המכללה האקדמית להנדסה בראודה
בכרמיאל

Software Engineering Department
Braude college

Course 61999: Capstone Project in Software Engineering

Parking reservation at the college

Shady Mansour - ID: 211732946 - E-mail : shady.mansour@e.braude.ac.il
Marwa Hamoud - ID: 212102263 - E-mail : marwa.hamoud@e.braude.ac.il

Supervisor:
Dr . Zeev Frenkel

Table of Contents

Abstract

The goal of the project is to build a smartphone-based application that enables efficient and fair management of parking reservations. Unlike existing solutions focused on maximizing revenue, this application emphasizes user-oriented design, prioritizing fairness and efficiency. For this purpose, the system analyzes the dynamics of parking slot usage and previous user behavior in parking slot reservations. Based on estimations of these dynamics and users behaviour the system assigns priorities to reservations, optimizing the utilization of limited parking slots. The system combines automated tools with manual management to ensure flexibility and reliability.

1.Introduction

Parking has become a significant challenge in our college, where the limited number of available spaces cannot keep up with the high demand. For students and staff, this daily struggle often results in frustration, wasted time, and inefficiency. To address this issue, we propose a smart parking reservation system. Our system brings order, fairness, and convenience to the process of parking management. In addition, the app introduces a framework for responsible sharing. While users are discouraged from unauthorized reservation transfers, the system is designed to allow flexible and approved sharing options, ensuring that spaces are utilized efficiently while maintaining fairness. By combining intuitive design, real-time updates, and machine learning algorithms, this app not only ensures fairness but also optimizes the use of limited parking spaces.

Through this book, we aim to share our journey of creating this innovative system. We present a comprehensive guide to the design and development of a parking app specifically tailored to our college's needs. The app enables users to register, check the availability of parking spots, and reserve spaces in advance, all from the convenience of their smartphones. Before embarking on the planning and development phase, we explore various topics critical to our research work. This includes examining the parking apps currently in use, analyzing their advantages and disadvantages, and identifying the key features that can be enhanced or added to address the specific needs of our college community. By learning from existing solutions, we aim to design an app that overcomes their limitations while introducing innovative features tailored to our goals.

This introduction establishes the framework for developing an intelligent parking management system to address the challenges of limited parking resources. The system integrates a mobile application for user interaction, server-side infrastructure for real-time data processing, and advanced algorithms for decision-making. Key methods include machine learning techniques such as K-Means clustering to analyze and classify user behavior patterns and Q-Learning, a reinforcement learning approach, to optimize parking allocation and reservation prioritization dynamically. Core technologies include smartphone applications for seamless user interaction, servers for handling real-time data processing, and a robust database for storing historical data and user profiles. algorithmic frameworks for priority allocation enhance system efficiency. By leveraging these methods and technologies, the project aims to transform parking management into an automated, data-driven, and user-focused solution.

1.1. Book structure

1. Introduction
A brief introduction to the problem of parking management in educational institutions. Presenting the idea of developing a smart application to provide an efficient and fair solution.

2. Background
An overview of the current state of parking management, including challenges faced by users and systems.

3. System Design
Description of the system structure, including the app, server, and database. Introduction to concepts like user categorization (New, Good, and Banned Users).

4. Technological Implementation
An outline of the tools and methods used, such as integrating advanced algorithms, real-time data processing, and database management.

2. Literature Review

The problem of parking shortages and the need for efficient management of available parking spaces has become increasingly discussed in the technological and managerial literature. A variety of technological solutions have been developed in recent years to address the challenge, but there are still limitations that justify the development of customized solutions, such as the application proposed in this project.

2.1. Existing Tools and Technologies

Numerous tools and technologies have been developed to address parking reservation challenges, each with distinct features and limitations:

2.1.1 Mobile Applications for Parking Management

Pango and Cellopark : Cellopark Website are widely used in Israel for managing parking payments. Both applications primarily serve public parking spaces and enable payment by linking a user's account to their vehicle's license plate. However, these apps are designed for generic public use and do not support features for customized reservations, prioritization, or tailored user management for institutional or private parking facilities.

Parkopedia : Parkopedia Website is an international platform providing extensive data on parking availability, locations, and pricing. However, it lacks real-time updates and features for reservation management, such as allocating priority for specific user groups or advanced scheduling capabilities.

Enhancing this system for real-time reservation could involve integrating sensors or IoT technology to track occupancy and automatically allocate spaces to users based on their needs.

DPM+:(DPM+ Website) manages parking lots and workstations within organizations. One of its unique features is the ability to notify drivers if their vehicle is blocking another and facilitate communication between the parties. This improves overall efficiency in private or corporate parking scenarios but does not extend to broader, more complex user prioritization needs. Expanding such systems could include predictive analytics to anticipate high-demand times or identify repeat offenders.

2.1.2 Smart Parking Solutions with IoT Integration

ParKam: (ParKam Website) utilizes computer vision and AI to provide real-time parking availability updates. This solution is effective in large urban areas or shopping centers where managing vast parking spaces is critical. However, the high implementation cost makes it less suitable for smaller institutions or educational facilities.

2.1.3 Machine Learning in Parking Management

Machine learning (ML) has been extensively studied for optimizing parking systems, focusing on areas such as:

- Occupancy Predictions: ML models, such as linear regression and recurrent neural networks (RNN), analyze historical data to predict parking availability at specific times. These predictions help reduce wait times and congestion.
- User Classification: Clustering algorithms (e.g., K-Means, DBSCAN) classify users into categories such as frequent users, news or violators. This classification allows for tailored parking allocation.
- Optimization Algorithms: Decision trees, reinforcement learning, and genetic algorithms optimize resource allocation by prioritizing reservations based on predefined criteria such as time sensitivity, user type, or availability.
- To implement these methods, the system must collect data from user activity logs, and historical usage patterns. The data is then processed using algorithms to dynamically adjust allocations and reservations.

2.2. Limitations of Current Solutions

While existing tools offer partial solutions, they fail to address the unique challenges faced by academic institutions:

- Absence of Behavior-Based Management: Current systems do not reward responsible parking behavior or penalize rule violations, such as unauthorized space transfers or no-shows [1][3].
- Lack of User Prioritization: No differentiation between user types (e.g., students, faculty, visitors), leading to inefficiencies.
- High Infrastructure Costs: Most solutions require significant investments in hardware, such as sensors or cameras, which may not be feasible for smaller campuses.
- Limited Real-Time Accuracy: Many apps rely on outdated data, resulting in unreliable parking availability information.

2.3. The Role of AI in Smart Parking

Artificial intelligence (AI) technologies are gaining traction in parking management:

- Behavior Prediction Models: AI can analyze user behavior to predict peak parking times and optimize space allocation. These models are especially relevant for environments with consistent user patterns, such as academic campuses [8].
- Facial Recognition and License Plate Recognition: Used to monitor parking lot usage and ensure rule compliance. However, privacy concerns remain a significant barrier to adoption in educational settings [9].
- Machine Learning Integration: AI systems often leverage machine learning (ML) to process large datasets, identify patterns in parking usage, and improve predictions and decision-making over time, ensuring adaptive and efficient parking management.

3.Innovative Approaches in Academic Parking Systems

Recent studies highlight innovative solutions specifically designed for academic institutions to address the unique challenges of parking management. These approaches aim to balance fairness, efficiency, and user accountability through advanced technological and behavioral frameworks:

- ## Dynamic User Classification

  One of the key innovations is dynamic user classification, which categorizes users into groups such as "Good," "New," or "Banned" based on their parking behavior. This classification system fosters a sense of fairness and encourages responsible use of limited parking spaces by rewarding compliance and penalizing violations. Machine learning algorithms can be employed to continuously analyze user behavior patterns, ensuring classifications remain accurate and adaptable to changes over time.

