Implementing this requirement may lead to the user seeing incorrectly filtered results due to the risk of missing parking area attribute data.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-024

- **Requirement ID:** REQ-024
- **Title:** Ability to Report Problems Regarding the Parking Area
- **Category:** This requirement belongs to the functional and user interface (UI)/user experience (UX) requirement categories.
- **Description:** The system should provide notifications when a parking problem occurs, allowing drivers to report their problems to the relevant authorities. This requirement includes sending a description of the problem along with photographic evidence, and allowing users to track the status of the notification. The system should also provide a feedback mechanism to improve service quality.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by Drivers, Software Developer, Operations Support Team, and System Administrator.
- **Acceptance Criteria:**
    - The system should enable users to create a complaint form via the "Report a Problem" tab.
    - Submitted problem notifications should be trackable by the user with statuses such as "open", "resolved" or "closed".
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-108 (Feedback and Complaints Display)
    - «satisfy» Feedback Module
    - «satisfy» Media Upload Module
    - «verify» Issue Report Flow Test
- **Restrictions:** Due to technical limitations, this requirement must be implemented with a maximum photo upload size of 3MB.

- **Assumptions:** For this requirement to be implemented, it is assumed that the user has an active internet connection.
- **Risks:** Implementing this requirement may unnecessarily increase system load due to the risk of malicious users sending misleading or spam notifications.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-025

- **Requirement ID:** REQ-025
- **Title:** Saving and Deleting Favorite Parking Areas
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system should allow drivers to save frequently used parking spaces to a "favorite" list so they can quickly access them. This requirement includes the ability for users to add and delete spaces from this list and to quickly access their favorite spaces from the home screen. This allows the system to personalize and streamline the user experience.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by Drivers, Software Developer and System Administrator.
- **Acceptance Criteria:**
    - The system should list favorited parking areas in a quick access section on the home screen.
    - The user should be able to easily edit (add/delete) the favorites list from the application interface.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-001 (Driver Registration and Login to Mobile Application)
    - «derivedReqt» REQ-045 (Receiving Availability Notifications for Favorite/Past Parking Spaces)
    - «satisfy» User Database
    - «verify» Favorite Area Test
- **Restrictions:** Due to system performance constraints, this requirement must be implemented such that a user can add a maximum of 10 favorite areas.
- **Assumptions:** For this requirement to apply, it is assumed that the user's account is active and logged in.

- **Risks:** Implementing this requirement may lead to loss of the favorites list and user dissatisfaction due to the risk of a synchronization error during device switching.
- **Status:** The current status of this requirement is Draft.
- **Responsible for:** Frontend implementation of this requirement The team is responsible.

REQ-026

- **Requirement ID:** REQ-026
- **Title:** Providing AI-Based Personalized Parking Recommendations
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** To provide drivers with a proactive and intelligent experience, the system should provide personalized parking suggestions based on the user's past parking habits. This requirement includes generating suggestions based on the user's frequent locations and behavioral patterns (e.g., visiting a cafe after a restaurant). This allows the system to provide user-specific and context-appropriate suggestions.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by Drivers, Data Scientist, Software Developer, and System Administrator.
- **Acceptance Criteria:**
    - The system should generate at least one personalized parking recommendation per week based on the user's historical data.
    - The generated recommendations should be displayed in a prominent area on the home page and feedback should be collected from the user.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-412 (Recommendation Engine with Driver History, Current Context and Community Data)
    - «derivedReqt» REQ-437 (Smart Post-Parking Activity Recommendations with Semantic Place Recognition)
    - «satisfy» Recommendation Engine Module
    - «satisfy» User History Database
    - «verify» Recommendation Test
- **Restrictions:** This requirement must be implemented in a way that cannot generate recommendations for new users without sufficient usage history due to the restriction of insufficient data.

- **Assumptions:** For this requirement to be implemented, it is assumed that the user's data sharing permissions are active and that there is sufficient historical data to generate recommendations.
- **Risks:** Implementing this requirement may lead to irrelevant parking suggestions and reduced recommendation quality due to the risk that the recommendation algorithm will misinterpret the user's routine.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** AI and Backend Teams are responsible for the implementation of this requirement.

REQ-027

- **Requirement ID:** REQ-027
- **Title:** Re-booking from Park History
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** To expedite the booking process for frequently visited locations, the system should allow drivers to rebook with a single click from the parking history page. This requirement allows the user to quickly start a new reservation for the same parking space by selecting a previous session. This should save the user time.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by Drivers, Software Developer and System Administrator.
- **Acceptance Criteria:**
    - When the user selects a session in the parking history, the system should provide an option to quick book the same space.
    - When quick booking is initiated, the system should prompt the user to enter a new time slot.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-012 (Viewing Park History)
    - «derivedReqt» REQ-016 (Parking Space Reservation in Advance)
    - «satisfy» Historical Booking Database
    - «verify» Quick Booking Test
- **Restrictions:** Due to business rules restrictions, this requirement must be implemented in a way that only works if the previously parked space is still active and bookable in the system.
- **Assumptions:** For this requirement to apply, it is assumed that the user's account has not been deleted.

- **Risks:** Implementing this requirement may lead to attempts to create a reservation for a parking space that is no longer in service due to the risk of not performing the necessary checks.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-028

- **Requirement ID:** REQ-028
- **Title:** Preventing Conflicting Reservations
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** To ensure fair resource allocation and prevent operational errors, the system must prevent a driver from having multiple active reservations within the same timeframe. This requirement includes checking a user's existing reservations when creating a new reservation. This prevents a user from booking multiple parking spaces simultaneously.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by the Drivers, Software Developer and Backend Team.
- **Acceptance Criteria:**
    - When the user tries to create a new reservation, the system must check if the user has another active reservation in that time slot.
    - When a conflict is detected, the system should display a warning message to the user and block the new booking process.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-016 (Parking Space Reservation in Advance)
    - «satisfy» Reservation Database
    - «verify» Clash Check Test
- **Restrictions:** This requirement must be implemented under the assumption that a user can only book one vehicle at a time due to business rules restrictions.
- **Assumptions:** For this requirement to be implemented, it is assumed that the reservation times are entered correctly by the user.
- **Risks:** Implementing this requirement may lead to duplicate bookings in the system and operational issues due to the risk of a logic error in the conflict control mechanism.
- **Status:** The current status of this requirement is Draft.
- **Responsible for:** Frontend implementation of this requirement and Backend Teams are responsible.

REQ-029

- **Requirement ID:** REQ-029
- **Title:** Real-Time Notification When a Reservation Is Invalid
- **Category:** This requirement belongs to the functional and user interface (UI)/user experience (UX) requirement categories.
- **Description:** To prevent driver inconvenience, the system must notify the user when a reservation is invalidated, such as due to a system error or cancellation by the parking area. This requirement includes sending a real-time notification explaining the cancellation reason. This prevents the user from driving to the parking area with an invalid reservation.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is affected by the Drivers, Software Developer, and Operations Team.
- **Acceptance Criteria:**
    - When a reservation is voided, the system should display an instant push notification to the user.
    - The notification sent must explain in detail why the reservation is cancelled.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-019 (Reservation Cancellation)
    - «derivedReqt» REQ-215 (Parking Attendant Override of Reservation)
    - «derivedReqt» REQ-415 (Multi-Channel Notification Support)
    - «satisfy» Notification Module
    - «verify» Notification Flow Test
- **Restrictions:** This requirement must be implemented in such a way that, due to technical limitations, the notification cannot be delivered in environments where the user does not have an internet connection.
- **Assumptions:** For this requirement to apply, it is assumed that the user has consented to application notifications.
- **Risks:** Implementing this requirement may lead to the risk of a failure in the notification system, leading to the user going to the parking area and experiencing inconvenience.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-030

- **Requirement ID:** REQ-030
- **Title:** Displaying Electric Vehicle (EV) Charging Station Status
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** To enable EV owners to meet their charging needs, the system must display the availability of EV charging stations in a parking area and their current occupancy status. This requirement includes identifying EV-supported parking spaces on the map and providing real-time charging station availability information. This system should facilitate charging planning for EV users.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by Drivers, Electric Vehicle Owners, Parking Space Providers, IoT Provider and Software Developer.
- **Acceptance Criteria:**
    - The system should distinguish parking spaces with EV charging stations with a special icon on the map.
    - The availability (empty/occupied) information of a charging station in a parking area should be displayed instantly.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-420 (Prioritization of Parking Spaces Suitable for Electric Vehicles)
    - «derivedReqt» REQ-112 (EV Charging Station Identification and Capacity Determination in Parking Areas)
    - «satisfy» IoT Charging Station Data Module
    - «satisfy» Map Module
    - «verify» Charge Compliance Test
- **Restrictions:** Due to technical limitations, this requirement should only be implemented to operate in parking areas with charging infrastructure and IoT integration.
- **Assumptions:** For the implementation of this requirement, it is assumed that charging station systems are integrated with IoT and provide data.
- **Risks:** Implementing this requirement could lead to the user being inconvenienced by encountering a full station at the location where they go to charge, due to the risk of incorrect availability information from the charging station.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-031

- **Requirement ID:** REQ-031
- **Title:** Providing Information on EV Charging Type and Power
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must provide technical details of charging stations in parking areas so that electric vehicle owners can select charging points suitable for their vehicles. This requirement includes displaying information such as charging station type (AC/DC) and power capacity (kW) on the parking area detail screen. The system should allow drivers to check vehicle compatibility.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by Drivers, Electric Vehicle Owners, Parking Space Providers, IoT Provider and Software Developer.
- **Acceptance Criteria:**
    - The system must display the type of a charging point (e.g. Type2, CHAdeMO) and its power rating (kW) on the parking area detail screen.
    - Charge type and power information must be clearly and visibly displayed on the parking area detail screen.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derive» REQ-030 (Electric Vehicle (EV) Charging Station Status Display)
    - «derivedReqt» REQ-112 (EV Charging Station Identification and Capacity Determination in Parking Areas)
    - «satisfy» Charger Database
    - «verify» Charging Information Test
- **Restrictions:** Due to data restrictions, this requirement must be implemented in a way that only works if parking providers enter this information into the system.
- **Assumptions:** For the implementation of this requirement, it is assumed that the technical data of EV charging stations are received from the providers in an up-to-date manner.
- **Risks:** Implementing this requirement could lead to the user's vehicle being directed to an incompatible charging point due to the risk that the charging type information entered by the provider is outdated or incomplete.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-032

- **Requirement ID:** REQ-032
- **Title:** Providing Predictive Recommendations When Real-Time Data Is Not Available
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** To ensure service continuity even when real-time sensor data is unavailable, the system must provide parking suggestions based on historical usage data. This requirement includes clearly informing the user that the suggestion is based on predictions. Thus, the system must provide an alternative even in the event of a data outage.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by Drivers, Data Scientist, and Software Developer.
- **Acceptance Criteria:**
    - When real-time occupancy data is not available, the system should provide an estimated parking recommendation based on historical usage data.
    - When a predictive recommendation is presented, the system should display a clear warning in the interface, such as "Live data is not available, this is a prediction."
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-413 (Working with Partial Data and Reservation-Free Parking Space Recommendation)
    - «derivedReqt» REQ-623 (Service Continuity During Partial Data Inaccessibility)
    - «satisfy» Recommendation Engine Module
    - «satisfy» User History Database
    - «verify» Suggestion Test
- **Restrictions:** Due to data limitations, this requirement should be implemented to make predictive recommendations only in locations with sufficient historical data.
- **Assumptions:** To implement this requirement, it is assumed that the system is capable of analyzing past parking traffic.
- **Risks:** Implementing this requirement may lead to user dissatisfaction due to the risk that the parking space may be full at the time the predictive recommendation is made.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** AI and Backend Teams are responsible for the implementation of this requirement.

REQ-033

- **Requirement ID:** REQ-033
- **Title:** Access to Weekly Usage and Expenditure Reports
- **Category:** This requirement belongs to the functional and business requirement categories.
- **Description:** The system should provide weekly usage reports to help drivers understand their parking habits and spending. This requirement includes providing users with a report that includes the most frequently visited areas, average parking times, and total spending. This provides users with a personalized analytics tool.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by the Drivers, Software Developer, and Data Scientist.
- **Acceptance Criteria:**
    - The system should automatically generate a usage and spending report each week.
    - The generated report should include ranking and summary information based on expenditure, duration and region.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-012 (Viewing Park History)
    - «derivedReqt» REQ-106 (Viewing Payment History and Defining Campaigns)
    - «satisfy» Reporting Module
    - «satisfy» User Parking History Database
    - The «verify» Reporting Test cannot be applied until the Reporting Test with the «verify» relationship is completed.
- **Restrictions:** Due to performance constraints, this requirement must be implemented such that reporting data is only processed for the last 30 days of activities.
- **Assumptions:** For this requirement to be implemented, it is assumed that the user has sufficient historical data in the system to generate a report.
- **Risks:** Implementing this requirement may lead to reports containing inaccurate or incomplete data due to the risk that some parking sessions may not be recorded in the system.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-034

- **Requirement ID:** REQ-034
- **Title:** Permanent Deletion of User Profile and Data
- **Category:** This requirement belongs to the functional, legal and regulatory requirement categories.
- **Description:** To ensure users can exercise their data privacy rights, the system must enable them to permanently delete their profiles and all associated data within the application. This requirement includes ensuring that the deletion request is irreversible and that the process complies with legal regulations (KVKK/GDPR). This allows users to exercise their "right to be forgotten."
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by the Drivers, Software Developer, Data Security Team, and System Administrator.
- **Acceptance Criteria:**
    - When the user gives the "Delete My Account" command and subsequent confirmation, the system should permanently delete all personal data belonging to the user.
    - The system should provide the user with at least one warning and a double-confirmation mechanism before this irreversible action.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-425 (Anonymization of User Data on Account Deletion)
    - «derivedReqt» REQ-441 (KVKK & GDPR Compliant Personal Data Processing and Storage)
    - «satisfy» User Management Module
    - «satisfy» Data Deletion Module
    - «verify» Account Deletion Flow Test
- **Restrictions:** This requirement must be implemented in a way that does not provide an undo option within 30 days of deletion due to regulatory restrictions.
- **Assumptions:** For this requirement to be implemented, it is assumed that GDPR/KVKK compliance is fully ensured by the system.
- **Risks:** Implementing this requirement may lead to the user permanently losing all their historical data and settings due to the risk of inadvertently deleting their account.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Backend Team and Data Security Team are responsible for the implementation of this requirement.

REQ-035

- **Requirement ID:** REQ-035
- **Title:** Getting Post-Park Activity Suggestions
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** To provide drivers with a value-added experience in the area where they park, the system should offer nearby activity or venue recommendations after parking. This requirement includes personalized recommendations for restaurants, cafes, and shopping malls based on the user's preferences or past behavior. This allows the system to provide users with not only a parking experience but also a city guide experience.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested or is impacted by the Drivers, Software Developer, AI Team, and Mobile UI Team.
- **Acceptance Criteria:**
    - The system should suggest places such as the nearest restaurants, cafes and shopping malls based on the user's parking location.
    - Recommendations should be personalized based on the user's weekly usage habits.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-437 (Smart Post-Parking Activity Recommendations with Semantic Place Recognition)
    - «derivedReqt» REQ-507 (Public Transport Route and Time Integration)
    - «satisfy» External Place API (Google Places etc.)
    - «satisfy» Recommendation Engine Module
    - «verify» Activity Suggestion Test
- **Restrictions:** Due to technical limitations, this requirement must be implemented to only offer recommendations when the user has an internet connection and location permissions.
- **Assumptions:** For this requirement to apply, it is assumed that the user has consented to suggestion notifications.
- **Risks:** Implementing this requirement could lead to a recommended location being closed and user dissatisfaction due to the risk of data from the external API being out of date.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** AI and Frontend Teams are responsible for implementing this requirement.

REQ-036

- **Requirement ID:** REQ-036
- **Title:** Accessing Help and Frequently Asked Questions
- **Category:** This requirement belongs to the non-functional/usability requirement category.
- **Description:** The system must provide a help center and frequently asked questions (FAQ) section so users can quickly find answers to questions about using the application. This requirement includes easily accessible information about basic operations and features. This allows users to self-support and reduces customer service overhead.
- **Priority:** This requirement has a low priority.
- **Stakeholders:** This requirement was requested or influenced by the Drivers, Software Developer, UX/UI Team, and Content Team.
- **Acceptance Criteria:**
    - The system must ensure that users can access the "Help Center" screen from the main menu of the application.
    - The help center should include a Frequently Asked Questions (FAQ) section with at least 10 different topics.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-604 (Dynamic In-App Help Based on User Behavior)
    - «satisfy» Content Management System (CMS)
    - «verify» Help Screen Test
    - «trace» FAQ Database
- **Restrictions:** This requirement must be implemented in a way that does not offer offline access due to technical limitations.
- **Assumptions:** It is assumed that the FAQ contents are regularly updated to implement this requirement.
- **Risks:** Implementing this requirement may lead to dissatisfaction and the user not finding the solution they are looking for due to the risk of the help content being inadequate or outdated.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** The Frontend Team is responsible for the implementation of this requirement.

REQ-037

- **Requirement ID:** REQ-037

- **Title:** Penalty Applicability in Case of Time Outages
- **Category:** This requirement belongs to the functional and business requirement categories.
- **Description:** To ensure fair use of parking spaces and prevent revenue loss, the system should implement a penalty mechanism for drivers who remain in the space after the expiration of the parking period. This requirement includes sending a warning before the expiration date and applying a penalty or surcharge in accordance with the rules set by the parking provider in case of overstay. This system should deter violations of the rules.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by Drivers, Providers, Software Developer, Operations Team, and Financial Rules Team.
- **Acceptance Criteria:**
    - The system should send a warning notification to the user 5 minutes before the end of the parking time.
    - In case of overtime, the system should automatically apply the penalty or surcharge policy defined by the parking provider.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-011 (Notification Receiving Before Parking Time Expires)
    - «derivedReqt» REQ-013 (Ability to Make Payments via Application)
    - «derivedReqt» REQ-721 (Warning and Provider-Defined Penalty Application in Case of No Show)
    - «satisfy» Timer Module
    - «satisfy» Payment Module
    - «verify» Criminal Practice Test
- **Restrictions:** This requirement must be enforced due to business rules restrictions, only imposing penalties in areas with the approval and agreement of the parking provider.
- **Assumptions:** It is assumed that if the user does not extend the time period for the implementation of this requirement, a penalty will be applied.
- **Risks:** Implementing this requirement may lead to incorrect user fines and customer loss due to the risk of a bug in the scheduler module.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-038

- **Requirement ID:** REQ-038
- **Title:** Parking Operations with Voice Assistant
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** To enhance driving safety, the system must enable drivers to perform basic parking operations hands-free using a voice assistant. This requirement includes the ability to start, stop, and steer using voice commands. This allows the driver to use the app without being distracted from the road.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested or is impacted by the Drivers, NLP Team, Software Developer, and Mobile UI Team.
- **Acceptance Criteria:**
    - The system should enable the user to perform basic voice commands such as “start parking” or “start navigation.”
    - The accuracy of perceiving and processing voice commands must be at least 90%.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derive» REQ-004 (Accessing Application Operations with Voice Command)
    - «derivedReqt» REQ-606 (Providing Hands-Free Experience with Natural Voice Commands)
    - «satisfy» Voice Recognition Module
    - «satisfy» NLP Module
    - «verify» Voice Command Test
- **Limitations:** This requirement must be implemented due to technical limitations where performance may be degraded in noisy in-vehicle environments.
- **Assumptions:** For this requirement to apply, it is assumed that the microphone permission is enabled on the user's device.
- **Risks:** Implementing this requirement may lead to an unintended action being triggered or security vulnerabilities due to the risk of incorrect recognition of the voice command.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** AI and Frontend Teams are responsible for implementing this requirement.

REQ-039

- **Requirement ID:** REQ-039
- **Title:** Customizing Notification Type Preferences

- **Category:** This requirement belongs to the functional and user interface (UI)/user experience (UX) requirement categories.
- **Description:** To respect users' communication preferences, the system must allow users to choose which channels they want to receive notifications from the system. This requirement includes the ability to choose between channels such as push notifications, email, or SMS, and manage their preferences. This should increase user satisfaction and prevent unnecessary notifications.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by the Drivers, Software Developer and Backend Team.
- **Acceptance Criteria:**
    - The system should enable users to select notification preferences (push, email, SMS) from the application settings tab.
    - The system should only deliver notifications through channels that align with the user's chosen preferences.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-002 (Driver Ability to Manage Profile and Vehicle Information)
    - «derivedReqt» REQ-415 (Multi-Channel Notification Support)
    - «satisfy» Notification Module
    - «verify» Notification Preference Test
- **Restrictions:** This requirement must be implemented due to technical limitations, where supported notification types may vary depending on the user's location.
- **Assumptions:** For this requirement to be implemented, it is assumed that the user keeps at least one communication channel open for notifications.
- **Risks:** Implementing this requirement may lead to critical information not reaching the user, such as expiration, due to the risk of the user closing all notification channels.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-040

- **Requirement ID:** REQ-040
- **Title:** Viewing the Facilities Around the Park Point
- **Category:** This requirement belongs to the functional requirement category.

- **Description:** The system should display key amenities (restaurants, ATMs, restrooms, etc.) around a selected parking area to provide drivers with additional information about the area they are parking in. This requirement includes displaying these amenities as icons on the application map. This should help the system facilitate travel planning.
- **Priority:** This requirement has a low priority.
- **Stakeholders:** This requirement is requested or affected by Drivers, Map API Providers, and Software Developer.
- **Acceptance Criteria:**
    - When the user clicks on a selected parking area, the system should display nearby facilities on the map.
    - Facility types (restaurant, market, pharmacy, etc.) should be displayed with standard and understandable icons.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «satisfy» REQ-504 (Google Maps and Google Places API Integration)
    - «verify» Facility Information Test
    - «trace» Map Module
- **Restrictions:** Due to data restrictions, this requirement should be implemented in a way that is limited to the regions where the external API provides data.
- **Assumptions:** For this requirement to apply, it is assumed that the user has given permission to location services.
- **Risks:** Implementing this requirement may lead to the user being redirected to a closed venue due to the risk that the facility information from the external API may be out of date.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-041

- **Requirement ID:** REQ-041
- **Title:** Accessing Historical Data in Offline Mode
- **Category:** This requirement belongs to the functional and user interface (UI)/user experience (UX) requirement categories.
- **Description:** The system must offer limited access to previously saved data to ensure access to essential information even when an internet connection is unavailable. This requirement includes storing recent bookings and recommendations in the device's local memory and displaying them in offline mode. This should enhance service availability.

- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested or is impacted by the Drivers, Software Developer, and Mobile Application Team.
- **Acceptance Criteria:**
    - The system should store the last 10 reservations and recommendations in the device's local memory.
    - In offline mode, the system should display a clear warning that "Data may be out of date."
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-438 (Robust User Experience with Smart Offline Backup Mode)
    - «satisfy» Device Local Storage Module
    - «verify» Offline Access Test
    - «trace» Reservation Module
- **Restrictions:** Due to technical limitations, this requirement must be implemented in such a way that only certain screens and cached data are supported offline.
- **Assumptions:** For this requirement to be implemented, it is assumed that the data is synchronized before the application is closed.
- **Risks:** Implementing this requirement may lead to the risk of the user making an incorrect parking decision by relying on outdated data seen in offline mode.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-042

- **Requirement ID:** REQ-042
- **Title:** Sending Application Error Notification
- **Category:** This requirement belongs to the functional and user interface (UI)/user experience (UX) requirement categories.
- **Description:** To improve the quality and stability of the application, the system must enable drivers to report any technical errors they encounter to the development team. This requirement includes reporting events such as application crashes or malfunctions, along with device information and screenshots. This should speed up the debugging process.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by Drivers, Software Developer, QA Team and Support Team.
- **Acceptance Criteria:**

    - The system should enable the user to submit a report with device, time, description and optional screenshot via the "Report Error" tab.
    - The submitted error report should be forwarded to the development team along with the relevant system logs.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-422 (Logging of System Events and Errors)
    - «derivedReqt» REQ-217 (Sensor Error Reporting)
    - «satisfy» Error Reporting Module
    - «verify» Error Reporting Test
- **Restrictions:** Due to technical limitations, this requirement must be implemented in such a way that it only works if the user has an active internet connection.
- **Assumptions:** For this requirement to apply, it is assumed that the user has granted the device permissions.
- **Risks:** Implementing this requirement may lead to unnecessary notifications (spam) creating a system load due to the risk of misuse of this feature by users.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-043

- **Requirement ID:** REQ-043
- **Title:** Warning of Parking Spaces Incompatible with Vehicle Height
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** To prevent physical damage to drivers' vehicles, the system must display a warning if the ceiling height of a selected parking space is inappropriate for the user's vehicle. This requirement involves comparing the vehicle height information in the user's profile with the ceiling height data of the parking space. This system should prevent potential accidents.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is affected by Drivers, Software Developer, Security Team and Parking Space Providers.
- **Acceptance Criteria:**
    - During the reservation process, the system must compare the height information in the profile of the user's selected vehicle with the ceiling height of the parking area.
    - If the vehicle height is greater than the parking space ceiling height, the system should display a warning message to the user.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-002 (Driver Ability to Manage Profile and Vehicle Information)
    - «derivedReqt» REQ-102 (Defining and Adding Parking Areas and Parking Spots)
    - «satisfy» Parking Database
    - «verify» Altitude Control Test
- **Restrictions:** Due to data constraints, this requirement should only work in cases where the parking provider enters the height information and the user defines the height of their vehicle.
- **Assumptions:** For this requirement to be implemented, it is assumed that the user has entered complete vehicle information into his profile.
- **Risks:** Implementing this requirement may lead to physical damage to the user's vehicle due to the risk of incorrectly entering ceiling height data.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-044

- **Requirement ID:** REQ-044
- **Title:** Providing Information About Park Area Ground Slope
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must provide information about the ground slope of a selected parking space to enable drivers to make more informed maneuvering and safety decisions. This requirement includes displaying a warning such as "ground slope present" for sloped areas. This should provide additional safety information, particularly for novice drivers or sensitive vehicles.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested or is affected by Drivers, Software Developer and Parking Space Providers.
- **Acceptance Criteria:**
    - The system should display a warning saying "ground slope present" for areas marked as sloped by the parking space provider.
    - Slope information should also be displayed in percentage (%), if data is available.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:

    - «derivedReqt» REQ-102 (Defining and Adding Parking Areas and Parking Spots)
    - «satisfy» Parking Database
    - «verify» Ground Slope Information Test
- **Restrictions:** Due to data limitations, this requirement should only be implemented to work in cases where slope information is manually entered by the parking provider.
- **Assumptions:** It is assumed that the physical characteristics of all parking areas are entered into the system for the implementation of this requirement.
- **Risks:** Implementing this requirement may lead to driver safety or maneuvering difficulties due to the risk that the provider enters incorrect slope information.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-045

- **Requirement ID:** REQ-045
- **Title:** Receiving Availability Notifications for Favorite/Past Parking Spaces
- **Category:** This requirement belongs to the functional and user interface (UI)/user experience (UX) requirement categories.
- **Description:** To help drivers keep track of frequently used parking spaces, the system should automatically send notifications when these spaces become available again. This requirement includes sending push notifications when a user-favorite or previously used parking space becomes available. This allows the system to provide a proactive and personalized user experience.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is affected by the Drivers, Software Developer, IoT Team, and Notification Module Team.
- **Acceptance Criteria:**
    - The system must detect when a parking space in the user's favorite list or one that the user has used in the past becomes available.
    - When availability is detected, the system should send an instant push notification to the user.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-025 (Save and Delete Favorite Parking Areas)
    - «derivedReqt» REQ-415 (Multi-Channel Notification Support)
    - «satisfy» IoT Sensor Module
    - «satisfy» Notification Module

    - «verify» Conformity Declaration Test
- **Restrictions:** This requirement should be implemented in a way that it will only work if the user grants notification permission for this feature, due to user permissions restriction.
- **Assumptions:** For this requirement to apply, it is assumed that the user has an active favorites list.
- **Risks:** Implementing this requirement may lead to incorrect availability information being sent and unnecessary inconvenience to the user due to the risk of a momentary error in sensor data.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

REQ-046

- **Requirement ID:** REQ-046
- **Title:** Receiving Weather-Based Security Alerts
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** To enhance driver safety, the system should provide advance warnings of parking spaces that could be risky due to adverse weather conditions. This requirement includes regionally identifying conditions such as slippery surfaces, icy conditions, or storms and displaying warnings for those spaces. This should enable drivers to make safer parking decisions.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested or is affected by the Drivers, Software Developer, AI Module Team, and Weather Service.
- **Acceptance Criteria:**
    - The system must identify adverse weather conditions regionally with data from external API.
    - When a risky situation is detected, the system should display a warning for the affected parking areas, such as "This area is risky due to slippery ground."
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «satisfy» REQ-506 (Weather API Integration)
    - «derivedReqt» REQ-230 (Monitoring Environmental Risks and Taking Precautions in Park Areas)
    - «verify» Security Alert Test
    - «trace» Risk Assessment Module

- **Restrictions:** Due to technical limitations, this requirement should only be implemented to work for regions where an external weather service provides data.
- **Assumptions:** For the implementation of this requirement, it is assumed that external weather services provide current and accurate data.
- **Risks:** Implementing this requirement could lead to a lack of warnings and potential safety issues due to the risk of inaccurate weather data.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** AI and Frontend Teams are responsible for implementing this requirement.

REQ-047

- **Requirement ID:** REQ-047
- **Title:** Secure Login and Password Reset Operations
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must provide secure login and password reset mechanisms to ensure users can securely access their accounts and recover them if they lose access. This requirement includes email/password login and an OTP-verified reset process for forgotten passwords. The system must thus ensure basic account security and availability.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is affected by the Drivers, Software Developer, and Security Team.
- **Acceptance Criteria:**
    - When the user provides incorrect login information, the system should display a warning message.
    - When the "Forgot My Password" flow is initiated, the system must verify the user's identity with OTP verification and only then allow the password change.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derive» REQ-001 (Driver Registration and Login to Mobile Application)
    - «derivedReqt» REQ-616 (Providing OTP-Based Password Reset Support)
    - «satisfy» Authentication Module
    - «satisfy» OTP Module
    - «verify» Password Reset Test
- **Restrictions:** Due to security restrictions, this requirement must be implemented in such a way that the validity of the OTP code sent for password reset is a maximum of 5 minutes.

- **Assumptions:** For this requirement to be implemented, it is assumed that the user's registered email or phone information is correct.
- **Risks:** Implementing this requirement could lead to unauthorized account takeover due to the risk of the user's email account being compromised.
- **Status:** The current status of this requirement is Draft.
- **Responsible for:** Backend implementation of this requirement The team is responsible.

REQ-048

- **Requirement ID:** REQ-048
- **Title:** Completion and Evaluation of the Park Session
- **Category:** This requirement belongs to the functional and user interface (UI)/user experience (UX **)** requirement categories.
- **Description:** The system must allow drivers to end an active parking session before they can complete their parking process. This requirement involves ending the session with the "End Parking" button and then redirecting the user to a summary screen that provides parking time, pricing, and experience evaluation. The system must then complete the parking session lifecycle.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested or influenced by Drivers, Software Developer, Parking Space Providers, Mobile UI Team, and Feedback Team.
- **Acceptance Criteria:**
    - When the user presses the "End Parking" button, the system should display a summary screen containing the parking time and fee.
    - The summary screen should allow the user to rate their experience with a 1-5 rating and comments.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - «derivedReqt» REQ-005 (Park Experience Evaluation and Feedback)
    - «derivedReqt» REQ-013 (Ability to Make Payments via Application)
    - «satisfy» Reservation Module
    - «satisfy» Scoring Module
    - «verify» Parking Session Completion Test
- **Restrictions:** This requirement must be implemented to be accessible only to users with an active parking session due to business rules restrictions.
- **Assumptions:** For this requirement to be implemented, it is assumed that the user will participate in the evaluation process after exiting the parking area.

- **Risks:** Implementing this requirement may lead to the failure to collect valuable data about the park experience due to the risk of the user bypassing the evaluation screen and closing the app.
- **Status:** The current status of this requirement is Draft.
- **Responsible:** Frontend and Backend Teams are responsible for the implementation of this requirement.

B) Parking Provider

Requirement REQ-101

- **Requirement ID:** REQ-101
- **Title:** Parking Lot Owner Mobile Registration and Login
- **Category:** Functional, User Interface (UI) and User Experience (UX)
- **Description:** The parking provider must create their Account, complete OTP verification via email and phone during registration, and then securely log in to their account with a password and two-factor authentication.
- **Priority:** High
- **Stakeholders:** Parking lot owners, Authentication Service Providers, Mobile UI/UX Team, QA Team, Security Policy Makers
- **Acceptance Criteria:**
    - Successfully create an account and log in
    - Account should not be opened without OTP verification
    - Must be able to view the parking provider profile
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-003 (OTP verification process)
    - «deriveReqt» REQ-004 (Two-factor authentication)
    - «trace» REQ-302 (Session token generation)
    - «verify» REQ-305 (Testing session scenarios)
    - «trace» REQ-039 (Notification preference management)
- **Restrictions:**
    - Restrictions on mobile or web panel use (ease of use)
    - Compatible with old parking systems (if any)
- **Assumptions:** User enters valid contact information, internet connection is active.
- **Risks:** If security measures do not work well enough
- **Status:** Draft
- **Responsible:** Mobile Backend Team, UI/UX team

Requirement REQ-102

- **Requirement ID:** REQ-102
- **Title:** Defining and adding parking areas and parking spots
- **Category:** Functional, Non-Functional, Technical
- **Description:** The parking lot owner should be able to customize the parking area and parking spot by providing detailed information about its location, capacity, fee information and operating hours, and choosing whether the parking area is sensor-based or staffed with a parking lot attendant and arrange it when necessary.
- **Priority:** High
- **Stakeholders:** Parking lot owners, Drivers, Sensor providers, Mobile backend team, Mobile UI/UX Team
- **Acceptance Criteria:**
    - Parking lot owners can easily create and add parking spaces.
    - Should be able to choose whether the parking space is sensor-based or with a parking attendant
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «satisfy» Location Map API
    - «trace» REQ-013 (Payment System via Application) t
    - «trace» REQ-108 Third-party service integrations (e.g. sensor systems)
    - «trace» REQ-100 – Parking lot owner profile
    - «deriveReqt» REQ-103 – Editing and deleting parking spots.
- **Limitations:** Interface ease of use when adding parking spaces
- **Assumptions:** The parking lot owner must provide accurate and up-to-date information.
- **Risks:** Driver grievances resulting from the parking lot owner's failure to provide accurate and up-to-date information.
- **Status:** Draft
- **Responsible:** Mobile Backend Team, UI/UX team

Requirement REQ-103

- **Requirement ID:** REQ-103
- **Title:** View real-time traffic status and historical usage statistics
- **Category:** Functional
- **Description:** The parking provider should be notified in real time, using sensor data if the parking space is sensor-based, or manually entered by the parking controller if the space is controlled by a parking controller. In addition, historical usage statistics based on the data collected in real time should be made available to the parking space owner.
- **Priority:** Medium
- **Stakeholders:** Parking lot owner, Drivers, Parking lot controller, Sensor provider
- **Acceptance Criteria:**
    - Real-time density data should be displayed
    - Historical usage statistics should be displayed
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:

    - «trace» REQ-216 – Requirement for parking provider to enter manual occupancy data
    - «satisfy» Sensor provider API
    - «trace» REQ-109 – Device (Sensor/Camera) Connection
    - «deriveReqt» REQ-110 – Historical performance reports
- **Restrictions:**
    - Internet connection and speed for instant data transmission
    - Focusing the software development team on higher priority requirements
- **Assumptions:** Correct operation of sensors and manual entry of instant and accurate data by the parking controller
- **Risks:** Data security
- **Status:** Draft
- **Responsible for:** Backend and UI/UX team, Sensor provider, Parking lot controller

---

Requirement REQ-104

- **Requirement ID:** REQ-104
- **Title:** Managing Parking Reservations
- **Category:** Functional
- **Description:** The parking provider must be able to manage parking reservations, view reservation requests, approve or reject them, filter and track past and current reservations, and send notifications to the user when necessary. The parking space must be automatically deactivated during reserved time slots through the system. The provider must also be able to manually change the space to reserveable/inactive based on availability.
- **Priority:** High
- **Stakeholders:** Parking lot owners, Drivers, Reservation Management System, Mobile Backend Team
- **Acceptance Criteria:**
    - The provider must be able to view reservation requests via the system interface.
    - Past, active and future reservations should be trackable and filterable.
    - The provider must be able to manually confirm and cancel reservations.
    - The system should automatically update parking space status based on confirmed reservations.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-003 – Driver can view manually changed availability status
    - «trace» REQ-102 – Parking Area Description and Status Data
    - «trace» REQ-016 – Advance Parking Reservation
- **Restrictions:** System checks for reservation time conflicts
- **Assumptions:**
    - The parking lot owner actively uses the application and has an internet connection.
    - Reservations are made correctly by the drivers through the system.
- **Risks:**

    - Double booking risk
    - Driver grievances due to late notification of reservation cancellation
    - In case of interface delay, no response is possible
- **Status:** Draft
- **Responsible:** Mobile Backend Team, UI/UX Team

Requirement REQ-105

- **Requirement ID:** REQ-105
- **Title:** Parking Provider's Ability to Receive Payments
- **Category:** Functional, Technical, Security
- **Description:** The parking provider must be able to receive payments after reservations are made. Payments must be transferred to the provider instantly or at predetermined intervals (e.g., weekly) and must be transmitted via the system-defined IBAN or digital wallet. The provider must be able to track payments on the dashboard.
- **Priority:** High
- **Stakeholders:** Parking provider, Drivers, Payment Infrastructure Providers (Stripe, iyzico, etc.), Security Team, Mobile Backend team
- **Acceptance Criteria:**
    - The provider must be able to define the payment account.
    - Payments should be transferred automatically.
    - Payment history should be viewable in the panel.
    - A lump sum payment option should be offered.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «satisfy» Payment infrastructure integration
    - «trace» REQ-013 – Payment via app, Driver-side payment system
- **Restrictions:** Bank API limits, Transaction fees
- **Assumptions:** The parking provider's account is verified, the Driver's payment is successful.
- **Risks:** Incorrect IBAN entry, Payment infrastructure interruption
- **Status:** Draft
- **Responsible:** Backend Team

Requirement REQ-106

- **Requirement ID:** REQ-106
- **Title:** Viewing Payment History and Defining Campaigns
- **Category:** Functional, Non-Functional, Technical
- **Description:** The parking provider should be able to view payment history for all transactions through the system and generate daily, weekly, and monthly revenue reports. Furthermore, the system should be able to define discount campaigns for specific time periods, user types, or special circumstances, and activate these campaigns by specifying active/start/end dates. Financial reports should be available in

exportable formats (PDF, Excel), and the revenue impact of the campaigns should be reportable.