- ## Reservation-Based Systems

  Allowing users to pre-book parking spaces has proven effective in reducing uncertainty and increasing overall efficiency. This feature is particularly valuable in academic environments, where high demand during peak hours creates significant challenges. By integrating real-time data on space availability with reservation functionality, institutions can allocate resources more effectively and reduce congestion.

- ## Incorporation of Incentives

  Incentive-based systems have gained traction as a method to promote punctuality and adherence to parking regulations. Examples include awarding priority access or discounted fees to users who consistently follow rules. Such incentives foster a culture of accountability and responsible behavior, which is essential in academic settings where fairness and equitable resource distribution are critical.

3.1 Unique Features of This System

What sets this parking management system apart is its emphasis on user behavior and accountability. The app integrates multiple innovative features to classify users based on their parking habits, ensuring equitable access and discouraging rule violations. The classification is as follows:

1. **Good Users**:

- Description: Users who consistently arrive on time, park correctly in their allocated spaces, and maintain a history of responsible parking behavior.
- Benefits:
  - Priority access to high-demand parking spots.
  - Possible rewards such as fee discounts or reserved spaces during peak times.
- Purpose: Encourage compliance and recognize users who contribute to an efficient parking environment.

2. **Banned Users**:

- Description: Users who violate parking rules repeatedly, such as failing to show up for reservations, occupying multiple spaces, or transferring reservations without permission.
- Penalties:
  - Temporary restrictions from making reservations.
  - A flag system (e.g., users are banned after accumulating three violations).
- Purpose: Enforce rules and deter misuse of parking resources.

3. Neutral Users:

Description: Users who recently registered and are in the observation phase while the system learns their behavior. They are prioritized above "Bad Users" but below "Good Users."

- Benefits**:**

-Neutral users are given access to general parking spots with limited prioritization, ensuring fair treatment without disrupting the prioritization structure.

-Eligibility for "Good User" status after one month of responsible behavior, based on system evaluations.

-Encourages responsible parking habits and adherence to rules during the observation period.

- **Purpose:**

- To create a balanced integration phase for new users, allowing the system to learn their behavior before granting higher priority.
- Ensures fairness by prioritizing neutral users above "Bad Users" but below "Good Users."

3.2 Relevance of the Proposed Solution

The proposed smart parking app directly addresses the limitations of existing solutions:

- User-Centric Design: Focused on creating a seamless and efficient parking experience by tailoring features to meet the needs of diverse user groups.
- Behavior-Based Management: Integrates machine learning algorithms to analyze parking behavior, enabling classification and management of users to promote accountability and fairness.
- Cost-Effective Implementation: Leverages existing infrastructure while incorporating advanced software capabilities, reducing reliance on costly hardware solutions.

3.3 Summary of Literature Review

This literature review highlights the gaps in current parking management solutions and underscores the need for a tailored approach in academic settings. By incorporating advanced technologies and user-focused features, the proposed solution aims to transform parking management into an efficient and fair system for all stakeholders.

4. System Achievements

The current version of the project includes major components that have already been implemented. These features lay the foundation for an efficient and fair smart parking system.

4.1 Implemented Components

4.1.1 Mobile Application

- **Real-Time Parking Space Availability**: Users can view available parking spots in real time, minimizing search time.
- **Reservation System**: The app allows users to reserve parking spots by selecting a specific date and time range.

4.1.2 Server-Side Infrastructure

- **Database Management**: A centralized and secure database is in place, storing user details, reservations, and violation records with real-time synchronization.
- **User Scoring System**: A dynamic system is used to rank users based on behavior—such as punctuality and rule compliance—to help determine parking priority.
- **Admin Dashboard**: Administrators can manage reservations, monitor user behavior, and generate statistical reports for system analysis and optimization.

These components form the backbone of the system and will continue to be refined based on feedback and further testing in the next stage.

4.2 Algorithm and System Description

The system is built around several key components, each designed to work together seamlessly to ensure efficient and user-friendly parking management.

Data Collection

- **Sources of Data**:
  - User interactions through the mobile app, such as booking, cancellation, or violation reporting.
  - Real-time occupancy data collected from manual inputs by administrators.
  - Historical data on usage patterns, reservation trends, and peak demand times.
  - Machine based data simulation to test the system
- **Data Storage**:
  - A centralized database stores all incoming data, ensuring secure and scalable data management for analysis and decision-making.
  - The data is structured for easy access and is periodically backed up for reliability.

User Scoring and Management

- **Scoring Criteria**:
    - Scores are calculated based on punctuality, adherence to parking rules (e.g., parking within assigned spaces), and reservation reliability.

Optimization of parking space usage

- **Purpose**:
    - Optimize parking space allocation.
    - Predict peak parking times to preemptively manage demand.
    - Classify user behavior to improve accountability and fairness.
- **Implementation Details**:
    - **Supervised Learning**: Uses labeled historical data to train the model on parking behavior patterns, such as peak demand times and compliance trends.
    - **Behavior Classification**: Implements clustering algorithms (e.g., K-Means) to group users based on behavior, allowing for better customization of user experiences.
    - **Real-Time Updates**: Continuously incorporates real-time data to refine predictions and dynamically adjust resource allocation, ensuring responsiveness to changing conditions.

4.3 Success Criteria

Functionality

Smooth operation of the app's core features, including real-time parking availability search, user classification, and reservation management.

Unit Testing: Test individual components like parking availability, user classification, and reservation systems to ensure they work independently.

Integration Testing: Combine the components and test their interaction, such as real-time updates of available spots after a reservation.Simulated Scenarios: Simulate user activities (e.g., multiple users reserving spots simultaneously, cancellations, and last-minute reservations) to identify potential system bottlenecks.

Stress Testing: Test the system under peak load conditions to ensure smooth operation during high user activity.

Efficiency

Demonstrated reduction in parking conflicts and underutilized spaces.

Testing Data: Use simulated parking data based on typical usage patterns, such as entry/exit times and peak hours. Generate synthetic data using known distributions (e.g., Gaussian for

arrival times). Create scenarios of overbooking and unused reservations to evaluate system performance.

Comparison Algorithm(s):
Compare against simple rule-based approaches (e.g., First-Come-First-Serve) to demonstrate improvements in parking allocation. Evaluate against baseline clustering models like static K-Means (without daily updates) to highlight the benefits of adaptive user classification. Use metrics like reservation success rate, parking spot utilization percentage, and user waiting times to compare performance.

Scalability

The system's scalability ensures it can adapt to diverse environments and grow with minimal adjustments to its infrastructure.

Explanation:
To achieve scalability, the app will employ:

1.Modular Architecture: Key components, such as parking spot management and user classification, are designed independently, allowing for seamless scaling of individual modules without impacting the entire system.

2.Flexible Database Schema: The database will be structured to easily support the addition of new parking locations, users, and features without requiring significant reconfiguration or downtime.

This approach ensures the system remains efficient and adaptable as demand increases or new requirements emerge.

4.3.1 Performance Evaluation Using Data Simulation:

To assess the system's efficiency, we will compare the smart algorithm, which utilizes machine learning techniques, with a simple algorithm that does not incorporate predictive mechanisms.

Methodology:

1. Generating Simulated Data:

- A probability-based model will be used to simulate realistic parking behavior.

- Data such as arrival times, cancellations, and tardiness rates will be generated based on known distributions (e.g., a normal distribution for arrival times).

- A basic rule-based allocation strategy (e.g., first-come-first-serve) will serve as a benchmark for comparison.

2. Performance Metrics:

- Percentage of utilized parking spaces – A measure of parking space optimization.

- Average waiting time for users on the waiting list – Indicates the system's load and efficiency.

- Overall efficiency of the algorithm*, calculated using the following metric:

$$E = \frac{\text{Accepted Reservations}}{\text{Total Requests}} - \lambda \cdot W$$

- We will run simulations for both the smart model and the basic model, analyzing the differences in performance.