- **Priority:** Medium
- **Stakeholders:** Parking providers, Finance Team, Drivers, Campaign/Administration Panel Developers
- **Acceptance Criteria:**
    - The provider must be able to filter and list past payment records by date range.
    - Automatic income reports should be generated through the system.
    - When defining a campaign, the provider must be able to enter information such as target audience, duration, and discount rate.
    - Defined campaigns must be updated and listed as active/passive.
    - The impact of campaigns should be analyzed separately in revenue reports.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-105 (Parking Provider Ability to Receive Payment)
    - «satisfy» Payment System Integration
    - «satisfy» Reporting and Export Module
    - «satisfy» Campaign Management Interface
- **Restrictions:**
    - It requires a powerful server and backend infrastructure for real-time data streaming.
    - Campaign definition screens should fit into limited space on the mobile interface.
- **Assumptions:**
    - Payment data is kept securely in the system.
    - The provider defines the campaigns under his own responsibility.
- **Risks:**
    - Incomplete or incorrect presentation of data in reports
    - Slow loading of historical data due to database density
    - Loss of revenue due to incorrect definition of campaigns
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Panel Developers

Requirement REQ-107

- **Requirement ID:** REQ-107
- **Title:** Creating a Dynamic Pricing Request
- **Category:** Functional, Technical
- **Description:** The parking provider should be able to set different pricing for specific hours or based on density. For example, the fee could be increased during peak hours and discounted during off-peak hours. These rules should be defined on a per-space basis and automatically applied by the system.
- **Priority:** Low
- **Stakeholders:** Parking provider, Drivers, Dynamic Pricing Infrastructure Developers (Backend)

- **Acceptance Criteria:**
    - The provider must be able to set fees based on hourly/time intervals.
    - The specified rules should be applied automatically by the system.
    - Drivers must see the correct price before booking.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «satisfy» Pricing Engine
    - «trace» REQ-003 (Real-Time Parking Availability Display)
- **Limitations:** Complex rules may be difficult to define on the mobile interface, App must process price updates in real time
- **Assumptions:** [Assumptions made for the implementation of the requirement]
- **Risks:** Incorrect time definitions may reduce user satisfaction, sudden price changes may cause transparency problems.
- **Status:** Draft
- **Responsible for:** Backend Team, Pricing Engine Team, UI/UX Team

---

Requirement REQ-108

- **Requirement ID:** REQ-108
- **Title:** Viewing Feedback and Complaints
- **Category:** Functional
- **Description:** The parking provider must be able to view driver feedback and complaints in the system dashboard. These reports must be listed by date, subject, and parking area; the provider must be able to respond optionally.
- **Priority:** Medium
- **Stakeholders:** Parking provider, Drivers
- **Acceptance Criteria:**
    - The provider must be able to see incoming notifications on the panel.
    - The provider must be able to respond (optional).
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-005 (Park Experience Evaluation and Feedback)
    - «trace» REQ-024 (Reporting Problems Regarding Parking Area)
- **Restrictions:** Push notification stream may experience delays
- **Assumptions:** Drivers provide feedback by logging into the system.
- **Risks:** Ignoring complaints may reduce user satisfaction.
- **Status:** Draft
- **Responsible:** Mobile UI/UX Team, Mobile Backend team

Requirement REQ-109

- **Requirement ID:** REQ-109
- **Title:** Device (Sensor/Camera) Connection and Health Status Management
- **Category:** Functional, Technical, Hardware Integration

- **Description:** The parking provider must be able to identify sensors and cameras connected to the system, disconnect them, or reconnect them as needed. They must also be able to monitor and intervene in real-time via the system panel, allowing for the operation status of these devices (active/passive, offline, error, etc.).
- **Priority:** High
- **Stakeholders:** Parking Provider, Sensor Data Source / Camera Data Source, System Administrator, Software Developer
- **Acceptance Criteria:**
    - The provider must be able to identify a new device or disable an existing device.
    - The status of connected devices (active, error, no connection) should be displayed on the panel.
    - The system should be able to notify when the device connection or status changes.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-103 (Real Time Data)
    - «satisfy» Sensor/Camera API
- **Restrictions:** Data cannot be received from offline devices. Some parking areas may use sensor systems and parking controllers.
- **Assumptions:** Devices are compatible with the system and are operational.
- **Risks:** Accidental device deactivation, Delayed detection of hardware errors, Data security
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Team, Hardware provider

---

Requirement REQ-110

- **Requirement ID:** REQ-110
- **Title:** Parking Area Layout Creation and Artificial Intelligence-Assisted Optimization
- **Category:** Functional, Technical, Artificial Intelligence
- **Description:** The parking provider must be able to digitally design the parking space layout, including entry/exit points, physical barriers, and parking spots. The system can visually edit and update this layout using the tools provided. The system should also provide AI-based layout recommendations based on historical usage data and space efficiency. The provider can review and accept these suggestions or continue with manual editing.
- **Priority:** High
- **Stakeholders:** Parking Provider, Software Developer, Data Scientist, System Administrator
- **Acceptance Criteria:**
    - The provider must be able to regulate entrance, exit and obstacle locations on the site.
    - The AI recommendation system should be able to suggest alternative placements based on historical data.

    - The recommendations should be comparable to the manual design.
    - The provider can accept the proposal or continue with his own design.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «satisfy» AI Recommendation Engine
    - «trace» REQ-102 (Defining Parking Areas and Parking Points)
- **Limitations:** Visual editing on mobile interface may be limited, AI recommendations may be based on static data and may not reflect instantaneous changes.
- **Assumptions:** The provider has basic use of design tools, the AI recommendation engine has sufficient historical data.
- **Risks:** Incorrect layouts can negatively impact driver traffic, AI recommendations may be incomplete or lack context.
- **Status:** Draft
- **Responsible for:** UI/UX Team, Backend Team, Artificial Intelligence Team

---

Requirement REQ-111

- **Requirement ID:** REQ-111
- **Title:** Multi-Storey Parking Structure Definition
- **Category:** Functional, Technical
- **Description:** The provider must be able to enter the number of floors, the layout of each floor, its capacity and entry/exit points for multi-storey car parks into the system.
- **Priority:** High
- **Stakeholders:** Parking Provider, Software Developer, System Administrator
- **Acceptance Criteria:**
    - The provider must be able to add and edit new floors.
    - Capacity and access points for each floor should be recorded.
    - Drivers must be able to see the appropriate floor information on the reservation screen.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-102 (Defining Parking Areas and Parking Points)
    - «satisfy» Location Map API
- **Restrictions:** Multi-layered drawing may be limited on mobile screens
- **Assumptions:** Provider provides accurate floor information
- **Risks:** Incorrect capacity data may lead to incorrect bookings
- **Status:** Draft
- **Responsible:** Backend team, UI/UX team

Requirement REQ-112

- **Requirement ID:** REQ-112
- **Title:** EV Charging Station Identification and Capacity Determination in Parking Areas
- **Category:** Functional, Technical

- **Note:** The parking provider must specify whether EV charging stations are available for each parking space and, if so, enter the number of units and power capacity (kW). This information is displayed to drivers and should be included in pricing/booking rules.
- **Priority:** High
- **Stakeholders:** Parking Provider, Sensor Data Source, Software Developer, System Administrator
- **Acceptance Criteria:**
    - The provider must be able to select the “EV Charging Available” option in the Parking area and enter the number of units and power value (kW).
    - The information entered should appear in the parking area details in the driver application.
    - The system should warn when a missing or invalid capacity value is entered.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-102 (Defining Parking Areas and Parking Points)
    - «satisfy» IoT Charging Data API
- **Limitations:** Some older stations may not automatically transmit power capacity data; the provider must perform a manual update.
- **Assumptions:** The provider has correct technical data and the internet connection is active.
- **Risks:** Incorrect capacity information may not meet the driver's charging expectations.
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Team

Requirement REQ-113

- **Requirement ID:** REQ-113
- **Title:** Suspicious Activity Alert System
- **Category:** Functional, Security
- **Description:** The system should send an alert to the parking lot owner when suspicious activity (an unexpected number of failed login attempts, etc.) is detected in the parking lot's bollard sensor, camera, or application session data.
- **Priority:** High
- **Stakeholders:** Parking Provider, Sensor Data Source, Camera Data Source, System Administrator, Software Developer
- **Acceptance Criteria:**
    - A notification should be sent via in-app and/or email immediately after the suspicious event is detected.
    - The notification must include the incident type, parking area, time, and recommended action.
    - All events must be stored with timestamps and identification information.
    - The provider should be able to mark it as “seen/action completed” from the notification center.

- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-109 (Device Connectivity and Health Management)
    - «satisfy» Security Sensor API
- **Limitations:** Threshold values for alert triggering may require fine tuning.
- **Assumptions:** All security devices are online in the network and threshold values are pre-configured.
- **Risks:** False alerts may lead to unnecessary intervention.
- **Status:** Draft
- **Responsible:** Backend Team

Requirement REQ-114

- **Requirement ID:** REQ-114
- **Title:** Sending Message from Provider to Parking Attendant
- **Category:** Functional
- **Description:** The parking provider must be able to send a 500-character, 24-hour instant text message from the admin panel to the parking attendant. The message will appear as a push notification in the parking attendant's mobile app and in the chat screen; a read receipt must be returned.
- **Priority:** Medium
- **Stakeholders:** Parking Provider, Parking Controller, Software Developer, System Administrator
- **Acceptance Criteria:**
    - The provider should be able to select an agent, type a message, and press Send.
    - The officer should see the message via push notification in their app.
    - Read status should be visible on both sides.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «satisfy» Messaging Infrastructure
- **Restrictions:** Requires network connection; messages are automatically deleted after 24 hours.
- **Assumptions:** Both parties have logged into the application.
- **Risks:** Wrong receiver selection or transmission delay may disrupt the operation.
- **Status:** Draft
- **Responsible:** Backend Team, Mobile UI/UX Team

Requirement REQ-115

- **Requirement ID:** REQ-115
- **Title:** Special authority assignment for parking lot controller
- **Category:** Functional
- **Description:** The parking provider must be able to assign access levels such as view, edit, or administrator to staff accounts.

- **Priority:** Low
- **Stakeholders:** Parking Controller, Parking Provider, Software Developer, System Administrator
- **Acceptance Criteria:**
    - The provider must be able to select staff and assign authority levels.
    - Role changes should be reflected in the system immediately.
    - Authorizations should be listed and filtered in the panel.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «satisfy» Role Management System
- **Restrictions:** Incorrect role assignment may lead to unauthorized access.
- **Assumptions:** Role templates are predefined, the parking lot manager knows the roles he needs to assign.
- **Risks:** Incorrect authorization assignments may create security vulnerabilities.
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Team

Requirement REQ-116

- **Requirement ID:** REQ-116
- **Title:** Granting Temporary IoT Access to Visitors
- **Category:** Functional, Security
- **Description:** In a parking lot using smart bollards, the parking provider should be able to grant temporary access to visitors for a specific period of time using IoT devices (e.g., license plate recognition). Access should automatically terminate when the time expires, and all access should be recorded in the system.
- **Priority:** Medium
- **Stakeholders:** Parking Provider, Sensor Data Source, System Administrator, Software Developer
- **Acceptance Criteria:**
    - The provider must be able to enter the visitor's license plate and set a start and end time.
    - When the time expires, the right of way should be automatically cancelled.
    - All access records must be kept in the system.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «satisfy» IoT Access Control API
- **Restrictions:** Cannot be used in areas without smart buoys.
- **Assumptions:** IoT devices are operational and visitor information has been entered correctly.
- **Risks:** Incorrect permission period may lead to unauthorized access.
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Team

Requirement REQ-117

- **Requirement ID:** REQ-117
- **Title:** Definition of Reservation Duration, Discount and Cancellation Rules
- **Category:** Functional
- **Description:** The provider must be able to define minimum and maximum reservation times, special discount conditions, and cancellation policies for the parking area. These rules must be automatically applied by the system.
- **Priority:** Medium
- **Stakeholders:** Parking Provider, Software Developer, System Administrator
- **Acceptance Criteria:**
    - The provider must be able to set time limits (min-max).
    - A discount rate based on the reservation period should be defined.
    - The refund rate should be adjusted according to the cancellation period. (?)
    - Defined rules should be automatically reflected in the reservation flow in the system.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-104 (Parking Reservations Management)
- **Limitations:** It can be difficult to simplify and display complex rules on a mobile interface.
- **Assumptions:** The provider enters rule definitions accurately and consistently.
- **Risks:** Wrong rules can reduce user satisfaction or lead to lost revenue.
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Team

Requirement REQ-118

- **Requirement ID:** REQ-118
- **Title:** Dynamic Price Optimization Suggestions Based on Real-Time and Historical Data
- **Category:** Functional, Artificial Intelligence
- **Description:** The system analyzes historical and current occupancy data to provide the provider with recommendations for action, such as areas with low utilization, vacancies occurring during specific time periods, and price reductions for these situations. These recommendations are displayed in the system interface by time, location, and recommended action; the provider reserves the right to evaluate, implement, or reject these recommendations.
- **Priority:** Medium
- **Stakeholders:** Parking Provider, Data Scientist, Software Developer, System Administrator
- **Acceptance Criteria:**
    - The system should generate recommendations by analyzing areas and hours with low usage.
    - The offer should be presented in the provider panel in the form of time, location and suggested action (e.g. discount).
    - The provider can accept, reject, or schedule the proposal.

- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «satisfy» Data Analysis Engine
    - «trace» REQ-103 (Real-time density status)
- **Limitations:** The AI model may not produce meaningful outputs if there is not enough historical data.
- **Assumptions:** Occupancy data must be collected accurately and completely in real time.
- **Risks:** Misleading recommendations may undermine price competitiveness or result in lost revenue.
- **Status:** Draft
- **Responsible for:** Backend team, UI/UX team, Data science - artificial intelligence team

---

Requirement REQ-119

- **Requirement ID:** REQ-119
- **Title:** Vehicle Blacklist for Repeat Violations and Debt Cases
- **Category:** Functional, Security
- **Description:** The parking provider should be able to blacklist license plates that violate regulations or fail to pay, and these vehicles should be automatically disabled by the system. The system should send an explanatory message to the user explaining why the reservation was not made.
- **Priority:** High
- **Stakeholders:** Parking Provider, System Administrator, Software Developer
- **Acceptance Criteria:**
    - The provider must have a blacklist management interface where it can add and remove license plates.
    - Vehicles on the blacklist should be automatically rejected by the system at the reservation stage.
    - The reason for the reservation rejection should be displayed to the user as a message.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-106 (Viewing payment history)
- **Restrictions:** Blacklisting operations can only be performed by the authorized provider or system administrator.
- **Assumptions:** Violation and debt data are recorded accurately.
- **Risks:** Incorrect blacklisting can lead to customer loss and legal risks.
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX team

Requirement REQ-120

- **Requirement ID:** REQ-120
- **Title:** Providing predictive maintenance reports for devices

- **Category:** Functional, Non-Functional, Technical
- **Description:** The system should analyze the usage data of hardware such as barriers, ticket machines, sensors, and estimate the remaining lifespan and inform the provider with a warning notification when maintenance time approaches (e.g., “Turnstile has reached 120,000 passes, maintenance is recommended.”).
- **Priority:** Low
- **Stakeholders:** Parking Provider, Sensor Data Source, System Administrator, Software Developer
- **Acceptance Criteria:**
    - The system should calculate estimated maintenance times by collecting device usage data.
    - The provider should be able to see the remaining life and maintenance recommendation per device.
    - The system should generate automatic notification when the critical threshold is crossed.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «satisfy» Device Usage Monitoring
- **Limitations:** Lifetime estimation algorithms may differ depending on the hardware type
- **Assumptions:** Devices transmit data regularly and historical usage records are available.
- **Risks:** Forecast errors can increase the risk of unplanned downtime or cause unnecessary maintenance.
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX team

Requirement REQ-121

- **Requirement ID:** REQ-121
- **Title:** Event-Based Demand Forecasting and Capacity Planning Recommendations
- **Category:** Functional, Non-Functional
- **Description:** The system detects large events like concerts, matches, and fairs, and provides the provider with demand forecasts and capacity management recommendations for the expected increase in density around the parking area. These recommendations should include time- and date-based alerts. (e.g., "There's a match at the stadium tomorrow, and a 200% increase in demand is expected.")
- **Priority:** Medium
- **Stakeholders:** Parking Provider, Event Data Source, System Administrator, Data Scientist, Software Developer
- **Acceptance Criteria:**
    - The system should predict demand growth based on the event date.
    - The provider should be shown the estimated rate of increase and the recommended measures.
    - Recommendations may include capacity increase, routing or dynamic price recommendation.

- **Dependencies:** —
- **Limitations:** Predictions may be invalid if event data is not up to date.
- **Assumptions:** Event information is received correctly and activated in the system.
- **Risks:** Events that do not occur may result in unnecessary cost of measures.
- **Status:** Draft
- **Responsible:** Backend Team, Data Science-Artificial Intelligence Team

---

Requirement REQ-122

- **Requirement ID:** REQ-122
- **Title:** Traffic Flow Optimization Suggestions Based on Parking Lot Entry-Exit Usage
- **Category:** Functional, Non-Functional, Technical
- **Description:** The system detects density patterns by analyzing usage data of entry and exit points and should provide traffic flow improvement suggestions to the provider (e.g., “West gate is busy in the evening, consider increasing entrance capacity.”).
- **Priority:** Low
- **Stakeholders:** Parking Provider, Sensor Data Source, Data Scientist, Software Developer, System Administrator
- **Acceptance Criteria:**
    - The system should determine the peak hours based on the entry/exit data.
    - The provider must be notified with the time, location, and recommended solution.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «satisfy» Input-Output Log API
- **Restrictions:** Physical arrangements may not be under the control of the provider
- **Assumptions:** Input-output data is collected accurately and without interruption.
- **Risks:** Incorrect analysis may cause unnecessary structure changes or system load.
- **Status:** Draft
- **Responsible for:** Backend Team, Data Science Team, UI/UX Team, System Administrator

---

Requirement REQ-123

- **Requirement ID:** REQ-123
- **Title:** Tracking and Warning Users Who Cancel Excessive Bookings
- **Category:** Functional, Technical
- **Description:** The provider should be able to monitor the reservation cancellation rates of users through the system, and the system should be able to apply automatic warnings or restrictions for users who exceed the specified threshold (e.g., “This user’s cancellation rate is 30%; sending a warning is recommended.”).
- **Priority:** Medium
- **Stakeholders:** Parking Provider, Software Developer, System Administrator, Reservation Engine
- **Acceptance Criteria:**
    - The provider must be able to see user-based cancellation rates.

    - The system should be able to automatically display a warning message when a certain rate is exceeded.
    - If necessary, temporary reservation restrictions should be applied to the user.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-019 (Reservation Cancellation)
- **Limitations:** The threshold value must be set fairly, otherwise it may lead to incorrect penalties.
- **Assumptions:** User cancellation data is recorded accurately.
- **Risks:** Unjustified penalties may reduce user satisfaction.
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Team

Requirement REQ-124

- **Requirement ID:** REQ-124
- **Title:** Time-Limited Campaign Definition and Performance Analysis
- **Category:** Functional
- **Description:** The parking provider should be able to define time-limited campaigns and/or discounts with parameters such as start-end date, usage limit, target audience; and should also be able to analyze metrics such as the total number of uses of these campaigns, their impact on revenue, and peak usage hours.
- **Priority:** Medium
- **Stakeholders:** Parking Provider, Software Developer, System Administrator, Data Scientist
- **Acceptance Criteria:**
    - When defining a campaign, the provider must be able to enter parameters such as duration, discount rate, and usage limit.
    - Active and past campaigns should be listed and filtered separately.
    - For each campaign, metrics such as total usage, revenue variance, and peak hours should be visually presented.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-106 (Campaign Identification)
    - «satisfy» Analytical Reporting Module
- **Limitations:** Excessive number of campaign definitions may impact system performance
- **Assumptions:** The Provider has entered Campaign usage and payment data correctly.
- **Risks:** Incorrect campaign parameters can lead to lost revenue or system abuse.
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Team, Data Science - Artificial Intelligence Team

---

Requirement REQ-125

- **Requirement ID:** REQ-125

- **Title:** Visually Customizing the Parking Area and Adding Components
- **Category:** Functional, Technical, UI/UX
- **Description:** The provider should be able to visually design their own parking area through the application and place elements such as standard parking, disabled parking, charging station, payment area, entry/exit point and new floors by dragging and dropping during editing.
- **Priority:** High
- **Stakeholders:** Parking Provider, UI/UX Team, Software Developer, System Administrator
- **Acceptance Criteria:**
    - The provider must be able to start a new parking area drawing in the edit panel.
    - Components such as disabled space, charging station, payment area should be added with certain icons.
    - Changes must be recorded in the system and reflected in the driver application.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «satisfy» Visual Editing Interface
- **Restrictions:** Usage on mobile screens may be limited.
- **Assumptions:** Provider has basic digital editing experience.
- **Risks:** Complex areas can cause confusion with incorrect drawing
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Team

Requirement REQ-126

- **Requirement ID:** REQ-126
- **Title:** Disabling Reservations When Necessary
- **Category:** Functional, Non-Functional, Technical
- **Description:** The provider must be able to temporarily disable parking reservations due to technical issues, maintenance, or special circumstances. Drivers must be notified on the reservation screen when this is disabled.
- **Priority:** High
- **Stakeholders:** Parking Provider, Software Developer, UI/UX Team, System Administrator
- **Acceptance Criteria:**
    - The provider must be able to cancel reservation acceptance on a per-parking basis.
    - The disabled status should be indicated to drivers via an explanation message.
    - The system must record the disabled status with date and time information.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-104 (Parking Reservations Management)
- **Restrictions:** Accidental disabling may unnecessarily hinder user access.
- **Assumptions:** The provider performs this operation with the authorized user account.

- **Risks:** Parking areas that remain permanently closed may lead to loss of confidence in the system.
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Team

Requirement REQ-127

- **Requirement ID:** REQ-127
- **Title:** Accessing the Administration Panel and Summary Dashboard via Secure Login Screen
- **Category:** Functional, Security, UI/UX
- **Description:** The provider must be able to access the administration panel via a screen that allows them to log in with their email address and password, optionally supporting 2FA authentication. Upon successful login, the summary panel displays parking spaces, controller assignments, revenue analytics, and live occupancy graphs. Incorrect logins will result in a warning pop-up, and forgotten passwords should be reset using the registered email address.
- **Priority:** High
- **Stakeholders:** Parking Provider, Software Developer, UI/UX Team, System Administrator, Authentication Provider
- **Acceptance Criteria:**
    - Provider must be able to log in with email/password + optional 2FA.
    - After successful login, the system should direct you to the summary dashboard screen.
    - In case of incorrect information, a warning should be displayed to the user.
    - In case of forgetting the password, the user should be able to request a reset via the registered e-mail.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «satisfy» Identity Verification Service
    - «satisfy» Dashboard Reporting System
- **Restrictions:** In order to use the 2FA system, it must be enabled in the user definition beforehand.
- **Assumptions:** User information is already registered in the system and the e-mail is active.
- **Risks:** Weak password policy or open use of 2FA can lead to security vulnerabilities.
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Team

Requirement REQ-128

- **Requirement ID:** REQ-128
- **Title:** Step-by-Step Guide to Adding a New Parking Space
- **Category:** Functional, UI/UX

- **Description:** The provider must be able to define a new parking area by typing an address from the "Add Area" menu, selecting a location from a map, or using live GPS. The guided form that opens allows for entering information such as open/closed space selection, capacity, operating hours, EV charging point, and disabled parking. Photo uploads can be previewed, and the process is completed with a "Save and Continue" screen. The area must be added to the provider's management list of areas.
- **Priority:** High
- **Stakeholders:** Parking Provider, UI/UX Team, Software Developer, System Administrator, Map Provider
- **Acceptance Criteria:**
    - The provider must be able to choose location using one of 3 methods (address, map, GPS).
    - Parking type, capacity, hours, EV and disabled space should be entered in the step-by-step form.
    - Photos should be uploaded and previewed, and the process should be completed on the summary screen.
    - The successfully added domain should be listed in the admin panel.
- **Dependencies:** —
- **Restrictions:** GPS option can only be activated via the mobile application.
- **Assumptions:** The provider's location access is enabled and the connection is active.
- **Risks:** Wrong location selection can mislead drivers.
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Team

Requirement REQ-129

- **Requirement ID:** REQ-129
- **Title:** Authority to View, Edit and Delete Reservations
- **Category:** Functional
- **Description:** The provider can view, filter, edit, or delete reservations in the parking spaces they manage. The reservation list includes filters such as date, user, license plate, and payment status. Vehicle information, time slot, or spot ID can be edited from the details screen. A confirmation window will appear for each edit, and the transaction is recorded. If the reservation is deleted, a reason must be selected (e.g., maintenance, unauthorized access), and the driver must be automatically notified.
- **Priority:** High
- **Stakeholders:** Parking Provider, Software Developer, UI/UX Team, System Administrator, Notification Service
- **Acceptance Criteria:**
    - The provider must be able to list and filter reservations.
    - Reservation details must be editable, and each transaction must include a checkbox and a log record.
    - The deletion must be done with a reason, and an automatic notification must be sent to the driver.

- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «trace» REQ-104 (Parking Reservations Management)
    - «satisfy» Notification Service
- **Restrictions:** Only reservations in provider-owned areas can be edited.
- **Assumptions:** Authorized user has access to the correct fields.
- **Risks:** Incorrect editing or deletion may lead to user inconvenience.
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Team

Requirement REQ-130

- **Requirement ID:** REQ-130
- **Title:** Assigning Active Controllers to the Parking Area and Shift Planning
- **Category:** Functional
- **Description:** The provider should be able to assign active controllers to registered areas; users should be selected from the drop-down list during staff selection, and then working hours and the area of responsibility should be defined via the resulting shift planner. The system should prevent conflicting assignments and provide a warning if the user is assigned to another area. After the assignment, the controller is notified, and the process should be logged with the time.
- **Priority:** High
- **Stakeholders:** Parking Provider, Parking Controller, Software Developer, UI/UX Team, Notification Service, System Administrator
- **Acceptance Criteria:**
    - The provider should be able to assign areas and shifts by selecting personnel from the drop-down list.
    - The system should warn about conflicting tasks and not allow assignment.
    - After the assignment, a system notification containing shift information must be sent to the controller.
    - All appointments must be recorded with a timestamp and provider ID.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - «satisfy» Personnel Management System
    - «satisfy» Notification Service
- **Restrictions:** An employee can only be assigned to one area at a time.
- **Assumptions:** Personnel information and work areas are pre-registered.
- **Risks:** Assignment error may cause operational disruptions.
- **Status:** Draft
- **Responsible:** Backend Team, UI/UX Team

C) Parking Controller

Requirement REQ-201

- **Requirement ID:** REQ-201
- **Title:** System Should Notify Parking Attendant of New Reservation
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system will automatically send a notification through the app when a driver makes a reservation, so that the parking attendant is instantly aware of this.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the parking attendant, the driver, and the software developer
- **Acceptance Criteria:**
  - The driver needs to make a reservation.
- **Dependencies:** This requirement cannot be implemented without completing the following system components and requirements:
  - REQ-016 «trace» (Driver must make a reservation before the notification is triggered)
  - REQ-202 «trace»(Officer can see driver information)
  - REQ-206 «trace»(may require access to Reservation List)
- **Restrictions:** Must work when system is connected to the internet
- **Assumptions:** Users are assumed to have an internet connection on their mobile devices. User notifications are assumed to be enabled.
- **Risks:** Notification may be delayed due to connection problems.
- **Status:** Draft
- **Responsible:** Development Team – Backend

Requirement REQ-202

- **Requirement ID:** REQ-202
- **Title:** Parking Attendant's Ability to See Reservation Information
- **Category:** This requirement belongs to the functional requirement category.

- **Explanation:** The system should allow the parking attendant to access basic information about the driver making the reservation, such as their license plate, name, and profile photo. This requirement serves the attendant's need to be able to inspect and verify vehicles in the area. The system should be designed to display this data only if the driver has consented to sharing the information. The shared information should be used by the attendant solely for viewing and should not be shared with third parties.
- **Priority:** This requirement has been set to **high priority** .
- **Stakeholders:** This requirement was requested by and directly impacts **the parking attendant** and **driver stakeholder groups.**
- **Acceptance Criteria:**
    - The system should only display this information to the parking attendant if the driver making the reservation has entered their profile information and approved the sharing.
    - The parking attendant must confirm through the system that they will use this information for display purposes only.
    - The driver's information must be listed accurately and completely on the officer's screen.
- **Dependencies:** This requirement cannot be implemented without completing the following system components and requirements:
    - REQ-001«trace»(Notification infrastructure)
    - REQ-002 «trace» (Driver profile and vehicle information),
    - REQ-012«trace»(Parking history data)
- **Restrictions:** This requirement is applicable provided that the driver has entered the necessary information into the system and the system is capable of operating with an internet connection.
- **Assumptions:** For the implementation of this requirement, it is assumed that the mobile devices of the users (driver and parking attendant) have an active internet connection.
- **Risks:** Implementing this requirement may carry the risk that data may not be loaded onto the officer's screen in a timely manner due to delays or interruptions in the internet connection.
- **Status:** The current status of this requirement is **Draft .**
- **Responsible: Development Team – Backend** and **Development Team – Frontend are responsible** for the implementation of this requirement .

Requirement REQ-203

- **Requirement ID:** REQ-203
- **Title:** Parking Attendant Sends Message to Park Owner and Driver

- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system should enable parking attendants to communicate via text message with parking owners or drivers in their area whenever necessary. This requirement supports faster and more efficient on-site communication. Messaging should only be active between mutually consenting parties; attendants should only be able to send messages to drivers registered with their parking lot and the parking owner defined in the system. Messages should be supported by in-app notifications and managed in accordance with security and privacy principles.
- **Priority:** This requirement has been determined to be **a high priority** .
- **Stakeholders:** This requirement was requested by and directly impacts **the parking attendant** , **parking owner** , and **driver stakeholder groups.**
- **Acceptance Criteria:**
    - The driver to whom the parking attendant will send a message must have an active reservation for the parking space under the attendant's control.
    - The parking attendant should only be able to communicate with the park owner assigned to their parking space.
    - Both the message sender and the message receiver must consent to using the messaging feature.
    - Once sent, messages should be displayed in both parties' message boxes and supported by system notifications.
- **Dependencies:** This requirement cannot be implemented without completing the following requirements:
    - REQ-202«trace»(Access to driver information)
    - REQ-039«trace»(Notification preference management)
    - REQ-114«trace»(Provider-attendant messaging)
- **Restrictions:** This requirement should be implemented on the condition that messages are only kept in the system for 24 hours, as per the privacy policy.
- **Assumptions:** For this requirement to be implemented, it is assumed that users' mobile devices have an active internet connection.
- **Risks:** Implementing this requirement carries the risk that message transmission between the parties cannot occur when there is no internet connection.
- **Situation:** The current status of this requirement is **Draft .**
- **Responsible: Development Team – Backend** and **Development Team – Frontend are responsible** for the implementation of this requirement .

Requirement REQ-204

- **Requirement ID:** REQ-204
- **Title:** Parking Attendant Warns Driver About Parking Illegally
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system should enable parking attendants to send warning messages via the app to drivers who park their vehicles incorrectly or overstep the designated area. This feature allows the attendant to maintain communication and parking order even when the driver is not physically present in the parking area. The attendant should only issue this warning when a reservation is available for the relevant vehicle. Warnings should be recorded in the system and, if necessary, supported by a photo report.
- **Priority:** This requirement has been determined to be **a high priority** .
- **Stakeholders:** This requirement was requested by **the parking attendant** and **driver stakeholder groups and directly impacts these stakeholders.**
- **Acceptance Criteria:**
    - Alerts should only be sent to a vehicle belonging to a driver who has a reservation.
    - Warnings should be transmitted via the messaging infrastructure and communicated to the driver via system notification.
    - The officer must be able to document the situation by attaching photos before sending the warning (optional).
    - Warnings sent to the driver should be recorded in the system with a timestamp.
- **Dependencies:** This requirement depends on and cannot be implemented without the completion of the following other system components:
    - REQ-203«trace»(messaging infrastructure)
    - REQ-207 «trace» (Can be supported by photo reporting)
    - REQ-007«trace»(Verify vehicle location)
    - REQ-113 «trace» (Suspicious activity warning system)
- **Restrictions:** This requirement should only be implemented on the mobile app and with an active internet connection.
- **Assumptions:** This requirement should only be implemented on the mobile app and with an active internet connection.
- **Risks:** Implementing this requirement could create unnecessary dissatisfaction with drivers, such as if the parking attendant misuses the alert feature or sends an alert in error.
- **Situation:** The current status of this requirement is **Draft .**
- **Responsible: The Development Team – Backend is responsible** for the implementation of this requirement .

Requirement REQ-205

- **Requirement ID:** REQ-205
- **Title:** Parking Attendant Closes Parking Area to Reservations
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system should allow the parking attendant to temporarily close the parking area to reservations in the event of a problem (maintenance, damage, emergency, etc.) in the parking area under their control. This feature aims to systematically control sudden situations that may arise in physical spaces and prevent misleading drivers. When closing a space, the attendant is responsible for entering details such as the reason for closure and the duration of the closure into the system. Only vacant and unreserved spaces can be closed.
- **Priority:** This requirement has been determined to be **a high priority** .
- **Stakeholders:** This requirement was requested by **the parking attendant** and **driver stakeholder groups and directly impacts these stakeholders.**
- **Acceptance Criteria:**
    - The parking attendant must write down the reason before closing the reservation.
    - The park attendant needs to write down how many hours the park will be closed.
    - The parking area must be vacant and not reserved.
- **Dependencies:** This requirement cannot be implemented without completing the following system components:
    - REQ-016 «trace»(Closing the area requires the reservation system to detect the reservation)
    - REQ-206 «deriveReqt»(To check reservation status)
    - REQ-126 «refine»(Reservation deactivation)
- **Restrictions:** This requirement only applies to internet-connected devices and **vacant, unreserved** parking spaces.
- **Assumptions:** To implement this requirement:
    - It is assumed that users' mobile devices have an internet connection.
    - It is assumed that the parking space is not reserved.
- **Risks:** This requirement may cause drivers to have difficulty finding suitable parking if multiple parking spaces are closed at the same time.
- **Situation:** The current status of this requirement is **Draft .**
- **Responsible: Development Team – Backend** and **Development Team – Frontend are responsible** for the implementation of this requirement .

Requirement REQ-206

- **Requirement ID:** REQ-206

- **Title:** Parking Attendant's Ability to See Made Reservations
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system must enable parking attendants to view all active reservations in their designated parking area. This requirement requires the attendant to have access to the number of drivers who have made reservations, the locations where they are booked, and the current status of this information. Reservation data should be presented both digitally and visually on a map, and the information should be updated regularly.
- **Priority:** This requirement has been determined to be **a high priority** .
- **Stakeholders:** This requirement was requested by or directly impacts **the parking attendant** , **driver** , **IoT Team** , and **Mobile Application Team .**
- **Acceptance Criteria:**
    - The system should display the total number of active reservations in the parking attendant's area in real time.
    - Reservation locations should be instantly viewable on the map.
    - Reservation data must be updated at most every **30 seconds** .
- **Dependencies:** This requirement is contingent upon completion of the following systems or requirements:
    - REQ-003«trace»(Integration of an external reservation system)
    - REQ-016«trace»(Reservation data)
    - REQ-103 «satisfy»(Real-time density data)
- **Restrictions:** Real-time data will only work on mobile devices with an internet connection.
- **Assumptions:** For this requirement to be implemented, it is assumed that users' mobile devices have an internet connection.
- **Risks:** In the event of system errors or internet disconnection, the reservation information displayed to the agent may be out of date, which may negatively impact field operations.
- **Situation:** The current status of this requirement is **Draft .**
- **Responsible: Development Team – Backend** and **Development Team – Frontend are responsible** for the implementation of this requirement .

Requirement REQ-207

- **Requirement ID:** REQ-207
- **Title:** Parking Attendant Reporting with Photograph
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system should allow parking attendants to take photos to document various issues they encounter on-site (e.g., vehicle damage,

incorrect parking), and report them to the system. This feature will ensure that immediate on-site problems are documented and accurately communicated to higher-level systems. Reports should be visually supported and can be integrated into penalty or warning processes when necessary.

- **Priority:** This requirement has been determined to be **a high priority** .
- **Stakeholders:** This requirement was requested by **the parking attendant** and **mobile app team or directly impacts these groups.**
- **Acceptance Criteria:**
    - The entire vehicle or parts related to the incident must be clearly included in the frame in the photo taken by the officer.
    - The problematic situation must be clearly visible in the photograph taken (for example: if the vehicle is damaged, the damaged part must be visible; if the vehicle was parked incorrectly, the photograph must clearly show how the vehicle was parked).
- **Dependencies:** This requirement depends on the following system components:
    - «satisfy» Media Upload Service
    - REQ-206«trace»(Reservation information)
    - REQ-213 «trace»(Integration into the penalty/warning process)
    - REQ-108«trace»(Feedback management)
- **Restrictions:** For this requirement to work, the mobile application must have camera access permission.
- **Assumptions:**
    - It is assumed that the mobile application's camera access is provided by the system.
    - It is assumed that users' mobile devices have an internet connection.
- **Risks:** During the implementation of this requirement, situations such as blurry, incomplete or irrelevant photographs taken by the officer may lead to inaccurate or incomplete reporting.
- **Situation:** The current status of this requirement is **Draft .**
- **Responsible: Development Team – Backend** and **Development Team – Frontend are responsible** for the implementation of this requirement .

Requirement REQ-208

- **Requirement ID:** REQ-208
- **Title:** Parking Attendant's License Plate Scanning Feature
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system should allow parking attendants to scan a vehicle's license plate using their mobile device and instantly view its reservation status and payment information. This feature allows attendants to conduct a fast, accurate, and secure on-site check. License plate scanning should be

performed using camera-assisted optical character recognition (OCR) technology.

- **Priority:** The priority of this requirement is set to **medium** .
- **Stakeholders:** This requirement is intended for use by **the parking attendant .**
- **Acceptance Criteria:**
  - The plate must be visible in its entirety
  - The license plate must be in the camera's field of view.
- **Dependencies:** For this requirement to apply, the following requirements must be met:
  - REQ-202 «trace»(Matching license plate with driver information)
  - REQ-002«trace»(Vehicle license plate database)
- **Restrictions:** For this requirement to be implemented, the mobile application must have permission to access the device's camera.
- **Assumptions:**
  - It is assumed that the mobile app has permission to access the camera .
  - It is assumed that users' mobile devices have an active internet connection.
- **Risks:** Due to hardware camera issues or software OCR errors, the plate scanning process may fail and the officer may not be able to access the necessary information.
- **Situation:** The current status of this requirement is **Draft .**
- **Responsible: Development Team – Hardware** and **Development Team – Backend are responsible** for the implementation of this requirement .

Requirement REQ-209

- **Requirement ID:** REQ-209
- **Title:** Parking Attendant's Ability to Control Instant Parking Areas
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** Parking attendants should be able to view the current occupancy status of all parking spaces under their control through the system. This information helps them manage parking more effectively.
- **Priority:** This requirement has been determined to be a high priority.
- **Stakeholders:** This requirement is requested or influenced by the parking attendant, drivers, and sensor and camera data sources.
- **Acceptance Criteria:**
  - Data from the sensor needs to be processed into the system.
- **Dependencies:** This requirement is contingent upon the following other requirements being met:
  - REQ-206«trace»(Reservation data)

    - REQ-216«trace»(Manual update support)
    - REQ-003«trace»(Real-time parking lot display)
- **Restrictions:** This requirement should only be implemented on devices that work online due to the need for an internet connection.
- **Assumptions:** To implement this requirement, it is assumed that the parking attendant's device has an internet connection and the hardware providing occupancy data is active.
- **Risks:** Implementing this requirement may create the risk of the officer not being able to access accurate information due to sensor/camera malfunctions and connection issues.
- Due to system density, data flow may slow down or be interrupted.
- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** Development Team – Backend and Development Team – Frontend are responsible for the implementation of this requirement.

Requirement REQ-210

- **Requirement ID:** REQ-210
- **Title:** Notification to Parking Attendant in Case of Reservation Exceeding Time and Rule Violations
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system should detect reservation overruns and violations through sensor and camera data, sending alerts so parking attendants can intervene in a timely manner. This requirement contributes to maintaining order in the parking area.
- **Priority:** This requirement has been determined to be a high priority.
- **Stakeholders:** This requirement is requested or influenced by the parking attendant, the sensor data source, and the camera data source.
- **Acceptance Criteria:**
    - The system should notify the parking attendant when the reservation expires.
    - The system should generate instant alerts in case of rule violations detected through sensors.
- **Dependencies:** This requirement is dependent on the completion of the following requirements:
    - REQ-203 «deriveReqt» (To send a warning message)
    - REQ-223«trace»(Manual input/output verification)
    - REQ-011«trace»(User receives notification without timeout)
- **Restrictions:** This requirement should be implemented in parking areas where sensors are installed.

- **Assumptions:** For this requirement to be implemented, it is assumed that the sensors are working properly and the system has an internet connection.
- **Risks:**
    - Implementing this requirement may result in a rule violation not being detected due to sensor malfunction.
    - Implementing this requirement may result in delays in notifications regarding reservations due to poor internet connectivity.
- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** The Development Team – Backend is responsible for the implementation of this requirement.

Requirement REQ-211

- **Requirement ID:** REQ-211
- **Title:** Parking Attendant Marks Parking Space as Problematic
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must allow parking attendants to mark areas on the map that pose maintenance and safety risks. This requirement should ensure that users are aware of these areas, thus promoting safe and orderly use of the parking area.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or influenced by parking attendants and drivers.
- **Acceptance Criteria:**
    - The system should allow the parking attendant to open the map via the app.
    - The system should allow entry of the problem cause for the selected area.
- **Dependencies:** This requirement is dependent on the completion of the following requirements:
    - REQ-205 «refine» (Parking Attendant's Ability to Close the Parking Area to Reservations)
    - REQ-046«trace»(Weather risk warnings)
    - REQ-110«trace»(Layout optimization)
- **Restrictions:** This requirement should be implemented provided that the mobile device has an internet connection.
- **Assumptions:** To implement this requirement, it is assumed that users' mobile devices are connected to the Internet.
- **Risks:** Implementing this requirement may lead to misinformation of users due to incorrect reason selection.
- **Status: The current status of** this requirement is determined as a draft.
- **Responsible:** Development Team – Backend and Development Team – Frontend are responsible for the implementation of this requirement.

Requirement REQ-212

- **Requirement ID:** REQ-212
- **Title:** Reservation and Entry Tracking for Parking Attendants via Mobile Panel
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system should provide a dashboard that allows the parking attendant to access the reservation list, upcoming entries, and expiring driver information via their mobile device. This requirement supports the attendant's efficient parking management.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or influenced by the parking attendant, the driver, and the software developer.
- **Acceptance Criteria:**
    - In the panel, the queued reservations should be displayed as a list.
    - Vehicles approaching their reservation time (e.g. within the last 30 minutes) should be highlighted in a separate section.
    - The information displayed on the panel (login times, session durations, etc.) should be automatically updated at certain intervals (e.g., every 30 seconds).
    - When there is no information to display (e.g. no active reservation), an appropriate information message should be displayed to the user: “There are currently no reservations to display.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-202«trace»(Driver information)
    - REQ-206 «refine»(Reservation list)
    - REQ-028«trace»(Overlapping reservation blocking)
- **Restrictions:** This requirement must be implemented with authenticated user access and persistent internet connectivity.
- **Assumptions:** To implement this requirement, it is assumed that users are connected to the Internet and that the data is correctly retrieved from the central database.
- **Risks:**
    - Reservation and session information on the panel may be out of date due to internet connection interruptions or server-side delays.
    - If unauthorized persons access the panel, data security may be compromised.