5.Engineering Process

5.1 Study existing applications and adaptation of some ideas in our project:

This project took inspiration from several existing parking systems to understand effective features and their limitations:

- Parkopedia:
  - Adopted: Real-time updates on parking availability.
  - Limitations: Does not include user behavior analysis or prioritization.
- ParKam:
  - Adopted: Efficient updates on parking availability.
  - Limitations: High costs and complexity unsuitable for academic institutions.
- DPM +:
  - Adopted: Clear communication features between users and the system.
  - Limitations: Primarily tailored for corporate parking, not educational settings.

5.2 Constraints Impacting Development:

- Budgetary Constraints:
  - Leveraging open-source tools and software to minimize costs.
  - Avoiding expensive systems like IoT sensors or high-resolution cameras.
- Time Constraints:
  - Developing and testing features within a defined timeline.
  - Ensuring a minimum viable product (MVP) is ready for deployment.
- Privacy and Simplicity:
  - Focusing on practical solutions without storing sensitive data.
  - Ensuring that the system respects user privacy while remaining efficient.

5.3 Parking reservations processing optimisation algorithm

Within our project we use the following idea. Reservations with high priorities are accepted automatically, while reservations with low priorities are added to waiting lists, ordered by their priority. The cutoff threshold for accepting reservations is dynamically optimized.

5.3.1.Priority calculation:

The priorities depend on:

- User type (Good, Neutral, or Bad).
- Time of reservation (early vs. late).
- Desired parking place (close vs. far).

The priority of a reservation is determined as follows:

$$P=\alpha \cdot U+\beta \cdot T+\gamma \cdot L$$

Where:

- U: User type (e.g., Good = 10, Neutral = 5, Bad = -5).
- T: Time of reservation (normalized, e.g., earlier reservations receive higher weights).
- L: Location type (e.g., Close = 10, Far = 5).
- α,β,γ: Weighting coefficients that balance the importance of user type, time, and location.

5.3.2 User type estimation

In our project we can estimate user types based on similarity of their behavior to users with known types.

- User behaviour representation

We represent the user behavior only by three numbers:

- **The percentage of reservations the user arrived at on time:**

  P_on-time = (Number of reservations arrived on time / Total reservations) × 100

- **The percentage of cancellations made at least one hour in advance:**

  P_cancel-on-time = (Number of timely cancellations / Total cancellations) × 100

- **User Role :**
  A categorical value representing the user's role:
  - Lecturer (Role = 1).
  - Student (Role = 2).
  - Other employee (Role = 3).

Each user is represented by a feature vector:

x_i=[P_on-time,P_cancel-on-time,Role]

- Centroids of clusters

Distance Calculation: For each user, the distance to all cluster centers is computed. The formula for calculating the distance is:

$$C_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$$

where (x1,y1) represents the user's coordinates and (x2,y2) represents the cluster center's coordinates.

- Distance between the users

For each user, the distance to all cluster centers is computed. The formula for calculating the distance is:

$$d = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$$

where (x1,y1) represents the user's coordinates and (x2,y2) represents the cluster center's coordinates.

- Cluster Assignment

Each user is assigned to the cluster with the smallest distance. The formula representing the distance between a data point xi and a cluster center Cj is:

$$d(x_i, C_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - C_{jk})^2}$$

where x-ik is the value of the data point in dimension k, and C{jk} is the value of the cluster center in dimension k.

- Recalculation of user type based on daily running of k-means

K-Means is a clustering algorithm used to group data points into k distinct clusters based on their shared features. In this project, the algorithm classifies users into clusters (types) based on their parking behavior represented by a feature vector (see 5.3.2.1). To maintain an updated classification system, the K-Means algorithm is run daily with updating feature vectors, classifying both new and existing users. This approach ensures the parking management system remains accurate and responsive to dynamic user behaviors.

New users remain in a neutral category for the first 30 parking reservations, during which their behavior is monitored:

- Observation Period:
  Data is collected daily for punctuality, cancellations, and user type.

  Users remain in the neutral group, without prioritization or penalties.

At the end of 30 days, the user's feature vector is incorporated into the K-Means algorithm

Iteration Until Convergence: This process is repeated until the cluster assignments no longer change (convergence).

Advantages of Daily Implementation:

- Dynamic Updates: Daily clustering ensures the system reflects current user behaviors.
- Neutral Monitoring: New users are observed independently, preventing their data from skewing the clustering of existing users.
- Accurate Classification: Regular updates improve the reliability of user classifications and parking allocation.

5.3.3 cutoff threshold for accepting reservations calculation

Classical Q-Learning algorithm

Q-Learning is a Reinforcement Learning algorithm that learns the optimal actions for a given state in a dynamic environment. The goal is to maximize a reward over time by determining the best policy (sequence of actions).

Components of Q-Learning

1. States(s): Represents the current situation or configuration.
2. Actions (a): Choices the system can make in a given state.
3. Rewards (r): Numerical feedback for performing an action in a state.
4. Q-Values (Q(s,a)): A table storing the estimated utility of performing action a in state s. These values are updated iteratively.

Q-Learning Update Formula

$$Q(s,a) = Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

- Q(s,a): Current value of taking action aaa in state sss.
- r: Immediate reward for performing action aaa.
- s': Next state after performing action aaa.
- max′Q(s′,a′): The highest Q-value of all possible actions in the next state s′.
- α: Learning rate, controls how much new information influences current values.
- γ: Discount factor, determines how much importance to give to future rewards.

Steps of Q-Learning

1. Initialization: Start with all Q-values Q(s,a)=0 or random values.
2. Action Selection:
    - Use an exploration-exploitation strategy:
        - Exploration: Randomly choose actions to explore new possibilities.
        - Exploitation: Choose the action with the highest Q-value for the current state.
3. Perform Action and Observe Reward:
    - Execute the action *a* and transition to the next state *s′*.
    - Receive immediate reward r.
4. Update Q-Value:
    - Update Q(s,a) using the formula above.
5. Repeat:
    - Continue until the Q-values stabilize or for a predefined number of iterations.

Advantages of Q-Learning:

- Adds real-time adaptability to dynamic parking scenarios.
- Considers immediate and future rewards for efficient parking allocation.

5.4 Parking reservations processing

Q-Learning After K-Means Classification:
Once users are classified using K-Means into groups + Neutral group, Q-Learning is applied to optimize the processing of parking reservations based on non-random priorities.

- Algorithm for Priority Cutoff Optimization:

The priority cutoff is a crucial parameter in the Q-Learning process for parking management. It defines the minimum priority score (P) that a user's reservation must exceed to be automatically accepted. Reservations with a score below this threshold are added to the waiting list. The Q-Learning algorithm dynamically updates Pcutoff off to optimize parking resource allocation based on:

  - Parking availability (close and far spots).
  - Time slots (peak hours vs. off-peak hours).
  - Current system load (number of users waiting).

Algorithm Steps:

  - Initialize P-cutoff with an arbitrary value.
  - For each reservation request:
    - Calculate the priority P using the formula above.
    - Compare P with P-cutoff:
      - If P≥P-cutoff, accept the reservation.
      - If P<P-cutoff < P, add the user to the waiting list, ordered by their priority.
        - Update P-cutoff based on system feedback (e.g., parking usage, waiting list size) using Q-Learning.

- Formula for Efficiency Evaluation (E):
The efficiency of the algorithm is calculated as follows:

$$\text{Efficiency} = \left(\frac{\text{Arrived}}{\text{Total}}\right) - \gamma \cdot \left(\frac{\text{Illegal}}{\text{Total}}\right)$$

Where:

Arrived – Number of users who actually showed up on time.
Illegal – Number of **unlawfully canceled** reservations (e.g., no-show without approval).
Total – Total number of reservations.
γ – Penalty factor (e.g., γ=2).

This formula ensures that both positive behavior (arrival) and negative behavior (illegal cancellation) are considered.

5.4.1 Combined Workflow:

Step 1: K-Means for User Classification

Collect user behavior data:

- Punctuality Percentage (Pon-time).
- Cancellation Percentage (Pcancel-on-time).
- User Type (type): Student, Lecturer, Employee.

Apply K-Means clustering:

- Users are classified into clusters (Good, Bad).
- Neutral users remain in their group until enough data is collected (30 days).

Calculate Priority (P) for each user:

- Use the formula: $P=\alpha \cdot U+\beta \cdot T+\gamma \cdot LP$ Where U, T, and L represent user type, reservation time, and location preference respectively.

step 2: Priority-Based Q-Learning for Dynamic Decision Making

For every parking request:

- Calculate the priority score (P) for the user based on their type, reservation time, and location preference.
- Compare the score with the system's priority cutoff :
  - If $P \geq P\text{-cutoff}$, accept the request automatically.
  - If $P < P\text{-cutoff}$, add the user to the waiting list, sorted by priority.

States (s):

Define the states based on the percentage of parking spots available, in increments of 10%:

- **s1**: 0% parking spots available
- **s2**: 10% parking spots available
- **s3**: 20% parking spots available
- **s4**: 30% parking spots available
- **s5**: 40% parking spots available
- **s6**: 50% parking spots available
- **s7**: 60% parking spots available

- **s8**: 70% parking spots available
- **s9**: 80% parking spots available
- **s10**: 90% parking spots available
- **s11**: 100% parking spots available

Q-Learning to dynamically adjust P-cutoff:

Actions (a):

Each action determines how users are managed based on the availability state and their group priority.

1. **a1**: Assign a parking spot to users above **Cutoff 1**.
2. **a2**: Assign a parking spot to users above **Cutoff 2**.
3. **a3**: Assign a parking spot to users above **Cutoff3.**

It is important to note that due to time constraints for system development and training, we opted for a limited number of states (11 values) and a relatively small number of actions (3). This simplification allows the system to operate effectively at an initial level and achieve satisfactory results. However, the model can be further expanded in the future by increasing the number of states and actions to improve decision-making accuracy and provide a more efficient solution.

Step 3: Efficiency Evaluation and Feedback

1. Calculate efficiency (E)
2. Optimize weights (α,β,γ) and P-cutoff:
   - Continuously refine the priority formula and Q-Learning parameters based on system feedback to maximize efficiency and user satisfaction.

5.4.2.Advanced Cutoff Optimization Approaches

Our system employs Q-Learning for dynamically optimizing the cutoff threshold, but additional strategies can enhance its effectiveness:

- *For Real-World Data* – Before applying Q-Learning, we can first perform *Model Fitting*, allowing us to estimate optimal parameters based on historical data. The Q-Learning algorithm can then be executed using simulations derived from this fitted model rather than starting from random values.
- Hybrid Approach – Combining statistical methods to determine an initial cutoff value with dynamic adjustments using Q-Learning.

By incorporating these enhancements, the system can improve decision-making efficiency and ensure optimal parking allocation based on both simulated scenarios and real-time data observations. improve decision-making efficiency and ensure optimal parking allocation based on both simulated scenarios and real-time data observations.

5.5 Simulation and Visualization of User Behavior

To evaluate how the system handles a variety of user behaviors and to validate the effectiveness of the classification mechanism, we implemented a large-scale simulation that generates reservation activity over a simulated week.

Using a custom simulation engine, the system created **randomized reservations for each user**, mimicking real-world patterns of use. This allowed us to observe how different types of users interact with the system and how their behavior influences parking availability, system load, and user status.

Reservation Simulation Logic:

```
For each user in the system:
    Choose a random number of reservations to generate (e.g., between 1 and 4)

    Repeat for the chosen number of times:
        Pick a random date within the next 7 days
        Pick a random start time between 08:00–18:00
        Set a fixed duration (e.g., 1 hour)
        Calculate end_time = start_time + 1 hour

        If user.status == "BLOCKED":
            Assign reservation to distant parking lot (distant_zone = true)
        Else:
            Assign reservation to regular parking lot (preferred_zone = true)

        Check if the user has any overlapping reservations
        If no overlap exists:
            Add the reservation with the following details:
                - user_id
                - date
                - start_time
                - end_time
                - location_type ("regular" or "distant")
                - status = "simulated"
```

This simulation allowed us to:

- Populate the system with realistic data for testing.
- Observe system performance under variable demand.
- Validate that users with a “BLOCKED” status are assigned less favorable locations.
- Ensure that no overlapping reservations are made, maintaining data integrity.

This approach provided a reliable and flexible tool to analyze the behavior-driven dynamics of the system and to support further development of smart prioritization and fairness algorithms.

5.5.1 User/System Grade Formula

This scoring mechanism quantifies the user/system's behavior by awarding or deducting points for each action.

$$\text{Score} = (\text{Arrived} \times p_{\text{arrived}}) + (\text{Cancelled} \times p_{\text{cancelled}}) + (\text{Illegal} \times p_{\text{illegal}})$$

Where:

p_arrived = Points for arriving on time (e.g., +10)
p_cancelled = Points for legal cancellation (e.g., 0)
p_illegal = Penalty points for illegal cancellation (e.g., −20)

$$\text{MaxScore} = \text{Total Reservations} \times p_{\text{arrived}}$$

-> This represents the ideal score if **all users arrived on time**.

$$\text{Grade} = \left(\frac{\text{Score}}{\text{MaxScore}} \right) \times 100$$

This scoring mechanism translates user behavior into a numerical value by assigning positive or negative points for specific actions. It allows the system to objectively classify users based on punctuality, cancellations, and rule violations.

Simulated User Profiles:

The simulation used the `User` class from `user_model.py`, which defines:

- Weekly time tables based on configurable behavior parameters (e.g., average arrival/departure times). Each user is assigned a weekly schedule that reflects average arrival and departure behavior, based on probability distributions

  For each user:

  Define weekly_schedule = {

  "Monday": {"arrival": 08:30, "departure": 16:00},

  "Tuesday": {"arrival": 09:00, "departure": 17:00},

  ...

  }

  Set behavior_parameters = {

  "avg_arrival_time": 08:30,

  "avg_departure_time": 16:00,

  "arrival_stddev": 0.5,

  "departure_stddev": 0.5

  }

- Realistic reservation statuses such as `arrived`, `cancelled`, and `illegal_cancelled`.->
  For each reservation:

  Generate random number between 0 and 1

  If rand < arrival_probability:

  status = "ARRIVED"

  Else if rand < arrival_probability + cancel_probability:

  status = "CANCELLED"

  Else:

  status = "ILLEGAL_CANCELLED"

- Behavioral randomness including no-shows and late cancellations based on defined probabilities.

  For each user:

  Define behavior_profile = {

  "arrival_probability": 0.8,

  "cancel_probability": 0.15,

  "no_show_probability": 0.05

  }

  For each reservation:

  Randomly assign outcome based on profile

Each simulated user had:

- A general weekly schedule.
- A full year of simulated reservation activity.
- A behavior score computed automatically.

  If reservation.status == "ARRIVED":

  user.score += 10

  Else if reservation.status == "CANCELLED":

  user.score += 0

  Else if reservation.status == "ILLEGAL_CANCELLED":

  user.score -= 20

  # Update status

  If user.score < threshold:

  user.status = "BLOCKED"

  Else:

  user.status = "ACTIVE"

Graphical Dashboard:

The administrative interface includes:

- **Bar charts** displaying the distribution of reservation outcomes.

  On dashboard load:

    Fetch stats from /stats

    Display bar chart:

      X-axis: Status types (Arrived, Cancelled, Illegal)

      Y-axis: Counts

- Visualization of system usage, updated via real-time data from `/stats`.

  Set interval timer (e.g., every 30s):

    Fetch updated stats from /stats

    Update chart values dynamically

- Tooltip-based display of both **percentage** and **actual counts** per status.

  On chart hover:

    Show tooltip:

      "32 ARRIVED (64%)"

      "10 CANCELLED (20%)"

      "8 ILLEGAL (16%)"

- Exportable **PDF reports** summarizing user behavior.