    - Even though a vehicle appears as “not entered” in the system, it may be physically inside; such situations may be caused by a data synchronization error.
- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** Development Team – Backend and Development Team – Frontend are responsible for the implementation of this requirement.

Requirement REQ-213

- **Requirement ID:** REQ-213
- **Title:** Digital Fine and Warning Authority by Parking Attendant
- **Category:** The system should enable parking attendants to issue fines and send warnings digitally to vehicles that violate the rules.
- **Priority:** This requirement has been set to low priority.
- **Stakeholders:** This requirement is requested or influenced by the parking attendant and the driver.
- **Acceptance Criteria:**
    - The system should present a special transaction screen to the officer.
    - The system should allow the officer to select the reason for the violation.
    - The system should offer “Penalty” and “Warning” options as transaction types.
    - The system must record the transactions in detail and administrative users must be able to view these records.
    - The system should allow the officer to cancel the transaction within 5 minutes.
- **Dependencies:** This requirement is dependent on the completion of the following requirements:
    - REQ-207 «deriveReqt»(Photo evidence)
    - REQ-210 «deriveReqt» (Rule violation detection)
    - REQ-013«trace»(Payment system integration)
- **Restrictions:** This requirement is limited so that the parking officer can only operate for defined reasons and in areas for which he or she is authorized.
- **Assumptions:** The applicability of this requirement is based on the assumption that the officer is an authorized user, has an internet connection, and acts in accordance with the rules.
- **Risks:** This requirement carries risks such as incorrect processing and damage to system reliability if the officer selects the wrong license plate.
- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** Development Team – Backend and Development Team – Frontend are responsible for the implementation of this requirement.

Requirement REQ-214

- **Requirement ID:** REQ-214
- **Title:** Parking Attendant's Note-Taking and Sharing Information with the Next Attendant
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system should enable the parking attendant to record notes about the events he or she encounters during his or her shift via a free text field, and these notes should be automatically forwarded to the next officer taking over the shift after the shift is completed, allowing for complete and uninterrupted information transfer between shifts.
- **Priority:** This requirement has been set to low priority.
- **Stakeholders:** This requirement was requested or influenced by the parking attendant and software development team.
- **Acceptance Criteria:**
    - The parking attendant must be able to access the “Shift Cycle” section after logging into the system.
    - The officer must be able to enter shift notes (events, actions taken, warnings, etc.) in the free text field.
    - When the officer presses the “Transfer” button, the system should record the date/time and officer ID and display a “Successfully transferred” notification.
    - When the new officer enters the system, a pop-up notification or modal containing the delegated notes should be automatically displayed.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-226«trace»(Shift schedule information)
    - REQ-047«trace»(Secure session management)
    - REQ-115 «refine» (Authority assignment)
- **Restrictions:** This requirement should be implemented in a way that only allows the attendant to delegate their shift to the parking areas they are responsible for due to technical limitations. Additionally, the process will not be supported offline and will not be completed in the event of a system outage. The notes field should be limited to a maximum of 2,000 characters, allowing only text entry; media attachments should not be supported.
- **Assumptions:** The functionality of this requirement depends on the officer having an internet connection and the officer who will take over the shift being pre-defined.
- **Risks:** If the exiting officer does not enter the events completely, the new officer may make a wrong decision, such as an internet outage or a server error.

- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** Development Team – Backend and Development Team – Frontend are responsible for the implementation of this requirement.

Requirement REQ-215

- **Requirement ID:** 215
- **Title:** Parking Attendant Invalidates Reservation
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system should ensure that the parking attendant cancels the reservation before the driver arrives due to potential parking problems. This requirement was developed to effectively manage parking space capacity and prevent potential usage problems. Cancellations made by the parking attendant should be recorded in the system along with the reason for the cancellation and communicated to the driver.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is directly affected by parking attendants and drivers and will be implemented by software developers.
- **Acceptance Criteria:**
    - The system should require the reason for cancellation to be entered when a reservation is cancelled.
    - The park attendant should only be able to cancel the reservation within the first 15 minutes after the reservation is created.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-019«refine» (Cancel reservation)
    - REQ-206«trace»(Reservation information)
- **Restrictions:**
    - Due to technical limitations, this requirement must only be implemented within a specific timeframe after the reservation is made. Additionally, the user (both the parker and the driver) must have access to the mobile app with an internet connection.
    - **Assumptions:** For this requirement to apply, it is assumed that the driver has made a reservation and both parties (the attendant and the driver) have a mobile internet connection.
- **Risks:** Implementation of this requirement may lead to user dissatisfaction due to parking attendant misuse or misuse of the system.
- **Situation:** The current status of this requirement is outlined as a draft.

- **Responsible:** Development Team – Backend and Development Team – Frontend are responsible for the implementation of this requirement.

Requirement REQ-216

- **Requirement ID:** REQ-216
- **Title:** Manual Update of Parking Space Status
  Category: This requirement belongs to the functional requirement category.
  Explanation: When sensor-based occupancy detection is not available, the system must enable the parking controller to manually update the occupied or unoccupied status of the relevant parking space via the control panel. This requirement is critical for maintaining system functionality in the event of automated system failures, such as sensor malfunctions. Manual updates must maintain real-time system visibility.
- **Priority:** When sensor-based occupancy detection is not available, the system must enable the parking controller to manually update the occupied or unoccupied status of the relevant parking space via the control panel. This requirement is critical for maintaining system functionality in the event of automated system failures, such as sensor malfunctions. Manual updates must maintain real-time system visibility.
- **Stakeholders:** This requirement is requested or influenced by parking controllers and system administrators.
- **Acceptance Criteria:**
  - The system should allow only authorized users to manually change parking space occupancy information via the administration panel.
  - Manual changes must be reflected in the system within 10 seconds at most.
- **Dependencies:** This requirement is dependent on the completion of the following requirements:
  - REQ-102 «trace» (Parking area identification)
- **Restrictions:** This requirement should be accessible only to specific users due to authorization requirement.
- **Assumptions:** To implement this requirement, it is assumed that the sensor system may fail and manual updates may be required.
- **Risks:** Implementing this requirement may lead to misleading occupancy information in the system due to the risk of incorrect data entry.
- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** The Development Team – Management Panel is responsible for the implementation of this requirement.

Requirement REQ-217

- **Requirement ID:** REQ-217
- **Title:** Sensor Error Reporting
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** In scenarios where sensors cannot automatically detect parking space occupancy, the system should provide a manual error notification to the parking controller. Following this notification, the system should record the error in the system logs and automatically forward it to the technical support team. This functionality is critical for ensuring ongoing troubleshooting and maintaining system accuracy.
- **Priority:** This requirement has been determined to be a high priority.
- **Stakeholders:** This requirement is requested or influenced by park rangers, software developers, and sensor data sources.
- **Acceptance Criteria:**
    - The system should provide an option to report an error to the parking attendant when automatic detection fails.
    - Notifications should be kept in system logs and automatically forwarded to the relevant technical team.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-207 «refine» (Can be supported by photo reporting)
    - REQ-229 «trace» (similar infrastructure with equipment fault notification)
    - REQ-042 «trace» (Error reporting infrastructure)
- **Restrictions:** This requirement must be implemented so that it is available only to authorized users.
- **Assumptions:** To implement this requirement, it is assumed that a failure may occur in the sensor system.
- **Risks:** Implementing this requirement could lead to a loss of confidence in the user experience due to the risk of notification skipping.
- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** The Development Team – Backend is responsible for the implementation of this requirement.

Requirement REQ-218

- **Requirement ID:** REQ-218
- **Title:** Suitable Parking Area Recommendations for Large Vehicles

- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system should analyze physically suitable parking spaces for large vehicles like minibuses and pickup trucks and provide recommendations to the parking attendant. Recommendations should be generated by matching the vehicle's dimensions with the physical dimensions of the parking spaces, and only suitable spaces should be displayed. The parking attendant should be able to guide the driver by displaying the suggested spaces through the system interface.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested or influenced by parking officials, software developers, municipal officials, and parking providers.
- **Acceptance Criteria:**
    - When the large vehicle type is selected, the system should only display parking spaces suitable for these vehicles.
    - The parking attendant should be able to verify the suggested parking spaces and guide the user through the system interface.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-102 «trace» (Parking area identification and parking area physical characteristics)
    - REQ-209 «deriveReqt» (Instant occupancy data)
- **Restrictions:** This requirement has limited applicability due to the limited parking spaces available for large vehicles.
- **Assumptions:** For this requirement to be implemented, it is assumed that the size and type information of all parking areas in the system are entered correctly.
- **Risks:** Implementing this requirement may result in the system's guidance functionality being limited due to the risk of not being able to find suitable parking spaces.
- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** Development Team – Backend and Frontend is responsible for the implementation of this requirement.

Requirement REQ-219

- **Requirement ID:** REQ-219
- **Title:** Overriding Automatic Systems in Emergencies
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system should allow the parking attendant to temporarily disable automated system operation in emergency scenarios such as fire, accidents, or security threats. This process should override sensor data,

automatic routing algorithms, and active reservations. The system should record such interventions in the system logs and notify the system administrator. This feature is designed to support system security and crisis management.

- **Priority:** This requirement has been determined to be a high priority.
- **Stakeholders:** This requirement has been requested or is influenced by parking officials, system administrators, software developers, municipal officials, and government agencies.
- **Acceptance Criteria:**
    - The system should only allow the parking attendant to override automatic operations when an emergency is signaled.
    - After the override process, all automatic routing, sensor information and reservation operations in the system must be temporarily suspended.
    - The override operation should be recorded in the system logs and an automatic notification should be sent to the system administrator.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
  REQ-220«trace» (Receiving and Acknowledging Security Alerts)
  REQ-221«trace» (Direct Communication with Emergency Services)
  REQ-222«trace» (Broadcasting Emergency Messages to All Users)
- **Restrictions:** This requirement should be limited to activation only in emergency scenarios and for authorized users.
- **Assumptions:** For the implementation of this requirement, it is assumed that emergency situations are identifiable on the system.
- **Risks:** Implementing this requirement may temporarily destabilize the system due to the risk of override abuse.
- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** Development Team – Backend and Frontend is responsible for the implementation of this requirement.

Requirement REQ-220

- **Requirement ID:** REQ-220
- **Title:** Receiving and Approving Security Alerts
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** When the system instantly detects security threats such as unauthorized entry, camera/sensor warnings, or emergency notifications, it should send alerts to the parking attendant via push notification. The parking attendant should indicate that they have responded to each incident by marking these alerts as "read" or "intervened" via the system interface. Every

action taken should be recorded in system logs to ensure traceability. This mechanism aims to expedite the management and simplify tracking of security incidents in the parking area.

- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or influenced by parking staff, system administrators, software developers, and camera and sensor data providers.
- **Acceptance Criteria:**
    - The system should provide security alerts to the parking attendant as instant notifications.
    - The parking attendant must be able to respond to every notification through the system.
    - Response information should be automatically recorded in system logs.
- **Dependencies:** This requirement is dependent on the completion of the following requirements:
    - REQ-219«trace» (Disabling automatic systems in emergency situations)
- **Restrictions:** This requirement can only be viewed and answered by authorized users.
- **Assumptions:** For this requirement to be implemented, it is assumed that security events can be detected and reported correctly by the system.
- **Risks:** Implementing this requirement may undermine system reliability due to false positive alerts or notification delays.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Development Team – Backend and Frontend is responsible for the implementation of this requirement.

Requirement REQ-221

- **Requirement ID:** REQ-221
- **Title:** Direct Communication with Emergency Services
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** When a parking attendant detects an emergency (e.g., fire, suspicious package, assault, etc.), they must be able to contact emergency services such as police, fire department, or paramedics directly through the system. This communication should be provided through predefined communication channels (e.g., emergency call button, text messaging).

- **Priority:** This requirement has been set to high priority.
  Stakeholders: Park rangers, government and municipal officials, system administrators, software developers, and integrated external system providers are affected by this requirement.
- **Acceptance Criteria:**
  - The parking attendant should be able to send a notification to the relevant emergency services with a single click via the emergency panel in the system.
  - The content of the notification sent must include the event type, location information and timestamp.
  - The system administrator should be automatically notified as soon as the notification is sent.
- **Dependencies:** This requirement is dependent on the completion of the following requirements:
  - REQ-219«trace» (Disabling automatic systems in emergency situations)
  - REQ-220«trace»(Security warnings)
  - REQ-203«trace»(messaging infrastructure)
- **Restrictions:** Only authorized park staff can access this communication channel.
- **Assumptions:** It is assumed that the necessary integration between the system and emergency services has been achieved.
- **Risks:** Misuse of this requirement could result in waste of public resources.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Development Team – Backend and Frontend is responsible for the implementation of this requirement.

Requirement REQ-222

- **Requirement ID:** REQ-222
- **Title:** Emergency Message Broadcast to All Users
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system must ensure that if the parking attendant detects an emergency, it will broadcast an instant alert message to all user devices, such as mobile apps and digital signage. This requirement is designed to quickly and effectively inform users in emergencies. The message content must include information such as the type of hazard, its location, and the necessary precautions, and the messages must be recorded in system logs.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by the Parking Attendant, Driver, System Administrator, Software Developer, and the Municipality.
- **Acceptance Criteria:**

    - The system should enable the parking attendant to send notifications to all active users by entering the urgent message text from the system interface.
    - Messages must appear on mobile applications and LED screens within 5 seconds at the latest.
    - All published messages should be automatically recorded in the system logs.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-219«trace»(Disabling automatic systems in emergency situations)
    - REQ-203«trace»(Messaging infrastructure)
- **Restrictions:** This requirement should be enforced so that only authorized parking attendants can use it.
- **Assumptions:** To implement this requirement, it is assumed that user devices have an internet connection and digital displays are operational.
- **Risks:** Implementing this requirement may carry the risk of causing panic in the public domain due to false message broadcasting and the reliability of the system may be undermined.
  Situation: The current status of this requirement is outlined as a draft.
- **Responsible:** Development Team – Backend and Development Team – Frontend are responsible for the implementation of this requirement.

Requirement REQ-223

- **Requirement ID:** REQ-223
- **Title:** Manual Verification of Vehicle Entry/Exit
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system should allow parking attendants to manually verify parking lot entry/exit movements in the event that IoT devices (sensors, cameras, etc.) malfunction or fail to transmit data. This requirement was developed to address the need to maintain up-to-date occupancy information and should allow transactions to be processed by entering license plate or vehicle information.
- **Priority:** This requirement has been determined to be a high priority.
- **Stakeholders:** This requirement is requested or influenced by the parking attendant, system administrator, and software development teams.
- **Acceptance Criteria:**

    - The system should enable the creation of a vehicle entry/exit record by entering the license plate or vehicle type information on the manual verification screen.
    - Entry/exit registration should be processed in the same way as IoT data, and parking lot occupancy status should be updated.
    - All transactions must be recorded in the system logs and viewable by the system administrator.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-216 «refine» (Manual Update of Parking Space Status)
    - REQ-208«trace»(License plate identification)
- **Restrictions:** This requirement is only applicable when a “no data retrieval” situation is detected by the system.
- **Assumptions:** For this requirement to apply, it is assumed that the parking attendant is in a position where he can physically see and identify the vehicle.
- **Risks:** Implementing this requirement may lead to deviations in occupancy information and reporting in the system due to the risk of incorrect manual recording.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Development Team – Backend and Development Team – Frontend are responsible for the implementation of this requirement.

Requirement REQ-224

- **Requirement ID:** REQ-224
  Title: Heat Map Display for Reservation Conflicts
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** To identify parking areas with high reservation density and a high probability of conflict, the system should generate a heat map based on time and location and make it available to the parking attendant. This visual representation will be used to facilitate staff adjustments.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or influenced by the parking attendant, system administrator, software developers, and data scientists.
  Acceptance Criteria:
    - The system should generate a heat map weekly by analyzing booking conflicts.

    - The parking attendant should be able to view this map from the panel and make arrangements according to density points.
    - The map should include information such as time intervals, parking area ID, and number of overlaps.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-016 «deriveReqt»(Advance Parking Reservation)
- **Restrictions:** This requirement is only available to authorized parking attendants.
- **Assumptions:** For this requirement to be implemented, it is assumed that reservation data is recorded completely and accurately in the central database.
- **Risks:** Implementing this requirement could run the risk of maps becoming outdated due to late data processing, potentially leading to misdirection by the parking attendant.
- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** Development Team – Backend and Development Team – Frontend are responsible for the implementation of this requirement.

Requirement REQ-225

- **Requirement ID:** REQ-225
- **Title:** Authorized Officials Ability to Edit and Cancel Reservations.
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** Parking attendants with editing authority can update or cancel active reservations. By clicking on the reservation detail, you can edit information such as start and end times, allocated parking space, and vehicle license plate. Clicking the "Cancel Reservation" button opens a confirmation window requiring you to select a reason for cancellation. All changes are immediately reflected in the system, and the transaction is recorded along with the attendant's user ID.
- **Priority:** This requirement has been set to low priority.
- **Stakeholders:** This requirement was requested or influenced by parking attendants, drivers, and software developers.
- **Acceptance Criteria:**
    - Only park attendants who have been assigned editing authority in the system can make changes to the reservation.

    - When you click on the reservation details, information such as start date, end date, allocated parking space and vehicle license plate should be editable.
    - Only park attendants who have been assigned editing authority in the system can make changes to the reservation.
    - The "Cancel Reservation" button must be visible and a confirmation window must appear requiring the user to select the reason for cancellation.
    - Any changes made should be updated instantly in the system and be viewable simultaneously by both the parking attendant and the driver.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-019«trace»(Reservation cancellation)
    - REQ-401«satisfy» (Hierarchical Role-Based Access Control)
    - REQ-215 «refine»(Reservation override logic)
- **Restrictions:** Only employees with "editing authority" defined in the system can perform these operations. Previous (completed) reservations cannot be edited or canceled. Cancellation reasons must be predefined; the employee cannot deviate from this list.
- **Assumptions:** In order to implement this requirement, it is assumed that the system has the capacity to process real-time data, the time information is working correctly, and the parking attendant is using an authorized mobile device with an internet connection.
- **Risks:** Implementing this requirement could create trust issues in the system if the parking attendant abuses their authority or chooses the wrong cancellation reason. Furthermore, concurrent edits could lead to inconsistencies in data synchronization and log integrity.
- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** Development Team – Backend and Development Team – Frontend are responsible for the implementation of this requirement.

Requirement REQ-226

- **Requirement ID:** REQ-226
- **Title:** View Parking Attendant's Shift Schedule and Receive Reminders
- **Category:** This requirement belongs to the functional requirement category.

- **Explanation:** Parking attendants should only be able to view their own shift schedule, including date, time, and location, via the system or mobile app. They should also receive automatic reminders from the system a certain amount of time before their shift starts. Shift information is predefined in the system, and the attendant is only authorized to view it.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** Parking attendants, parking owners, and software developers are affected by this requirement.
- **Acceptance Criteria:**
    - The parking attendant should only be able to view their own shift schedule via the system panel or mobile app.
    - Shift information should be presented clearly and concisely, with date, time and location details.
    - The system should send automatic reminders or notifications to the officer at a certain time before the start of the shift.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-231«trace»(User login and authorization)
    - REQ-101 «trace»(Parking lot owner mobile registration and entry)
    - REQ-214 «refine» (Shift handover notes)
- **Restrictions:**
    - A parking attendant can only view their own shift information; they cannot access the schedules of other attendants.
    - Only managers have the authority to adjust shift hours; the officer cannot change these hours.
- **Assumptions:** In order for this requirement to be implemented, it is assumed that notification permissions are enabled on the officer's device, the officer's personal account is defined and verified in the system, and the shift schedule is pre-defined through a central system.
- **Risks:** Implementing this requirement could lead to negative outcomes, such as the system failing to display updated shift information without an internet connection, or the officer missing their shift or being late if notifications are not delivered. Furthermore, the officer may not receive system alerts due to notification permissions being disabled.
- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** The Development Team – Backend and the Development Team – Frontend are responsible for fulfilling this requirement.

Requirement REQ-227

- **Requirement ID:** 227
- **Title:** Parking Attendant's Ability to Share Live Location
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** Parking attendants should be able to share their live location during patrols via the mobile app, and this location should be tracked in real time by superiors. This feature will be introduced to enhance officer safety and facilitate field coordination. For example, the system administrator should be able to receive notifications such as, "Officer Ahmet is currently patrolling the 2nd floor."
- **Priority:** This requirement has been set to low priority.
- **Stakeholders:** Parking attendant, parking lot owner, software developer, and data scientist stakeholders are affected by this requirement.
- **Acceptance Criteria:**
    - The parking attendant should be able to initiate live location sharing via the mobile app.
    - The shared location should be viewable in real time on the map by the responsible supervisor or system administrator.
    - Location data must be updated periodically (e.g. every 10 seconds) and accurately reflected in the system.
    - Location sharing should be able to be started and stopped manually.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-400«satisfy» (Access to Real-Time Location Data (GPS))
    - REQ-231«satisfy»(Secure login required)
- **Restrictions:** Live location should only be active during shifts and while in patrol mode. For this feature to work, the device must have location permission enabled and GPS enabled. Location accuracy may be limited within buildings (e.g., parking garage floors). The feature only works when there is an internet connection; the location cannot be updated if the connection is lost.
- **Assumptions:** In order for this requirement to be implemented, it is assumed that the officer's device is compatible with GPS feature and location sharing, the officer is logged into the system and is on an active shift.
- **Risks:** Implementing this requirement could disrupt on-site coordination if the officer's location is transmitted incorrectly or delayed. Manually disabling location sharing by the officer or due to a system error could pose a security risk. Furthermore, unauthorized access to or misuse of location data could raise serious concerns about data privacy and personal privacy.
- **Situation:** The current status of this requirement is designated as “draft.”
- **Responsible:** The Development Team – Backend is responsible for fulfilling this requirement.

Requirement REQ-228

- **Requirement ID:** REQ-228
- **Title:** Marking the Parking Area as Temporarily Reserved or Disabled Area
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must allow parking attendants to mark a parking space on the map as a temporary reservation or temporary disabled parking space via the mobile app. This requirement was developed to address the need for real-time site management, and marking actions must be reflected in the system in real time. Furthermore, marked spaces must be reversible to their original status if necessary.
- **Priority:** This requirement has been set to low priority.
- **Stakeholders:** This requirement was requested by the parking attendant and will be implemented by the software development team.
- **Acceptance Criteria:**
    - The system should allow the attendant to select any available parking space on the map via the mobile interface.
    - The selected parking area must be marked as a “Temporary Reservation” or “Temporary Disabled Space”.
    - The marking process should be visible in the system instantly.
    - The parking area should be able to be restored to its former state when necessary.
- **Dependencies:** This requirement is dependent on the completion of the following requirements:
    - REQ-211 «refine» (Parking Officer Marking the Parking Area)
    - REQ-205«trace»(Parking area closure logic)
- **Restrictions:** This requirement should only be applied if the officer is an authorized user. Furthermore, the temporary marking period may be limited for technical or operational reasons (e.g., a maximum of 2 hours).
- **Assumptions:** To implement this requirement, it is assumed that parking spaces are defined and up-to-date in the system, the attendant is using an internet-connected mobile device, and location services are active. It is also assumed that the attendant has received the necessary training and authorization.
- **Risks:** Implementing this requirement could lead to parking confusion and service disruptions due to attendants incorrectly marking spaces, either accidentally or maliciously. Furthermore, failure to promptly remove marked spaces could render the system outdated and negatively impact the user experience.
- **Situation:** The current status of this requirement is outlined as a draft.

- **Responsible:** Development Team – Backend and Development Team – Frontend are responsible for the implementation of this requirement.

Requirement REQ-229

- **Requirement ID:** REQ-229
- **Title:** Reporting Equipment Malfunctions via Mobile Device
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system should enable parking attendants to instantly notify their mobile devices of malfunctions in parking equipment such as barriers, ticket machines, or sensors. This requirement allows the technical team to quickly address issues. During the notification process, the attendant should be able to select the equipment type and location, and optionally enter a description. The notification should appear immediately in the system and be forwarded to the appropriate technical unit.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by the parking attendant and is impacted by its implementation by the software developer and data scientist.
- **Acceptance Criteria:**
    - The system should allow the parking attendant to access the “Report a Problem” option within the mobile app.
    - Equipment type and location selection must be possible.
    - The description field should be able to be filled in at the officer's discretion.
    - The fault notification should appear instantly in the system and be forwarded to the relevant technical team.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-217 «refine» (Sensor Error Reporting)
- **Restrictions:** This requirement is only available online due to the need for an internet connection. The function should be accessible only to authorized users. Multiple active fault reports cannot be made for the same piece of equipment.
- **Assumptions:** For this requirement to be implemented, it is assumed that all equipment in the parking area is defined in the system and the mobile device used by the attendant is compatible with the application.
- **Risks:** Implementing this requirement carries the risk of inaccurate notifications due to the officer selecting the wrong equipment or location. Furthermore, any technical delays in the system could result in delayed notifications.
- **Situation:** The current status of this requirement is outlined as a draft.

- **Responsible:** The Development Team – Backend is responsible for the implementation of this requirement.

Requirement REQ-230

- **Requirement ID:** REQ-230
- **Title:** Monitoring Environmental Risks and Taking Precautions in Park Areas
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must enable parking attendants to monitor environmental hazards (e.g., floods or smoke) through data from weather services and smart sensors. This requirement includes identifying at-risk parking zones to improve safety, temporarily disabling reservations when necessary, and sending alerts to appropriate individuals. The system must support both automatic and manual marking of risk zones.
- **Priority:** This requirement has been set to low priority.
- **Stakeholders:** This requirement was requested or influenced by parking attendants, drivers, data scientists, and software developers.
- **Acceptance Criteria:**
    - The system should be able to detect environmental risks such as floods and smoke in real time.
    - Risky areas should be determined automatically or by manual intervention.
    - Reservations should be temporarily disabled in affected parking areas.
    - The system should be able to send warnings and notifications to the relevant personnel.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-219 «refine» (Emergency management)
- **Restrictions:** This requirement must be implemented based on the accuracy and timeliness of weather and sensor data. Reservation cancellations can only be made by authorized users. Technical issues with sensor and weather API services may affect the operation of the automatic warning system.
- **Assumptions:** This requirement assumes the system has access to reliable weather APIs and a well-defined sensor infrastructure. It also assumes that all parking spaces are correctly located in the system.
- **Risks:** Implementing this requirement could lead to the risk of not detecting risky situations in a timely manner due to inaccurate or delayed environmental data collection. Furthermore, unnecessary reservation closures could lead to customer dissatisfaction. System failure or data loss could lead to critical

alerts not being sent. Sensor malfunctions or data misinterpretation could also create risks of both false alarms and underreporting.

- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** The Development Team – Backend is responsible for the implementation of this requirement.

Requirement REQ-231

- **Requirement ID:** REQ-231
- **Title:** Parking Attendant Login and Daily Control Panel Access
- **Category:** This requirement belongs to the functional requirement category.
- **Explanation:** The system must ensure that parking attendants can securely log in to the mobile app using the username and password assigned to them by the parking owner. This requirement supports efficient business processes by providing attendants with access to daily shift information, upcoming reservation numbers, and manual control tools. Furthermore, limiting failed login attempts by the hour and recording login records with timestamps and device IDs enhances security.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested and influenced by parking lot attendants and parking lot owners.
- **Acceptance Criteria:**
    - The officer should only be able to log in with the unique credentials assigned to them.
    - After logging in, the user panel should display the shift information for that day, the number of upcoming bookings, and manual control tools (e.g., space status update, violation reporting).
    - The system should monitor failed login attempts and limit them to 5 attempts per hour.
    - Each successful login should be recorded with a timestamp and the identification of the device used.
    - The authorization of the logged-in user must be verified by the system.
- **Dependencies:** This requirement is contingent upon completion of the following requirements:
    - REQ-101«trace» (Parking Owner Mobile Registration and Login)
    - REQ-226«trace»(Shift schedule integration)
    - REQ-115«trace»(Authorization assignment)

- **Restrictions:** Credentials must be pre-assigned by the provider. Login is not possible without an internet connection. Manual intervention may be required if your password is forgotten or your account is locked.
- **Assumptions:** Each parking attendant is assigned a unique user account in the system and their credentials are transmitted securely.
- **Risks:** An employee who is stuck with the hourly trial limit may not be able to complete their shift on time. Furthermore, incorrect device ID registration can make user tracking difficult.
- **Situation:** The current status of this requirement is outlined as a draft.
- **Responsible:** The Development Team – Backend is responsible for the implementation of this requirement.

D) System Requirements

REQ-400

- **Requirement ID:** REQ-400
- **Title:** The system should provide users with access to real-time GPS location data.
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must access real-time GPS data to determine users' current locations and provide location-based services. This requirement is critical for providing location-based services (navigation, parking suggestions, and location-based notifications) in parking management systems and must support core functionality such as navigation, parking suggestions, and location-based notifications. Without this capability, the system cannot recommend nearby parking spaces, provide directions, or provide personalized experiences to users.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by drivers, system administrators, parking providers, backend team, and frontend team.
- **Acceptance Criteria:**
    - The system must be able to successfully retrieve GPS data when the user grants mobile device location permissions.
    - GPS data should be updated within a maximum of 10 seconds.
    - Location data accuracy must be at least 10 meters.
    - The system should provide the ability to enter an alternative location when location permission is not granted.
    - The location service must provide 99.5% uptime.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:

    - REQ-G445 <<trace Reqt>> (Dynamic GPS Location Permission Management and Graceful Feature Downgrade) Required for mobile OS GPS APIs and location permission management
    - REQ-406 <<trace Reqt>> (Real-Time Map-Based Navigation) Required for navigation functionality using GPS data
    - REQ-407 <<trace Reqt>> (Redirect to Third Party Navigation Applications) Required for location-based routing services
    - REQ-008 <<trace Reqt>> (Receiving Navigation Directions to Selected Parking Area) GPS location data is required for parking lot navigation
    - REQ-504 <<satisfy>> (Google Maps and Google Places API Integration) Required for map and location services API integration
- **Restrictions:** This requirement should be implemented considering that GPS-based features will be disabled if the user does not grant location permission due to permission restrictions
- **Assumptions:** For this requirement to be implemented, it is assumed that roles will be predefined by the system administrator, each user will be assigned only one main role, and authorization will be checked on all API calls.
- **Risks:** Implementing this requirement may create privacy violations and data security issues due to the sensitive nature of location data; the accuracy of location-based services may be reduced due to areas with weak GPS signals, negatively impacting the user experience; and continuous location tracking may significantly shorten device battery life and lead to service interruptions.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team and Frontend Team are responsible for the implementation of this requirement.

REQ-401

- **Requirement ID:** REQ-401
- **Title:** The system must be able to manage user access with hierarchical role-based access control.
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must implement user access authorizations securely and manageably using a hierarchical Role-Based Access Control (RBAC) model. Each higher-level role must be configured to inherit all the authorizations of lower-level roles. This requirement is defined to ensure centralized management of system security and user authorization.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by system administrators, providers, auditors, drivers, and the backend team.
- **Acceptance Criteria:**
    - The higher-level role must inherit all the authority of the lower-level roles.

    - When role assignments are updated, the system must be able to instantly apply the new permissions.
    - Unauthorized access attempts should be logged and notifications should be generated.
    - Role changes should be traceable via audit trail.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-430 <<deriveReqt>> (Secure Session Management) Secure session management is required for role-based access control
    - REQ-402 <<trace Reqt>> (Optional Registration and Login with Third Party Authentication) Required for authentication system integration
    - REQ-408 <<trace Reqt>> (Configurable User Roles and Authorization Support) Required for role management and customizable authorization
    - REQ-225 <<deriveReqt>> (Authorized Officials to Edit and Cancel Reservations) Required for the authorization control of parking attendants
- **Restrictions:**
    - This requirement should be implemented by providing a modular structure so that the entire system is not affected in case of changes in the role structure due to architectural constraints.
    - This requirement should be implemented provided that authorization checks work without causing delays due to performance constraints.
    - This requirement should be implemented taking into account that role changes should be implemented in a way that does not affect past transactions due to data integrity restrictions.
- **Assumptions:** It is assumed that to implement this requirement, roles will be predefined by the system administrator, each user will be assigned only one main role, and authorization will be checked on all API calls.
- **Risks:** Implementing this requirement may create security vulnerabilities in the system due to the risk of incorrect authorization assignment; create management difficulties and maintenance complexity due to the risk of complex hierarchical structure; and decreases and delays in system performance may occur due to the risk of authorization updates.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-402

- **Requirement ID:** REQ-402
- **Title:** The system must support optional registration and login with third-party authentication.
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must allow users to optionally register and sign in with third-party authentication providers such as Google or Apple. This requirement is defined

to improve the user experience, speed up the registration process, and reduce password management overhead.

- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by the drivers, providers, system administrators, UI/UX design team, backend team, and frontend team.
  Acceptance Criteria:
    - The user must be able to log in/register with their Google or Apple credentials.
    - Secure authentication must be performed in accordance with the OAuth 2.0 protocol.
    - After third-party login, the system should request missing user information (phone, vehicle, payment).
    - Traditional and third-party input methods should have the same user experience standard.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-430 <<trace Reqt>> (Secure Session Management) Required for secure management of OAuth tokens
    - REQ-401 <<deriveReqt>> (Hierarchical Role-Based Access Control) Required for third-party post-authentication role assignment
    - REQ-448 <<trace Reqt>> (OAuth Integration with Smart Profile Pre-filling and Staged Enrollment) Required for Google and Apple OAuth integration
    - REQ-001 <<deriveReqt>> (Driver's Enablement to Register and Login to the Mobile Application) Required for basic authentication infrastructure
- **Restrictions:**
    - This requirement should be implemented taking into account the dependency on API policies of third-party services due to integration restrictions.
    - This requirement should be implemented provided that OAuth tokens are stored securely due to security restrictions.
    - This requirement must be implemented provided that alternative login methods are provided in the event of third-party service interruptions due to accessibility restrictions.
- **Assumptions:** For this requirement to be implemented, it is assumed that Google and Apple OAuth services will be accessible, users will understand the OAuth process and approve permissions, and the information provided by third-party services will be reliable.
- **Risks:** Implementing this requirement could lead to user access being blocked due to the risk of third-party service disruption; the risk of theft of OAuth tokens leading to a system compromise; and the risk of vendor-side policy changes rendering existing integrations dysfunctional.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Frontend Team and Backend Team are responsible for the implementation of this requirement.

REQ-403

- **Requirement ID:** REQ-403
- **Title:** The system must be able to collect and manage real-time occupancy data from IoT devices.
- **Category:** This requirement belongs to the functional and technical requirement category.
- **Description:** The system must collect and manage occupancy information (available, occupied, reserved) from IoT devices such as sensors, cameras, and bollards. This requirement is critical to providing drivers and parking providers with accurate and up-to-date parking status information.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by drivers, parking providers, system administrators, backend team, frontend team, and AI team.
- **Acceptance Criteria:**
    - The system must be able to receive real-time occupancy data from all connected IoT devices.
    - Data should be updated instantly in the user interface.
    - When erroneous data is detected, the system should create a log record and send a notification to the relevant personnel.
    - Protocols such as MQTT, HTTP and CoAP should be supported.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-501 <<trace Reqt>> (Receiving and Processing Real-time Parking Status Data from External IoT Sensors) Required for collecting parking status data from IoT sensors
    - REQ-003 <<deriveReqt>> (Real-time Parking Availability Display) Required for displaying IoT data in the user interface
    - REQ-217 <<trace Reqt>> (Sensor Error Reporting) Required for reporting IoT sensor errors
    - REQ-622 <<deriveReqt>> (Real-Time System Monitoring) Required for IoT network monitoring and system integrity
    - REQ-607 <<deriveReqt>>(Basic Data Collection for Advanced Real-Time System Integration)
- **Restrictions:**
    - This requirement should be implemented taking into account that manufacturer-specific IoT protocols may prolong the integration process due to integration limitations.
    - This requirement should be implemented considering that network interruptions due to connection limitations may cause data loss.
    - This requirement should be implemented under the condition that the energy consumption of IoT devices is continuously monitored due to energy efficiency constraints.

- **Assumptions:** For this requirement to be implemented, it is assumed that IoT devices will be constantly operational, standard protocols will be used, and the network infrastructure will have sufficient capacity.
- **Risks:** Implementing this requirement may lead to system slowdown and performance degradation due to the risk of heavy data traffic; system reliability may be compromised due to the risk of forged or manipulated data; and data flow may be interrupted and service continuity may be disrupted due to the risk of hardware failures.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** AI and Backend Team is responsible for the implementation of this requirement.

REQ-404

- **Requirement ID:** REQ-404
- **Title:** The system should provide future demand forecasting and early notification based on parking occupancy analysis.
- **Category:** This requirement belongs to the functional and business requirement category.
- **Description:** The system must predict future parking demand by analyzing historical occupancy data and provide proactive support to drivers and providers by providing early notification. This requirement is necessary to improve parking management efficiency through AI-based prediction mechanisms.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by drivers, parking providers, system administrators, backend team, frontend team, and AI team.
- **Acceptance Criteria:**
    - The system should be able to analyze the last 30 days of data.
    - The prediction accuracy rate must be at least 80%.
    - Predictive advance notice must be sent at least 15 minutes in advance.
    - Forecasts should be produced on hourly, daily and weekly basis.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-403 <<deriveReqt>> (Collecting and Managing Real-Time Occupancy Data from IoT Devices) Occupancy data source required for prediction model
    - REQ-415 <<trace Reqt>> (Multi-Channel Notification Support) Required for sending early notifications
    - REQ-501 <<trace Reqt>> (Receiving and Processing Real-time Parking Status Data from External IoT Sensors) Historical parking status data is required for analysis
    - «verify» Prediction accuracy test scenarios with historical data (80% target)

    - REQ-003 <<trace Reqt>> (Real-Time Parking Availability Display) Required to display the prediction results to the user
    - «trace» Multi-Channel Notification and Alert Module
- **Restrictions:**
    - This requirement should be implemented considering that model performance may degrade if there is insufficient data due to data limitations.
    - This requirement should be implemented taking into account that prediction calculations must be implemented in a way that does not strain system resources due to performance constraints.
    - This requirement should be implemented considering that uncontrollable environmental factors such as weather may affect forecast accuracy due to external factor constraints.
- **Assumptions:** For this requirement to be implemented, it is assumed that the data is uninterrupted and clean, and that users can understand and evaluate the predictions.
- **Risks:** Implementing this requirement could lead to decreased user satisfaction and questioned system reliability due to the risk of inaccurate predictions; could result in user irritation and increased likelihood of abandonment due to the risk of notification spam; and could lead to decreased prediction accuracy and compromised decision support processes due to the risk of model drift.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** AI Team and Backend Team are responsible for the implementation of this requirement.

REQ-405

- **Requirement ID:** REQ-405
- **Title:** The system should be able to work integrated with third-party parking management systems.
- **Category:** This requirement belongs to the technical requirement category.
- **Description:** The system must be able to exchange data with external parking management systems and reduce infrastructure costs by providing integration. This requirement is intended to ensure widespread system acceptance and compatibility with various platforms.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by parking providers, system administrators, integration partners, users, backend team, and frontend team.
- **Acceptance Criteria:**
    - Data exchange should be possible in JSON format via RESTful API.
    - Parking area layout, price information and reservation rules should be available.
    - Connection failures should be logged and notifications should be sent.
    - API response times should not exceed 3 seconds.
    - Data synchronization should occur at least every 5 minutes.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-420 <<trace Reqt>> (The system must be able to convert different formats in data exchange.) To convert XML, CSV, JSON formats from third-party systems in a system-appropriate format.
    - REQ-422 <<satisfy>> (The system should automatically log the user out when the session expires.) To securely manage API connections and ensure session security
    - REQ-635 <<deriveReqt>> (Checking Backend Latency) To meet the 3-second response time requirement for third-party API calls
    - REQ-629 <<satisfy>> (Ensuring Application Availability (%99.9 Uptime)) To ensure uninterrupted operation of integration connections and preservation of system availability
- **Restrictions:**
    - This requirement should be implemented taking into account that API access limits and version differences may complicate integration processes due to integration restrictions.
    - This requirement should be implemented taking into account that different data formats may require conversion due to data compatibility restrictions.
- **Assumptions:** For this requirement to be implemented, it is assumed that third-party APIs are always available and their documentation is up-to-date.
- **Risks:** Implementing this requirement may lead to system integration disruption and service disruption due to the risk of API changes; application functionality may be interrupted and user experience may be negatively impacted due to the risk of external system failures; and inconsistencies may occur between systems and lead to a loss of trust due to the risk of data synchronization errors.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-406

- **Requirement ID:** REQ-406
- **Title:** The system should provide real-time map-based navigation.
- **Category:** This requirement belongs to the functional requirement and user interface (UI) / user experience (UX) requirement category.
- **Description:** The system must provide real-time map-based navigation to help users reach their reserved or recommended parking spaces. This requirement is defined to improve the driver experience by calculating the optimal route, taking into account travel time, traffic conditions, and road conditions.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the drivers, UI/UX design team, system administrators, map service providers, frontend team, and backend team.
- **Acceptance Criteria:**

    - The user should be guided to the parking spot on the map.
    - Navigation should offer routes based on traffic and road conditions.
    - Route updates should occur within 10 seconds at most.
    - Voice guidance support should be provided.
    - The map interface must support day/night modes.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-400 <<trace Reqt>> (The system should provide users with access to real-time GPS location data.) To determine the user's current location for navigation and route calculation
    - REQ-421 <<satisfy>> (The system should support dark and light modes in the user interface.) For the map interface to support day/night modes
    - REQ-635 <<deriveReqt>> (Performing Backend Delay Control) For the requirement that route updates be performed within 10 seconds
    - REQ-629 <<satisfy>> (Ensuring Application Availability (99.9% Uptime)) To ensure uninterrupted navigation service and maintain map service accessibility
- **Restrictions:**
    - This requirement should be implemented considering that route accuracy may be reduced in areas where the GPS signal is weak due to environmental constraints.
    - This requirement should be implemented taking into account that pricing and usage policies of external services may be restrictive due to integration and cost constraints.
    - This requirement should be implemented taking into account that system functionality may be limited in offline mode due to offline operation restrictions.
- **Assumptions:** For this requirement to be implemented, it is assumed that the device can use location services effectively and that map service providers will provide uninterrupted service.
- **Risks:** Implementing this requirement may lead to users not being guided correctly and navigation failures due to the risk of external service interruptions; the system may suggest sub-optimal routes due to the risk of traffic data delays, resulting in loss of time; and security risks may arise due to the risk of GPS-related problems, and routing accuracy may be negatively affected.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Frontend Team and Backend Team are responsible for the implementation of this requirement.

REQ-407

- **Requirement ID:** REQ-407
- **Title:** The system should be able to direct users to third-party navigation applications.
- **Category:** This requirement belongs to the functional requirement and user interface (UI) / user experience (UX) requirement category.

- **Description:** The system should offer the option to redirect the user to their preferred third-party navigation application instead of their internal navigation system. This requirement is defined to reduce development and maintenance costs and to support user habits.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by drivers, UI/UX design team, system administrators, external map service providers, frontend and backend teams.
- **Acceptance Criteria:**
    - When redirecting the user to the external application, the target coordinates must be pre-filled.
    - Google Maps and Apple Maps support should be provided.
    - The user should be able to choose the default navigation app.
    - If the app is not installed, an alternative redirect should be offered.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-406 <<trace Reqt>> (The system must provide real-time map-based navigation.) To provide guidance to third-party applications as an alternative to the internal navigation system
    - REQ-400 <<trace Reqt>> (The system should provide users with access to real-time GPS location data.) For automatic filling of target coordinates and determination of the starting position
    - REQ-419 <<satisfy>> (The system must provide a consistent user experience across different platforms.) To manage URL scheme differences between iOS and Android platforms
    - REQ-421 <<satisfy>> (The system should support dark and light modes in the user interface.) To ensure theme consistency of the navigation guidance interface
- **Restrictions:**
    - This requirement should be implemented taking into account that the application to be redirected must be installed on the target device due to device restrictions.
    - This requirement should be implemented considering that different application invocation protocols may exist between iOS and Android operating systems due to platform restrictions.
- **Assumptions:** For this requirement to be implemented, it is assumed that popular map applications are installed on the device.
- **Risks:** Implementing this requirement may result in inconsistent user experiences and reduced satisfaction due to the risk of platform differences; may break system integration and lose functionality due to the risk of API or URL changes; and may lead to redirects to external applications, where users may refuse to switch and the process flow may be interrupted.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Frontend Team and Backend Team are responsible for the implementation of this requirement.

REQ-408

- **Requirement ID:** REQ-408
- **Title:** The system must provide configurable user roles and authorization support.
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must allow the addition of new user roles and the customization of their authorizations, in addition to the predefined roles. This requirement is defined to provide flexibility in user management to suit the needs of organizations.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by system administrators, managers, drivers, providers, backend and frontend teams.
- **Acceptance Criteria:**
    - The system must contain at least four predefined roles.
    - The system administrator must be able to define new roles.
    - Roles should be customizable with specific permissions.
    - Authorization changes should be tracked in the audit log.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-401 <<trace Reqt>> (The system must be able to manage user access with hierarchical role-based access control.) To ensure that the basic role structure is extensible and customizable
    - REQ-422 <<satisfy>> (The system should automatically log the user out when the session expires.) To implement role changes with secure session management
    - REQ-628 <<deriveReqt>> (Recording of Structural Error and Event Logging) To monitor role and authorization changes in the audit trail system
    - REQ-629 <<satisfy>> (Ensuring Application Availability (%99.9 Uptime)) To ensure uninterrupted operation of the authorization system and continuous accessibility of role controls
- **Restrictions:**
    - This requirement should be implemented considering that predefined roles cannot be deleted due to system integrity restrictions.
    - This requirement should be implemented taking into account that complex role configurations may negatively impact system performance due to performance constraints.
- **Assumptions:** It is assumed that managers have the necessary authority and knowledge to implement this requirement.
- **Risks:** Implementing this requirement may create security vulnerabilities and compromise system integrity due to the risk of misconfiguration; and the risk of overly complex system structure may make the system difficult to manage and maintain, and the likelihood of errors may increase.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-409

- **Requirement ID:** REQ-409
- **Title:** The system should be able to convert voice commands into structured commands with NLP and LLM.
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must convert user voice input into structured system commands through natural language processing and large language models. This requirement is defined to provide a hands-free experience and improve safety while driving.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the drivers, UI/UX team, system administrators, frontend, backend and AI teams.
- **Acceptance Criteria:**
    - The audio input must be converted to text and its meaning interpreted.
    - Structured command output must conform to the system command format.
    - The accuracy rate must be above 90%.
    - Turkish and English language support should be provided.
    - For ambiguous commands, explanatory dialogs should be initiated.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-410 <<trace Reqt>> (The system must be able to detect driver intent from voice commands.) In order to interpret the converted structured commands in accordance with the driver intent
    - REQ-635 <<deriveReqt>> (Checking Backend Latency) To complete NLP and LLM operations within the specified response time limits
    - REQ-421 <<satisfy>> (The system should support dark and light modes in the user interface.) To ensure theme consistency of the voice command feedback interface
    - REQ-629 <<satisfy>> (Ensuring Application Availability (%99.9 Uptime)) To ensure uninterrupted availability of voice command processing services
- **Restrictions:**
    - This requirement should be implemented taking into account that some functions may require an internet connection to function due to connection restrictions.
    - This requirement should be implemented taking into account that system performance may be degraded in noisy environments due to environmental constraints.
- **Assumptions:** For this requirement to be implemented, it is assumed that users will speak clearly and in accordance with the system language.
- **Risks:** Implementing this requirement may result in the system firing incorrect commands and exhibiting unexpected behavior due to the risk of low accuracy; may result in poor performance across different accents and reduced accessibility due to the

risk of biases in the model; may cause privacy concerns for users and negatively impact system adoption due to the risk of continuous location listening.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** The AI Team is responsible for implementing this requirement.

REQ-410

- **Requirement ID:** REQ-410
- **Title:** The system should be able to detect driver intent from voice commands.
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must determine the driver's parking intent based on the driver's voice command and match it with contextual goals (shopping, hospital visit, etc.). This requirement is defined to provide contextual parking recommendations.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by drivers, system administrators, parking providers, frontend, backend, and AI teams.
- **Acceptance Criteria:**
    - The system should be able to classify intent from voice commands.
    - The intent must be matched with the parking purposes defined in the system.
    - An accuracy rate of at least 85% must be achieved.
    - At least 15 different intent categories must be supported.
    - In case of unclear intentions, clarification should be obtained from the user.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-409 <<trace Reqt>> (The system should be able to convert voice commands into structured commands with NLP and LLM.) In order to perform intent analysis after converting the voice input into structured commands
    - REQ-400 <<trace Reqt>> (The system must provide access to real-time GPS location data for users.) For contextual intent detection and accurate matching of location-based parking suggestions
    - REQ-411 <<satisfy>> (The system should be able to offer parking suggestions to drivers based on intent and context.) In order to transfer the detected intent to the parking suggestion system and provide appropriate suggestions.
    - REQ-412 <<deriveReqt>> (The system should be able to provide personalized recommendations based on driver history and community data.) To optimize intent detection by combining it with the personalized recommendation algorithm
- **Restrictions:**
    - This requirement should be implemented taking into account that noise filtering is necessary for the proper operation of the system due to environmental constraints.
    - This requirement should be implemented taking into account that cultural differences may affect the interpretation of user intent due to contextual and cultural constraints.

- **Assumptions:** It is assumed that the user will clearly express his/her purpose for the implementation of this requirement.
- **Risks:** Implementing this requirement may result in incorrect parking recommendations being presented to users, resulting in a decrease in the quality of the experience, due to the risk of incorrect intent detection; and model performance may deteriorate over time, resulting in decreased recommendation accuracy, due to the risk of intent drift.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** The AI Team is responsible for implementing this requirement.

REQ-411

- **Requirement ID:** REQ-411
- **Title:** The system should be able to provide drivers with parking suggestions based on intent and context.
- **Category:** This requirement belongs to the functional and AI-based requirement category.
- **Description:** The system should recommend the most suitable parking space using the driver's stated intent (shopping, hospital visit, etc.), current location, and contextual data (date, time, weather). This requirement is defined to provide drivers with a personalized, efficient, and time-saving parking experience.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by drivers, parking providers, system administrators, AI team, backend and frontend teams.
- **Acceptance Criteria:**
    - The system should recommend parking spaces that match user intentions.
    - Suggestions should be sorted by location and date/time.
    - The recommendation accuracy must have at least an 85% success rate.
    - The user should be able to receive a cause-effect explanation regarding the proposed parking location.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-410 <<trace Reqt>> (The system should be able to detect driver intent from voice commands.) To provide the detected intent of the user as input to the parking suggestion algorithm.
    - REQ-400 <<trace Reqt>> (The system must provide access to real-time GPS location data for users.) To calculate contextual parking recommendations based on the user's current location
    - REQ-412 <<satisfy>> (The system should be able to provide personalized recommendations based on driver history and community data.) To enrich intent-based recommendations with personalization data
    - REQ-635 <<deriveReqt>> (Checking Backend Latency) To ensure that the recommendation calculation algorithms run at a speed that will not disrupt the user experience.

- **Restrictions:**
    - This requirement should be implemented considering that missing data in the system due to data limitations may reduce the quality of the recommendation.
    - This requirement should be implemented considering that an excessive number of recommendations may lead to user indecision due to user experience constraints.
- **Assumptions:** For this requirement to be implemented, it is assumed that the user's intent will be correctly identified and the parking lot data is up to date.
- **Risks:** Implementing this requirement may lead to user dissatisfaction and negatively impact traffic patterns due to the risk of incorrect recommendations; and users' trust in the guidance may decrease due to the risk of recommended fields being full and updates being delayed.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** AI Team, Backend Team

REQ-412

- **Requirement ID:** REQ-412
- **Title:** The system should be able to provide personalized recommendations based on driver history and community data.
- **Category:** This requirement belongs to the functional and AI-based requirement category.
- **Description:** The system should recommend the most suitable parking space using drivers' past parking preferences, current behavior, and community (similar drivers) data. This requirement was defined to increase user satisfaction through experience-based learning.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the drivers, AI team, backend and frontend teams.
- **Acceptance Criteria:**
    - The system should be able to offer suggestions based on past preferences.
    - Community data should be matched with driver behavior.
    - At least 80% personalization success must be achieved.
    - Recommendations should be updated according to changing conditions.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-400 <<trace Reqt>> (The system must provide users with access to real-time GPS location data.) To match personalized recommendations with location-based behavior patterns
    - REQ-413 <<trace Reqt>> (The system should enable users to save their favorite parking spaces.) To include favorite parking space preferences in the personalization algorithm

    - REQ-428 <<satisfy>> (The system must ensure the security and confidentiality of personal data.) For the secure processing of user history and behavior data
    - REQ-629 <<satisfy>> (Ensuring Application Availability (99.9% Uptime)) To ensure uninterrupted accessibility of personalized recommendation services
- **Restrictions:**
    - This requirement must be implemented taking into account that the confidentiality of personal data is subject to applicable regulations due to legal restrictions.
    - This requirement should be implemented considering that recommendation quality may be lower for users without historical data due to data limitations.
- **Assumptions:** For this requirement to be implemented, it is assumed that the user's previous data is recorded in the system.
- **Risks:** Implementing this requirement could lead to misleading users due to the risk that community recommendations could be inaccurate or malicious, and the recommendation engine could malfunction or crash completely due to the risk that data integrity could be compromised.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** The AI Team is responsible for implementing this requirement.

REQ-413

- **Requirement ID:** REQ-413
- **Title:** The system should enable users to save their favorite parking spots.
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system should provide easy access to frequently used or liked parking spaces by adding them to "favorites." This requirement is defined to enhance the user experience and save time.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by the drivers, UI/UX team, frontend and backend teams.
- **Acceptance Criteria:**
    - The user should be able to add and remove parking spaces from their favorite list.
    - The favorites list must be associated with the personal account.
    - The user should be able to filter and sort favorites.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-401 <<trace Reqt>> (The system must be able to manage user access with hierarchical role-based access control.) To securely associate favorite parking spaces with the user account
    - REQ-422 <<satisfy>> (The system should automatically log the user out when the session expires.) To protect access to favorite places with secure session management

    - REQ-412 <<satisfy>> (The system should be able to provide personalized recommendations based on driver history and community data.) Favorite parking data should be fed into the personalization algorithm.
- **Restrictions:**
    - This requirement should be implemented with the consideration that the number of favorites that can be defined for a user may be limited due to system design constraints.
- **Assumptions:** For this requirement to apply, it is assumed that the user is logged in.
- **Risks:** Implementing this requirement may reduce the accuracy and quality of the recommendation system due to the risk of outdated data in favorites.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Frontend Team and Backend Team are responsible for the implementation of this requirement.

REQ-414

- **Requirement ID:** REQ-414
- **Title:** The system should be able to display real-time occupancy status in the user interface.
- **Category:** This requirement belongs to the functional and user interface (UI) requirement category.
- **Description:** The system must be able to instantly display the occupancy status of selected or recommended parking spaces in the user interface using color, icon, and description. This requirement is defined to facilitate decision-making and inform the user.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by drivers, parking providers, frontend team, and system administrators.
- **Acceptance Criteria:**
    - Occupancy information should be displayed instantly and without delay.
    - Users should be able to see the parking lot status on the map.
    - Statuses such as full, empty, reserved should be distinguished with icons and colors.
    - The user should also be shown the last updated time.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-403 <<trace Reqt>> (Collecting and managing real-time occupancy data from IoT devices) To display real-time occupancy information on the interface by receiving it from IoT sensors
    - REQ-632 <<deriveReqt>> (Real-time occupancy data update) To update occupancy status instantly on the interface
    - REQ-635 <<deriveReqt>> (Performing Backend Delay Control) To display occupancy information on the interface without delay

    - REQ-629 <<satisfy>> (Ensuring Application Availability (%99.9 Uptime)) To ensure uninterrupted accessibility of real-time occupancy data
- **Restrictions:**
    - This requirement should be implemented considering that delays due to network constraints may prevent information from being displayed up to date.
- **Assumptions:** For the implementation of this requirement, it is assumed that the data flow is continuous and stable.
- **Risks:** Implementing this requirement may lead to misleading occupancy information, which could lead to users being misled and having difficulty finding a parking space; and the risk of errors in the graphical interface could lead to reduced user satisfaction and damage to the perception of professionalism of the system.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** The Frontend Team is responsible for the implementation of this requirement.

REQ-415

- **Requirement ID:** REQ-415
- **Title:** The system should be able to deliver notifications via multiple channels (mobile, email, in-app).
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must be able to deliver timely alerts, reservation reminders, and special offers to the user via the appropriate communication channel. This requirement is defined to increase information, security, and user interaction.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by drivers, system administrators, UI/UX team, frontend and backend teams.
- **Acceptance Criteria:**
    - The user should be able to determine his/her notification preference.
    - The system should send a reminder 10 minutes before the reservation time.
    - Notifications must be submitted in both Turkish and English.
    - It should be monitored whether the notifications are delivered or not.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-416 <<trace Reqt>> (The system should be able to store user preferences in the central profile.) To store and manage the user's notification channel preferences in the profile
    - REQ-401 <<satisfy>> (The system should be able to manage user access with hierarchical role-based access control.) To filter notification contents appropriate to different user roles
    - REQ-422 <<satisfy>> (The system should automatically log the user out when the session expires.) For the integration of session security and notification system

    - REQ-629 <<satisfy>> (Ensuring Application Availability (%99.9 Uptime)) To ensure uninterrupted availability of multi-channel notification services
- **Restrictions:**
    - This requirement should be implemented considering that push notifications cannot be sent without user consent due to permission restrictions.
- **Assumptions:** It is assumed that the user's preference for the implementation of this requirement is specified in the profile settings.
- **Risks:** Implementing this requirement may result in users becoming irritated and abandoning the application due to the risk of excessive notifications being sent; critical messages may not reach the user due to the risk of email notifications being caught in spam filters.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Frontend Team and Backend Team are responsible for the implementation of this requirement.

REQ-416

- **Requirement ID:** REQ-416
- **Title:** The system should be able to store user preferences in the central profile.
- **Category:** This requirement belongs to the functional and user experience (UX) requirement category.
- **Description:** The system must store users' selected language, notification preferences, favorite parking spots, and in-app settings within a centralized profile structure and maintain this information across sessions. This requirement is defined to provide a personalized and consistent user experience.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the drivers, frontend team, backend team, and UI/UX design team.
- **Acceptance Criteria:**
    - User preferences should be synced across multiple devices.
    - Changes should be saved instantly and reversible.
    - Preferences should be listed and editable in the profile interface.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-401 <<trace Reqt>> (The system should be able to manage user access with hierarchical role-based access control.) For secure storage of user preferences and role-based access control
    - REQ-413 <<trace Reqt>> (The system should enable users to save their favorite parking spaces.) To manage favorite parking space preferences within the central profile system
    - REQ-422 <<satisfy>> (The system should automatically log the user out when the session expires.) To protect user preferences with secure session management

    - REQ-629 <<satisfy>> (Ensuring Application Availability (%99.9 Uptime)) To ensure that user preferences are always accessible and synchronized
- **Restrictions:**
    - This requirement should be implemented provided that compliance with data protection regulations such as KVKK is ensured when storing user preferences due to legal restrictions.
- **Assumptions:** For this requirement to apply, it is assumed that the user is logged in.
- **Risks:** Implementing this requirement may reduce user satisfaction and question system reliability due to the risk of incorrectly recorded user preferences; may create inconsistencies in the user interface and reduce the quality of the experience due to the risk of loss of preference data.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Frontend Team and Backend Team are responsible for the implementation of this requirement.

REQ-417

- **Requirement ID:** REQ-417
- **Title:** The system must be able to record and monitor user activities on a daily basis.
- **Category:** This requirement belongs to the functional and traceability requirement category.
- **Description:** The system should record important actions performed by users within the application (login, reservation, cancellation, notification setting, etc.) in log files with timestamps, and this data should be available for auditing and support operations.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by system administrators, support team, security team and backend team.
- **Acceptance Criteria:**
    - Each action should be logged with the user ID and timestamp.
    - Logs should be archived periodically.
    - Erroneous transactions should be marked in a separate category.
    - Data integrity must be checked.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-401 <<trace Reqt>> (The system must be able to manage user access with hierarchical role-based access control) For logging user authentication and authorization data
    - REQ-430 <<trace Reqt>> (Secure Session Management) To securely track session-based activity records
    - REQ-609 <<trace Reqt>> (Encryption of Sensitive Data) For secure storage and encryption of log data
    - REQ-622 <<trace Reqt>> (Real-Time System Monitoring) For the integration of system performance and activity monitoring infrastructure

    - «satisfy» Activity recording module
    - «verify» Log accuracy and performance test scenarios
    - «trace» Security audit infrastructure and audit trail mechanism
- **Restrictions:**
    - This requirement should be implemented under the condition that only authorized persons can access the system logs due to security restrictions.
    - This requirement must be implemented taking into account that, due to legal restrictions, the data retention period must be defined within the limits set by the relevant legislation.
- **Assumptions:** For this requirement to be implemented, it is assumed that the application is used by logged-in users.
- **Risks:** Implementing this requirement may result in slower system performance and increased resource consumption due to the risk of excessive log data growth; and the risk of unauthorized access to log files may result in a privacy violation and the exposure of sensitive data.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-418

- **Requirement ID:** REQ-418
- **Title:** The system should be able to present content according to language preferences.
- **Category:** This requirement belongs to the functional and user experience (UX) requirement category.
- **Description:** The system must present all content and system messages translated into the appropriate language, taking into account the language preference defined in the user profile. This requirement is defined to support a multilingual user base.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by the drivers, frontend team, and UI/UX design team.
- **Acceptance Criteria:**
    - At least Turkish and English languages must be supported.
    - The language preference must be matched with the registered user profile.
    - System messages, labels, and error alerts must be provided with translation.
    - The infrastructure for adding new languages should be provided.
- **Dependencies:**
    - REQ-401 <<trace Reqt>> (The system must be able to manage user access with hierarchical role-based access control) For secure storage of user profile management and language preference data
    - REQ-421 <<trace Reqt>> (The system should support dark and light modes in the user interface) For language settings integrated with UI theme management
    - REQ-423 <<trace Reqt>> (The system must support accessibility standards for users with disabilities) For language settings to comply with accessibility standards

    - «satisfy» Multilingual content management module
    - «verify» Language change UI test cases
    - «trace» Profile preference infrastructure
- **Restrictions:**
    - This requirement should be implemented taking into account that the accuracy of translations can directly affect user satisfaction due to user experience constraints.
- **Assumptions:** For this requirement to be implemented, it is assumed that users will specify their language preference in the profile.
- **Risks:** Implementing this requirement may undermine user confidence and question the professionalism of the system due to the risk of incomplete or incorrect translations.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** The Frontend Team is responsible for the implementation of this requirement.

REQ-419

- **Requirement ID:** REQ-419
- **Title:** The system must provide a consistent user experience across different platforms.
- **Category:** This requirement belongs to the non-functional and user experience (UX) requirement category.
- **Description:** The system must provide a consistent experience for users when switching platforms by providing a similar appearance and behavior across the mobile app, web interface, and other platforms. This requirement is defined to reduce brand integrity and learning curve.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by the UI/UX team, frontend team, and drivers.
- **Acceptance Criteria:**
    - Colors, icons, menus, and buttons should be consistent across platforms.
    - The navigation structure should be similar on every platform.
    - Continuous improvement should be achieved based on user feedback.
- **Dependencies:**
    - «satisfy» UI component library
    - «verify» Multi-platform consistency tests
- **Restrictions:**
    - This requirement should be implemented taking into account that there may be technical limitations specific to each platform due to platform restrictions.
- **Assumptions:** It is assumed that the design guide will be implemented by all teams to implement this requirement.
- **Risks:** Implementing this requirement could undermine brand trust and negatively impact user experience due to the risk of incompatible appearances in the user interface.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** The Frontend Team is responsible for the implementation of this requirement.

REQ-420

- **Requirement ID:** REQ-420
- **Title:** The system must be able to convert different formats for data exchange.
- **Category:** This requirement belongs to the technical requirement category.
- **Description:** The system must be able to convert data received from or sent to third-party systems (e.g., XML, CSV, JSON) into a suitable internal format and present it in the appropriate format upon export. This requirement is defined to ensure compatibility in inter-system integrations.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by system administrators, integration teams, and the backend team.
- **Acceptance Criteria:**
    - The system must support XML, CSV and JSON formats.
    - Conversion operations must be performed without data loss.
    - Conversion speed between formats should not exceed 2 seconds.
- **Dependencies:**
    - «satisfy» Data transformation engine module
    - «verify» Format conversion test cases
    - «trace» Third-party integration mechanism
- **Restrictions:**
    - This requirement should be implemented with the consideration that incompatible data schemas may require special converters due to data compatibility restrictions.
- **Assumptions:** It is assumed that external system data formats are documented for the implementation of this requirement.
- **Risks:** Implementing this requirement may result in data loss and system reliability being compromised due to the risk of format mismatches; integration processes may fail and system operation may be interrupted due to the risk of data conversion errors.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-421

- **Requirement ID:** REQ-421
- **Title:** The system should support dark and light modes in the user interface.
- **Category:** This requirement belongs to the non-functional and user interface (UI/UX) requirement category.
- **Description:** The system must offer dark mode or light mode themes based on user preference. This requirement is defined to ensure visual comfort in different lighting conditions and to personalize the user experience.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by the drivers, frontend team, and UI/UX design team.
- **Acceptance Criteria:**

    - The user should be able to choose a theme in their profile settings.
    - The theme change should be applied instantly in the interface.
    - Colour contrasts must meet accessibility standards.
- **Dependencies:**
    - «satisfy» UI theme management module
    - «verify» Interface theme migration test cases
    - «trace» User preference configurations
- **Restrictions:**
    - This requirement must be implemented provided that the theme files provide consistent visual results across devices due to visual compatibility restrictions.
- **Assumptions:** For this requirement to be implemented, it is assumed that users will save their visual preferences in the profile.
- **Risks:** Implementing this requirement may lead to users experiencing eye strain and violating accessibility standards due to the risk of insufficient color contrast.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** The Frontend Team is responsible for the implementation of this requirement.

REQ-422

- **Requirement ID:** REQ-422
- **Title:** The system should automatically log the user out when the session expires.
- **Category:** This requirement belongs to the non-functional and security requirement category.
- **Description:** For security purposes, the system should terminate the session after a period of inactivity and redirect the user back to the login screen. This requirement is defined to prevent unauthorized access and increase session security.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the security team, system administrators, and the backend team.
- **Acceptance Criteria:**
    - The user session must be terminated after 10 minutes of inactivity.
    - A notification message should be displayed when the session expires.
    - The user should be able to continue operations by logging in again.
- **Dependencies:**
    - «satisfy» Session management module
    - «verify» Session duration test cases
    - «trace» Security audits
- **Restrictions:**
    - This requirement should be implemented considering that excessively short session duration may negatively impact user satisfaction due to user experience constraints.
- **Assumptions:** For this requirement to be implemented, it is assumed that users are likely to share the device with others.

- **Risks:** Implementing this requirement risks interrupting user operations and negatively impacting the experience due to session expiration; losing unsaved data and decreasing user satisfaction due to automatic session terminations.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-423

- **Requirement ID:** REQ-423
- **Title:** The system must support accessibility standards for users with disabilities.
- **Category:** This requirement belongs to the non-functional and accessibility requirement category.
- **Description:** The system must meet standards such as color blindness support, screen reader compatibility, and accessible labeling to ensure users with disabilities can easily use the application. This requirement is defined to support inclusive design.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by disabled users, the UX design team, the frontend team, and legal audit units.
- **Acceptance Criteria:**
    - WAI-ARIA labels must be used.
    - Color combinations should be suitable for color blindness.
    - Compatibility with screen readers must be ensured.
    - All user actions must be performed via keyboard.
- **Dependencies:**
    - «satisfy» Accessibility module
    - «verify» Accessibility test cases
- **Restrictions:**
    - This requirement should be implemented taking into account that due to accessibility restrictions, design flexibility must be balanced with accessibility requirements.
- **Assumptions:** For this requirement to be implemented, it is assumed that the application will be used by individuals with different user abilities.
- **Risks:** Implementing this requirement may result in violation of legal obligations due to the risk of lack of accessibility and users whose access is restricted abandoning the system.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** The Frontend Team is responsible for the implementation of this requirement.

REQ-424

- **Requirement ID:** REQ-424
- **Title:** The system must be able to maintain its performance under high traffic.
- **Category:** This requirement belongs to the non-functional and performance requirement category.

- **Description:** The system must maintain functionality and responsiveness while serving a large number of concurrent users. This requirement is defined to ensure application scalability and user experience.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by system administrators, infrastructure engineers, backend team and users.
- **Acceptance Criteria:**
    - The system must be able to serve 10,000 concurrent users with a response time of less than 2 seconds.
    - During traffic increases, system response time should deteriorate by a maximum of 20%.
    - Automatic load balancing mechanism should be activated.
- **Dependencies:**
    - «satisfy» Load balancing and caching modules
    - «verify» Load test scenarios
- **Restrictions:**
    - This requirement should be implemented taking into account that the system may operate with limited resources due to hardware limitations.
- **Assumptions:** For this requirement to be implemented, it is assumed that user density will increase on special days.
- **Risks:** Implementing this requirement may lead to decreased customer satisfaction and negatively impacting the brand image due to the risk of system crashes.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-425

- **Requirement ID:** REQ-425
- **Title:** The system must regularly take and store database backups.
- **Category:** This requirement belongs to the technical and security requirement category.
- **Description:** The system must automatically back up databases containing user data and configuration information daily and store the backups in a secure environment. This requirement is mandatory to prevent data loss and as part of the disaster recovery plan.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by system administrators, security teams, and database administrators.
- **Acceptance Criteria:**
    - The backup process should be performed automatically every day.
    - Backups should be stored in encrypted format.
    - Backup files should be retained for 30 days.
    - Restoration operations should be completed within 5 minutes.
- **Dependencies:**
    - «satisfy» Database backup module
    - «verify» Backup and restore test scenarios
    - «trace» Data security policy

- **Restrictions:**
    - This requirement should be implemented taking into account that backup operations should be implemented in a way that does not negatively impact system performance due to performance constraints.
- **Assumptions:** It is assumed that sufficient disk space is available during the backup process to implement this requirement.
- **Risks:** Implementing this requirement may result in critical data loss and compromise system restoreability due to the risk of failed backups; and may result in sensitive data being accessed by unauthorized parties due to the risk of backup files not being encrypted.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-426

- **Requirement ID:** REQ-426
- **Title:** Emergency Notifications
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must provide a comprehensive emergency notification system that will quickly and effectively notify all users in critical emergency scenarios (fire, earthquake, security breach, emergency evacuation, terrorist threat, natural disaster, etc.). This requirement was developed with the motivation to protect users' lives and property. Notifications will be delivered via multiple channels (in-app push notifications, email, SMS, voice alerts) and will be delivered to all users as quickly as possible. The system will send customized messages based on the type of emergency, guiding users to take the appropriate action.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by drivers, parking providers, system administrators, backend team, frontend team.
- **Acceptance Criteria:**
    - When an emergency notification is triggered, a simultaneous message should be sent via all predetermined communication channels (push, SMS, email).
    - Notifications must be delivered to users within 5 seconds from the moment they are triggered by the system administrator.
    - Notification content should be customized to the type of emergency and include clear, understandable, and action-oriented guidance.
    - The system should log all emergency notifications sent with timestamp, scope, content and delivery status.
    - The notification system must be able to operate without interruption even in high traffic situations and not exceed system resources.
- **Dependencies:**
    - «satisfy» Push Notification Service (Firebase, APNs)
    - «satisfy» Email Service Provider
    - «deriveReqt» REQ-415 (Multi-Channel Notification Support)
    - «satisfy» Email Service Provider

    - «satisfy» SMS Gateway Service
- **Restrictions:**
    - This requirement must be implemented in a way that ensures 24/7 uninterrupted operation and guaranteed notification delivery during high traffic times due to technical limitations.
    - This requirement is due to legal and ethical constraints; the content of emergency notifications should only be implemented in accordance with relevant regulations and ethical guidelines.
- **Assumptions:** For this requirement to be implemented, it is assumed that all users have at least one active communication channel (phone, email) and that this information is kept up to date in the system, emergency management teams have the authority to trigger notifications and are trained in this regard, and third-party notification service providers offer high availability.
- **Risks:** Implementing this requirement could endanger life and property due to the risk of notifications being delayed, not delivered, or incorrect content being sent; could create a critical security risk due to the risk of the emergency notification system failing in the event of a system failure; and could lead users to ignore future alerts due to the risk of false alarms or excessive notifications.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team and Frontend Team are responsible for the implementation of this requirement.

REQ-427

- **Requirement ID:** REQ-427
- **Title:** Multilingual User Interface Support (Turkish and English)
- **Category:** This requirement belongs to the category of functional requirements, user interface (UI) and user experience (UX) requirements.
- **Description:** The system must provide multilingual support in the user interface, with a minimum of Turkish and English. This requirement was motivated by the need to ensure that users with different language preferences can easily use the system and to appeal to an international user base. Language options should be instantly changeable based on user preferences and consistently applied throughout the application. The system's architecture will be flexible and modular, allowing for the easy addition of new languages (German, French, Spanish, etc.) in the future.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by international users, tourist drivers, multinational parking providers, Frontend Team, Backend Team, UI/UX Team, all system users, translation teams and customer support teams.
- **Acceptance Criteria:**
    - The system should detect the default language of the user device at the first startup and select the language automatically or take the user's preference into account.

    - When the user changes the language via the application settings, all interface elements (menus, buttons, labels, messages) should instantly switch to the selected language.
    - Both Turkish and English language packs must be able to fully cover all interface texts.
    - New languages should be integrated into the system dynamically via system updates or Over-The-Air (OTA) updates.
    - Language change should be preserved throughout the user session and should not require re-login
- **Dependencies:**
    - «satisfy» Multilingual language packs and translation files (JSON, XML formats)
    - «satisfy» Translation management system and localization infrastructure
    - «satisfy» Application settings and user preferences database
    - «satisfy» OTA update mechanism
    - «verify» UI/UX test scenario that checks whether all interface elements are updated instantly and completely when the user switches between different languages
- **Restrictions:**
    - This requirement must be implemented in such a way that all user interface text is not hard-coded in the source code, but is loaded dynamically only from the language files, due to technical constraints.
    - This requirement should be implemented in accordance with responsive design principles so that text lengths in different languages do not disrupt the user interface design due to design constraints.
- **Assumptions:** For this requirement to be implemented, it is assumed that users have basic technical knowledge of language preference management and can change the language from the application settings, that the translation quality will be professional and that technical terms will be correctly localized, that language packs will be updated regularly and that translations will be added for new features.
- **Risks:** Implementing this requirement may negatively impact the user experience and lead to user dissatisfaction due to the risk of incomplete, incorrect, or poor-quality translations; may lead to the user being unable to use the interface due to the risk of system errors during the language change process; and system stability may be negatively affected due to the risk of incompatibility or performance issues in the system when adding a new language.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team and Frontend Team are responsible for the implementation of this requirement.

REQ-428

- **Requirement ID:** REQ-428
- **Title:** User Consent and Preference Management Regarding the Use of Personal Data

- **Category:** This requirement belongs to the functional and legal/regulatory requirement categories.
- **Description:** The system must obtain explicit and informed consent from users regarding the collection, processing, and use of personal data, in accordance with legal regulations. This requirement was developed to ensure full compliance with data protection regulations such as KVKK and GDPR and to protect users' data rights. Users must be able to view, change, or revoke their consent at any time. The system will provide users with granular control by providing separate consent mechanisms for each data use purpose.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is affected by drivers.
- Parking Providers, System Administrators, Backend Team, Frontend Team, AI Team

- **Acceptance Criteria:**
    - The user must see a detailed and clear data usage confirmation screen upon initial registration or before using features that require data collection.
    - Each type, time and scope of approval given by the user must be stored securely in an encrypted manner along with the date/time information.
    - The user should be able to manage data usage consent preferences by category (marketing, analytics, personalization, etc.) through account settings.
    - When consent is revoked, the relevant data usage processes must be stopped immediately and the user must be informed.
    - Approval texts and processes can be updated when legal requirements change and the user can be notified for re-approval.
- **Dependencies:**
    - «satisfy» KVKK compliance modules and legal requirements database
    - «satisfy» User management system and approval tracking infrastructure
    - «satisfy» Encryption and secure data storage system
    - «trace» Data Privacy Preference Management Module (module that performs consent, cancellation, logging and monitoring operations)
    - «verify» Users' ability to change, cancel and resubmit their preferences in different consent scenarios should be verified through KVKK compliance tests.
- **Restrictions:**
    - This requirement should be implemented in such a way that approval texts and processes are regularly updated and subject to legal review due to legal constraints.
    - This requirement must be implemented so that no personal data processing activities are carried out without user consent due to legal restrictions.
- **Assumptions:** For this requirement to be implemented, it is assumed that users will consciously and voluntarily indicate their preferences by reading the consent texts, the legal team will regularly review and update the legal compliance of the consent texts, and system administrators will have adequate training on KVKK requirements.
- **Risks:** Implementing this requirement could lead to legal sanctions and penalties due to the risk of incomplete, unclear, or incorrect approval management; user rights could be

violated due to the risk of malfunctions in the approval system; and regulatory authorities could impose heavy penalties due to the risk of failure to follow legal requirements.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team, Frontend Team and AI Team are responsible for the implementation of this requirement.

REQ-429

- **Requirement ID:** REQ-429
- **Title:** Rate Limiting and Abuse Detection Mechanisms
- **Category:** This requirement belongs to the non-functional requirement category.
- **Description:** The system must implement comprehensive rate limiting and abuse detection mechanisms to prevent malicious attacks, excessive usage attempts, and misuse of system resources. This requirement was motivated by the need to protect system security, ensure fair resource usage for all users, and maintain optimal system performance. The system will prevent Denial of Service (DoS) attacks and brute force attacks by providing access control with limits defined on a per-user, per-IP address, and per-API key basis.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or impacted by system administrators, security teams, all system users, Backend Team, Security Team, API users, and third-party integrations.
- **Acceptance Criteria:**
    - Access should be temporarily blocked when API requests exceed predetermined thresholds (e.g., 10 requests per second, 100 requests per minute)
    - The abuse detection algorithm must recognize unusual access patterns (repeated failed login attempts, abnormal traffic spikes) in real time
    - The system should log all detected exploit attempts with timestamp, source IP, user information, and action type.
    - Blocking and warning mechanisms should be monitored via the admin panel and automatic notifications should be sent for critical situations.
    - There should be a whitelist mechanism to prevent legitimate users from being accidentally blocked.
- **Dependencies:**
    - «satisfy» API management platform and traffic monitoring infrastructure
    - «satisfy» Security monitoring system and anomaly detection algorithms
    - «trace» REQ-421 (Real Time System Monitoring Panel)
- **Restrictions:**
    - This requirement should be implemented in a balanced manner so that rate limiting limits do not negatively affect the user experience due to technical constraints.
    - This requirement should be implemented by defining different breakpoints for different user types (regular user, premium user, API partner) due to user profile restrictions.

- **Assumptions:** For this requirement to be implemented, it is assumed that the majority of legitimate users will remain within normal usage limits and use system resources reasonably, that system administrators have the ability to detect and respond to security incidents in a timely manner, and that rate limiting configurations will be regularly reviewed and optimized.
- **Risks:** Implementing this requirement could lead to legitimate users being blocked from the system and reduced user satisfaction due to the risk of false positive detections; the overall usability of the system could be negatively impacted due to the risk of overly stringent restrictions; and system security could be compromised due to the risk of advanced attacks bypassing rate limiting mechanisms.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-430

- **Requirement ID:** REQ-430
- **Title:** Secure Session Management
  Category: This requirement belongs to the non-functional requirement category.
- **Description:** The system must securely manage user sessions. This includes automatic token expiration after a certain period of time (token expiration), the use of refresh tokens, and forced logouts when suspicious or compromised sessions are detected. This requirement forms the foundation of the system's security infrastructure, protecting user accounts from unauthorized access and preventing session hijacking attacks. Secure session management, critical to modern web applications, must ensure maximum security without negatively impacting the user experience.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by drivers, parking providers, system administrators, and the Backend Team.
- **Acceptance Criteria:**
    - Access tokens should automatically become invalid after the specified time (maximum 15 minutes)
    - With the renewal token, the user should be able to get a new access token without logging in again (maximum 7 days validity)
    - When suspicious sessions are detected (different IP, device change), the relevant user should be automatically logged out from all devices.
    - Session management processes (login, logout, token renewal, forced logout) should be logged and auditable.
    - The user must be able to view their active sessions and terminate unwanted sessions
- **Dependencies:**
    - «derivedReqt» REQ-401 – Hierarchical Role-Based Access Control
    - «derivedReqt»REQ-402 – Third Party Authentication
    - «verify» Manual & automatic tests of login with different IP/device test, token expiration test and forced logout scenario must be completed successfully.