On 'Export PDF' click:

  Compile user behavior stats:

    - Total reservations per user

    - Average score

    - Number of violations

  Generate PDF file

  Prompt download

Benefits of the Simulation:

- Allowed performance testing without real users.
- Helped optimize the behavior score algorithm.
- Provided meaningful data to **feed into K-Means clustering**.
- Offered admin-friendly insights for monitoring system trends.

5.6 Impact of Parameter Changes on Simulation Outcomes

The system includes an interactive simulation tool that allows only the administrator to modify key parameters related to the user scoring mechanism. This feature is designed to evaluate how changes in point values influence user classification and overall system efficiency.

Simulation Components:

Three adjustable sliders are available to the admin:

`pointsArrived`: Points awarded for on-time arrival.

`pointsCancelled`: Points awarded or deducted for legal cancellations.

`pointsIllegal`: Penalty for illegal cancellations (no-show or last-minute violations).

When the admin changes a slider value:
The updated value is stored in corresponding state variables within the application.

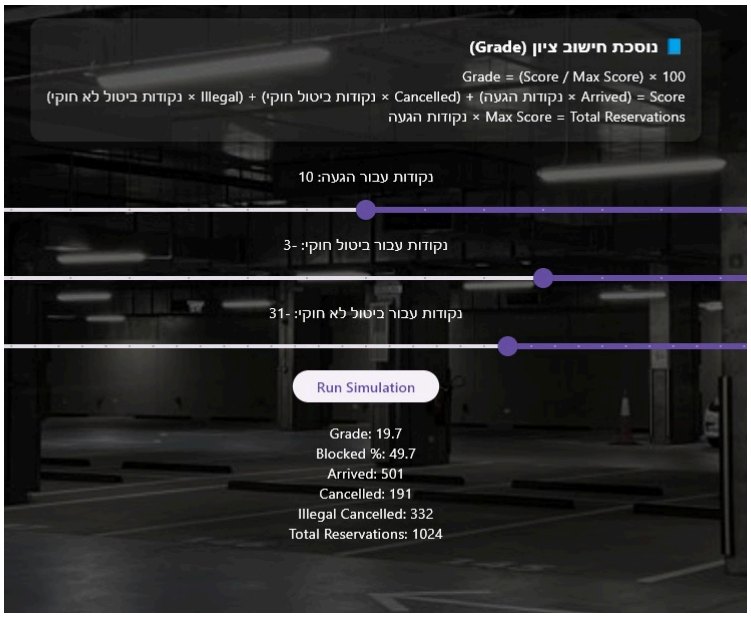
Upon clicking "Run Simulation":
A POST request is sent to the Flask server with the updated parameters:

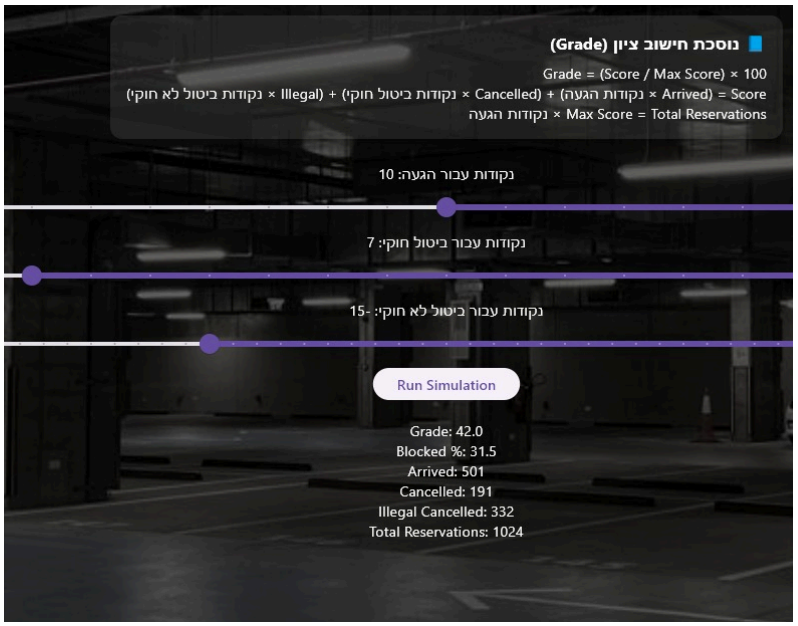
json
Copy code

```
{
  "points_arrived": <slider value>,
  "points_cancelled": <slider value>,
  "points_illegal": <slider value>
}
```

Once a response is received from the server, the following results are displayed on screen: Grade (overall system score), Blocked % (percentage of users marked as BLOCKED), Number of arrivals, Number of legal cancellations, Number of illegal cancellations, Total reservations

Example Comparison of Results:



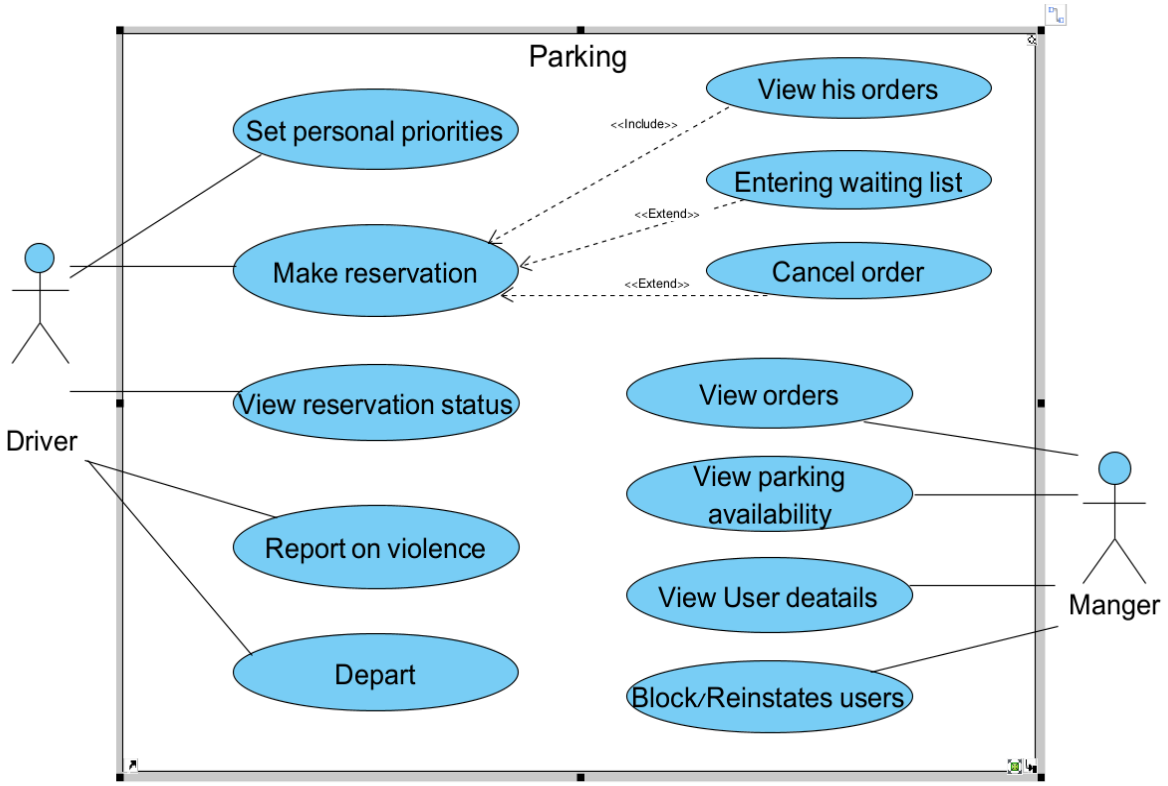


Conclusions

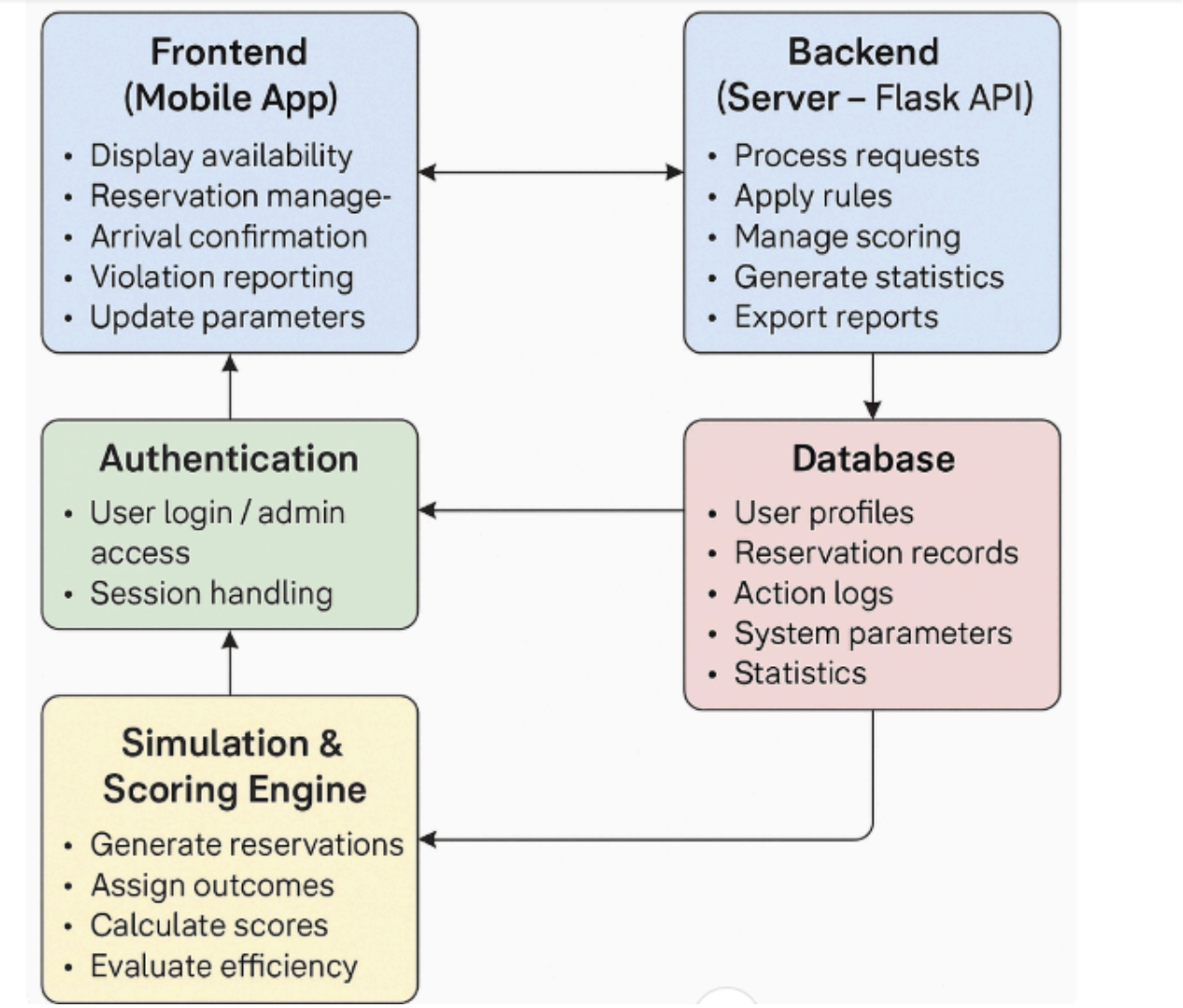
This simulation demonstrates that modifying point values directly impacts the system's behavior:

- Reducing the penalty for illegal cancellations decreases the number of blocked users.
- Increasing the reward for legal cancellations improves overall system scores.
- These dynamic settings allow the admin to fine-tune the system for fairness, adaptability, or strictness, depending on operational needs.

6.Use case


Parking
View his orders
Set personal priorities
<<Include>>
Entering waiting list
<<Extend>>
Make reservation
Cancel order
<<Extend>>
View reservation status
View orders
Driver
View parking availability
Report on violence
View User deatails
Manger
Depart
Block/Reinstates users

7.Architecture



1. Frontend (Mobile App – Flutter)

Purpose:
The user-facing application that enables interaction with the parking system.

Key Features:

- Display of real-time parking availability.

- Reservation creation and cancellation.
- Arrival confirmation within a specific time window.
- Violation reporting interface.
- Admin interface for managing users, reservations, and statistics.
- Dynamic update of parameters for simulations and scoring.

2. Backend (Server – Flask API)

Purpose:
Processes logic, manages business rules, and communicates between the frontend and the database.

3. Authentication Layer (Flask Sessions / Optional Firebase)

Purpose:
Manages secure login, session handling, and admin authentication.

Key Features:

- Supports user roles (regular user vs. admin).
- Protects sensitive admin endpoints.
- Handles login, logout, and session-based access control.

4. Database (PostgreSQL or MySQL)

Purpose:
Stores all persistent system data.

Data Includes:

- User profiles (ID, name, email, status, role, score).
- Reservation records (date, start/end time, status, assigned zone).
- Logs of actions and violations.
- System configuration parameters (scoring values, simulation settings).
- Statistics and efficiency logs.

5. Simulation & Scoring Engine

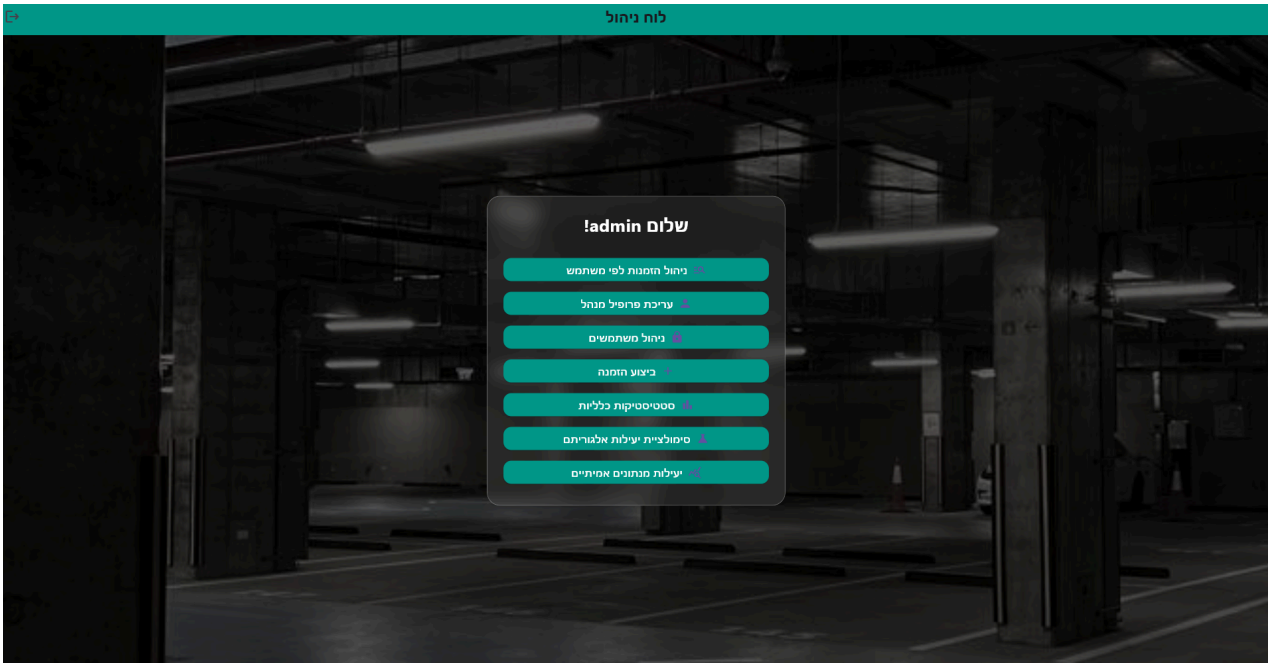
Purpose:
Generates simulated reservations and evaluates the system's efficiency under changing conditions.