    - «trace» Must be traceable by the security module that tracks session management and security events
    - «satisfy» Authentication module
    - «satisfy» Security monitoring system
    - «satisfy» JWT token management libraries
- **Restrictions:**
    - This requirement should be implemented by setting session duration limits in a way that does not negatively impact the user experience due to technical and user experience constraints.
    - This requirement must be implemented in a way that token sizes do not overload mobile network traffic due to network constraints.
    - This requirement must be implemented without compromising any security measures while ensuring multi-device support due to security constraints.
- **Assumptions:** For this requirement to be implemented, it is assumed that users log in from secure devices, the system time is synchronized on all servers, an encrypted connection is provided using the HTTPS protocol, and users have valid contact information for security notifications.
- **Risks:** Implementing this requirement may lead to unauthorized access due to the risk of token theft or misuse; user experience may be negatively impacted due to the risk of overly stringent security policies; user accounts may be compromised due to the risk of session fixation attacks; and the risk of false positive alarms in detecting suspicious activity may lead to unnecessary system interventions and user dissatisfaction.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-430

- **Requirement ID:** REQ-431
- **Title:** Feature Flag Management
- **Category:** This requirement belongs to the technical requirement category.
- **Description:** The system should support feature flag management. This allows specific features to be dynamically enabled or disabled for different user segments or environments (development, test, production, etc.). This framework should enable the implementation of A/B testing, gradual rollout, beta testing, and rollback strategies. The feature flag system minimizes risk by supporting continuous delivery practices, enabling controlled software development processes. This approach allows teams adopting a trunk-based development methodology to safely experiment in a production environment.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or affected by drivers, parking providers, system administrators, Backend Team, and Frontend Team.
- **Acceptance Criteria:**
    - The system should be able to define separate flags for each property (boolean, string, JSON object types)

    - Flags can be enabled or disabled in real time (with a maximum delay of 30 seconds)
    - Special flag configurations can be made for specific user groups, device types or environments.
    - Flag changes should be trackable and logged with timestamp and user information.
    - Percentage-based distribution support should be provided (e.g., new feature for 10% of users)
- **Dependencies:**
    - «satisfy» Configuration management module
    - «satisfy» User segmentation system
    - «satisfy» Distributed configuration management (Redis, Consul)
    - «verify» Feature flag test cases
    - «trace» System module that tracks feature activation/deactivation process
- **Restrictions:**
    - This requirement must be implemented carefully to ensure that misconfiguration due to configuration restrictions does not result in critical features being disabled for all users.
    - This requirement should be implemented by limiting the number of feature flags so that it does not increase system complexity due to technical constraints.
    - This requirement should be implemented with minimal system load, <1 ms per request, due to performance constraints.
- **Assumptions:** For this requirement to be implemented, it is assumed that developers are trained in the management of feature flags, feature flag cleanup processes will be performed regularly, an environment-specific configuration management infrastructure is available, and monitoring and alerting systems can monitor flag status changes.
- **Risks:** Implementing this requirement may lead to undesirable features being released to the wrong group of users due to the risk of incorrect flag usage; it may lead to the accumulation of technical debt due to the risk of retaining old flags in the system; complex rollback scenarios may occur due to the risk of dependency chains between flags; and inconsistent system behavior may occur due to the risk of configuration differences between environments.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team and Frontend Team are responsible for the implementation of this requirement.

REQ-432

- **Requirement ID:** REQ-432
- **Title:** Automatic Backup (Daily/Weekly)
- **Category:** This requirement belongs to the non-functional requirement category.
- **Description:** The system must automatically back up all critical data (user information, parking transaction records, system configurations, log files) daily and/or weekly. These backups must be made to a secure storage area to ensure business continuity in the

event of potential system failures, data loss, natural disasters, or cyberattacks. The backup strategy should follow the 3-2-1 rule (3 copies, 2 different media, 1 external location). Storage optimization should be achieved using incremental and differential backup techniques, while point-in-time recovery capability should be provided.

- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by drivers, parking providers, system administrators, and the Backend Team
- **Acceptance Criteria:**
    - The system should perform automatic backups according to the defined schedule (daily at 02:00, weekly on Sunday at 01:00).
    - Backup files must be stored in a secure environment (encrypted storage) and protected from unauthorized access.
    - A full backup (weekly) and incremental backup (daily) strategy should be implemented
    - The restore test must be completed successfully monthly and should not take less than 4 hours.
    - Backup integrity checks should be performed automatically and alerts should be issued on detection of corruption.
- **Dependencies:**
    - «satisfy» Storage infrastructure (AWS S3, Azure Blob Storage)
    - «satisfy» Backup software or services (pg_dump, mysqldump, automated scripts)
    - «satisfy» Cryptographic libraries
    - «trace»Backup Management Module (system component that tracks backup scheduling, execution and restore processes)
    - «verify» Backup and Restore Test Scenarios (full backup, incremental backup, integrity check and recovery tests.
- **Restrictions:**
    - This requirement should be implemented taking into account additional storage costs and network bandwidth requirements for backup operations due to budget and infrastructure constraints.
    - This requirement should be implemented with the understanding that temporary decreases in system performance may be experienced during the backup window due to performance constraints.
    - This requirement must be implemented in accordance with the requirement that backed up data be retained for at least 5 years due to regulatory restrictions.
    - This requirement should be implemented taking into account any delays that may occur during inter-region data replication due to technical constraints.
- **Assumptions:** For this requirement to be implemented, it is assumed that system administrators will configure backup schedules and retention policies, sufficient storage capacity and network bandwidth will be available, disaster recovery procedures will be documented and tested, and the cloud provider will meet SLA requirements.
- **Risks:** Implementing this requirement may result in data loss and service interruption due to the risk of backup failure; restore failure due to the risk of corrupted backups;

budget allocations may be exceeded due to the risk of increased backup storage costs; and legal compliance violations due to the risk of inadequate data retention policies.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-433

- **Requirement ID:** REQ-433
- **Title:** Separate Software Environments (Development, Test, Staging, Production)
- **Category:** This requirement belongs to the technical requirement category.
- **Description:** To ensure reliable and secure software delivery, the system must maintain separate development, testing, staging, and production environments. Each environment must operate with different configurations, data sets, and access permissions. This separation of concerns approach ensures quality control at every stage of the software delivery pipeline while maintaining the stability of the production environment. Environment parity should be ensured using Infrastructure as Code (IaC) principles, and configuration drift should be prevented.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is affected by drivers, parking providers, Backend Team, Frontend Team, and AI Team.
- **Acceptance Criteria:**
    - Each environment must be isolated from each other, either physically or virtually (containers, virtual machines)
    - A clear upgrade pipeline must be defined to avoid data and configuration confusion between environments
    - Only authorized personnel should have access to the production environment (role-based access control)
    - Separate database instances and configurations must be used for each environment
    - The staging environment must be an exact replica of the production environment (data anonymized)
- **Dependencies:**
    - «satisfy» Container orchestration platform (Kubernetes, Docker Compose)
    - «satisfy» Infrastructure as Code tools (Terraform, CloudFormation)
    - «satisfy» CI/CD pipeline automation (Jenkins, GitLab CI, GitHub Actions)
    - «trace» CI/CD Process Execution Module (system component that manages feature flag propagation, pipeline triggering, and environment migrations)
    - «verify» Pipeline Test Cases (verify feature deployment, rollback and cross-environment transitions)
- **Restrictions:**
    - This requirement should be implemented taking into account that environmental equality may be limited due to budget and resource constraints.
    - This requirement must be implemented considering cross-environment network connectivity and latency issues due to technical constraints.

    - This requirement should be implemented considering the cost implications of maintaining multiple environments due to budget constraints.
    - This requirement should be implemented considering the difficulties in data synchronization across environments due to technical limitations.
- **Assumptions:** To implement this requirement, it is assumed that the development team will adhere to environment management best practices, sufficient infrastructure budget and resources will be allocated, automated deployment processes will be implemented, and environment-specific secret information management will be configured correctly.
- **Risks:** Implementing this requirement may lead to inconsistencies between environments due to the risk of configuration drift; security breaches and data leaks may occur due to the risk of unauthorized access to the production environment; system performance may be negatively affected due to the risk of competition between resources; and delays in development and deployment may occur due to the risk of complexity in the environment setup and teardown processes.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team is responsible for the implementation of this requirement.

REQ-434

- **Requirement ID:** REQ-434
- **Title:** IoT Device Failure Management and Gentle Degradation
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** When IoT devices (sensors, smart bollards, etc.) fail to report data for more than 60 seconds, the system should temporarily mark them as disabled, use the last known status in decision-making, log the event, and notify relevant personnel via the dashboard. This self-healing capability maintains service continuity by ensuring system resilience against inevitable device failures in the IoT ecosystem. Proactive interventions should be enabled through pattern recognition using predictive maintenance algorithms.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is affected by drivers, parking providers, Backend Team, and AI Team.
- **Acceptance Criteria:**
    - If the device is not able to receive data for 60 seconds, it should be marked as "temporarily disabled".
    - The last known device state should be cached and used for decision making
    - Device failure events must be recorded in structured log format (timestamp, device_id, failure_type)
    - Real-time alerts and notifications should be sent on the dashboard
    - The automatic recovery mechanism should return the device to active state when it is restored.
- **Dependencies:**
    - «satisfy» IoT device communication protocols (MQTT, HTTP, CoAP)
    - «satisfy» Real-time messaging system (Apache Kafka, RabbitMQ)
    - «satisfy» Time series database (InfluxDB, TimescaleDB)

    - «deriveReqt» REQ-403 (Real-Time Occupancy Data Collection from IoT Devices)
    - «trace» Device Data Collection and Publishing Module (software component that processes data from the device in real time and transmits it to other systems)
    - «trace» Fault and Anomaly Tracking Module (module that analyzes repetitive error signals from devices)
- **Restrictions:**
    - This requirement should be implemented in a way that prevents network latency and packet loss due to technical constraints from negatively impacting IoT connectivity.
    - This requirement must be implemented considering device battery levels and power management due to hardware limitations.
    - This requirement must be implemented to support large IoT deployments due to technical and scalability constraints.
    - This requirement must be implemented by utilizing edge computing capabilities in local decision-making processes due to architectural constraints.
- **Assumptions:** For this requirement to be implemented, it is assumed that IoT devices support the heartbeat/keepalive mechanism, the network infrastructure provides sufficient capacity for IoT traffic, device firmware updates can be managed remotely, and maintenance teams have the ability to intervene in a timely manner.
- **Risks:** Implementing this requirement may lead to unnecessary maintenance dispatches due to the risk of false-positive alerts; the overall operation of the system may be impaired due to the risk of cascading failures affecting multiple devices in the IoT network; sub-optimal parking recommendations may be generated due to the risk of data staleness, which may reduce user satisfaction; and maintenance costs may increase due to the risk of frequent device failures.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team and AI Team are responsible for the implementation of this requirement.

REQ-435

- **Requirement ID:** REQ-435
- **Title:** Dialog Manager and Authentication System for Voice Inputs
- **Category:** This requirement belongs to the functional requirements and user interface (UI) and user experience (UX) requirements categories.
- **Description:** The system should include an advanced dialogue manager that can detect ambiguous or potentially misunderstood input from drivers' voice commands and request verification or clarification from the user. This feature works with a conversational AI paradigm to precisely determine user intent by asking contextual clarification questions like, "Did you want to park near Hospital X?" The dialogue manager supports multi-turn conversations, strengthening the natural language understanding pipeline and optimizing the voice-first experience. The system implements a proactive clarification strategy by

analyzing acoustic confidence scores, semantic disambiguation, and user behavior patterns.

- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is affected by the drivers, UX/UI Design Team, Backend Team, Frontend Team, and AI Team.
- **Acceptance Criteria:**
    - The system should initiate an automatic explanation dialogue when it detects the uncertainty confidence threshold (below 80%) by processing voice inputs
    - Verification questions should be clear and understandable, appropriate to the user's context (clarity score >4.5/5.0)
    - User intent should be resolved in a maximum of 3 explanation cycles with multi-turn dialog support
    - The dialog manager must persist the conversation state throughout the session
    - The audio feedback delay should not exceed 2 seconds.
- **Dependencies:**
    - «satisfy» Voice recognition module (Speech-to-Text API, Azure Cognitive Services)
    - «satisfy» Natural language processing (NLP) service and intent classification engine
    - «trace» Voice Command Interpretation Module (software component that analyzes the user's voice commands and extracts the driver's intention)
    - «verify» Multimodal interaction test scenarios (voice command, text command, system response interaction chain tests).
- **Restrictions:**
    - This requirement should be implemented considering that the accuracy of voice verification may decrease in noisy environments (>70dB) due to environmental constraints.
    - This requirement should be implemented by limiting the dialog complexity to avoid user confusion (maximum 3 rounds) due to user experience constraints.
    - This requirement must be implemented taking into account the dependence of audio processing latency on network connectivity due to technical constraints.
    - This requirement should be implemented in such a way that multilingual support is initially limited to Turkish and English only due to language support restrictions.
- **Assumptions:** For this requirement to be implemented, it is assumed that the user device has microphone access and the voice recognition service operates reliably, that users have reasonable patience for explanation dialogues, that the speech context is rich enough to accommodate user uncertainties, and that background noise management algorithms are effective.
- **Risks:** Implementing this requirement may cause user frustration and reduce system adoption due to the risk of excessive clarification requests; incorrect decisions may be made in critical security scenarios due to the risk of inaccurate disambiguation; user engagement may be negatively impacted due to the risk of privacy concerns when using a voice interface; and the risk of dialogue complexity may increase user cognitive load and make it difficult to use.

- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** AI Team and Frontend Team are responsible for the implementation of this requirement.

REQ-436

- **Requirement ID:** REQ-436
- **Title:** No-Code Drag-and-Drop Visual Parking Lot Layout Design Editor
- **Category:** This requirement belongs to the functional requirements and user interface (UI) and user experience (UX) requirements categories.
- **Description:** The system provides a comprehensive visual editor with an intuitive drag-and-drop interface, allowing parking providers and facility managers to create complex parking layouts without requiring any technical programming knowledge. This no-code solution creates a digital twin of the parking infrastructure, supporting visual modeling of entry/exit points, multi-level parking structures, special zones (EV charging stations, disabled parking, VIP areas, motorcycle bays), and traffic flow patterns. It features real-time preview, collision detection, spatial validation, and automatic layout optimization, all with a "what you see is what you get" paradigm.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is affected by parking providers, facility managers, building managers, urban planners, Backend Team, and Frontend Team.
- **Acceptance Criteria:**
    - Parking elements (slots, lanes, barriers, signs) must be responsive under 50ms latency with drag-and-drop functionality
    - A minimum of 10-storey vertical layout design should be possible with multi-storey parking support.
    - Special zones (EV, disabled, premium) must be clearly marked with visual differentiation
    - The layout validation engine must detect invalid configurations in real time
    - Undo/redo functionality must maintain a minimum history of 50 actions
    - Seamless integration must be provided for export capabilities (JSON, CAD formats)
- **Dependencies:**
    - «satisfy» Graphic rendering libraries (Canvas API, WebGL, SVG)
    - «satisfy» Parking lot data model and geometric algorithms
    - «verify» Cross-browser compatibility test suites
    - «trace» IoT Occupancy Data Integration Module
- **Restrictions:**
    - This requirement should be implemented taking into account that browser graphics capabilities may be a limiting factor for complex layouts due to technical limitations.
    - This requirement must be implemented by optimizing large-scale plant designs to avoid creating bottlenecks in the system due to performance constraints.

    - This requirement should be implemented carefully to ensure that the limited screen space on mobile devices due to device restrictions does not restrict the drag-and-drop user experience.
    - This requirement should be implemented with the understanding that legacy browser support may limit advanced graphics features due to compatibility restrictions.
- **Assumptions:** To implement this requirement, it is assumed that the facility managers who will use the editor have basic digital literacy, the park infrastructure will conform to standardized dimensions, the visual editor outputs will produce formats compatible with operational systems, and user feedback will be sufficient for iterative design improvements.
- **Risks:** Implementing this requirement may result in decreased editor performance due to the risk of complex parking structures; operational inefficiencies due to the risk of irregular user-created structures; errors in downstream systems due to the risk of invalid configurations; and user adoption may be inhibited due to the risk of increased design complexity.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team and Frontend Team are responsible for the implementation of this requirement.

REQ-437

- **Requirement ID:** REQ-437
- **Title:** Smart Post-Parking Activity Suggestions with Semantic Place Recognition
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system uses a complex multimodal AI pipeline to generate semantic, context-aware activity recommendations after a user completes their parking. This comprehensive recommendation engine delivers personalized recommendations through user intent mining, contextual data fusion (temporal patterns, weather conditions, location semantics, social activities), behavioral analytics, and collaborative filtering algorithms. Semantic location recognition technology generates contextually relevant recommendations, such as "visit recommendations if you park near a hospital" or "dining recommendations in a shopping mall context," by deeply understanding the geographic context. It integrates NLP models, computer vision, and geospatial analytics to deliver a hyper-personalized user experience.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is affected by drivers, local partners, parking providers, Backend Team, and AI Team.
- **Acceptance Criteria:**
    - Must be able to provide a minimum of 3 contextually relevant suggestions within 5 seconds of park completion
    - Recommendations should be personalized based on the user's past behavior, current context, and semantic understanding of place.

    - Recommendation accuracy should achieve a 75% satisfaction rate based on user feedback
    - Localized content recommendations should be provided with multi-language support.
    - Dynamic recommendation improvement should be possible with real-time context adaptation
- **Dependencies:**
    - «satisfy» NLP models and Large Language Models (LLM) infrastructure
    - «satisfy» Geospatial databases and Points of Interest (POI) APIs
    - «satisfy» Weather APIs, event databases and real-time context data feeds
    - «satisfy» User profile analysis module (contextual learning based on past preferences)
      «trace» Recommendation Engine Module (software component that provides dynamic recommendations based on driver intent, context and user data)
    - Recommendation optimization scenarios with the «verify» A/B testing infrastructure (for example: the recommendation acceptance rate should be tested comparatively)
- **Restrictions:**
    - This requirement should be implemented carefully to ensure that the AI model's inference latency due to technical constraints does not negatively impact the real-time user experience.
    - This requirement should be implemented taking into account that contextual data may be limited in some geographic areas due to data availability restrictions.
    - This requirement should be implemented taking into account that privacy regulations may limit user behavior tracking due to legal restrictions.
    - This requirement should be implemented taking into account the dependence of recommendation quality on sufficient historical data due to data limitations.
- **Assumptions:** For this requirement to apply, it is assumed that users will provide sufficient consent for location tracking and behavior analysis, contextual data sources (weather, events, POIs) will be reliable and up-to-date, user interaction with recommendations will provide a sufficient feedback loop, and semantic location recognition accuracy will maintain an acceptable threshold.
- **Risks:** Implementing this requirement may result in irrelevant or unrelated recommendations due to the risk of incorrect contextual interpretation; may undermine user trust and lead to legal issues due to the risk of privacy violation; may result in users being trapped in a filter bubble and unable to access alternative information due to the risk of over-personalization; and may result in unfair or unbalanced recommendations due to the risk of bias in recommendation algorithms.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team and AI Team are responsible for the implementation of this requirement.

REQ-438

- **Requirement ID:** REQ-438
- **Title:** Resilient User Experience with Smart Offline Backup Mode
  Category: This requirement belongs to the functional requirements and system reliability and performance requirements categories.
- **Description:** The system must implement sophisticated offline fallback mechanisms that ensure a seamless downgrade service experience in the event of network connectivity outages, API service failures, or server-side errors. This resiliency architecture leverages intelligent caching strategies, predictive data preloading, and graceful downgrade policies to provide cached recommendations, historical reservations, offline maps, and alternative service paths. Progressive Web App (PWA) capabilities ensure user experience continuity by providing local storage optimization, service worker integration, and background synchronization functionality. It includes transparent communication with data stale indicators and automatic synchronization recovery mechanisms.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by drivers, parking providers, Backend Team, Frontend Team, system administrators, and business continuity teams.
- **Acceptance Criteria:**
  - Network connection loss detection should be triggered within 3 seconds and seamlessly switch to offline mode
  - Cached data must support a minimum of 24 hours of user activity (bookings, preferences, maps)
  - Data staleness warnings should be displayed clearly and user-friendly
  - Once connectivity is restored, automatic sync recovery should complete in <10 seconds
  - 85% of core functionality should be available in offline mode
- **Dependencies:**
  - «satisfy» Progressive Web App (PWA) infrastructure and service workers
  - «satisfy» Local storage management systems (IndexedDB, Cache API)
  - «verify» Network connection simulation test scenarios
  - «trace» **OfflineExperienceHandler** (application module for data display, synchronization and temporary process recording in offline mode)
- **Restrictions:**
  - This requirement should be implemented taking into account that device storage limitations may restrict offline data capacity due to hardware constraints.
  - This requirement should be implemented considering that the accuracy of cached data may decrease over time due to data freshness constraints.
  - This requirement should be implemented taking into account that complex business logic may create difficulties in offline execution processes due to technical limitations.
  - This requirement should be implemented considering that background synchronization processes may increase battery consumption due to hardware and energy consumption constraints.
- **Assumptions:** To implement this requirement, it is assumed that the facility managers who will use the editor have basic digital literacy, the park infrastructure will conform to

standardized dimensions, the visual editor outputs will produce formats compatible with operational systems, and user feedback will be sufficient for iterative design improvements.
- **Risks:** Implementing this requirement may result in decreased editor performance due to the risk of complex parking structures; operational inefficiencies due to the risk of irregular user-created structures; errors in downstream systems due to the risk of invalid configurations; and user adoption may be inhibited due to the risk of increased design complexity.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team and Frontend Team are responsible for the implementation of this requirement.

REQ-439

- **Requirement ID:** REQ-439
- **Title:** Advanced Security: Credential Security and Session Management
- **Category:** This requirement belongs to the non-functional requirements and security and compliance requirements categories.
- **Description:** The system must implement comprehensive credential protection and complex session management with enterprise-grade security standards. This security framework includes end-to-end encryption, multi-factor authentication (MFA), adaptive authentication, advanced threat detection, and zero trust security principles. It uses industry-leading hash algorithms (Argon2id, scrypt), salt generation, pepper mechanisms, and hardware security module (HSM) integration for credential storage. It provides JWT tokens, refresh token rotation, session hijacking detection, concurrent session control, and automated threat response mechanisms for session management. It fully meets the requirements of OWASP security guidelines and regulatory compliance (GDPR, KVKK, PCI DSS).
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is affected by the Backend Team, Frontend Team, AI Team, system administrators, users, and legal team.
- **Acceptance Criteria:**
    - Password hashing with Argon2id algorithm should use minimum 12 iterations, 64MB memory
    - Strong password policy enforcement: minimum 12 characters, complexity requirements, breach database control
    - Session timeout should expire in 15 minutes when idle, absolute session lifetime should expire in 24 hours
    - Suspicious activity detection should be triggered in real time and automatic session termination should be performed.
    - Multi-factor authentication support should be available for TOTP, SMS, and hardware tokens
    - The security audit log must provide comprehensive event capture in a tamper-proof format

- **Dependencies:**
    - «satisfy» Authentication infrastructure (OAuth 2.0, OpenID Connect)
    - «satisfy» Cryptographic key management systems and Hardware Security Modules
    - «satisfy» Threat intelligence feeds and anomaly detection systems
    - «verify» Penetration testing frameworks and security assessment tools
    - «trace» SecurityComplianceMonitor (system component that performs regulatory compliance tracking and real-time security auditing)
- **Restrictions:**
    - This requirement should be implemented taking into account that high security requirements can create friction in the user experience due to security restrictions.
    - This requirement should be implemented taking into account that compliance requirements may lead to geographic operational limitations due to legal and regional restrictions.
    - This requirement should be implemented taking into account that encryption key management can significantly increase system complexity due to operational constraints.
    - This requirement should be implemented considering that advanced security features may create computational overhead and processing latency due to performance and security constraints.
- **Assumptions:** For this requirement to apply, it is assumed that users will be trained in and comply with security best practices, the regulatory environment will maintain reasonable stability, the security infrastructure will deliver reliable and scalable performance, and the evolution of the threat landscape will follow predictable patterns.
- **Risks:** Implementing this requirement could result in bypassing advanced security measures and breaching the system due to the risk of complex attack scenarios; could lead to significant legal and financial penalties due to the risk of compliance violations; could damage brand reputation and lose customer trust due to the risk of security incidents; and could lead to reduced user adoption due to the risk of overly complex security measures.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team, AI Team and Frontend Team are responsible for the implementation of this requirement.

REQ-440

- **Requirement ID:** REQ-440
- **Title:** Modular and Scalable System Architecture
- **Category:** This requirement belongs to the technical requirements and system architecture and infrastructure requirements categories.
- **Description:** The system should be built with a modular microservices architecture and scalable design principles to quickly and cost-effectively adapt to future expansion

needs. This enterprise-level architectural approach enables the addition of new parking areas, IoT sensor types, payment systems, and third-party integrations without requiring major system changes. It provides horizontal and vertical scalability by working with container-based deployment, service mesh architecture, API-first design, and domain-driven design (DDD) principles. With this cloud-native approach, auto-scaling, load balancing, and fault tolerance mechanisms are built-in.

- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or impacted by the Backend Team, Frontend Team, AI Team, system administrators, users, parking providers, system integrators, and product managers.
- **Acceptance Criteria:**
    - New parking zones should be added to the system within 30 minutes via configuration.
    - New sensor types should be integrated with a plug-and-play approach, requiring minimal code changes.
    - New microservices must be deployable with zero downtime deployment
    - The system must be able to support a 300% traffic increase with automatic scaling
    - Backward compatibility must be guaranteed with the API versioning strategy
- **Dependencies:**
    - «satisfy» Microservices architecture (Docker, Kubernetes orchestration)
    - «satisfy» API Gateway and service registration infrastructure
    - «verify» Load testing and scalability test suites
    - «trace» Modular architectural component that manages sensor expandability and service connections
- **Restrictions:**
    - This requirement should be implemented taking into account that the complexity of the microservices architecture may increase the operational load due to architectural constraints.
    - This requirement should be implemented taking into account that network latency can have a negative impact on performance in distributed system calls due to network constraints.
    - This requirement should be implemented taking into account that container-based architecture may increase operational costs along with resource consumption due to resource and cost constraints.
    - This requirement should be implemented considering that service mesh complexity can make debugging and monitoring processes difficult due to observability constraints.
- **Assumptions:** To implement this requirement, it is assumed that the developer team must be able to apply microservice design principles and cloud-native patterns, the cloud infrastructure will provide sufficient scalability and reliability, API consumers will adopt versioning strategies and support backward compatibility, and the DevOps team has expertise in container orchestration and service management.

- **Risks:** Implementing this requirement may create high future refactoring costs and technical debt due to the risk of insufficient modularity; troubleshooting processes may become difficult due to the risk of distributed systems complexity; tight coupling issues may arise due to the risk of incorrectly defining microservice boundaries; and incorrect autoscaling settings may lead to increased costs or performance issues.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team, AI Team and Frontend Team are responsible for the implementation of this requirement.

REQ-441

- **Requirement ID:** REQ-441
- **Title:** KVKK & GDPR Compliant Personal Data Processing and Storage
- **Category:** This requirement belongs to the categories of legal and regulatory requirements and data security and privacy requirements.
- **Description:** The system must process and store personal data in full compliance with the Personal Data Protection Law (KVKK) and the GDPR (General Data Protection Regulation). This comprehensive privacy framework implements the principles of data minimization, purpose limitation, legal basis for processing, user consent management, data subject rights (access, correction, deletion, portability), and privacy by design. It provides proactive compliance management with automated compliance monitoring, data lineage tracking, consent audit trails, and regulatory reporting capabilities. It offers full lifecycle data governance with data retention policies, automated anonymization processes, and secure data destruction procedures.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by the driver, parking controller, municipality, parking provider, software developer, system administrator, data scientist, sensor data source, camera data source, government, external system, AI Team, legal team, and compliance officers.
- **Acceptance Criteria:**
    - The user should be able to view the purposes for which their data is processed via a transparent control panel.
    - With detailed consent management, users should be able to manage data processing consents on a category basis.
    - Data minimization: only necessary data should be processed and automatically deleted at the end of the retention period
    - Complete deletion must be made within 30 days of the data deletion request and an audit trail must be kept.
    - GDPR Article 25 compliance: Privacy by Design and Privacy by Default must be implemented
- **Dependencies:**
    - «satisfy» Approval management platform and user preference engine
    - «satisfy» Data encryption systems and secure storage infrastructure
    - «verify» Compliance audit frameworks and regulatory assessment tools

    - Data security module integrated into the «trace» authentication and authorization infrastructure
- **Restrictions:**
    - This requirement should be achieved by implementing invalidation mechanisms in data deletion processes, as retention is mandatory for certain data, such as financial and breach records, due to legal restrictions.
    - This requirement should be implemented provided that adequate protection mechanisms are provided for cross-border data transfers due to data security and legal restrictions.
    - This requirement should be implemented taking into account that real-time approval updates may create additional load on the system due to performance constraints.
    - This requirement should be implemented with the understanding that reporting requirements may impose additional storage and processing load due to compliance and resource constraints.
- **Assumptions:** For this requirement to be implemented, it is assumed that system administrators have received comprehensive training in privacy regulations, that updates to the legal framework will be communicated with reasonable advance notice, that users will actively use privacy controls and make informed decisions, and that regulatory authorities will maintain reasonable expectations of compliance.
- **Risks:** To implement this requirement, it is assumed that system administrators have received comprehensive training in privacy regulations, that updates to the legal framework will be communicated with reasonable advance notice, that users will actively use privacy controls and make informed decisions, and that regulatory authorities will maintain reasonable expectations of compliance.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team, AI Team and Frontend Team are responsible for the implementation of this requirement.

REQ-442

- **Requirement ID:** REQ-442
- **Title:** Full Transparency in Personal Data Use and Third Party Sharing
- **Category:** This requirement belongs to the categories of legal and regulatory requirements and user interface (UI) and user experience (UX) requirements.
- **Description:** The system must provide full transparency into personal data usage and third-party sharing for all user types (drivers, parking controllers, parking providers). This comprehensive transparency framework includes detailed privacy notices, real-time data sharing visualizations, third-party data processor lists, data flow diagrams, and interactive privacy dashboards. Layered privacy notices provide information at appropriate levels of detail for both expert and average users. Data sharing agreements, processor due diligence, and transfer impact assessments are presented in a user-friendly format.
- **Priority:** This requirement has been set to high priority.

- **Stakeholders:** This requirement is requested or influenced by the Frontend Team, Backend Team, AI Team, users, UX/UI designers, data protection authorities, third-party partners, and customer support.
- **Acceptance Criteria:**
    - The user should be able to visualize the data usage policy and data flows through the interactive privacy dashboard
    - Third-party data recipients must be clearly listed with information on purpose, duration and legal basis.
    - Users should be able to control their data sharing preferences in detail with real-time approval status tracking.
    - Data sharing activity records should be transparently displayed in a chronological timeline
    - Privacy policy changes should be highlighted with a difference view and notified to the user.
- **Dependencies:**
    - «satisfy» User consent management systems and preference tracking
    - «satisfy» Third-party integration monitoring and data flow mapping
    - «verify» Transparency compliance verification procedures
    - «trace» Data security module that monitors user data processing processes
- **Restrictions:**
    - This requirement should be implemented taking into account that certain data sharing may be required to institutions such as law enforcement or tax authorities due to legal restrictions.
    - This requirement should be implemented in a simplified manner to ensure that the density of information does not overwhelm users and lead to decision fatigue due to user experience constraints.
    - This requirement should be implemented considering that visualizing complex data relationships can be challenging due to technical limitations.
    - This requirement must be implemented in a way that requires additional localization efforts for transparency content while providing multilingual support due to language and localization constraints.
- **Assumptions:** For this requirement to apply, it is assumed that users will read and understand privacy information and make informed decisions, that third-party partners will comply with transparency requirements and provide accurate information, that regulatory expectations remain consistent for transparency standards, and that user interface design will facilitate effective information presentation.
- **Risks:** Implementing this requirement could lead to regulatory violations due to the risk of incomplete or misleading transparency information; users may not understand the information and make informed decisions due to the risk of confusing privacy information presentation; the system's transparency commitments may be undermined due to the risk of incompatibilities with third parties; and competitively sensitive information may be exposed due to the risk of excessive transparency.
- **Status:** The current status of this requirement is determined as a draft.

- **Responsible:** Backend Team and Frontend Team are responsible for the implementation of this requirement.

REQ-443

- **Requirement ID:** REQ-443
- **Title:** UKOME / General Security API Compliant License Plate and Violation Data Management
- **Category:** This requirement belongs to the legal and regulatory requirements and technical requirements categories.
- **Description:** The system securely records and stores license plate numbers and parking violation data for a period of 5 years in formats fully compliant with UKOME (Transportation Coordination Center) and General Directorate of Security API standards. This specialized compliance system meets municipal parking regulations, traffic ticket procedures, and law enforcement integration requirements. It ensures regulatory compliance with immutable audit trails, tamper-proof logging, secure data archiving, and automatic retention policy enforcement. It provides real-time violation detection, automatic fine calculation, and municipal system integration capabilities.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is affected by the Frontend Team, Backend Team, AI Team, municipal authorities, General Directorate of Security, UKOME, traffic police, parking operators, law enforcement agencies, legal teams and data governance teams.
- **Acceptance Criteria:**
    - License plate and violation data must be recorded in a format that is 100% compatible with UKOME/Safety API specifications.
    - 5-year retention must be guaranteed with tamper-proof logging and audit trail integrity must be maintained
    - Data that has expired must be securely deleted by the automatic archiving system.
    - Authorized institutions should be able to access data instantly through real-time API connection.
    - 99.9% uptime with high availability guarantee must be maintained for critical law enforcement operations
- **Dependencies:**
    - «satisfy» UKOME / Police General API integration infrastructure
    - «satisfy» Immutable data storage systems and blockchain-like audit trails
    - «verify» API compliance verification frameworks
    - «trace» Regulatory compliance module that monitors public data sharing and record integrity
- **Restrictions:**
    - This requirement should be implemented with the consideration that feature changes to government APIs due to integration restrictions may create backward compatibility challenges.

    - This requirement should be implemented considering that long-term data storage will require significant infrastructure costs due to infrastructure and budget constraints.
    - This requirement must be implemented based on the requirement that legal evidence requirements be met with the highest security standards and data integrity due to legal and security constraints.
    - This requirement should be implemented taking into account that, due to regulatory constraints, different regulations across jurisdictions may impose varying compliance requirements.
- **Assumptions:** For this requirement to be implemented, it is assumed that the UKOME and Safety API formats will maintain reasonable stability or provide transition support, government systems will provide reliable connectivity and reasonable response times, the legal framework will maintain consistent enforcement procedures for parking violations, and municipal partners will have the necessary technical integration capabilities.
- **Risks:** Implementing this requirement could lead to system incompatibility, compliance errors, and legal violations due to the risk of API format changes; the validity of legal evidence could be weakened due to the risk of data integrity being compromised; critical law enforcement operations could be disrupted due to the risk of system failure; and significant legal penalties and operational shutdowns could occur due to the risk of system non-compliance with regulations.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team, AI Team and responsible for the implementation of this requirement.

REQ-444

- **Requirement ID:** REQ-444
- **Title:** AI-Powered Smart Campaign Recommendations and Revenue Optimization
  Category: This requirement belongs to the functional requirements and artificial intelligence and machine learning requirements categories.
- **Description:** The system should provide intelligent campaign recommendations and revenue optimization strategies for parking providers using advanced machine learning algorithms. This advanced recommendation engine provides hyper-personalized business intelligence by leveraging predictive demand analysis, driver behavior pattern recognition, seasonal trend analysis, competitive intelligence, and dynamic pricing optimization. It offers comprehensive business growth solutions through time-series forecasting, customer segmentation, A/B testing frameworks, and marketing automation integration. It generates actionable insights by incorporating real-time market conditions, weather patterns, local events, and competitor analysis.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** Frontend Team, Backend Team, AI Team, Parking Providers, Parking Operators, Revenue Management Teams, Business Development, Data Scientists,

Machine Learning Engineers, Business Intelligence Analysts, Marketing Teams, Financial Analysts, Competitive Intelligence Teams

- **Acceptance Criteria:**
  - Campaign recommendation accuracy should achieve an 85% satisfaction rate based on user feedback
  - Revenue impact estimates must be within a 20% margin of error and return on investment projections must be reliable
  - Campaign adjustments sensitive to hourly demand fluctuations should be made through real-time market analysis.
  - Seasonal pattern recognition should support 3-month advance campaign planning
  - Campaign performance should be continuously optimized with A/B testing integration.
- **Dependencies:**
  - «satisfy» Comprehensive data analytics platform and big data processing infrastructure
  - «satisfy» Machine learning model training pipelines and MLOps frameworks
  - «verify» Campaign effectiveness measurement and return on investment tracking systems
  - «trace» Recommendation engine analytics module that analyzes user interaction data
- **Restrictions:**
  - This requirement should be implemented taking into account that insufficient historical data due to data limitations may limit forecast accuracy for some market segments.
  - This requirement should be implemented considering that real-time processing requirements may impose high computational load due to performance constraints.
  - This requirement should be implemented taking into account that market volatility and unpredictable variables may unexpectedly affect model estimates due to exogenous factor constraints.
  - This requirement should be implemented taking into account that adapting complex business rules to different provider types may create technical difficulties due to customization restrictions.
- **Assumptions:** For this requirement to be implemented, it is assumed that parking providers will trust and demonstrate willingness to implement AI-generated recommendations, market data sources will provide reliable, accurate, and timely information, the business environment will allow for rapid AI adoption while maintaining reasonable predictability, and provider feedback will be systematically collected and used for model improvement.
- **Risks:** Implementing this requirement could lead to financial losses for providers and undermine system reliability due to the risk of inaccurate revenue forecasts; the risk of algorithm bias could unfairly disadvantage certain types of providers; the risk of market disruptions could render AI models obsolete and require significant retraining; and the

risk of competition could result in AI-powered insights being copied and competitive advantage being eroded.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team, AI Team and Frontend Team are responsible for the implementation of this requirement.

REQ-445

- **Requirement ID:** REQ-445
- **Title:** Dynamic GPS Location Permission Management and Graceful Feature Downgrade
- **Category:** This requirement belongs to the functional requirements and user privacy and permission management requirements categories.
- **Description:** The system should implement an advanced permission management system that can request users' permission to access GPS location data at appropriate times and in a contextual manner. This privacy-focused approach uses a phased permission request strategy to avoid disrupting the user experience and intelligently restricts location-dependent features with graceful downgrades when permission is denied. It provides a clear value proposition to users with a just-in-time permission model, while maintaining functionality continuity through alternative input methods (manual location entry, city selection, favorite locations). It maintains a minimum-optimal experience through a privacy-first design.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is impacted by the Frontend Team, Backend Team, AI Team, drivers, privacy advocates, UX researchers, mobile platform experts, customer support team, analytics teams, location-based service providers, and compliance team.
- **Acceptance Criteria:**
    - GPS permission should only be requested contextually when the relevant feature is being used.
    - When permission is denied, the system should immediately offer alternative solutions (manual location, city selection)
    - Location-based features should adapt in real time based on permission status
    - Permission changes should be managed through user settings and take effect immediately
    - 80% basic functionality should be preserved in downgraded mode
- **Dependencies:**
    - «satisfy» Mobile OS permission management APIs (iOS Location Services, Android Location)
    - «satisfy» Alternative location entry systems and geocoding services
    - «verify» Permission flow user test scenarios
      «trace» Location verification and access module that manages location data from the device
- **Restrictions:**

    - This requirement should be implemented taking into account that permission behavior may differ between iOS and Android operating systems due to platform restrictions.
    - This requirement should be implemented considering that background location tracking must be implemented with additional battery optimization considerations due to hardware and energy constraints.
    - This requirement should be implemented taking into account that regional privacy regulations may require additional consent mechanisms due to legal restrictions.
    - This requirement should be implemented taking into account that alternative location methods may only provide limited functionality in offline scenarios due to technical constraints.
- **Assumptions:** For this requirement to apply, it is assumed that users will understand the reasons for consent and make informed decisions, that alternative location input methods will provide reasonable accuracy, that there will be sufficient system responsiveness to gracefully handle consent status changes, and that users will find the privacy-enhanced experience acceptable in exchange for functionality trade-offs.
- **Risks:** Implementing this requirement could lead to user disapproval or abandonment due to the risk of aggressive permission requests; the core value proposition of the system could be significantly diminished due to the risk of limited location access; the risk of a complex permission flow could lead to user confusion and abandonment; and the risk of using alternative location determination methods could lead to degraded quality of data collected and compromised accuracy.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team, AI Team and Frontend Team are responsible for the implementation of this requirement.

REQ-446

- **Requirement ID:** REQ-446
- **Title:** Multi-Device Session Management and Secure Batch Logout
  Category: This requirement belongs to the functional requirements and security and session management requirements categories.
- **Description:** The system must provide a comprehensive session management solution that allows users to centrally manage sessions across all active devices and terminate them with a single action, aligning with security requirements. This enterprise-level security feature provides comprehensive account security with device tracking, session monitoring, suspicious activity detection, and remote session termination capabilities. Multi-factor authentication is designed to protect against unauthorized access by protecting critical actions. Real-time session synchronization ensures immediate effectiveness and audit logging meets compliance requirements.
- **Priority:** This requirement has been set to high priority.