Key Functions:

- Randomized reservation generation for virtual users.
- Behavior-based outcome assignment (ARRIVED, CANCELLED, ILLEGAL).
- Score calculation and dynamic classification (ACTIVE / BLOCKED).
- Supports parameter tuning by the admin and visual feedback.

8.user interface

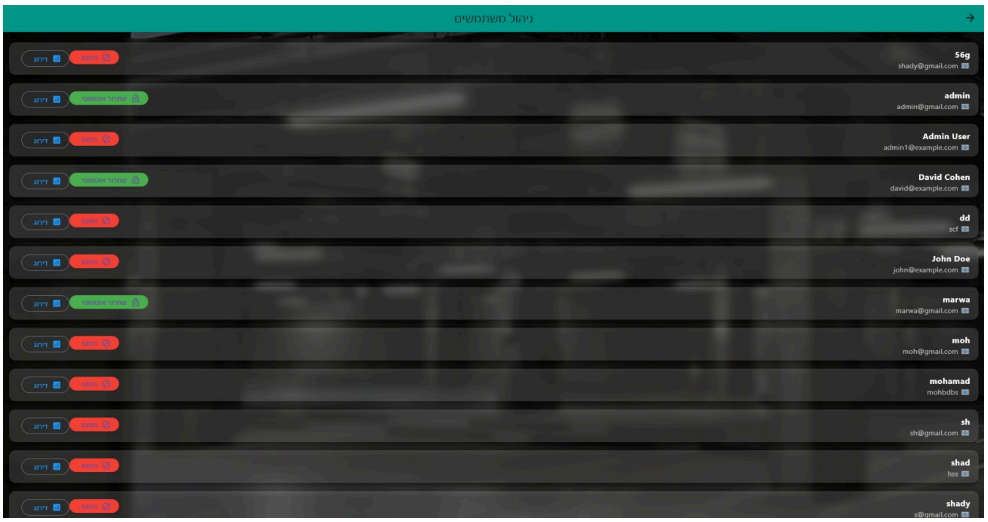
Manager Screen:
Description: Displays administrative options such as viewing user orders.
Purpose: Provides tools for administrators to manage the system efficiently.

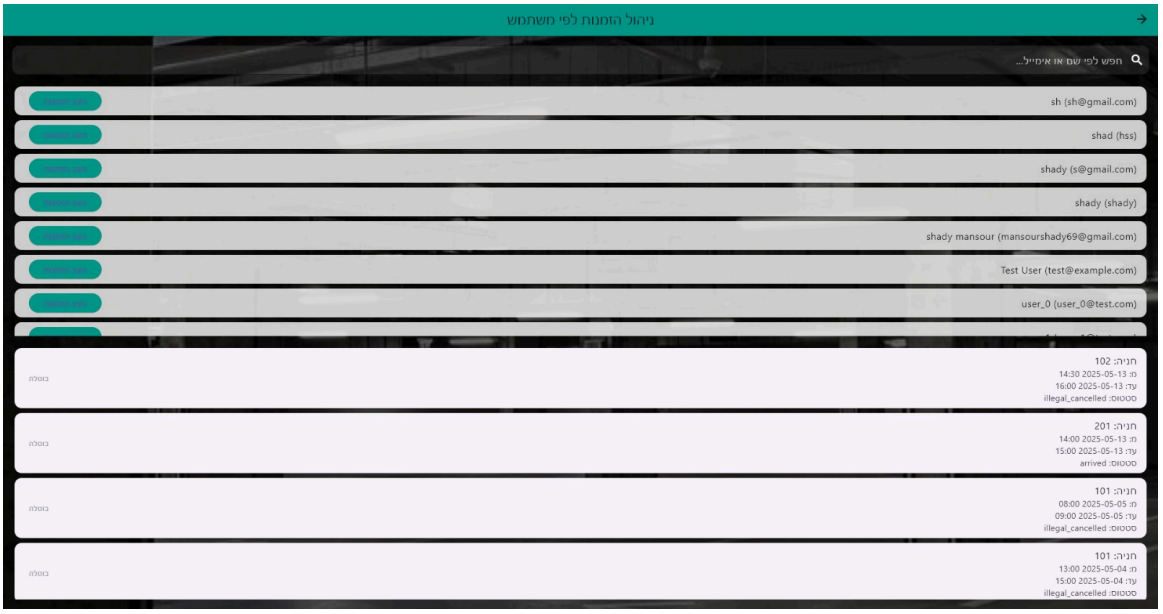


After selecting an action like "View user orders" or "Block user," the manager is redirected to a screen where they can input the user's ID and choose to view the profile, check orders, or block the user.

If the user is blocked, the manager can also release them by selecting the **Reinstate user** option in the first image, entering the user's ID, and performing the unblocking action

The screen shows two user categories in the system: Good Users and Bad Users, with their names listed under each category. For each user, the manager has the option to transfer them between the two groups: from Bad Users to Good Users and vice versa, using the "Transfer" button provided next to each user's name. The current system cutoff value, P_Cutoff: 45, used to determine priority when confirming an order