- **Stakeholders:** This requirement is requested or impacted by the Frontend Team, Backend Team, AI Team, drivers, security experts, compliance officers, customer support team, and system administrators.
- **Acceptance Criteria:**
    - The user should be able to view active sessions and device details (platform, location, last activity) from their account settings.
    - The "Log Out of All Sessions" action must be protected with multi-factor authentication
    - Session termination must take effect on all devices within 30 seconds
    - After the transaction, confirmation should be sent to the user via email and push notification.
    - Session activity logs should be recorded comprehensively for audit trail
- **Dependencies:**
    - «satisfy» Distributed session management infrastructure and Redis/database clusters
    - «satisfy» Multi-factor authentication systems and real-time notification services
    - «verify» Cross-device session termination test suites
      «trace» Session security module that tracks session ID matching and device management
- **Restrictions:**
    - This requirement should be implemented taking into account that connectivity issues due to network constraints may delay session termination.
    - This requirement should be implemented considering that immediate session termination may not be possible when the device is offline due to offline scenario restrictions.
    - This requirement should be implemented taking into account that an eventual consistency approach may be required for system-wide synchronization of session state due to distributed system constraints.
    - This requirement should be implemented considering that high-frequency session control operations may create additional load on the system due to performance constraints.
- **Assumptions:** For this requirement to be implemented, it is assumed that users will understand the concepts of device and session management, the network infrastructure will be sufficient for reliable session synchronization, multi-factor authentication will keep friction in the user experience at acceptable levels, and users will receive session termination notifications in a timely manner.
- **Risks:** Implementing this requirement could cause significant user inconvenience and loss of productivity due to the risk of accidental session termination; the risk of synchronization errors could lead to partial session terminations and disrupted workflow; the risk of complex session management could lead to user confusion and increased support calls; and the risk of improper storage of session recordings could lead to the exposure of sensitive information and raise privacy concerns.
- **Status:** The current status of this requirement is determined as a draft.

- **Responsible:** Backend Team, AI Team and Frontend Team are responsible for the implementation of this requirement.

REQ-447

- **Requirement ID:** REQ-447
- **Title:** Flexible Invoice Information Management and Automatic Invoice Creation
- **Category:** This requirement belongs to the functional requirements and financial transactions and taxation requirements categories.
- **Description:** The system must provide an advanced financial data management solution that allows users to optionally manage comprehensive invoice information and use it for automated invoice generation. This business-focused feature supports flexible invoice profiles for individual and corporate users, meeting tax compliance, expense tracking, and accounting integration requirements. It meets global business needs with dynamic form validation, international address support, taxpayer authentication, and multi-currency invoicing capabilities. It also provides professional business documentation with template-based invoice generation and customizable branding options.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested or is impacted by the Frontend Team, Backend Team, AI Team, individual users, enterprise customers, accounting teams, tax professionals, compliance team, and customer support.
- **Acceptance Criteria:**
    - User must be able to enter comprehensive billing information (individual/corporate) from profile settings
    - Tax ID format verification and address standardization must be done through form validation.
    - Invoice generation should be triggered automatically after transaction completion
    - Generated invoices should be downloadable in PDF format and can be sent via email.
    - Invoice information updates should not allow retroactive invoice reproduction
- **Dependencies:**
    - «satisfy» Payment processing systems and transaction recording infrastructure
    - «satisfy» Tax authentication services and international address verification APIs
    - «verify» Invoice accuracy and legal compliance verification procedures
    - «trace» Invoice management module that monitors financial transaction auditing and address verification processes

- **Restrictions:**
    - This requirement should be implemented taking into account that tax regulations may vary significantly between jurisdictions due to legal restrictions.

    - This requirement should be implemented taking into account that real-time tax authentication may create dependency on external services due to integration limitations.
    - This requirement should be implemented with the understanding that past invoice changes may pose a risk of non-compliance due to regulatory compliance restrictions.
    - This requirement should be implemented taking into account that multi-currency support may create exchange rate fluctuations and accounting complexity due to financial constraints.
- **Assumptions:** For this requirement to apply, it is assumed that users will provide accurate and up-to-date invoice information, tax compliance requirements will maintain reasonable consistency, invoice templates will meet legal and professional standards, and users will understand the importance of invoice information management.
- **Risks:** Implementing this requirement could lead to tax compliance issues and legal liability due to the risk of inaccurate invoice information; customer dissatisfaction and increased administrative workload due to the risk of invoice generation errors; unauthorized access to financial information and exposure of sensitive data due to the risk of data security breaches; and the risk of complex invoicing scenarios could exceed the technical capacity of the system and cause functionality issues.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team, AI Team and Frontend Team are responsible for the implementation of this requirement.

REQ-448

- **Requirement ID:** REQ-448
- **Title:** Smart Profile Pre-filling and Staged Enrollment with OAuth Integration
- **Category:** This requirement belongs to the functional requirements and user experience and identity integration requirements categories.
- **Description:** The system should provide seamless integration with third-party OAuth providers (Google, Apple, Facebook, Microsoft) to deliver intelligent profile pre-filling and a progressive sign-up experience. This modern authentication approach minimizes user initial friction while implementing intelligent data collection strategies for comprehensive profile completion. It maximizes social login benefits while maintaining regulatory compliance (data minimization, consent management). It ensures a superior user experience with adaptive form completion, duplicate detection, and data quality validation. It strategically defers phone verification and payment information collection, enabling conversion optimization.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or impacted by the Frontend Team, Backend Team, AI Team, drivers, UX researchers, growth marketing teams, product managers, customer support, data analytics teams, and compliance officers.
- **Acceptance Criteria:**

    - OAuth registration flow should work fine for major vendors (Google, Apple)
    - Existing profile data (name, email, avatar) should be pre-filled automatically
    - Missing critical information (phone, vehicle, payment) should be collected with progressive disclosure
    - Duplicate account detection should be prevented by existing phone/email
    - Full profile completion should be a mandatory requirement for booking eligibility
- **Dependencies:**
    - «satisfy» OAuth 2.0/OpenID Connect provider integrations and secure token handling
    - «satisfy» Progressive commit flow management and state persistence
    - «verify» Cross-platform OAuth flow test cases
    - «trace» Authentication module that traces phone number verification and deduplication flows

- **Restrictions:**
    - This requirement should be implemented with the consideration that policy differences among OAuth providers may restrict data accessibility and terms of use due to integration restrictions.
    - This requirement should be implemented considering that platform-specific applications such as 'Sign in with Apple' on iOS may create additional integration complexity due to platform restrictions.
    - This requirement should be implemented with the understanding that abandonment rates in phased registration processes may be higher than traditional registrations due to user behavior restrictions.
    - This requirement should be implemented considering that the accuracy and integrity of social profile information cannot be guaranteed due to data quality limitations.
- **Assumptions:** For this requirement to be implemented, it is assumed that OAuth providers will maintain reliable service and consistent data formatting, users will prefer the convenience of social login to traditional registration, progressive information collection will significantly improve the user experience, and social profile data will generally be accurate and up-to-date.
- **Risks:** Implementing this requirement may lead to provider API changes that could disrupt OAuth integration and disrupt user access; operational issues and service limitations due to the risk of missing profile data; user adoption may be limited due to privacy concerns around social logins; and the complexity of the staged signup process could lead to user confusion and increased support calls.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team, AI Team and Frontend Team are responsible for the implementation of this requirement.

REQ-449

- **Requirement ID:** REQ-449
- **Title:** Phone Number Uniqueness Enforcement and Authentication Security
  Category: This requirement belongs to the functional requirements and security and authentication requirements categories.
- **Description:** The system must implement a robust anti-duplication system that ensures account security and authentication integrity through unique ID enforcement based on the phone number. This critical security feature provides a foundational security layer for protecting against phone number spoofing attacks, preventing account takeovers, and user authentication. It provides comprehensive fraud prevention through advanced duplication detection algorithms, phone number normalization, international format processing, and suspicious record pattern detection. It also includes intelligent processing mechanisms for account recovery scenarios, number portability cases, and legitimate number reuse cases.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or impacted by the Frontend Team, Backend Team, AI Team, drivers, security experts, fraud prevention teams, customer trust officers, regulatory compliance teams, customer support, risk management unit, legal teams, and system administrators.
- **Acceptance Criteria:**
    - Real-time verification of phone number uniqueness must be performed during registration
    - International number formats should be standardized and normalized.
    - Attempted multiplexing detection should be communicated to the user with a clear error message.
    - There should be a grace period mechanism for legitimate number reuse scenarios (previous account deletion)
    - Phone number should be a mandatory requirement to complete the verification process
- **Dependencies:**
    - «satisfy» Phone number validation and normalization services (libphonenumber libraries)
    - «satisfy» SMS verification infrastructure and international operator support
    - «verify» Edge test cases (number recycling, international formats)
    - «trace» Authentication module that prevents phone number duplication and traces the registration verification flow
- **Restrictions:**
    - This requirement should be implemented taking into account that telephone number recycling may create legitimate multiplexing scenarios due to operator restrictions.
    - This requirement should be implemented with the understanding that complexities in international number formats can create edge cases and validation challenges due to validation restrictions.

    - This requirement should be implemented taking into account that SMS delivery reliability may vary across international markets due to regional infrastructure constraints.
    - This requirement should be implemented taking into account that high volumes of uniqueness checks performed on the database may require optimization due to performance constraints.
- **Assumptions:** For this requirement to be implemented, it is assumed that OAuth providers will maintain reliable service and consistent data formatting, users will prefer the convenience of social login to traditional registration, progressive information collection will significantly improve the user experience, and social profile data will generally be accurate and up-to-date.
- **Risks:** Implementing this requirement could lead to legitimate users being unfairly blocked from the registration process due to the risk of number recycling; the overall security of the system could be compromised due to the risk of the phone number-based security structure creating a single point of failure; additional verification steps and system complexity could be introduced due to the risk of international compliance requirements; and attempts to bypass phone number verification mechanisms could be made due to the risk of sophisticated attacks.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend Team, AI Team and Frontend Team are responsible for the implementation of this requirement.

E) External System Integration Requirements

Requirement REQ-500

• **Requirement ID:** REQ-500
• **Title:** Third-Party Payment Systems Integration
• **Category:** Technical, External System
• **Description:** The system must integrate with third-party payment providers such as Iyzico and payTR to enable drivers to securely pay their parking fees. This integration must utilize token-based authentication, support multiple currencies, and include refunds.
• **Priority:** High
• **Stakeholders:** Drivers, Payment Service Providers, Backend team
• **Acceptance Criteria:**
○ Payments must be completed successfully using token-based authentication.

○ The system must be able to handle transactions with at least three different currencies.
○ A confirmation message must be displayed to the user within 5 seconds after payment.
○ Refunds initiated through the application must be approved or rejected within 10 seconds.
• **Dependencies:**

«satisfy» Payment System APIs (e.g., payTR API, Iyzico API)

«satisfy» Payment System via App
«satisfy» Protection System Against Brute-Force and Denial of Service Attacks
«verify» REQ-037 – Penalty Application for Time Outs Test Scenario
«verify» REQ-701 – Define Default Payment Method Status
«verify» REQ-704 – Automatic Reservation Cancellation in Case of Late Arrival Test Scenario
«deriveReqt»REQ-018 – Pre-Booking Fee and Payment Method Confirmation
«deriveReqt»REQ-020 – Ability to Extend Active Parking Session
«deriveReqt»REQ-215 – Parking Attendant Ability to Override Reservation
«deriveReqt»REQ-225 – Authorized Officials to Make and Cancel Reservations

• **Restrictions:** Payment systems are limited to a certain number of payment providers.
• **Assumptions:** The user has a valid payment method and their device is connected to the internet.
• **Risks:** Interruptions in payment providers' API services may cause payment failures.
• **Status:** Draft
• **Responsible:** Backend team

Requirement REQ-501

• **Requirement ID:** REQ-501
• **Title:** Receiving and Processing Real-Time Parking Status Data from External IoT Sensors
• **Category:** Technical, External System
• **Description:** The system shall receive real-time parking lot occupancy data from external IoT sensors (e.g., ultrasonic, magnetic, camera-based) and store this data in the central system with standard timestamps. The obtained data shall be processable in a usable format for future analysis and reporting.
• **Priority:** High
• **Stakeholders:** Parking Operators, IoT Integration Team, Data Analysis Team
• **Acceptance Criteria:**
○ The system must be able to record sensor data to the central database in less than 10 seconds.

○ Each data record must include date, time, parking area ID, and status information.
○ Sensor data must be detected and parsed by the system with at least 98% accuracy.
• **Dependencies:**

«satisfy» Sensor Integration APIs (Ultrasonic, Magnetic, Camera-based IoT APIs)

«satisfy» Real-Time Parking Location Monitoring Module
«satisfy» Accessible Parking Area Monitoring System
«satisfy» System to See Available Parking Spaces on the Map
«satisfy» Density Status and Usage Statistics Module
«satisfy» IoT Device Health and Connection Management Module
«derive» REQ-424 – Parking Space Status Control with IoT
«satisfy» Real Time System Monitoring Panel
«deriveReqt»REQ-209 – Parking Officer Control of Areas
«deriveReqt»REQ-217 – Reporting Sensor Errors
«deriveReqt»REQ-404 – Demand Forecast Based on Occupancy Analysis
«verify» REQ-003 – Parking Location Visualization test case

• **Restrictions:** Limited to receiving data only from IoT sensor devices compatible with the system.
• **Assumptions:** It must be assumed that sensors are correctly positioned and functioning properly.
• **Risks:** Sensor failures, connection interruptions, or data loss may affect the accuracy of analysis results.
• **Status:** Draft
• **Responsible:** Backend Team

Requirement REQ-502

• **Requirement ID:** REQ-502
• **Title:** Secure Data Sharing and Receiving with Government Agencies
• **Category:** Technical, External System
• **Description:** The system must enable secure data sharing with law enforcement or public institutions. This sharing may include parking violation records, vehicle information, or data provided with user permission. Data exchange must occur through encrypted and authenticated API connections.
• **Priority:** Medium
• **Stakeholders:** City Administration, Police Departments, Backend Team
• **Acceptance Criteria:**
○ The system must be able to respond to data requests via the API after authorization is

verified.
○ Shared data must be transferred with end-to-end encryption.
○ Each transaction must be recorded as a log and stored for 6 months.
• **Dependencies:**

«satisfy» Government APIs / Data Sharing Interfaces (e-Government, municipal systems, police information systems, etc.)

«deriveReqt» REQ-441 – Personal Data Processing and Storage in Compliance with KVKK & GDPR
«satisfy» Secure Credential Storage and Session Management Module
«satisfy» Sensitive Data Encryption Module
«deriveReqt»REQ-220 – Receiving and Approving Security Alerts
«deriveReqt»REQ-221 – Ability to Communicate Directly with Emergency Services
«deriveReqt»REQ-222 – Ability to Broadcast Emergency Message to All Users

• **Restrictions:** Data sharing is limited to public institutions with legal permission only.
• **Assumptions:** Data to be shared is stored with user consent.
• **Risks:** Unauthorized access attempts, data leaks, or legal liabilities may arise.
• **Status:** Draft
• **Responsible:** Backend Integration Team

Requirement REQ-503

• **Requirement ID:** REQ-503
• **Title:** Automatic Data Exchange with Third-Party Parking Providers
• **Category:** Technical, External System
• **Description:** The system should integrate with external parking providers to automatically retrieve information such as parking availability, pricing, reservation rules, and service updates. This integration should be implemented through API-based data retrieval.
• **Priority:** Medium
• **Stakeholders:** Third-Party Parking Operators, Backend Team
• **Acceptance Criteria:**
○ The system must be able to retrieve real-time parking availability information from providers.
○ Price changes and reservation rules must be updated every 10 minutes.
○ Data from external providers must be aligned with the system data schema and displayed consistently.
• **Dependencies:**

«satisfy» Parking Provider APIs (e.g. Parkopedia, CityParking API etc.)

«satisfy» Advance Parking Reservation System
«satisfy»«verify» Parking Reservations Management Module
«satisfy»«deriveReqt» Integration with Third Party Parking Systems
«deriveReqt»REQ-019 – Reservation Cancellation
«deriveReqt»REQ-028 – Avoiding Conflicting Reservations
«deriveReqt»REQ-201 – Reporting Reservation to Parking Attendant
«deriveReqt»REQ-224 – Heat Map for Reservation Conflicts

• **Restrictions:** Integration is only possible with parking providers that meet the technical requirements.
• **Assumptions:** External providers provide consistent and consistent data.
• **Risks:** Data format mismatches, connection interruptions, and incorrect price/availability information may occur.
• **Status:** Draft
• **Responsible:** Backend Team

Requirement REQ-504

• **Requirement ID:** REQ-504
• **Title:** Google Maps and Google Places API Integration
• **Category:** Technical, External System
• **Description:** The system must integrate with the Google Maps and Google Places APIs to meet users' location-based search and navigation needs. This integration allows users to search for parking on a map, get directions, and view nearby points of interest (restaurants, shopping malls, etc.).
• **Priority:** High
• **Stakeholders:** Drivers, Front End Team
• **Acceptance Criteria:**
 ○ The user should be able to see nearby parking spaces based on their current location via the map interface.
○ The user should be able to get a route to the selected parking spot via Google Maps.
○ Nearby popular places (POIs) should be listed via the Google Places API.
• **Dependencies:**

«satisfy» Map APIs (Google Maps API, Google Places API)

«satisfy» Surrounding Facilities Display Module
«satisfy» EV Charging Station Identification Module
«satisfy»«deriveReqt» Parking Guidance with Map-Based Navigation
«satisfy»«deriveReqt» Access to Real-Time GPS Data
«satisfy» Map Based Navigation system
«satisfy» Redirection to Third Party Navigation Apps

«deriveReqt»REQ-107 – Dynamic Pricing Request (location-based)
«deriveReqt»REQ-218 – Parking Area Recommendation for Large Vehicles
«deriveReqt»REQ-227 – Parking Attendant Ability to Share Live Location
«deriveReqt»REQ-437 – Post-Park Activity Suggestions
«verify» REQ-008 – Navigation Guidance test case
«verify» REQ-605 – Map-Based Parking Guidance scenario

• **Restrictions:** API usage is limited by Google's quota and licensing restrictions.
• **Assumptions:** The user consents to access their location and has an internet connection on their device.
• **Risks:** Google APIs may become inaccessible due to quota exceedances, connectivity issues, or pricing changes.
• **Status:** Draft
• **Responsible:** Front End Team

Requirement REQ-505

• **Requirement ID:** REQ-505
• **Title:** Integration with Smart City Data
• **Category:** Technical, External System
• **Description:** The system should enrich parking recommendations and provide context-aware recommendations to users by integrating information from smart city data platforms such as open municipal data, air quality information, and traffic density.
• **Priority:** Medium
• **Stakeholders:** Municipal Systems, Data Analysis Team, Mobile Application Developers
• **Acceptance Criteria:**
 ○ The system should integrate data such as traffic density and air quality into parking recommendations.
○ Parking recommendations presented to the user should be ranked according to environmental data.
○ Locations excluded from recommendations due to weather or traffic conditions should be clearly explained to the user.
• **Dependencies:**

«satisfy» Smart City APIs (traffic data, air quality APIs, open municipal data sources)

«satisfy» Recommendation System Without Real-Time Data
«satisfy» Parking Space Layout and AI Optimization Module
«satisfy» «deriveReqt» Driver History and Contextual Recommendation Engine System

«satisfy» EV Parking Space Prioritization Module
«satisfy» Post-Parking Information System
«satisfy» Late Management System

«deriveReqt»REQ-107 – Dynamic Pricing Request
«deriveReqt»REQ-219 – Ability to Override Automatic Systems in Emergency Situations
«deriveReqt»REQ-230 – Monitoring Environmental Risks
«deriveReqt»REQ-404 – Demand Forecasting
«deriveReqt»REQ-418 – Indoor Mapping and Floor Plan
«deriveReqt»REQ-713 – Critical Event Detection
«deriveReqt»REQ-712 – Officer Notification
«deriveReqt»REQ-703 – Ability to Search for Parking Spaces in the Background
«verify» REQ-026 – Parking Proposal Test scenario

• **Restrictions:** Limited to the use of only publicly available and integration-friendly data.
• **Assumptions:** Smart city platforms provide regular and up-to-date data.
• **Risks:** Data outages, timeliness issues, or inaccurate environmental factor assessments may negatively impact the user experience.
• **Status:** Draft
• **Responsible:** Backend Team

Requirement REQ-506

• **Requirement ID:** REQ-506
• **Title:** Weather API Integration
• **Category:** Technical, External System
• **Description:** The system shall retrieve data from external weather APIs to predict when weather conditions might impact parking operations. This data shall be used for functions such as filtering parking suggestions, adjusting pricing strategies, and issuing user alerts.
• **Priority:** Medium
• **Stakeholders:** Drivers, Weather Data Providers, System Developers
• **Acceptance Criteria:**
 ○ The system must be integrated with a weather API that provides at least hourly forecast data.
○ In conditions such as heavy rain, snow, or storms, the application should notify the user with a warning message.
○ Dynamic pricing policies must be able to be applied based on the weather.
• **Dependencies:**

«satisfy» Weather APIs (e.g. OpenWeatherMap, AccuWeather)

«satisfy»«deriveReqt» Weather-Based Safety Alert System
«deriveReqt»REQ-026 – AI-Based Parking Recommendation Module
«deriveReqt»REQ-230 – Monitoring of Environmental Risks
«deriveReqt»REQ-623 – Service Continuity in the Event of Partial Data Accessibility
«deriveReqt»REQ-633 – Parking Data Intermittent Update
«deriveReqt»REQ-626 – Hassle-Free Software Updates

• **Restrictions:** API service is limited by quota and regional coverage.
• **Assumptions:** Weather services are assumed to provide regular and accurate data.
• **Risks:** Inaccurate forecast data may reduce user satisfaction; API outages may impact the recommendation system.
• **Status:** Draft
• **Responsible:** Backend team

Requirement REQ-507

• **Requirement ID:** REQ-507
• **Title:** Public Transport Route and Time Integration
• **Category:** Technical, External System
• **Description:** The system should integrate with public transport service providers' APIs to provide the nearest bus, tram, or metro schedule information so that drivers can continue their public transport journeys after parking.
• **Priority:** Medium
• **Stakeholders:** Drivers, Public Transport Operators, Map & Navigation Team
• **Acceptance Criteria:**
○ The nearest public transport stops within 500 meters of the parking space should be listed.
○ The next three departure times should be displayed for each stop.
○ The user should be able to access this information with a single click after completing the parking process.
• **Dependencies:** Public Transport APIs (e.g., Trafi, Navitia, local municipal data services)

«satisfy» Post-Parking Activity Suggestion Module
«satisfy» Driver Intention Detection System
«satisfy» Multi-Step Intention Processing and Route Planning System
«deriveReqt»REQ-008 – Navigation Guidance
«deriveReqt»REQ-110 – Parking Area Layout and AI Optimization

• **Restrictions:** API access is limited to integrated cities only.

• **Assumptions:** API data from public transportation providers is up-to-date and accessible.
• **Risks:** Changes in schedules may mislead the user if not updated in real time.
• **Status:** Draft
• **Responsible:** Frontend Team

F) Quality Requirements

- **Requirement ID:** REQ-600
- **Title:** Creating an Intuitive Interface for First-Time Users
- **Category:** This requirement belongs to the category of quality, user interface (UI) and user experience (UX) requirements.
- **Description:** The system should be intuitive for first-time users. The interface should use clear icons, and common tasks like booking should be completed in a minimum of steps. This requirement aims to reduce the learning curve and increase user satisfaction by providing a user-friendly experience. The intuitive interface should be designed to enable users to easily understand the system and perform actions quickly.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by end users, the UX/UI design team, the software development team, and customer support.
  Acceptance Criteria:
    - During user testing, 90% of first-time users should be able to complete the booking process within 3 minutes.
    - In the user satisfaction survey, the ease of use of the interface must reach an average score of at least 4/5.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-601 <<deriveReqt>>(Dark and Light Theme Selection Support)
    - REQ-602 <<deriveReqt>> (Automatic Language Adaptation to Device Language)

    - REQ-006 <<deriveReqt>> (Display of Accessible Parking Areas)
    - REQ-605 <<deriveReqt>> (Parking Lot Guidance with Real-Time, Map-Based Navigation)
    - REQ-025 <<deriveReqt>> (Save and Delete Favorite Parking Areas)
    - REQ-027 <<deriveReqt>> (Re-booking from Park History)
    - REQ-033 <<deriveReqt>> (Access to Weekly Usage and Expenditure Reports)
    - REQ-038 <<deriveReqt>> (Parking Operations with Voice Assistant)
    - REQ-047 <<deriveReqt>> (Performing Secure Login and Password Reset Operations)
- **Restrictions:** This requirement is not applicable if existing design patterns are deviated from.
- **Assumptions:**
    - The target user audience has basic digital literacy.
    - The system will be used on both mobile and desktop devices.
- **Risks:**
    - If the initial user experience is not good enough, user churn may occur.
    - Complex user tasks may not be simplified.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** The Frontend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-601
- **Title:** Support for Dark and Light Theme Selection
- **Category:** This requirement belongs to the category of quality, user interface (UI) and user experience (UX) requirements.
- **Description:** The system must support both dark and light mode themes in the user interface. Users should be able to choose a theme based on their preferences, and the system should remember this preference throughout the session, preferably permanently. This feature aims to increase user comfort and accessibility. The requirement aims to increase user satisfaction by improving the interface experience.
- **Priority:** The priority of this requirement is set to medium.

- **Stakeholders:** This requirement was requested by end users, UX/UI design team, and software development team.
- **Acceptance Criteria:**
    - The user should be able to change the theme preference from the settings menu.
    - When the theme preference is changed, the entire user interface should be updated instantly.
    - The selected theme should be preserved throughout the user session.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-601 <<deriveReqt>> (Creating an Intuitive Interface for First-Time Users)
- **Restrictions:**
    - All screen components must be fully compatible with both themes; otherwise, the design will not be considered valid.
    - If the theme transition exceeds 1 second, the performance criterion is not met.
- **Assumptions:**
    - Users can work in different lighting conditions on their personal devices.
    - The system may store user preferences via browser local storage or cookies.
- **Risks:**
    - Some UI components may not appear consistent across both themes.
    - Temporary UI errors may occur during theme changes.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** The Frontend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-602
- **Title:** Automatic Language Adaptation to Device Language
- **Category:** This requirement belongs to the quality requirements category.
- **Description:** The system should automatically adapt all text and labels based on the user's device language. This feature will enhance accessibility and user experience by reflecting the user's preferred language. The requirement aims to ensure users have a seamless interface experience in their native language.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the end users, the software development team, the localization team, and the product manager.

- **Acceptance Criteria:**
    - The app should automatically set the interface language if the device language is a supported language.
    - If an unsupported language is detected, the system should default to English.
    - The user should be able to manually select another language from the settings.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-600 <<deriveReqt>> (Creating an Intuitive Interface for First-Time Users)
    - REQ-427 <<deriveReqt>> (Multilingual User Interface Support (Turkish and English))
    - Language Detection Module <<satisfy>>
- **Restrictions:**
    - Initially, only certain languages (for example: English, Turkish, German) are supported. If a language other than these is selected, the system will not accommodate the request.
    - Inconsistencies may occur if translation files are not synchronized with app updates.
- **Assumptions:**
    - The user's device language settings are accessible.
    - All texts are managed from localizable sources (e.g. JSON language files).
- **Risks:**
    - Incomplete or incorrect translations may negatively impact user experience.
    - Failure to correctly detect the device language may result in incorrect language display.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-603
- **Title:** Voice Command Perception in Noisy Environments
- **Category:** This requirement belongs to the quality requirements category.
- **Description:** The system must be able to accurately recognize user voice commands, even in noisy urban environments such as traffic and construction zones. This requirement aims to increase the reliability of voice control and ensure the user

experience is sustainable in urban environments. The requirement aims to ensure that users can effectively control the system with voice commands, even in diverse environmental conditions.

- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by end users, the voice recognition/AI development team, the UX research team, and product management.
- **Acceptance Criteria:**
    - In tests conducted with noisy environment simulations, the voice command accuracy rate must be at least 85%.
    - Commands must be interpreted correctly by the system even when background noise is above 70 dB.
    - Incorrectly perceived commands should be given clear feedback to the user without having them repeat them.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-606 <<deriveReqt>> (Providing Hands-Free Experience with Natural Voice Commands)
    - REQ-004 <<deriveReqt>> (Accessing Application Operations with Voice Command)
- **Restrictions:**
    - If the tests are not performed with the predefined command set, the test results cannot be evaluated.
    - If the noise cancellation algorithms are not suitable for the mobile device processor capacity, system performance will be negatively affected.
- **Assumptions:**
    - The microphone on the user's device is active and working.
    - The system is capable of detecting environmental noise levels in real time.
- **Risks:**
    - Ambient noises may cause the system to incorrectly detect the voice command.
    - Voice recognition may be delayed if performance is not optimized.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** The Artificial Intelligence (AI) team is responsible for implementing this requirement.

- **Requirement ID:** REQ-604
- **Title:** Providing Dynamic In-App Help Based on User Behavior
- **Category:** This requirement belongs to the quality requirements category.
- **Description:** The system should dynamically provide in-app help messages and guidance based on user behavior. For example, if a user is making a payment for the first time, the system should automatically display descriptive instructions and tips. This help content should be context-aware to simplify the user experience and reduce transaction errors. The requirement aims to enable users to use the application more effectively and seamlessly.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by end users, UX design team, product management, and software development team.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-600 <<deriveReqt>> (Creating an Intuitive Interface for First-Time Users)
    - REQ-036 <<deriveReqt>> (Accessing Help and Frequently Asked Questions)
    - System modules that track user interaction history <<satisfy>>
- **Restrictions:**
    - Help messages can be used if they do not interrupt the user or interfere with their actions.
    - The requirement may be implemented if the help system does not consume too many resources.
- **Assumptions:**
    - The system can keep sufficient data about users' previous transactions.
    - Help contents are prepared according to predefined scenarios.
- **Risks:**
    - Triggering help messages at the wrong time or unnecessarily may disturb the user.
    - If user behavior cannot be analyzed correctly, help content may be ineffective.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Frontend team and Backend team are responsible for the implementation of this requirement.

- **Requirement ID:** REQ-605
- **Title:** Parking Guidance with Real-Time, Map-Based Navigation
- **Category:** This requirement belongs to the Quality and Business Requirements category.
- **Description:** The system should provide real-time, map-based navigation to help drivers quickly and easily find their chosen parking space. The navigation interface should include intuitive visuals, minimize user interaction, and provide clear directions. The goal is to enable drivers to reach their destinations efficiently without distraction.
- **Priority:** High
- **Stakeholders:** Drivers (end users), navigation and map service integration team, UX/UI design team, mobile application development team
- **Acceptance Criteria:**
    - After the user selects the parking lot, the system should automatically start routing via the map.
    - Navigation should show the most suitable route in real time, based on traffic density and road conditions.
    - The map interface should load in less than 3 seconds and be able to update the route without interruption.
    - It should be possible for the user to follow the guidance without any manual action.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-600 <<deriveReqt>> (Creating an Intuitive Interface for First-Time Users)
    - REQ-003 <<deriveReqt>> (Display of Real-Time Parking Availability)
- **Restrictions:**
    - The navigation function works if connected to the internet.
    - If map services depend on third-party providers (e.g. Google Maps API), the availability of these services is required.
- **Assumptions:**
    - The user's device can actively use location services.
    - The system can access location data with user permission.
- **Risks:**
    - When the GPS signal is weak, orientation accuracy may decrease.

    - Delays in the map service may negatively impact the user experience.
- **Status:** Draft
- **Responsible:** Frontend team, Backend team

- **Requirement ID:** REQ-606
- **Title:** Providing a Hands-Free Experience with Natural Voice Commands
- **Category:** This requirement belongs to the technical and quality requirements category.
- **Description:** The system should be able to understand user intent with high accuracy through natural voice commands and execute operations with minimal manual interaction. This should enable users to easily perform targeted operations without using their hands. The goal is to increase ease of use and safety, especially in situations requiring attention, such as driving.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by end users, AI development team, UX design team, and product management.
- **Acceptance Criteria:**
    - The system should be able to understand natural language commands containing defined user intentions with 90% accuracy.

    - Actions should be initiated by voice command only and completed without requiring manual interaction.

    - Voice commands should be able to get to the destination without requiring more than two steps (for example: “Guide me to the nearest available parking lot”).

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-603 <<deriveReqt>> (Voice Command Recognition in Noisy Environments)

    - REQ-004 <<deriveReqt>> (Accessing Application Operations with Voice Command)

    - REQ-409 <<deriveReqt>> (The system must be able to convert voice commands into structured commands with NLP and LLM)

    - REQ-410 <<deriveReqt>> (The system must be able to detect driver intent from voice commands)

- **Restrictions:**
    - The system will work fine if only certain languages are supported initially.

    - If the device's processing power is insufficient for some natural language operations, performance may decrease.

- **Assumptions:**
    - The microphone on the user's device is active and working properly.
    - Sufficient command variations are predefined to suit user intent.

- **Risks:**
    - Misunderstood commands may cause the user to lose time or experience operational errors.
    - Users can express their commands in natural ways that the system cannot understand.
    - A slow response time during voice operation may negatively impact the user experience.

- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** The Artificial Intelligence (AI) team and the Backend team are responsible for the implementation of this requirement.

- **Requirement ID:** REQ-607
- **Title:** Accessing the In-App Notification Center
- **Category:** This requirement belongs to the quality and user interface (UI) and user experience (UX) requirement categories.
- **Description:** The system must provide a unified and easily accessible in-app notification center where users can view all relevant alerts, reminders, and updates in one place. Notifications must be clearly, concisely, and organized, and delivered to the user in a timely and contextual manner.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by end users, software development team, product management, and UX/UI design team.
- **Acceptance Criteria:**
    - The user should be able to view all notifications from a single point in the application interface.
    - Notifications should be grouped by category (for example: system warning, user reminders, campaign).
    - Each new notification should be instantly displayed in the notification center by the system.

    - Unread notifications must be presented in a visually distinguishable manner.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-00 <<deriveReqt>> (Creating an Intuitive Interface for First-Time Users)

- **Restrictions:**
    - The requirement may be implemented if excessive notification displays do not negatively impact the user experience.

    - The module remains organized if notification data is automatically archived after a certain period of time.

- **Assumptions:**
    - The system has the infrastructure to correctly match user-specific and contextual notifications.

    - The content of the notifications is presented in JSON or a similar standard format.

- **Risks:**
    - Delayed delivery of notifications may result in a lack of information for the user.

    - If the notification center is cluttered or messy, users may miss important information.

- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Frontend team and Backend team are responsible for the implementation of this requirement.

- **Requirement ID:** REQ-608
- **Title:** Parking Lot Editing with Drag-and-Drop Visual Editor Without Requiring Technical Knowledge
- **Category:** This requirement belongs to the quality and user interface (UI) and user experience (UX) requirement categories.
- **Description:** The system should provide a no-code, drag-and-drop visual editor that allows parking providers to create and edit parking space layouts without requiring any technical knowledge. This editor should enable the visual placement of elements such as parking spaces, entry/exit points, and designated areas.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by parking providers, product management, software development team, and UX/UI design team.
- **Acceptance Criteria:**

    - Users should be able to arrange parking spaces using drag and drop and save the design.

    - New areas should be able to be added without any programming knowledge (for example: disabled parking space, entrance gate, direction sign).

    - Users should be able to view their edit history and undo actions.

    - Edited settlements must be verified by the system and incorrect definitions (e.g. settlement without an entry point) must be prevented.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-600 <<deriveReqt>> (Creating an Intuitive Interface for First-Time Users)

- **Restrictions:**
    - The expected performance will be achieved only if the visual editor runs fully functional on desktop devices.

    - If complex layouts require performance optimization, additional measures must be taken.

- **Assumptions:**
    - Parking providers are familiar with basic interface usage.

    - The system has the infrastructure to store the designs in the database.

- **Risks:**
    - In complex parking areas, the visual editor may be inadequate.

    - Users may cause functional problems by making incorrect placement.

    - Performance problems can negatively impact user experience.

- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** The Frontend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-609
- **Title:** Encryption of Sensitive Data
- **Category:** This requirement belongs to the technical and quality requirement category.
- **Description:** All sensitive data (passwords, etc.) must be encrypted using strong cryptographic algorithms, both during storage and transmission. This requirement is

implemented to ensure data security, prevent unauthorized access, and support compliance with data protection standards (KVKK, GDPR, etc.).

- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the information security team, software development team, system administrators, end users, and data scientists.
- **Acceptance Criteria:**
    - Sensitive data should be verified to be encrypted with AES-256 or a similar strong algorithm before being stored.
    - The use of a secure protocol such as TLS 1.2 or higher during data transmission must be documented with log records.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-449 <<deriveReqt>> (Phone Number Uniqueness Enforcement and Authentication Security)
- **Restrictions:**
    - The system operates efficiently as long as the performance impact of cryptographic algorithms is compatible with the encryption support in the existing infrastructure.
    - If the existing infrastructure has insufficient cryptographic support, cryptographic algorithms may negatively impact performance.
- **Assumptions:** For the implementation of this requirement, it is assumed that the system infrastructure supports the selected encryption algorithms.
- **Risks:** Inappropriate algorithm selection, incorrect key management, or performance degradation may negatively impact data security.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-610
- **Title:** Automatic Session Termination
- **Category:** This requirement belongs to the technical and quality requirement category.
- **Description:** For security purposes, users should be automatically logged out of the system after 15 minutes of inactivity. This requirement is mandatory to prevent unauthorized access and ensure session security. Before the session is terminated, the user should be notified that the session is about to expire, and the session duration should be extended upon request. The timeout should be reset based on user activity such as mouse movements, clicks, keyboard inputs, and touches on mobile devices.
- **Priority:** This requirement has been set to high priority.

- **Stakeholders:** This requirement was requested by the software development team, information security team and end users.
- **Acceptance Criteria:**
    - The system should log the user out after 15 minutes of inactivity.
    - When the session expires, the user should be shown a message that the system has logged out for security reasons.
    - If no further entry is made before the time expires, any unfulfilled reservations must be cancelled and the reserved resources released.
    - A warning should be displayed to the user 1 minute before the session is terminated, indicating that the time is about to expire, and the user should be able to continue the session if they wish.
    - The system should reset the time by detecting mouse movements, clicks, keyboard inputs, and touches on mobile as user activity **.**
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-609 <<deriveReqt>> (Encryption of Sensitive Data)
    - REQ-430 <<deriveReqt>> (Secure Session Management)
- **Limitations:** If a suitable warning mechanism is not developed, user experience may be negatively impacted.
- **Assumptions:** For this requirement to be implemented, it is assumed that the system can track user interactions (mouse movement, clicks, keyboard input, etc.).
- **Risks:** Users may experience data loss (for example, being suddenly logged out while filling out a form).
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-611
- **Title:** Authentication for Suspicious Login Attempts
- **Category:** This requirement belongs to the technical and quality requirements category.
- **Description:** When suspicious user activity, such as a high number of failed login attempts, is detected, the system must halt the relevant session and implement additional authentication mechanisms (e.g., two-factor authentication, email/SMS confirmation). This requirement is implemented to enhance account security and prevent unauthorized access.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the information security team, software development team and end users.

- **Acceptance Criteria:**
    - After 5 consecutive unsuccessful login attempts, the system should detect suspicious login.

    - After detection, the user should be asked for additional authentication via SMS, email or mobile application.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-610 <<deriveReqt>> (Automatic Session Termination)

    - REQ-609 <<deriveReqt>> (Encryption of Sensitive Data)

    - REQ-402 <<deriveReqt>> (The system must support optional registration and login with third-party authentication.)

    - REQ-429 <<deriveReqt>> (Rate Limiting and Abuse Detection Mechanisms)

    - REQ-448 <<deriveReqt>> (Smart Profile Pre-filling and Staged Enrollment with OAuth Integration)

- **Restrictions:** If the authentication system relies on external services (e.g. SMS provider), these services must be accessible.
- **Assumptions:** For this requirement to be implemented, it is assumed that the user's contact information is current and accessible in the system.
- **Risks:** False positive detections may result in users being unnecessarily blocked or having their access restricted.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-612
- **Title:** Enforcing Password Complexity Rules
- **Category:** This requirement belongs to the technical and quality requirements category.
- **Description:** The system must require user passwords to be generated according to specific complexity rules. Passwords must be at least 8 characters long and contain uppercase letters, lowercase letters, numbers, and special characters. This requirement will be implemented to protect against brute-force and dictionary attacks and to increase account security.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the software development team, information security team and end users.
- **Acceptance Criteria:**

    - The user should not be able to save a password that is at least 8 characters long and does not contain uppercase letters, lowercase letters, numbers or special characters.

    - Descriptive error messages should be provided to guide the user on the password creation or change screen.

- **Dependencies:**
    - REQ-611 <<deriveReqt>> (Authentication for Suspicious Login Attempts)

    - REQ-609 <<deriveR This requirement is contingent upon the completion of the following system components and requirements: eqt>> (Encryption of Sensitive Data)

    - REQ-439 <<deriveReqt>> (Enhanced Security: Credential Security and Session Management)

- **Restrictions:** Security measures must not unnecessarily complicate the user experience.
- **Assumptions:** To implement this requirement, it is assumed that the system can perform real-time verification during password entry.
- **Risks:** There is a possibility that users may struggle to create strong passwords and may compromise security by writing down passwords.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-613
- **Title:** Providing reCAPTCHA Integration on the Login Screen
- **Category:** This requirement belongs to the technical and quality requirements category.
- **Description:** The system must integrate a verification mechanism, such as Google reCAPTCHA, into the user login screen to protect against bots. This requirement is designed to ensure that only legitimate users can log in and to prevent automated attacks. The integration must ensure that login attempts are not made without completing the verification step and must record verification results in the system logs.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by the software development team and the information security team.
- **Acceptance Criteria:**
    - The system should display reCAPTCHA to the user on the login screen.

    - Login should not occur before reCAPTCHA verification is completed.

    - reCAPTCHA verification results (pass/fail) should be recorded in system logs.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-611 <<deriveReqt>> (Authentication for Suspicious Login Attempts)
- **Restrictions:** This requirement must be enforced by providing an alternative verification method in the event of an outage of the Google reCAPTCHA service.
- **Assumptions:** For this requirement to be implemented, it is assumed that the system has an internet connection and the API keys required for reCAPTCHA integration have been previously defined.
- **Risks:** Implementing this requirement may reduce user satisfaction due to the risk that reCAPTCHA will cause accessibility issues or negatively impact the user experience.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team and Frontend team are responsible for the implementation of this requirement.

- **Requirement ID:** REQ-614
- **Title:** Preventing the Uploading of Harmful File Types
- **Category:** This requirement belongs to the technical and quality requirements category.
- **Description:** The system must not accept executable files such as .exe files and other file types that could potentially cause harmful behavior when attempted by users. This requirement is implemented to increase system security and prevent the spread of malware. Additionally, checking not only the file extension but also the MIME type will increase accuracy.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the software development team, information security team, and system administrators.
- **Acceptance Criteria:**
    - The system should reject installations with file extensions such as .exe, .bat, .cmd, .js, .vbs, .scr.

    - In case of any attempt to upload an invalid file, a descriptive error message should be displayed to the user and the event should be logged.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-429 <<deriveReqt>> (Rate Limiting and Abuse Detection Mechanisms)
- **Restrictions:** This requirement may produce misleading results if applied based on file extension alone; it must be applied in conjunction with a MIME type check.
- **Assumptions:** To implement this requirement, it is assumed that the system has the infrastructure to perform file extension and MIME type analysis.
- **Risks:** There is a risk that malicious content may infiltrate the system through different means (for example, by changing the file extension).

- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-615
- **Title:** Email and Phone Number Verification Before Account Activation
- **Category:** This requirement belongs to the technical and quality requirements category.
- **Description:** The system must require verification of both the email address and phone number using a one-time password (OTP) before activating a user account. This requirement is implemented to ensure user identity and prevent the creation of fraudulent accounts. Verification must ensure that account activation does not occur until both communication channels are verified.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the software development team, information security team and end users.
- **Acceptance Criteria:**
    - When the user creates an account, the system should send an OTP to both email and phone number.

    - The user should not be able to activate the account without verifying both communication channels by entering the correct OTP codes.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-612 <<deriveReqt>> (Enforcement of Password Complexity Rules)

    - REQ-611 <<deriveReqt>> (Authentication for Suspicious Login Attempts)

- **Restrictions:** This requirement can be implemented without any issues as long as SMS and email service providers are functioning; however, service provider outages may negatively impact the verification process.
- **Assumptions:** For this requirement to be implemented, it is assumed that the email address and phone number provided by the user are valid and accessible, and that the system has the necessary infrastructure to send OTPs.
- **Risks:** There is a risk of user dissatisfaction or incomplete verification process due to delayed delivery of OTP.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-616
- **Title:** Providing OTP-Based Password Reset Support
- **Category:** This requirement belongs to the technical and quality requirements category.

- **Description:** The system must support password resets by authenticating with a one-time password (OTP) using a previously verified email address or phone number to allow users to reset forgotten passwords. This requirement is implemented to ensure account security and improve the user experience. The password reset process should not proceed without successful OTP verification.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the software development team, information security team and end users.
- **Acceptance Criteria:**
    - When the user creates a password reset request, the system should send an OTP to the registered email or phone number.
    - After entering the correct OTP, the user should be able to create a new password and the system should save it successfully.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-615 <<deriveReqt>> (Email and Phone Number Verification Before Account Activation)
- **Restrictions:** This requirement is dependent on SMS and email service providers being operational. Incorrect or incomplete contact information will cause the password reset to fail.
- **Assumptions:** For this requirement to apply, it is assumed that the user's contact information is already verified and up-to-date in the system.
- **Risks:** Delayed OTP delivery can lead to user dissatisfaction. Furthermore, malicious actors gaining access to the OTP can compromise account security.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-617
- **Title:** Collecting and Storing User Consent Preferences for Personal Data Use
- **Category:** This requirement belongs to the legal and regulatory and quality requirements categories.
- **Description:** The system must obtain explicit consent from the user regarding the processing of personal data, securely store these preferences, and comply with these consents when using the data. This requirement will be implemented in accordance with applicable data protection standards, such as the GDPR or KVKK. All data processing processes must be updated immediately when consents are withdrawn or changed by the user.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the information security team, software development team, legal department and end users.
- **Acceptance Criteria:**

    - The user must clearly see the purposes for which his personal data will be used, and no relevant transactions can be made without his consent.

    - The consent preferences given or withdrawn by the user must be securely recorded in the system and taken into account in all data processing processes.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-609 <<deriveReqt>> (Encryption of Sensitive Data)

- **Restrictions:** This requirement meets legal requirements provided local and international data protection laws are complied with.
- **Assumptions:** For the implementation of this requirement, it is assumed that the user interface is designed to be clear and accessible for consent management and that the system follows updates to legal standards.
- **Risks:** Processing data without consent carries a risk of legal sanctions. Incomplete or inaccurate records may lead to violations of user rights.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-618
- **Title:** Protection Against Denial of Service and Brute-Force Attacks
- **Category:** This requirement belongs to the quality and technical requirements categories.
- **Description:** To protect against exploitation and denial of service (DoS) attacks, the system must implement request rate limiting and include abuse detection mechanisms to detect suspicious activity. This requirement is implemented to prevent misuse of system resources and maintain system availability. Anomalies should be identified by analyzing incoming requests during specific time periods, and access should be temporarily restricted when necessary.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the information security team, software development team and system administrators.
- **Acceptance Criteria:**
    - Requests exceeding a certain number per second per user or IP should be blocked or limited.

    - The system should detect and log anomalies such as repeated failed login attempts and request fluctuations within a certain time period and, if necessary, temporarily stop access.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-611 <<deriveReqt>> (Authentication for Suspicious Login Attempts)

    - REQ-613 <<deriveReqt>> (Providing reCAPTCHA Integration on the Login Screen)
- **Restrictions:** When restriction thresholds are set too tightly, the risk of genuine users being blocked may increase.
- **Assumptions:** To implement this requirement, it is assumed that the system has the infrastructure to analyze user identities and IP addresses, and that logging and warning systems are active.
- **Risks:** Incorrectly configured breakpoints can reduce service availability or lead to attacks going undetected.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-619
- **Title:** Additional Authentication for Login from a New Device or Location
- **Category:** This requirement belongs to the quality and technical requirements categories.
- **Description:** The system must perform an additional authentication step when attempting to log in from a previously unused device or an unusual location. This authentication must be performed using a one-time password (OTP) sent to the user's pre-verified email address or phone number. This requirement is implemented to enhance account security and prevent unauthorized access.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the information security team, software development team and end users.
- **Acceptance Criteria:**
    - The system must detect login attempts from a previously unused device or an unusual location.
    - In this case, an OTP should be sent to the user and the login process should not be performed before the verification is completed.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-615 <<deriveReqt>> (Email and Phone Number Verification Before Account Activation)
    - REQ-616 <<deriveReqt>> (Providing OTP-Based Password Reset Support)
    - REQ-611 <<deriveReqt>> (Authentication for Suspicious Login Attempts)
    - REQ-402 <<deriveReqt>> (The system must support optional registration and login with third-party authentication)

- **Limitations:** False positives in location or device detection may negatively impact user experience.
- **Assumptions:** To implement this requirement, it is assumed that the system has the infrastructure to analyze device identification information and IP location information, and that user contact information has been previously verified.
- **Risks:** Prolonged login process due to security checks for legitimate users may lead to user dissatisfaction.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-620
- **Title:** Storing Passwords Securely
- **Category:** This requirement belongs to the quality, technical and legal/regulatory requirement categories.
- **Description:** All user passwords must be stored securely using cryptographic hash algorithms (e.g., bcrypt, scrypt, or Argon2), which are irreversible and strong encryption methods. Storing passwords in plain text is strictly prohibited. This requirement aims to reduce the risk of data breaches and ensure compliance with legal data protection standards (KVKK, GDPR, etc.) by maintaining the highest level of password security.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the software development team and the information security team.
- **Acceptance Criteria:**
    - User passwords should only be stored in the system encrypted with strong hashing algorithms (e.g. bcrypt, scrypt, Argon2).
    - Passwords that are not encrypted or stored with weak algorithms (e.g. MD5, SHA-1) should be detected and rejected by the system.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-611 <<deriveReqt>> (Authentication for Suspicious Login Attempts)
    - REQ-612 <<deriveReqt>> (Enforcement of Password Complexity Rules)
    - REQ-609 <<deriveReqt>> (Encryption of Sensitive Data)
- **Limitations:** System performance may be negatively impacted when hash algorithms place a high load on the processor. Incorrect configuration may lead to performance issues.
- **Assumptions:** To implement this requirement, it is assumed that the system has the infrastructure to support strong and modern hash algorithms.

- **Risks:** Misconfigured encryption algorithms or lack of salting can lead to security vulnerabilities.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-621
- **Title:** Providing High Availability (99.95% Uptime)
- **Category:** This requirement belongs to the quality requirement category.
- **Description:** The system must be designed with a minimum uptime target of 99.95%, with redundant servers, load balancers, and automatic failover mechanisms to ensure uninterrupted service even in the event of component failures. This requirement will be implemented to ensure the uninterrupted availability of critical services and maintain user satisfaction.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by system users and infrastructure administrators.
- **Acceptance Criteria:**
    - The system must provide over 99.95% availability.
    - Failover and load balancing systems must be actively operating.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-622 <<deriveReqt>> (Real-Time System Monitoring)
    - REQ-623 <<deriveReqt>> (Ensuring Service Continuity in Case of Partial Data Accessibility)
    - REQ-438 <<deriveReqt>> (Robust User Experience with Smart Offline Backup Mode)
- **Limitations:** System performance may be affected if there are hardware and network limitations.
- **Assumptions:** To implement this requirement, the installation of redundant structures will be completed.
- **Risks:** Infrastructure issues may cause outages and lower than planned availability.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-622
- **Title:** Real-Time System Monitoring

- **Category:** This requirement belongs to the quality and technical requirements categories.
- **Description:** The system must continuously monitor its critical components, such as the database, IoT network, and APIs, and send real-time alerts to administrators in the event of performance degradation or outages. This requirement is implemented to ensure high system availability, detect problems early, and prevent service disruptions.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by system administrators.
- **Acceptance Criteria:**
    - The system must continuously monitor critical components (database, IoT network, APIs, etc.).
    - In case of performance degradation or interruption, managers should be immediately notified.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-621 <<deriveReqt>> (Ensuring High Availability (99.95% Uptime))
    - REQ-632 <<deriveReqt>> (Updating Real-Time Occupancy Data)
    - REQ-403 <<deriveReqt>> (The system must be able to collect and manage real-time occupancy data from IoT devices)
    - REQ-404 <<deriveReqt>> (The system must provide future demand forecasting and early notification based on parking occupancy analysis)
    - REQ-414 <<deriveReqt>> (The system must be able to display real-time occupancy status in the user interface)
    - REQ-424 <<deriveReqt>> (The system must be able to maintain performance under high traffic)
    - REQ-438 <<deriveReqt>> (Robust User Experience with Smart Offline Backup Mode)
    - REQ-217 <<deriveReqt>> (Sensor Error Reporting)
- **Limitations:** Reliability of the monitoring infrastructure is necessary for uninterrupted monitoring of the system.
- **Assumptions:** For the implementation of this requirement, it is assumed that access to system components can be provided by the monitoring infrastructure.
- **Risks:** Malfunctions that may occur in the monitoring system itself may prevent events from being detected.

- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-623
- **Title:** Ensuring Service Continuity in Case of Partial Data Accessibility
- **Category:** This requirement belongs to the quality requirements category.
- **Description:** In situations where access to real-time data is partially unavailable, the system must maintain basic functionality using the latest valid data or summary information based on user behavior. This requirement is implemented to ensure uninterrupted system operation and preserve the user experience.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by end users.
- **Acceptance Criteria:**
    - In the absence of real-time data, the system must resort to historical data.
    - Essential functions must continue uninterrupted despite data loss.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-621 <<deriveReqt>> (Ensuring High Availability (99.95% Uptime))
    - REQ-632 <<deriveReqt>> (Updating Real-Time Occupancy Data)
    - REQ-438 <<deriveReqt>> (Robust User Experience with Smart Offline Backup Mode)
    - REQ-216 <<deriveReqt>> (Manual Update of Parking Space Status)
    - REQ-217 <<deriveReqt>> (Sensor Error Reporting)
- **Limitations:** Using outdated data may reduce information accuracy.
- **Assumptions:** For the implementation of this requirement, it is assumed that the system stores historical data.
- **Risks:** Outdated recommendations and guidance can negatively impact user experience.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-624
- **Title:** Ensuring Transaction Integrity

- **Category:** This requirement belongs to the quality and technical requirements categories.
- **Description:** The system must perform all associated database operations completely, or if the operation fails, roll back all changes to ensure data integrity. This requirement is implemented to maintain data consistency and increase system reliability in the event of an error.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by software and database developers.
- **Acceptance Criteria:**
    - When an error occurs during processing, the system must roll back all intermediate changes.
    - Rollback must be verified in automated tests.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-622 <<deriveReqt>> (Real-Time System Monitoring)
    - REQ-623 <<deriveReqt>> (Ensuring Service Continuity in Case of Partial Data Accessibility)
    - REQ-217 <<deriveReqt>> (Sensor Error Reporting)
- **Limitations:** The applicability of this requirement cannot be verified without testing the performance impacts.
- **Assumptions:** To implement this requirement, it is assumed that the database supports rollback.
- **Risks:** If the rollback mechanism fails, data inconsistency may occur.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-625
- **Title:** Keeping Detailed Error Records
- **Category:** This requirement belongs to the technical requirement category.
- **Description:** Error logs must be recorded, including timestamps, user actions, and system status information. This requirement is implemented to facilitate debugging, quickly identify causes of failures, and increase system reliability.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by software developers and system administrators.
- **Acceptance Criteria:**
    - Each error record must include a timestamp.

    - The error log should be saved along with user action and system status information.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-417 <<deriveReqt>> (The system must be able to record and monitor user activities daily)
    - REQ-421 <<deriveReqt>> (System must support dark and light mode in the user interface)
    - REQ-422 <<deriveReqt>> (The system should automatically log the user out when the session expires)
    - REQ-423 <<deriveReqt>> (The system must support accessibility standards for users with disabilities)
- **Limitations:** Growing log file size requires storage management.
- **Assumptions:** For this requirement to be implemented, it is assumed that the system can identify user actions and internal state.
- **Risks:** Incomplete or inaccurate log records can complicate the debugging process and increase the time it takes to resolve the issue.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-626
- **Title:** Trouble-Free Installation of Software Updates
- **Category:** This requirement belongs to the quality and technical requirements categories.
- **Description:** Software updates must be installed seamlessly without requiring reconfiguration by the user. This requirement is implemented to ensure the system remains up-to-date and secure without disrupting the user experience. User settings must be preserved during updates, and the installation process must complete without user intervention.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by end users and system administrators.
- **Acceptance Criteria:**
    - User settings should be preserved after the update.
    - The installation process should be completed without requiring user intervention.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:

    - REQ-632 <<deriveReqt>> (Updating Real-Time Occupancy Data)

    - REQ-633 <<deriveReqt>> (Updating Parking Data at Specific Intervals)

- **Restrictions:** In case of different loading behaviors depending on the platform, compatible solutions must be developed.
- **Assumptions:** For this requirement to be implemented, it is assumed that the system has an update mechanism.
- **Risks:** Faulty updates may negatively impact system stability and lead to service interruptions.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-627
- **Title:** Providing an Admin Monitoring Panel
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system must provide an administrator monitoring dashboard that displays system health, usage statistics, and error trends. This dashboard will be implemented to enable administrators to monitor system performance, detect potential problems early, and effectively manage the system.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by system administrators.
- **Acceptance Criteria:**
    - The panel should include visual displays of system health.

    - Usage and error data should be accessible via the panel.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-417 <<deriveReqt>> (The system must be able to record and monitor user activities on a daily basis.)

    - REQ-421 <<deriveReqt>> (The system must support dark and light modes in the user interface.)

    - REQ-422 <<deriveReqt>> (Logging System Events and Errors)
- **Restrictions:** In order for the panel to work, the system must have real-time data processing capacity.
- **Assumptions:** For this requirement to be implemented, it is assumed that the system generates monitoring data regularly.
- **Risks:** Panel malfunction may delay problem detection and negatively affect service quality.
- **Status:** The current status of this requirement is determined as a draft.

- **Responsible:** Frontend team, Backend team and Artificial Intelligence (AI) team are responsible for the implementation of this requirement.

- **Requirement ID:** REQ-628
- **Title:** Structural Error and Event Logging
- **Category:** This requirement belongs to the quality and technical requirements categories.
- **Description:** The system must continuously log critical events and errors in a structured format. This requirement is implemented to support system diagnostics, debugging, and performance analysis. The log format should be designed to be suitable for systematic analysis and reporting.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by developers and system maintenance personnel.
- **Acceptance Criteria:**
    - Critical incidents should be recorded in a timely and structured manner.
    - The log format should be suitable for systematic analysis and reporting.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-622 <<satisfy>> (Real-Time System Monitoring)
    - REQ-423 <<deriveReqt>> (The system must support accessibility standards for users with disabilities.)
- **Limitations:** Large data volumes may occur in case of high frequency events.
- **Assumptions:** For this requirement to be implemented, it is assumed that the system can categorize events.
- **Risks:** Structural corruption or data loss can complicate error resolution and delay analysis processes.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team and Artificial Intelligence (AI) team are responsible for the implementation of this requirement.

- **Requirement ID:** REQ-629
- **Title:** Ensuring Application Availability (99.9% Uptime)
- **Category:** This requirement belongs to the quality requirement category.
- **Description:** The application must be available 99.9% of the time during each 30-day period. This requirement is implemented to protect the user experience, ensure

uninterrupted service, and maintain high system reliability. Availability must be verified with independent monitoring tools (such as UptimeRobot, Pingdom, etc.).

- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by end users and system administrators.
- **Acceptance Criteria:**
    - The application must be available at least 99.9% of the time within 30 days.
    - Availability status should be verified with independent tools such as UptimeRobot or Pingdom.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-634 <<deriveReqt>> (Providing Caching for Performance Optimization)
    - REQ-632 <<deriveReqt>> (Updating Real-Time Occupancy Data)
- **Restrictions:** In case of dependency on external service providers, the availability of these services must be ensured.
- **Assumptions:** For this requirement to be implemented, it is assumed that the application runs real-time services without interruption.
- **Risks:** Access disruptions can severely negatively impact user experience and reduce service reliability.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-630
- **Title:** Integration of New Sensor Types with Reusable Interfaces
- **Category:** This requirement belongs to the quality and technical requirements categories.
- **Description:** The system must support the integration of new sensor types through modular interfaces. This requirement ensures that the core processing logic is reusable and the system can be expanded without requiring architectural changes. This ensures that adding new sensor types requires minimal changes to the existing architecture.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by the hardware developers and the software integration team.
- **Acceptance Criteria:**
    - When a new sensor type is introduced into the system, the existing processing logic must operate unchanged.
    - No structural changes at the architectural level should be required during integration.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-632 <<deriveReqt>> (Updating Real-Time Occupancy Data)
    - REQ-633 <<deriveReqt>> (Updating Parking Data at Specific Intervals)
- **Restrictions:** Backward compatibility may be limited if older sensor formats are used.
- **Assumptions:** To implement this requirement, it is assumed that the system has been developed in accordance with modular architecture principles.
- **Risks:** Integrating incompatible sensors may require additional development and prolong the integration process.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-631
- **Title:** Ensuring a Modular and Sustainable Code Structure
- **Category:** This requirement belongs to the quality and technical requirements categories.
- **Description:** The codebase must adhere to clean code practices and be modular. This requirement is implemented to increase software maintainability, reduce maintenance costs, and allow for the addition of new features with minimal refactoring. The code structure must be flexible enough to accommodate changes and include reusable components.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by software developers.
- **Acceptance Criteria:**
    - The modularity and reusability of the code should be measured with static analysis tools.
    - A low repetition rate and a high compliance score should be achieved in code reviews.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-632 <<deriveReqt>> (Updating Real-Time Occupancy Data)
- **Limitations:** Refactoring existing code can lengthen the development process.
- **Assumptions:** For the implementation of this requirement, it is assumed that the development team is familiar with clean code principles.
- **Risks:** If code modularity is not ensured, maintenance cost and error rate may increase.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team and Frontend team are responsible for the implementation of this requirement.

- **Requirement ID:** REQ-632
- **Title:** Updating Real-Time Occupancy Data
- **Category:** This requirement belongs to the technical and functional requirements categories.
- **Description:** The system must collect real-time occupancy data from IoT devices and transmit it to the user interface with minimal latency. Under normal operating conditions, this data should appear updated in the user interface within 500 milliseconds or less. This requirement is implemented to ensure users have access to real-time status information and ensure high system performance.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the software development team, IoT hardware team, and end users.
- **Acceptance Criteria:**
    - Occupancy data from IoT devices should become visible in the user interface within 500 ms.
    - Under normal operating conditions, the system should not exceed 500 ms update time at 95%.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-622 <<deriveReqt>> (Real-Time System Monitoring)
    - REQ-424 <<deriveReqt>> (The system must maintain its performance under high traffic.)
    - REQ-434 <<deriveReqt>> (IoT Device Fault Management and Gentle Degradation)
- **Limitations:** Performance may be impacted when latency is limited by network infrastructure and device response time.
- **Assumptions:** For the implementation of this requirement, it is assumed that IoT devices are working properly and can transmit data instantly.
- **Risks:**
    - Extended data update time due to network delay or connection issues
    - Hardware failures or software errors in IoT devices
    - Decreased system performance under heavy traffic
    - Incorrect information is presented to the user due to data loss or corruption

- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-633
  Title: Updating Parking Data at Periodic Intervals
- **Category:** This requirement belongs to the technical and quality requirements category.
- **Description:** The system must refresh parking-related data, such as parking availability and traffic density, at configurable intervals (e.g., every 10-30 seconds) to maintain accuracy without overloading data sources. This requirement is implemented to ensure users have access to up-to-date information and to ensure the system is compatible with the data sources.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by the software development team, system operations team, and end users.
- **Acceptance Criteria:**
  - The system should automatically refresh parking-related data at configured time intervals.
  - The update interval should be adjustable between 10-30 seconds by the system administrator.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
  - REQ-632 <<deriveReqt>> (Updating Real-Time Occupancy Data)
- **Restrictions:** The system operates stably as long as the query frequency limits of the data sources are not exceeded.
- **Assumptions:** For the implementation of this requirement, it is assumed that the data sources can provide reliable data at certain intervals.
- **Risks:** Too frequent data updates may lead to performance degradation or excessive usage of data resources.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-634
- **Title:** Providing Caching for Performance Optimization
- **Category:** This requirement belongs to the technical and quality requirements category.
- **Description:** The system must implement a caching mechanism for frequently accessed data (e.g., parking availability and user preferences). This requirement is implemented to improve system performance by reducing response times and reducing the load on the back-end system. The caching strategy should be designed to maintain data freshness.

- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested by the software development team and the system operations team.
- **Acceptance Criteria:**
- In a caching system like Redis, the cache hit rate should be monitored.

- It should be observed that the response time decreases significantly for repeated data requests.

- **Dependencies:-**
- **Limitations:** If data is not kept current, the caching strategy will be implemented incorrectly.
- **Assumptions:** To implement this requirement, it is assumed that frequently used data is in a easily cacheable structure.
- **Risks:** Incorrect caching policies can lead to data inconsistency or outdated information being displayed.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

- **Requirement ID:** REQ-635
- **Title:** Checking Backend Latency
- **Category:** This requirement belongs to the technical and quality requirement category.
- **Description:** All backend API calls must return a response within 200 milliseconds under normal network conditions. This requirement is implemented to protect the user experience and prevent functional issues (e.g., reservation errors, data not loading). Performance monitoring should be verified through testing with specific tools.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested by the software development team and the system testing team.
- **Acceptance Criteria:**
    - In load tests under typical load, 95% of API responses should be under 200 ms.

    - Test results with tools such as Postman and JMeter should confirm this period.

- **Dependencies:** -
- **Restrictions:** Measurements are only valid under normal network conditions and with optimized system configuration.
- **Assumptions:** For the implementation of this requirement, it is assumed that the system infrastructure and network connectivity meet the minimum performance requirements.
- **Risks:** Under high traffic, API response time may increase, which may negatively impact user satisfaction and system reliability.
- **Status:** The current status of this requirement is determined as a draft.
- **Responsible:** Backend team is responsible for the implementation of this requirement.

G)Flow Requirements

Requirement REQ-700

- **Requirement ID:** REQ-700
- **Title:** Ability to upload vehicle photos
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** Drivers should be able to upload one or more photos for each vehicle registered on their profile through the app. This feature should support system verification, license plate verification, and vehicle type verification. Users can upload photos for existing vehicles, as well as optionally during vehicle registration. Uploaded photos should be viewable, deletable, and editable later.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or impacted by Drivers, Mobile Development Teams (UI/UX), Backend Teams, Data Security Team, QA (Quality Assurance) Team, Other Service Providers, Security Policy Makers.
- **Acceptance Criteria:**
    - The system must support uploading photos, up to 5 photos, during user vehicle registration or on the current vehicle profile page.
    - Each uploaded photo must be added to the gallery section within the application and presented to the user in a listable format.
    - The system should provide the user with the ability to view, delete and modify uploaded photos.
    - The system should display a notification message to the user after each successful installation.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-002 «deriveReqt» (Driver Ability to Manage Profile and Vehicle Information)
    - REQ-034 «trace Reqt» (Permanent Deletion of User Profile and Data)
    - REQ-202 «trace Reqt» (Parking Officer's Ability to See Reservation Information)
    - REQ-502 «trace Reqt» (Secure Data Sharing and Receiving with Official Institutions)
    - «satisfy» Media upload module
    - «verify» Vehicle photo upload test scenario
- **Restrictions:**
    - This requirement should only be applied to certain formats and files less than 10 MB due to system installation size and format limitations.
    - This requirement should be implemented to allow the first 5 images to be uploaded due to the maximum limit of 5 photos per vehicle.
- **Assumptions:**
    - This requirement is assumed to have sufficient storage space and internet connection on the user's device.

    - For this requirement to apply, it is assumed that camera or gallery access permission has been granted.
- **Risks:**
    - Implementing this requirement may negatively impact processing continuity due to the risk of timeouts when uploading large photos.
    - Implementing this requirement could create privacy violations and legal issues under the KVKK due to the risk of the user uploading photos containing personal information or images of other individuals.
    - Implementing this requirement may lead to system abuse due to the risk of uploading inappropriate content (e.g. obscene, offensive or irrelevant images).
    - Implementing this requirement may lead to failures in plate checks and vehicle type identification due to the risk that the uploaded photo is not clear enough.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-701

- **Requirement ID:** REQ-701
- **Title:** Defining the Default Payment Method
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** A driver can select one of the methods linked to their account as the default payment method. The default method will automatically be the suggested option during checkout. The user can change the default method at any time.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or impacted by Drivers, Frontend Teams, UI/UX Design Teams, Backend Teams, Other Service Providers.
- **Acceptance Criteria:**
    - The user should be able to mark one of the available saved payment methods as the “default payment method”.
    - The system should automatically display the default payment method as selected when the payment screen opens.
    - When the user changes or deletes the default payment method, the system should display a confirmation message.
    - Once the default payment method is successfully completed, the system should instantly reflect this transaction in the user interface.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-500 «deriveReqt» (Third Party Payment Systems Integration)
    - REQ-013 «trace Reqt» (Payment via App)
    - «satisfy» Payment system
    - «verify» Add default payment test case
- **Restrictions:**

    - This requirement must be implemented in such a way that more than one method cannot be assigned as the default due to the restriction that each user can only choose one default method.
    - This requirement should be implemented in a way that disables empty payment lists due to the restriction that at least one valid payment method must be present before a default method can be assigned.
- **Assumptions:** For this requirement to apply, it is assumed that the system can correctly retrieve all registered payment methods of the user.
- **Risks:** Implementing this requirement may lead to failed payments due to the risk of selecting the wrong payment method by default.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-702

- **Requirement ID:** REQ-702
- **Title:** Getting Manual Parking Approval
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** In parking areas where there are no sensors or parking attendants, the driver should be able to confirm manually by pressing the parked button.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or impacted by Drivers, Mobile Development Teams (UI/UX), Backend Teams.
- **Acceptance Criteria:**
    - The system should only activate the “I Parked” button in parking areas where there are no sensors or parking attendants.
    - When the user presses the “Parked” button, the system should immediately display a confirmation notification message.
    - When the user tries to access the button in areas with sensors or where a parking attendant is present, the system should display this button as passive.
    - When an error occurs during a manual parking operation, the system should present the user with a meaningful error message explaining the cause.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - REQ-632 «trace Reqt» (Real-Time Occupancy Data Update)
    - REQ-705 «trace Reqt» (Post Parking Information Notification)
    - «satisfy» Parking control module
    - «verify» Manual parking confirmation test scenario
- **Restrictions:** This requirement must be implemented in a way that a new session cannot be started while a new one is in progress due to the restriction of supporting only one active reservation at a time.
- **Assumptions:** For this requirement to be implemented, it is assumed that user location information can be obtained accurately.

- **Risks:**
    - Implementing this requirement may lead to false parking sessions starting when the driver manually confirms before reaching the actual parking space, due to the risk of the location service detecting incorrect or inaccurate locations.
    - Implementing this requirement may lead to early parking session start and charging errors when the driver manually confirms the session without parking due to the risk of the location service being disabled.
    - Implementing this requirement may lead to manual approval outside the parking area due to the risk of incorrect user location information being received and may lead to accidental or intentional abuse of the system.
    - Implementing this requirement may lead to the user being unable to determine whether the transaction was successful or not and attempting to perform the transaction again, due to the risk of the confirmation notification being delayed or not arriving at all.
    - Implementing this requirement may cause this feature to be active in the wrong locations due to the risk of not correctly identifying areas without sensors and parking attendants.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-703

- **Requirement ID:** REQ-703
- **Title:** Ability to Search for Parking in the Background
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** If a driver can't find a suitable parking spot, they can activate Background Search mode through the app. When this mode is active, the system automatically searches for available parking spaces while the driver continues driving. When a suitable spot is found, the driver is notified. The notification offers the option to accept or decline. If accepted, the reservation is automatically made; if rejected, the search is stopped.
- **Priority:** This requirement has a low priority.
- **Stakeholders:** This requirement is requested or impacted by Drivers, Mobile Development Teams (UI/UX), Backend Teams, Other Service Providers.
- **Acceptance Criteria:**
    - When the user enables Background Search mode, the system must indicate this mode as active with an icon or status indicator on the screen.
    - The system should send instant push notification to the user when it finds a suitable parking space while the background search mode is active.
    - In the notification, the user should be presented with the options of Accept and Decline; if Accept is selected, automatic booking should be made, if Decline is selected, the mode should be disabled.
- **Dependencies:** This requirement is contingent upon completion of the following system components and requirements:
    - REQ-400 «trace Reqt» (Access to Real-Time Location Data (GPS))

    - REQ-039 «trace Reqt» (Customization of Notification Type Preferences)
    - REQ-416 «trace Reqt» (Unified In-App Notification System for All User Types)
    - «satisfy» Parking space finding module
    - «satisfy» Notification module
    - «allocate» Real-time GPS data
    - «verify» Real-Time Background Search test case
- **Restrictions:**
    - This requirement should only be implemented when GPS and internet access are available, as location services and network connectivity must be active.
    - This requirement should be limited to making calls at set intervals only when the app is running in the background, for device battery consumption and performance management reasons.
- **Assumptions:** For this requirement to be implemented, it is assumed that the system has access to up-to-date parking lot data.
- **Risks:**
    - Implementing this requirement may increase the device's battery consumption due to the system constantly performing searches in the background.
    - Enforcing this requirement may lead to unnecessary resource usage because of the risk that the user may unknowingly leave this mode enabled.
    - Implementing this requirement may result in missing suitable parking spaces due to the risk of temporary interruptions in GPS or internet connection.
    - Implementing this requirement may result in incorrect parking suggestions being presented to the user due to the risk of incorrect location detection.
    - Implementing this requirement could lead to distraction and pose a safety risk due to the risk of the driver receiving notifications while in motion.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-704

- **Requirement ID:** REQ-704
- **Title:** Automatic Cancellation of Reservation in Case of Late Arrival
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** If the driver does not arrive at the reserved parking space within the specified time, the system automatically cancels the reservation.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by Drivers, Backend Teams, Frontend Teams, Other Service Providers.
- **Acceptance Criteria:**
    - If the system does not detect vehicle entry within 10 minutes of the reservation start time, the relevant reservation will be automatically canceled.
    - When a reservation is cancelled, the system should send an instant push notification to the user.

    - The cancelled reservation should be marked as cancelled and listed in the user booking history.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-632 «trace Reqt» (Real-Time Occupancy Data Update)
    - REQ-716 «trace Reqt» (Reservation Cancellation Restrictions and Justification Collection Process)
    - REQ-500 «trace Reqt» (Third Party Payment Systems Integration)
    - REQ-415 «trace Reqt» (Multi-Channel Notification Support)
    - «satisfy» Automatic Reservation Cancellation Service
    - «verify» Automatic cancellation with timeout test scenario
- **Restrictions:** This requirement should only be implemented reliably on devices that are correctly time synchronized, as it measures time based on the system clock.
- **Assumptions:** For this requirement to be implemented, it is assumed that system time synchronization is functioning properly.
- **Risks:**
    - Implementing this requirement may result in the user being subject to unfair cancellation of bookings due to traffic congestion, roadworks or unexpected delays.
    - Implementing this requirement risks delayed or incorrect transmission of GPS data, which could result in the system failing to detect the vehicle entering and inadvertently cancelling the reservation.
    - Implementing this requirement may lead to the user arriving at the parking area to find the space full due to the risk of not receiving the reservation cancellation notification in time.
    - Implementing this requirement could result in loss of revenue for parking providers due to the risk of frequent reservation cancellations and a negative perception of the system's reliability.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-705

- **Requirement ID:** REQ-705
- **Title:** Post-Parking Information Notification Can Be Sent
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** After parking the vehicle, the driver should receive a notification from the system containing basic information about the parking session. This notification should include the name of the parking space, its exact location (floor/area), session start and end times, and the parking area's operating hours.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by Drivers, Backend Teams, Frontend Teams, Other Service Providers.
- **Acceptance Criteria:**

    - The system should send instant push notification to the user when the parking session is started.
    - The notification must include the name of the parking space, floor/area information, session start time, planned end time and parking area operating hours.
    - If the user has push notification permissions turned off, the system should display this information to the user via an in-app message.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-415 «trace Reqt» (Multi-Channel Notification Support)
    - REQ-416 «trace Reqt» (Unified In-App Notification System for All User Types)
    - «satisfy» Notification module
    - «satisfy» Parking information service
    - «verify» Park session initiation test scenario
- **Restrictions:** This requirement must be implemented in a way that only works when push notification permissions are granted, as the device must have push notification permissions enabled to send notifications.
- **Assumptions:** To implement this requirement, it is assumed that the system has information about the parking area and the relevant session and can display this information correctly.
- **Risks:**
    - Implementing this requirement could lead to drivers being unable to track their parking session due to the risk of the notification arriving late or not at all.
    - Implementing this requirement may result in incomplete or incorrect information being displayed and misleading the user due to the risk of incompatibility in data formats between different parking providers.
    - Implementing this requirement may mislead the user because there is a risk that this information will not be updated if there is a change to the scheduled end time included in the notification.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-706

- **Requirement ID:** REQ-706
- **Title:** Creating Template-Based Parking Area Layout
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** A provider can create a parking lot layout by selecting a template from a library of pre-built parking lot layouts in the system. These templates can be listed with filters such as total capacity, number of floors, and parking type, and can be customized after selection.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement was requested or is impacted by Parking Providers, UI/UX Design Teams, Frontend Teams.

- **Acceptance Criteria:**
    - The system should present a template gallery containing at least 10 different layouts to the user when they want to create a parking area layout.
    - The user should be able to narrow down the templates in the gallery with the help of filters such as total capacity, number of floors and parking type.
    - On the selected template, the user should be able to add, remove and edit objects using drag and drop.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-102 «deriveReqt» (Defining and adding Parking areas and Parking spots)
    - REQ-110 «deriveReqt» (Parking Area Plan Creation and Artificial Intelligence-Assisted Optimization)
    - REQ-125 «trace Reqt» (Visual Customization of Parking Area and Adding Components)
    - REQ-418 «trace Reqt» (Indoor Mapping and Floor Plan Integration)
    - «satisfy» Parking Space Creation Module
    - «verify» Parking area creation test scenario
- **Restrictions:** This requirement must be implemented in a way that the relevant areas are disabled in single-story plans, as multi-story structure support will only be active in multi-story plans.
- **Assumptions:** For this requirement to apply, it is assumed that the template library contains up-to-date and complete templates.
- **Risks:**
    - Implementing this requirement may create a mismatch with the actual physical structure of the parking area due to the risk of choosing the wrong template.
    - Implementing this requirement may lead to the parking provider not being able to find a suitable template due to the risk that the filtering system will not work correctly.
    - Implementing this requirement may limit the user by providing limited choice due to the risk that the template library may be outdated or may not cover some scenarios.
    - Implementing this requirement can result in wasted effort and time due to the risk that edits made to the template may not be saved permanently.
    - Implementing this requirement may reduce the efficiency of parking space use due to the risk of incorrectly designed plans and may lead to user dissatisfaction.
- **Status:** The status of this requirement is draft.
- **Responsible:** The Frontend team is responsible for the implementation of this requirement.

Requirement REQ-707

- **Requirement ID:** REQ-707
- **Title:** Creating Park Layout with Artificial Intelligence-Assisted Photography
- **Category:** This requirement belongs to the functional requirement category.

- **Description:** A provider can upload one or more photos of a parking area, which the system can then process to automatically detect parking spaces, obstacles, and road flows. The detected layout can then be manually edited.
- **Priority:** This requirement has a low priority.
- **Stakeholders:** This requirement is requested or impacted by Parking Space Providers, AI/AI Teams, UI/UX Design Teams, Frontend Teams, Backend Teams.
- **Acceptance Criteria:**
    - After the user uploads at least one photo, the AI edit option must be active.
    - The system must detect parking lot boundaries and obstacles from the photo with 80% accuracy.
    - The detected parking pattern must be manually editable by the user.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-102 «deriveReqt» (Defining and adding Parking areas and Parking spots)
    - REQ-110 «deriveReqt» (Parking Area Plan Creation and Artificial Intelligence-Assisted Optimization)
    - REQ-125 «trace Reqt» (Visual Customization of Parking Area and Adding Components)
    - REQ-418 «trace Reqt» (Indoor Mapping and Floor Plan Integration)
    - REQ-436 «trace Reqt» (No-Code Drag-and-Drop Visual Parking Lot Design Editor)
    - «satisfy» Parking Space Creation Module
    - «verify» Parking area creation test scenario
- **Restrictions:** This requirement must be implemented in a way that does not work with low images due to the restriction that the photo resolution must be at least 1080x720 for the AI-supported model to be able to analyze successfully.
- **Assumptions:** To implement this requirement, it is assumed that the AI can evaluate the photo and create an appropriate layout.
- **Risks:**
    - Implementing this requirement may result in an incorrect layout due to the risk that the AI may incorrectly detect parking space boundaries, obstacles, or road flows in the image.
    - Implementing this requirement may prevent the system from operating successfully due to the risk of poor photo quality.
    - Implementing this requirement may lead to the analysis performance of the AI being negatively affected due to factors such as light, shadow and weather conditions.
    - Implementing this requirement may create user dissatisfaction due to the risk of not providing sufficient explanation or feedback to the user if photos uploaded to the system cannot be processed due to system errors such as AI service interruption or timeout.
    - Implementing this requirement may cause AI to make incorrect plans due to the risk of scaling errors in images taken from different camera angles or perspectives.

- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend, Backend and AI teams are responsible for the implementation of this requirement.