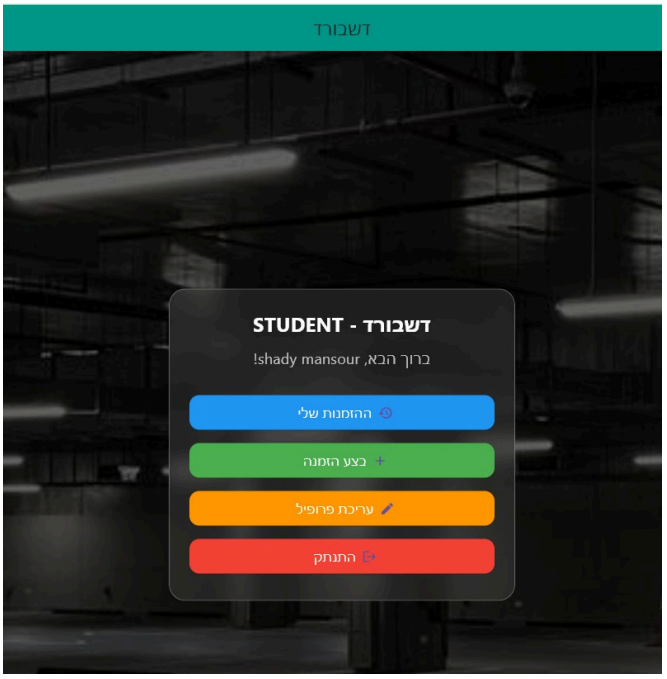




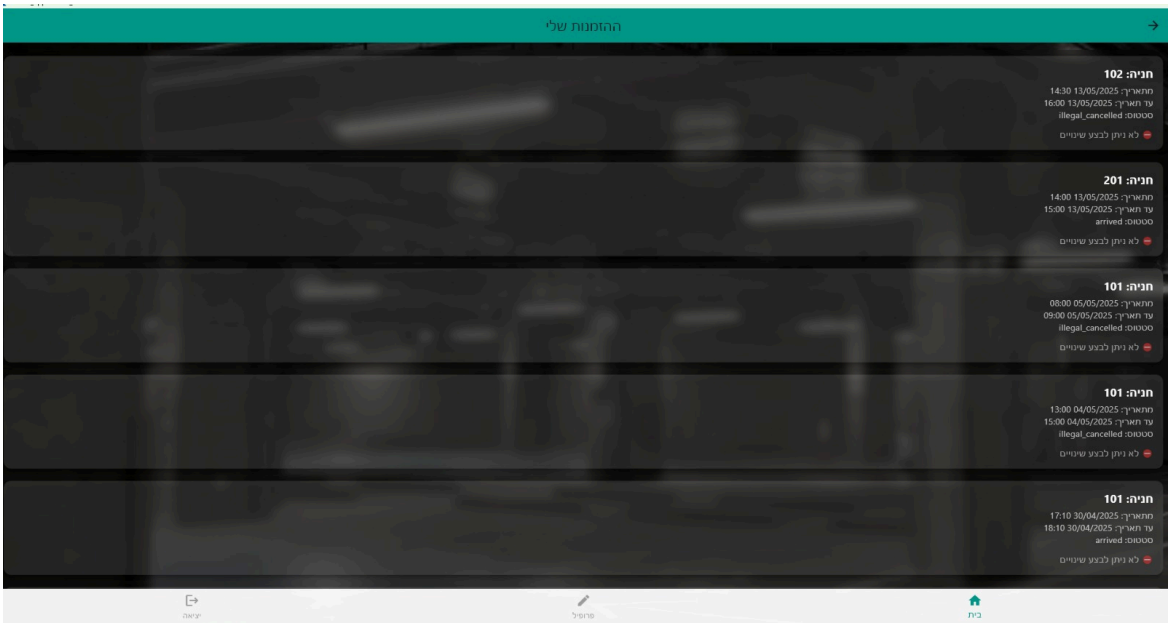
Reservation Status Overview

Administrative View: Statistics of All Users' Behavior

User Screen:
Description: The main screen for regular users, providing options such as viewing past orders, placing a new parking order, accessing their profile information and choosing the time he prefer .
Purpose: Acts as the central hub for users to navigate and interact with the app's primary features efficiently and conveniently.



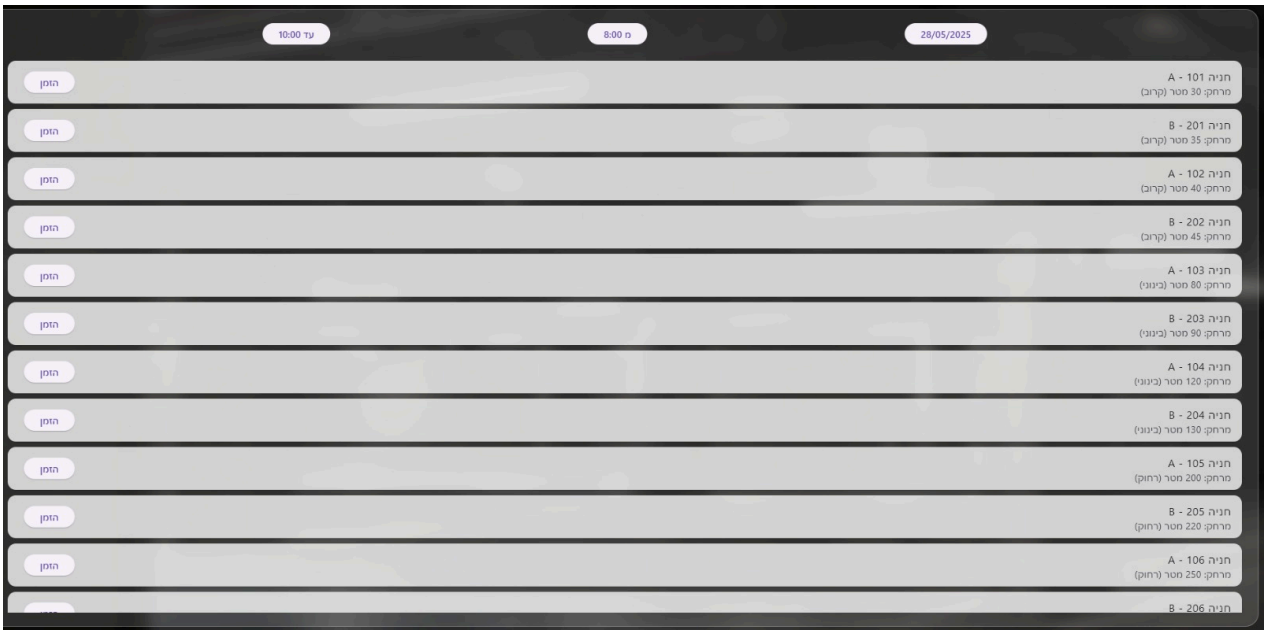
When the user clicks on the view orders button, he receives a table with all his orders:



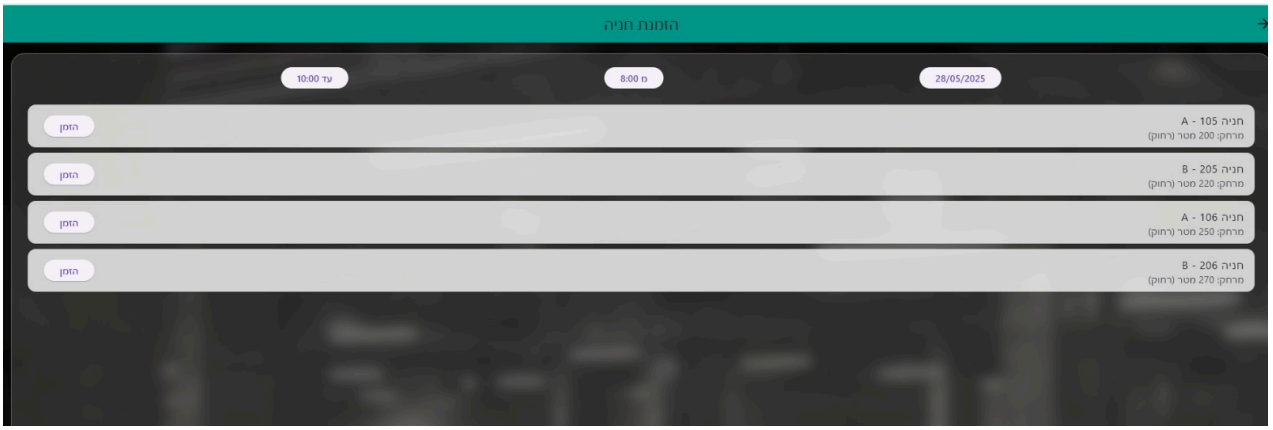
Parking Selection Screen:

Description**:** Users can select from available parking spots. For restricted (bad) users, nearby parking lots are blocked, and they are only allowed to choose spots in distant parking lots.
Purpose**:** Provides real-time feedback on parking availability while enforcing user-specific restrictions to promote fairness and discourage misuse.
(*Insert the Parking Selection screen for Good and Bad Users here.*)

Good user:



Bad user :



9.REFERENCES

- https://www.pango.co.il/
- https://www.cello-app.com/
- https://business.parkopedia.com
- https://www.dpm-parking.com/en/
- https://parkam.com
- https://www.mypango.com/

- https://www.dpm-parking.com/en/
- https://www.sciencedirect.com/science/article/abs/pii/S0167739X19322496
- https://www.intuz.com/blog/iot-in-smart-parking-management-benefits-challenges
- https://www.mdpi.com/1424-8220/25/2/427
  Real-Time On-Device Continual Learning Based on a Combined Nearest Class Mean and Replay Method for Smartphone Gesture Recognition by Heon-Sung Park, Min-Kyung Sung, Dae-Won Kim and Jaesung Lee, Published: 13 January 2025.
- https://www.sciencedirect.com/science/article/pii/S1366554523002223
  Yubin Liu, Qiming Ye, Jose Escribano-Macias, Yuxiang Feng, Eduardo Candela, Panagiotis Angeloudis, Available online 5 August 2023.
- https://www.sciencedirect.com/science/article/pii/S1319157819312613
  Stéphane Cédric Koumetio Tekouabou, El Arbi Abdellaoui Alaoui, Walid Cherif , Hassan Silkan, Available online 1 February 2020.