Requirement REQ-708

- **Requirement ID:** REQ-708
- **Title:** Parking Space Type and Object Placement
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The provider should be able to drag and drop parking spaces (standard, EV, disabled, reserved, etc.) into the plan, and also select from the icon library items such as security booths, trees, cameras, and fire extinguishers to include them in the plan. If the floor plan is multi-story, floors should be added.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is impacted by Parking Providers, UI/UX Design Teams, Frontend Teams, Backend Teams.
- **Acceptance Criteria:**
    - The user should be able to drag the object from the icon panel and drop it onto the plan.
    - Different colors and icons should be assigned for each parking space type and object.
    - In multi-storey plans, the user should be able to add and delete floors.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-436 «deriveReqt» (No-Code Drag-and-Drop Visual Parking Design Editor)
      REQ-102 «deriveReqt» (Defining and adding parking areas and spots)
      REQ-110 «deriveReqt» (Parking Area Plan Creation and AI-Assisted Optimization)
      REQ-125 «deriveReqt» (Visual Customization of Parking Area and Adding Components)
      REQ-111 «trace Reqt» (Defining Multi-Storey Parking Structure)
      REQ-112 «trace Reqt» (Defining EV Charging Stations and Capacity Determination in Parking Areas)
      REQ-418 «trace Reqt» (Indoor Area Mapping and Floor Plan Integration)
      «satisfy» Parking Area Creation Module
      «verify» Parking area creation test case
- **Restrictions:** This requirement should be implemented to place only non-collision locations to prevent multiple objects from being placed at the same coordinates.
- **Assumptions:** For this requirement to apply, it is assumed that the icon library contains up-to-date and complete icons.
- **Risks:**
    - Implementing this requirement may lead users to make incorrect placements due to the risk of clutter in the user interface due to icon density or object diversity.
    - Implementing this requirement may negatively impact map readability and standardization due to the risk of inconsistent color and icon assignments.

    - Implementing this requirement may cause usability issues during drag-and-drop operations due to browser incompatibilities or screen size differences.
    - Implementing this requirement may lead to loss of functionality if the icon library fails to meet user needs due to the risk of it being incomplete or outdated.
    - Implementing this requirement may cause the plan to not correspond to physical reality and lead to implementation problems due to the risk of incorrect placement of objects out of scale.
    - Implementing this requirement may lead to freezing or slowdowns when editing floor plans containing large numbers of objects due to the risk of insufficient system performance.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-709

- **Requirement ID:** REQ-709
- **Title:** Artificial Intelligence-Powered Automatic Layout Suggestion
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The provider should be able to receive automatic layout recommendations from AI based on traffic flow and usage data, save draft versions of the current layout, and compare them with past versions. The user should be able to directly implement the suggested layout or make manual changes. Furthermore, the system should briefly explain the reasons for the recommendations (e.g., increased efficiency, traffic relief, etc.) to the user.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or impacted by Parking Space Providers, AI/AI Teams, UI/UX Design Teams, Frontend Teams, Backend Teams.
- **Acceptance Criteria:**
    - When the "Get Suggestion" button is clicked, the system should generate an artificial intelligence suggestion within 30 seconds and display it on the drawing.
    - The user should be able to save the recommendation and switch between different versions.
    - The system must store the creation date and notes for each version.
    - The user should be able to preview the suggested layout before applying it and cancel the suggestion if necessary.
    - The system should provide the rationale for the AI recommendation to the user in the form of a short explanation.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-102 «deriveReqt» (Defining and adding Parking areas and Parking spots)
    - REQ-110 «deriveReqt» (Parking Area Plan Creation and Artificial Intelligence-Assisted Optimization)
    - REQ-125 «deriveReqt» (Visual Customization of Parking Area and Adding Components)

    - REQ-436 «trace Reqt» (No-Code Drag-and-Drop Visual Parking Lot Design Editor)
    - REQ-111 «trace Reqt» (Multi-Storey Parking Structure Identification)
    - REQ-112 «trace Reqt» (EV Charging Station Identification and Capacity Determination in Parking Areas)
    - REQ-418 «trace Reqt» (Indoor Mapping and Floor Plan Integration)
    - «satisfy» Parking Space Creation Module
    - «verify» Parking area creation test scenario
- **Restrictions:**
    - This requirement must be implemented in a way that the recommendation engine only generates answers within this time limit due to the maximum processing time of 30 seconds.
    - This requirement should be implemented in such a way that only a certain number of recommendation versions can be stored due to the limited capacity of the system to maintain version history.
- **Assumptions:**
    - For this requirement to apply, it is assumed that the user has granted access to historical usage data.
    - For this requirement to be implemented, it is assumed that sufficient resources are available in the system to run the recommendation engine.
- **Risks:**
    - Implementing this requirement could lead to inaccurate placement recommendations due to the risk of AI analyzing incomplete or inaccurate parking/usage data.
    - Implementing this requirement may lead to a decrease in confidence in the reliability of the system due to the risk that automatic recommendations may not meet user needs.
    - Implementing this requirement can create high storage costs and performance loss in the system due to the risk of storing a large number of draft versions.
    - Implementing this requirement can present the user with a complex comparison process due to the risk of conflict between the automatic suggestion and the existing layout.
    - Implementing this requirement may lead to the proposal not being implemented due to the risk that the design proposed by the AI does not meet the technical, physical or legal requirements.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend, Backend and AI teams are responsible for the implementation of this requirement.

Requirement REQ-710

- **Requirement ID:** REQ-710
- **Title:** Accepting the Privacy Policy and Terms of Service During the Registration Process

- **Category:** Functional, Legal and Regulatory
- **Description:** The system should require drivers to read and agree to the privacy policy and terms of service during registration. This consent should be obtained through separate checkboxes. Mechanisms such as swipe-up or read verification should be integrated to prevent users from consenting without reading the texts. The system should record the consent with a timestamp. User accounts should not be allowed to be created without consent.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement was requested or is impacted by Drivers, Mobile Development Teams (UI/UX), Backend Teams, Data Security Team, Security Policy Makers.
- **Acceptance Criteria:**
    - The user should be able to see both the privacy policy and terms of service in full text on the registration screen or access them via a link; the checkbox should not activate until the texts are read (e.g., scrolling is complete).
    - The registration process cannot be completed without checking the separate checkboxes indicating that you have read and accepted both texts.
    - The user's consent must be securely stored in the system database along with the timestamp and user ID.
    - Consent storage must comply with legal requirements (e.g., KVKK/GDPR) and confirmation of consent must be sent to the user via email/SMS.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-428 «trace Reqt» (User Consent and Preference Management Regarding the Use of Personal Data)
    - REQ-439 «trace Reqt» (Secure Credential Storage and Secure Session Management)
    - REQ-441 «trace Reqt» (KVKK & GDPR Compliant Personal Data Processing and Storage)
    - REQ-442 «trace Reqt» (Transparency in Personal Data Use and Third Party Sharing)
    - REQ-617 «deriveReqt» (User Consent Preferences for Personal Data Use)
    - REQ-402 «trace Reqt» (Optional Registration and Login with Third Party Authentication)
    - «satisfy» Approval Recording & Versioning Module
    - «allocate» User registration interface component
    - «verify» Record confirmation test scenario
- **Restrictions:** This requirement should be limited to the languages in which the privacy policy and terms of service are available; these must be translated to provide multilingual support.
- **Assumptions:** For this requirement to apply, it is assumed that the user can read the privacy policy and terms of service in a language they understand and has the legal capacity.
- **Risks:**

    - The implementation of this requirement may become obsolete due to the risk of not updating the privacy policy and terms of service texts in a timely manner, which may lead to the need for an automatic update mechanism.
    - Implementing this requirement could lead to legal disputes because of the risk of users checking the box without reading it, so requiring “read” verification may be necessary.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-711

- **Requirement ID:** REQ-711
- **Title:** Viewing Selected Parking Details
- **Category:** This requirement belongs to the functional requirement and UI&UX requirement categories.
- **Description:** The system should provide a user-friendly interface that displays detailed and up-to-date information about the selected parking space. Each parking space should be presented in the system in a list or map view, and users should be directed to a separate information screen by clicking the "view details" button on the relevant parking space card or icon. This screen should include the following information:
    - General Information: name, full address, type (closed/open), total capacity, current occupancy/vacancy rate, fee information (hourly, daily, dynamic), working hours, estimated time of arrival (ETA), estimated waiting time
    - Accessibility and Equipment Features: disabled parking area information, valet service, security camera, ceiling height, floor slope, weather conditions
    - Electric Vehicle Information: EV charging station availability, charging type [AC/DC], power capacity [kW], number of stations
    - Environmental and Current Status Information: surrounding facilities, temporary status information [disabled, reserved, problematic], manual occupancy status update
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or impacted by Drivers, Mobile Development Teams (UI/UX), Backend Teams, Frontend Teams.
- **Acceptance Criteria:**
    - Each parking card or listing must include a "view details" button and the button must be clickable.
    - When the driver clicks on this button, he/she should be redirected to a new page containing only the details of the relevant parking space within a maximum of 2 seconds.
    - The information displayed should include the parking area name, address, total capacity, occupancy/emptiness ratio, pricing information, and hours; data should be retrieved from the database in real time.

    - The display must comply with accessibility standards (e.g., WCAG) and use responsive design on mobile devices.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-009 «deriveReqt» (Estimated Time of Arrival Calculation)
    - REQ-010 «deriveReqt» (Estimated Waiting Time Display)
    - REQ-030 «deriveReqt» (Electric Vehicle Charging Station Status)
    - REQ-031 «deriveReqt» (EV Charging Type and Power Information)
    - REQ-040 «deriveReqt» (Display of Facilities Around the Parking Point)
    - REQ-102 «deriveReqt» (Parking Area and Parking Point Definition)
    - REQ-112 «deriveReqt» (EV Charging Station Identification and Capacity Determination)
    - REQ-211 «deriveReqt» (Marking a Parking Area as Problematic)
    - REQ-216 «deriveReqt» (Manual Update of Status)
    - REQ-228 «deriveReqt» (Disabled/Temporary Parking Area Marking)
    - REQ-003 «trace Reqt» (Real-Time Parking Location Display)
    - REQ-022 «trace Reqt» (Seeing Available Parking Spaces on the Map)
    - «satisfy» Parking Database Module
    - «allocate» UI Interface Component
    - «verify» Detail display test case
- **Restrictions:** This requirement should only be applied to parking spaces defined and active in the system; data freshness depends on IoT integration.
- **Assumptions:** To implement this requirement, it is assumed that parking area information is current and complete in the system database and that users' internet connection is stable.
- **Risks:**
    - Implementing this requirement could mislead drivers due to the risk of parking area information being outdated, which requires real-time data synchronization.
    - Implementing this requirement can negatively impact the user experience due to the risk of errors or slowness in the user interface, so performance testing is mandatory.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-712

- **Requirement ID:** REQ-712
- **Title:** Automatic Notification of Parking Control Officers About Vehicle Arrival
- **Category:** This requirement belongs to the functional requirement and technical requirement categories.

- **Description:** The system should automatically notify the parking control officer responsible for the parking area when a vehicle enters a parking space. This notification should be detected in real time by the system and communicated to the parking control panel or to the officers via mobile notification. Detection should be achieved through entrance sensors or camera systems; manual verification should be optionally supported.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or impacted by Parking Providers, IoT Team / IoT Integration Team, Backend Teams, Mobile Development Teams (UI/UX), Operations and System Management, QA (Quality Assurance) Team, Other Service Providers, Data Security Team, Security Policy Makers.
- **Acceptance Criteria:**
    - Vehicle arrival must be detected with high accuracy via entrance sensors or camera systems.
    - When an entry is detected, a notification must be sent to the control officer via the system, with p95 ≤ 3 seconds.
    - The notification must include the vehicle license plate (masked in accordance with KVKK/GDPR requirements) and entry time.
    - An event log should be kept in the notification system, access records should be kept and should be auditable when necessary.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-403 «deriveReqt» (Collecting and Managing Real-Time Occupancy Data from IoT Devices)
    - REQ-415 «deriveReqt» (Multi-Channel Notification Support)
    - REQ-416 «deriveReqt» (Unified In-App Notification System for All User Types)
    - REQ-424 «deriveReqt» (Parking Space Status Control with IoT Integration)
    - REQ-632 «deriveReqt» (Real-Time Occupancy Data Update)
    - REQ-208 «trace Reqt» (Parking Officer's License Plate Scanning Feature)
    - REQ-223 «trace Reqt» (Manual Verification of Vehicle Entry/Exit)
    - REQ-434 «trace Reqt» (IoT Device Error Status Management)
    - «satisfy» Notification Module
    - «allocate» Parking Control Panel Component
    - «verify» Input detection test case
- **Restrictions:** This requirement should only be implemented in system-integrated parking areas and scenarios with active officer identification; network connectivity is mandatory.

  Assumptions: This requirement assumes that entrance sensors or camera systems are continuously and correctly functioning and that officers' mobile devices are ready to receive notifications.
- **Risks:**
    - Implementing this requirement may result in a delay or failure to send notification at all due to the risk of sensor or network failure, in which case a backup manual mode must be provided.

    - Implementing this requirement may lead to unnecessary intervention by the controller due to the risk of false alarms; therefore, verification algorithms should be optimized.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-713

- **Requirement ID:** REQ-713
- **Title:** Detecting Critical Post-Parking Events and Notifying the Driver
- **Category:** This requirement belongs to the functional requirement and technical requirement categories.
- **Description:** The system must detect critical events (e.g., collisions, attempted theft, unauthorized entry, emergency scenarios) after parking and immediately notify the driver. Critical events can be manually reported by parking attendants and/or detected through integrated IoT sensors and surveillance cameras. The notification can be sent via mobile app, SMS, or email and should include basic information about the event type, location, and time.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or impacted by Drivers, Parking Space Providers, IoT Team / IoT Integration Team, AI / Artificial Intelligence Teams, Backend Teams, Mobile Development Teams (UI/UX), Operations and System Management, QA (Quality Assurance) Team, Other Service Providers, Data Security Team, Security Policy Makers.
- **Acceptance Criteria:**
    - The system must be able to accurately relay to the driver the events manually reported by the officers.
    - Events detected by IoT sensors and cameras (if any) must be detected and reported with high accuracy.
    - Personal data included in the notifications (license plate, facial image, etc.) must be transmitted in a masked manner in compliance with KVKK/GDPR.
    - The notification must include the type of incident, parking lot ID, and time.
    - When using AI-based detection, validation mechanisms (e.g., second sensor confirmation) should be implemented to reduce false positives.
    - Event reporting and detection must be reported to the driver with an SLA target of p95 ≤ 5 seconds.
    - When there is a conflict between officer reporting and AI detection (if any), the officer report should take precedence.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-219 «deriveReqt» (Overriding Automatic Systems in Emergency Situations)

    - REQ-221 «deriveReqt» (Direct Communication with Emergency Services)
    - REQ-222 «deriveReqt» (Broadcast Emergency Message to All Users)
    - REQ-424 «deriveReqt» (Parking Space Status Control with IoT Integration)
    - REQ-416 «deriveReqt» (Unified In-App Notification System for All User Types)
    - REQ-426 «deriveReqt» (Emergency Notifications)
    - REQ-113 «trace Reqt» (Suspicious Activity Alert System)
    - REQ-203 «trace Reqt» (Parking Officer Sends Message to Park Owner and Driver)
    - REQ-210 «trace Reqt» (Notification to Parking Attendant in Case of Reservation Exceeding Time and Violation of Rules)
    - REQ-220 «trace Reqt» (Receiving and Acknowledging Security Alerts)
    - REQ-230 «trace Reqt» (Monitoring Environmental Risks and Taking Precautions in Park Areas)
    - REQ-415 «trace Reqt» (Multi-Channel Notification Support)
    - REQ-434 «trace Reqt» (IoT Device Error Status Management)
    - REQ-502 «trace Reqt» (Secure Data Sharing and Receiving with Official Institutions)
    - REQ-506 «trace Reqt» (Weather API Integration)
    - REQ-209 «trace Reqt» (Parking Officer's Ability to Check Instant Parking Areas)
    - REQ-211 «trace Reqt» (Parking Officer Marks Parking Area as Problematic)
    - REQ-214 «trace Reqt» (Parking Officer's Note-Taking and Information Sharing to the Next Officer)
    - «satisfy» Notification Module
    - «satisfy» IoT Sensor Module
    - «allocate» Security Camera Component
    - «verify» Event detection test case
- **Restrictions:**
    - This requirement is applicable only if the manual reporting feature is staffed at the parking areas.
    - This requirement should be implemented under the condition that AI-based detection is only applicable in parking areas integrated with IoT sensors and cameras.
- **Assumptions:**
    - For this requirement to apply, it is assumed that officers can report incidents quickly and accurately.
    - To implement this requirement, it is assumed that IoT sensors and surveillance systems (when used) are constantly active and open to data communication.
- **Risks:**
    - Implementing this requirement may lead to delays in reporting due to the risk of officers missing incidents or reporting them late.
    - Implementing this requirement could undermine driver confidence in the system if AI-based detection (when used) produces false alarms.
- **Status:** The status of this requirement is draft.

- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-714

- **Requirement ID:** REQ-714
- **Title:** Verification Checking of Location Conflicts
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** When registering a new parking space, the system must check whether the specified location overlaps with the locations of other parking spaces already registered in the system. This verification should be done through a coordinate-based space overlap check. If such an overlap is detected, the user should be warned and the registration process should be stopped.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or affected by Parking Space Providers, Backend Teams, Frontend Teams, Operations and System Management, QA (Quality Assurance) Team.
- **Acceptance Criteria:**
    - The coordinates of the newly recorded area must be compared with all active parking areas in the system via the WGS84 coordinate system.
    - If a conflict is detected, the user should be shown a warning saying “Location overlaps existing area” and the action should be canceled.
    - The collision check should be completed within p95 ≤ 2 seconds and there should be no performance loss on high data sets.
    - The system should log the conflict warning and retain historical control records for auditing.
    - A tolerance range for possible measurement errors (GPS drift, etc.) in location data accuracy should be defined and false positive/negative results should be minimized.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-102 «deriveReqt» (Defining and adding Parking areas and Parking spots)
    - REQ-110 «deriveReqt» (Parking Area Layout Creation and Artificial Intelligence-Assisted Optimization)
    - REQ-418 «trace Reqt» (Indoor Mapping and Floor Plan Integration)
    - REQ-707 «trace Reqt» (Artificial Intelligence-Assisted Park Layout Creation with Photography)
    - REQ-709 «trace Reqt» (Automatic Layout Suggestion)
    - REQ-722 «trace Reqt» (Defining a Parking Area by Setting Boundaries on the Map)
    - «satisfy» Location Verification Module
    - «allocate» Map and Coordinate Control Component

    - «verify» Location conflict test case
- **Restrictions:** This requirement should be applied provided that location data is processed only via the WGS84 coordinate system.
- **Assumptions:** For this requirement to apply, it is assumed that the location data for all registered parking spaces is accurate and up-to-date.
- **Risks:**
    - Implementing this requirement may result in valid records being mistakenly rejected by the system due to the risk of inaccurate location data.
    - Implementing this requirement may lead to system slowdowns on large datasets due to the risk of performance issues in the collision algorithm.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-715

- **Requirement ID:** REQ-715
- **Title:** Ability to Search and Add Officers Among Existing Users
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The parking provider should be able to search among users registered in the system for their parking spaces and assign appropriate individuals as attendants. The search should be based on first name, last name, or phone number, and the users found should be listed. Once the appropriate user is selected, they should be assigned as attendants and added to the "Assigned Attendants" list. The appointment should apply only to the provider's own parking spaces, and the user should be notified of the appointment.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or impacted by Parking Providers, Backend Teams, Frontend Teams, Mobile Development Teams (UI/UX), QA (Quality Assurance) Team, Operations and System Management.
- **Acceptance Criteria:**
    - The provider should be able to search for attendants only for their own parking areas, and the search results should be displayed in real time.
    - The search should be made by name-surname or phone number; results should be filtered when at least 3 characters are entered.
    - If the user does not exist in the system, a "User not found" warning should be displayed and the assignment should not be possible.
    - Once the assignment is successful, the officer should be added to the “Assigned Officers” tab and an email or in-app notification should be sent to the user (officer).
    - User roles and permissions must be updated after assignment; a double-approval mechanism must be implemented to prevent unauthorized access.

    - The appointment process and notification submissions must be logged by the system and kept open to audit.
    - User contact information (phone/email) must be protected under KVKK/GDPR and used for assignment purposes only.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-101 «deriveReqt» (Parking Owner Mobile Registration and Login)
    - REQ-115 «trace Reqt» (Special Authority Assignment for Parking Lot Controller)
    - REQ-408 «trace Reqt» (Configurable User Roles and Authorization)
    - REQ-231 «trace Reqt» (Parking Officer Login and Daily Control Panel Access)
    - «satisfy» User Database Module
    - «allocate» User Search and Assignment Component
    - «verify» Officer assignment test scenario
- **Restrictions:** This requirement should be implemented with the condition that only users with an active record in the system and who have not previously been assigned as officers can be assigned; the user role must be updated after assignment.
- **Assumptions:** It is assumed that the persons who will be responsible for the implementation of this requirement have already registered as users in the system and their contact information is correct.
- **Risks:** Implementing this requirement may lead to operational errors (e.g., unauthorized access) due to the risk of assigning tasks to the wrong user; therefore, a double-approval mechanism should be added.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-716

- **Requirement ID:** REQ-716
- **Title:** Reservation Cancellation Restrictions and Justification Collection Process
- **Category:** This requirement belongs to the functional requirement and UI&UX requirement categories.
- **Note:** Drivers can only cancel their reservations within 10 minutes of their reservation being created. After this period, the system will not allow cancellations. When the driver initiates the cancellation, the system should ask for a reason for the cancellation. Reasons should be presented in general terms; if the driver cannot find their situation in these options, a text box should be provided for personal explanations. Once the cancellation is confirmed, the parking space becomes available again. However, a specific waiting period must be defined before the same driver can make a new reservation.
- **Priority:** This requirement has been set to high priority.

- **Stakeholders:** This requirement is requested or impacted by Drivers, Parking Space Providers, Backend Teams, Frontend Teams, Mobile Development Teams (UI/UX), QA (Quality Assurance) Team, Operations and System Administration, Data Security Team.
- **Acceptance Criteria:**
    - The reservation can only be cancelled within the first 10 minutes after it is created.
    - When the driver clicks cancel, the system should ask "What is the reason for your cancellation?"
    - The system should present a list of common cancellation reasons and also provide a description box in the “Other” option.
    - After successful cancellation, the relevant parking space should become available again.
    - The driver who cancels cannot make a new reservation for 15 minutes (with a variable system parameter).
    - Cancellation operations must be completed within p95 ≤ 2 seconds, and the result must be reflected to the user instantly.
    - All cancellations must be logged, including the reason for cancellation and a timestamp, and stored in accordance with KVKK/GDPR requirements.
    - Reasons for cancellation must be stored in an anonymized form for statistical reporting purposes.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-704 «deriveReqt» (Automatic Cancellation of Reservation in Case of Late Arrival)
    - REQ-016 «deriveReqt» (Advance Parking Reservation)
    - REQ-029 «deriveReqt» (Real Time Notification When Reservation Is Invalid)
    - REQ-010 «deriveReqt» (Estimated Waiting Time Display)
    - REQ-117 «deriveReqt» (Definition of Reservation Period, Discount and Cancellation Rules)
    - REQ-210 «deriveReqt» (Notification of Reservation Excess and Violation of Rules)
    - REQ-215 «deriveReqt» (Parking Attendant Override of Reservation)
    - REQ-225 «deriveReqt» (Authorized Officials Ability to Edit and Cancel Reservations)
    - REQ-201 «deriveReqt» (Reservation Notification to Parking Attendant)
    - REQ-209 «trace Reqt» (Control of Instant Parking Areas)
    - REQ-206 «trace Reqt» (Parking Officer Can See Made Reservations)
    - REQ-202 «trace Reqt» (Parking Officer's Ability to See Reservation Information)
    - REQ-203 «trace Reqt» (Parking Officer Sends Message to Park Owner and Driver)
    - REQ-019 «trace Reqt» (Reservation Cancellation)
    - REQ-212 «trace Reqt» (Reservation and Entry Tracking via Mobile Panel)
    - REQ-213 «trace Reqt» (Parking Officer Ticket)

    - REQ-231 «trace Reqt» (Parking Officer Login and Daily Control Panel Access)
    - «satisfy» Reservation Module
    - «satisfy» Timer Module
    - «satisfy» Notification Module
    - «allocate» UI Cancel UI Component
    - «verify» Cancellation process test scenario
- **Restrictions:**
    - This requirement must be implemented in a way that does not rely on client device time, as booking creation and cancellation tracking must be performed based solely on system time.
    - This requirement must be implemented in a way that prevents unauthorized intervention due to the restriction that the waiting time and cancellation time can only be changed by authorized users from the system configuration.
- **Assumptions:** For this requirement to apply, it is assumed that the driver has an internet connection on their mobile device and the current version of the application is installed.
- **Risks:**
    - Implementing this requirement could negatively impact user satisfaction due to the risk of the driver attempting to cancel after the cancellation period has passed.
    - Implementing this requirement could lead to system vulnerabilities due to the risk of mismanaging the waiting period.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-717

- **Requirement ID:** REQ-717
- **Title:** Reservation Verification and Area Matching with Automatic License Plate Recognition
- **Category:** This requirement belongs to the functional requirement and technical requirement categories.
- **Description:** The system should scan the license plate of a vehicle entering a parking space using IoT sensors or camera-based automatic license plate recognition (ANPR) technology and compare it to the existing reservation database. If the license plate and reservation match and the vehicle parks in the correct space, the system should immediately mark the space as "occupied," record the entry time, and send a confirmation notification to the driver. If the vehicle parks in the wrong space, the system should display a warning to the driver and provide guidance options such as "Change Spot."
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or impacted by Drivers, Parking Space Providers, IoT Team / IoT Integration Team, AI / Artificial Intelligence Teams, Backend

Teams, Mobile Development Teams (UI/UX), Operations and System Management, QA (Quality Assurance) Team, Data Security Team, Security Policy Makers, Other Service Providers.

- **Acceptance Criteria:**
    - The system must detect the vehicle's license plate with p95 ≥ 95% recognition accuracy at the time of entry.
    - If the license plate matches the reservation in the system, the parking space should be marked as “occupied” and the entry time should be recorded.
    - If the pairing is successful, a verification notification should be sent to the driver within p95 ≤ 3 seconds.
    - If the vehicle parks in the wrong space, the driver should be presented with a warning notification and the option to “Change Spot”.
    - All license plate scanning and matching operations must be logged, including event time and result information, and stored in accordance with KVKK/GDPR requirements.
    - An AI-based verification mechanism should be activated in cases of false recognition or incorrect matching.
    - Reservation and license plate matching algorithms must have passed performance tests (p95 ≤ 2 s processing time in high density scenarios).
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-002 «deriveReqt» (Driver Ability to Manage Profile and Vehicle Information)
    - REQ-016 «deriveReqt» (Advance Parking Reservation)
    - REQ-017 «deriveReqt» (Entering Start and End Information for Reservations)
    - REQ-102 «deriveReqt» (Parking Area and Parking Point Definition)
    - REQ-403 «deriveReqt» (Collecting and Managing Real-Time Occupancy Data from IoT Devices)
    - REQ-416 «deriveReqt» (Unified In-App Notification System for All User Types)
    - REQ-424 «deriveReqt» (Parking Space Status Control with IoT Integration)
    - REQ-501 «deriveReqt» (Receiving and Processing Real-time Parking Status Data from External IoT Sensors)
    - REQ-632 «deriveReqt» (Real-Time Occupancy Data Update)
    - REQ-415 «trace Reqt» (Multi-Channel Notification Support)
    - «satisfy» License Plate Recognition Module
    - «allocate» Image Processing Component
    - «verify» License plate matching test scenario
- **Restrictions:**
    - This requirement only applies to drivers with active reservations.
    - This requirement does not apply in areas where the system does not have an active camera or sensor infrastructure.
    - This requirement must be implemented under the condition that the recognition module can only be used in systems compatible with the supported camera and sensor protocols.
- **Assumptions:**

    - For this requirement to be implemented, it is assumed that the recognition infrastructure operates uninterruptedly and correctly at all entry and exit points.
    - For this requirement to apply, it is assumed that the reservation data is up-to-date and kept in a format that can be matched with the license plate.
- **Risks:**
    - Implementing this requirement may cause the system to consider the reservation invalid due to the risk of license plate recognition failure.
    - Implementing this requirement may negatively impact the user experience due to notification delays or the risk of incorrect parking spot detection.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend, Backend and AI teams are responsible for the implementation of this requirement.

Requirement REQ-718

- **Requirement ID:** REQ-718
- **Title:** Notification Restriction and Warning Message When Location Permission is Denied
- **Category:** This requirement belongs to the quality requirement and UI&UX requirement categories.
- **Description:** If a mobile app user denies location permission, the system should display a clear and informative warning message within the app. This message should clearly state which features (e.g., nearest parking suggestion, directions, real-time distance calculation, etc.) will not be available if location data is disabled. The user should also be provided with instructions to re-enable location permission in the app settings. The warning message should be designed to minimize user experience impact, yet be noticeable to the user.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or impacted by Drivers, Mobile Development Teams (UI/UX), Backend Teams, QA (Quality Assurance) Team.
- **Acceptance Criteria:**
    - The system shall display a warning message within p95 ≤ 3 seconds when location access is denied.
    - The message must list at least 3 disabled features.
    - The user should be instructed to re-enable permissions from the settings menu.
    - The warning message must be placed in the application interface in a clearly visible manner that will not obstruct the user.
    - The language of the message must comply with WCAG accessibility standards and understandability criteria.
    - All warning message displays should be logged by the system, and the situation and time when they were triggered should be open to audit.
    - To reduce the risk of misdirection, the content of the message should be periodically checked and updated according to mobile operating system versions.

- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-400 «deriveReqt» (Real-Time Location Data Access)
    - REQ-416 «deriveReqt» (Unified In-App Notification System for All User Types)
    - REQ-445 «deriveReqt» (Location Access Permission Management)
    - REQ-008 «trace Reqt» (Navigation Guidance)
    - REQ-009 «trace Reqt» (Estimated Time of Arrival Calculation)
    - REQ-022 «trace Reqt» (Seeing Available Parking Spaces on the Map)
    - REQ-415 «trace Reqt» (Multi-Channel Notification Support)
    - «satisfy» Mobile Location Service Module
    - «allocate» Alert Message Component
    - «verify» Location Permission Denial Test Scenario
- **Restrictions:** This requirement should only be implemented when location access is completely turned off at the mobile operating system level.
- **Assumptions:** For this requirement to be implemented, it is assumed that the application can instantly query the status of the location access permission from the device.
- **Risks:**
    - Implementing this requirement may reduce user satisfaction because of the risk of the user experiencing the app negatively without seeing the warning.
    - Implementing this requirement may lead to incorrect user settings due to the risk of incorrect redirect messages.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-719

- **Requirement ID:** REQ-719
- **Title:** Reservation Blocking and Redirection with Incomplete Profile Information
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** Before initiating a parking reservation, the system should verify that the driver's profile is fully filled with the required information. If any of the required fields are missing (for example: first name, last name, phone number, registered vehicle, payment method), the system should not allow the reservation. Instead, the user should be shown a list of the missing fields and directed directly to the relevant editing screen.
- **Priority:** This requirement has been set to high priority.
- **Stakeholders:** This requirement is requested or impacted by Drivers, Backend Teams, Frontend Teams, Mobile Development Teams (UI/UX), QA (Quality Assurance) Team.
- **Acceptance Criteria:**
    - The system must check the profile before clicking the reservation button.

    - If there is missing information, the user should be shown a warning saying "Your profile is incomplete. Please complete the following fields."
    - Missing fields should be listed, and each should include a link to the relevant edit screen.
    - Once the profile is complete, the user should be able to continue with the booking process.
    - Profile data must be protected under KVKK/GDPR and processed only for verification purposes.
    - The missing area check p95 must be completed within ≤ 1 second and should not cause processing delay.
    - All routing and missing area detection operations should be logged by the system and open to auditing when necessary.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-001 «deriveReqt» (Driver Registration and Login to Mobile Application)
    - REQ-002 «deriveReqt» (Driver Ability to Manage Profile and Vehicle Information)
    - REQ-016 «trace Reqt» (Advance Parking Space Reservation)
    - «satisfy» Reservation Availability Check Service
    - «allocate» Profile Management Module
    - «allocate» Edit Screen Component
    - «verify» Profile control test case
- **Restrictions:** This requirement should only be applied to information that has been defined by the system as a “required field.”
- **Assumptions:** To implement this requirement, it is assumed that the system can perform profile verification synchronously during the reservation initiation process.
- **Risks:**
    - Implementing this requirement may negatively impact user satisfaction due to the risk of defining too many mandatory fields.
    - Implementing this requirement may lead to the user being taken to the wrong edit screen due to the risk of misdirection.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-720

- **Requirement ID:** REQ-720
- **Title:** Viewing Parking Fee Information, Estimated Costs, and Discounts
- **Category:** This requirement belongs to the functional requirement and UI&UX requirement categories.
- **Description:** Before selecting a parking space, the system should allow drivers to review the hourly and daily rates, the estimated total cost, and any applicable discounts for that space. The estimated cost should be calculated based on the driver's selected default duration or default ETA and clearly presented to the user. The system should

clearly indicate any temporary discount campaigns or subscription discounts on this screen.

- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or impacted by Drivers, Backend Teams, Frontend Teams, Mobile Development Teams (UI/UX), QA (Quality Assurance) Team, Operations and System Management.
- **Acceptance Criteria:**
    - Hourly and daily fee information must be visible on each parking area card.
    - Once the driver selects a time, the system should automatically calculate and display the estimated total fare.
    - If there is a temporary promotional discount on parking, this information must be clearly stated (for example: “20% Discount – Today Only”).
    - All price, discount and estimated cost information displayed should be logged, and past price changes should be open to audit.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-015 «deriveReqt» (Displaying Parking Fee and Availability Information)
    - REQ-014 «deriveReqt» (Discounted Parking with Subscription System)
    - REQ-013 «deriveReqt» (Payment via App)
    - REQ-009 «deriveReqt» (Estimated Time of Arrival Calculation)
    - REQ-102 «deriveReqt» (Parking Area and Parking Point Definition)
    - REQ-107 «deriveReqt» (Dynamic Pricing)
    - REQ-105 «deriveReqt» (Campaign Management)
    - REQ-117 «trace Reqt» (Definition of Reservation Period, Discount and Cancellation Rules)
    - REQ-022 «trace Reqt» (Seeing Available Parking Spaces on the Map)
    - «satisfy» Pricing Module
    - «satisfy» IoT Sensor Module
    - «allocate» Map Component
    - «verify» Fee calculation test scenario
- **Restrictions:** This requirement should be applied provided that campaign discounts are only valid for the period and conditions defined in the system.
- **Assumptions:** To implement this requirement, it is assumed that the system can retrieve up-to-date fee and discount data from the relevant parking provider.
- **Risks:**
    - Implementing this requirement could undermine user trust due to the risk of misleading price display.
    - Implementing this requirement may lead to surprise costs at the checkout screen due to the risk of incorrect calculation of the estimated time.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-721

- **Requirement ID:** REQ-721
- **Title:** Warning and Provider-Defined Penalty Application in Case of No-Show
- **Category:** This requirement belongs to the functional requirement category.
- **Description:** The system should ensure that if a driver fails to arrive on time for their reservation, a warning and a flat-rate penalty are applied if the behavior repeats. The first time, the driver is only shown a descriptive warning message. For subsequent similar violations, the penalty policy defined by the parking provider is implemented. Under this policy, the system automatically charges the driver a percentage of the reservation fee. The penalty rate can be defined through the parking provider's dashboard. When the driver attempts to make a new reservation, the previous violation and any applicable penalty should be clearly displayed.
- **Priority:** The priority of this requirement is set to medium.
- **Stakeholders:** This requirement is requested or impacted by Drivers, Parking Space Providers, Backend Teams, Frontend Teams, Mobile Development Teams (UI/UX), QA (Quality Assurance) Team, Operations and System Management, Data Security Team, Security Policy Makers.
- **Acceptance Criteria:**
    - If the driver does not enter the parking area at the reservation time, the reservation will be automatically canceled.
    - For the first violation, the driver should be shown a warning notice and no penalty will be imposed.
    - For subsequent violations, the penalty rate set by the parking provider (e.g., 30%) should be automatically charged.
    - The penalty rate must be defined and changeable by the provider from the administration panel for each parking area.
    - When the driver tries to make a booking, the previous penalty message and the amount charged should be displayed, if any.
    - Fine collection and warning notifications must be logged by the system and kept open to audit.
    - Processing of user payment information must be KVKK/GDPR compliant, and penalty application must be triggered only by authorized roles.
    - To prevent incorrect/faulty penalty application, a double verification mechanism should be implemented before collection.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-016 «deriveReqt» (Advance Parking Reservation)
    - REQ-013 «deriveReqt» (Payment via App)
    - REQ-117 «deriveReqt» (Definition of Reservation Period, Discount and Cancellation Rules)
    - REQ-210 «deriveReqt» (Notification of Reservation Excess and Violation of Rules)

    - REQ-704 «deriveReqt» (Automatic Cancellation of Reservation in Case of Late Arrival)
    - REQ-215 «deriveReqt» (Parking Attendant Override of Reservation)
    - REQ-029 «deriveReqt» (Real Time Notification When Reservation Is Invalid)
    - REQ-019 «trace Reqt» (Reservation Cancellation)
    - REQ-213 «trace Reqt» (Parking Officer – Issuing a Ticket)
    - REQ-102 «trace Reqt» (Parking Area and Parking Point Identification)
    - REQ-225 «trace Reqt» (Authorized Officials Ability to Edit and Cancel Reservations)
    - REQ-201 «trace Reqt» (Reservation Notification)
    - «satisfy» Automatic Reservation Cancellation Service
    - «satisfy» Payment Module
    - «allocate» Penalty Policy Management Panel
    - «allocate» Alert Notification Component
    - «verify» Violation tracking test scenario
- **Restrictions:** This requirement should only be implemented in sensor or camera-enabled areas where entry into the parking area can be verified.
- **Assumptions:** For this requirement to apply, it is assumed that the user has a payment method registered in the system and the parking provider has defined its penalty policy.
- **Risks:**
    - Implementing this requirement may lead to user dissatisfaction due to the risk of incorrect application of the penalty.
    - Implementing this requirement may negatively impact user behavior due to the risk that the provider sets the penalty rate too high.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.

Requirement REQ-722

- **Requirement ID:** REQ-722
- **Title:** Defining a Parking Area by Determining the Boundary on the Map
- **Category:** This requirement belongs to the functional requirement and UI&UX requirement categories.
- **Description:** When adding a new parking space, the parking provider must be able to draw space boundaries using the interactive map interface on the system. The map must allow the provider to define the parking space as a polygon; corner points must be draggable. The provider can choose to mark the space center automatically using GPS location or manually enter an address to locate it on the map. The system must check the boundaries to ensure they do not overlap with existing registered spaces.
- **Priority:** This requirement has been set to high priority.

- **Stakeholders:** This requirement is requested or affected by Parking Providers, Backend Teams, Frontend Teams, Mobile Development Teams (UI/UX), QA (Quality Assurance) Team, Operations and System Management, Other Service Providers.
- **Acceptance Criteria:**
    - The map must support polygon drawing.
    - The provider must be able to draw an area with at least 3 corners and edit the corners with the mouse or touch.
    - The map interface should have zoom in/out and location fixing features.
    - After the area drawing is completed, the system should check for collisions (related to REQ-714).
    - If the user wishes, he/she should be able to automatically determine the center point using the current GPS location.
- **Dependencies:** This requirement is dependent on the completion of the following system components and requirements:
    - REQ-102 «deriveReqt» (Parking Area and Parking Point Definition)
    - REQ-110 «deriveReqt» (Parking Area Layout Creation and Artificial Intelligence-Assisted Optimization)
    - REQ-111 «deriveReqt» (Multi-Storey Parking Structure Identification)
    - REQ-714 «deriveReqt» (Location Conflict Validation Check)
    - REQ-022 «trace Reqt» (Seeing Available Parking Spaces on the Map)
    - REQ-418 «trace Reqt» (Indoor Mapping and Floor Plan Integration)
    - REQ-709 «trace Reqt» (Automatic Layout Suggestion)
    - «satisfy» Map and Location Service
    - «satisfy» GPS Service Module
    - «allocate» Parking Area Definition Interface
    - «allocate» Map Interface Component
    - «verify» Border drawing test case
- **Restrictions:** This requirement is only applicable on platforms with map service integration enabled (e.g. Google Maps, OpenStreetMap).
- **Assumptions:** For this requirement to be implemented, it is assumed that parking providers' devices support location services and have internet connectivity.
- **Risks:**
    - Implementing this requirement may lead to performance issues while drawing due to the risk of map API restrictions or quota limits.
    - Implementing this requirement may result in erroneous data recording due to the risk of incorrect boundary definition, which may lead to the occupation of areas that do not actually exist.
- **Status:** The status of this requirement is draft.
- **Responsible:** Frontend and Backend teams are responsible for the implementation of this requirement.