WePool

The prefect balance between

Privacy & convenience

Created by: Aviel Smolanski & Elior Uzan

June 2025

Table of Contents:

1. Literature Review……………………………………………………………… 3 - 4
2. Competitor review……………………………………………………………. 5 - 6
3. Stakeholders…………………………………………………………………... 7
4. Use-Case for example………………………………………………………. 8 - 10
5. Functional requirements…………………………………………………… 11 -13
6. Non-functional requirements……………….……………………….…... 14
7. Introduction………………………………………………………………….... 15
8. Key Components………………………………………………………....… 15 – 16
9. Design Considerations……………………………………………………… 16
10. System Architecture………………………………………………………… 17 – 18
11. Data Design……………………………………………………………………. 18 – 20
12. Detailed class Design……………………………………………………… 20 – 22
13. STP………………………………………………………………………………. 23 – 41

13.1 System Description…………………………………………………… 24

13.2 Test objectives…………………………………………………………. 25

13.3 STP document goals…………………………………………………. 25

13.4 Reference……………………………………………………………….. 25

13.5 Glossary…………………………………………………………………. 26 – 28

13.6 Test levels………………………………………………………………. 29 – 30

13.7 Test planning…………………………………………………………... 30

13.8 Test execution…………………………………………………………. 30

13.9 Test tree…………………………………………………………………. 30 - 34

13.10 Environmental Requirements for System Testing………… 34

13.11 Work Environments for System Testing……………………… 34

13.12 Configuration and version management…………………….. 34 - 35

13.13 Success metrics…………………………………………………….. 35 - 36

13.14 Technology and authority…………………………………………. 36

13.15 Bug report protocol…………………………………………………. 37 - 39

13.16 Bug severity…………………………………………………………… 39 - 40

13.17 Tools for bug report and test management…………………. 41

13.18 Bug Tracking and Management…………………………………. 41

13.19 Test deliverables…………………………………………………….. 41

14. STD………………………………………………………………………………. 42 - 52

Literature Review

**Carpooling** is the sharing of [car](https://en.wikipedia.org/wiki/Automobile) journeys so that more than one person travels in a car, and prevents the need for others to have to drive to the same location themselves.

Carpooling reduces each person's travel costs such as fuel costs, stress and boredom of traveling alone. It is also environmentally friendly by reducing  air pollution and [carbon emissions](https://en.wikipedia.org/wiki/Carbon_emissions).

Carpooling faces a major setback, which is travelling with strangers. However, according to an article that was published on 08.07.2022 with the name "Carpooling in times of crisis: Organizational identification as a safety belt"[[1]](#footnote-1) states that the creation of carpooling programs restricted to members of a single organization represents an interesting opportunity that several organizations have seized. Indeed, various studies have shown that individuals are more likely to trust and interact with members of their own organization, using group identity to create personal connections. In this scenario, individuals do not necessarily share common interests but may share, to an extent, an organizational identity. In that respect, organization-based carpooling programs would benefit from the identification to the organization of members, which would help overcome the psychological barriers to carpooling.

Another advantage of carpooling is getting to know new people and you can also save time by discussing work related issues during the ride. An article that was published on 11.09.2019 with the name "Carpooling could be the traffic solution we already have"[[2]](#footnote-2) spotted this topic: "[Bosch" company had a 50% enrollment rate](https://www.theguardian.com/small-business-network/2017/sep/25/carpooling-mainstream-congestion-splt-gokid) in its Mexican offices and reports improved employee retention. Carpooling has also increased their inter-departmental collaboration as employees who don't usually meet can converse on the way to work. ["Bosch" has saved](https://www.theguardian.com/small-business-network/2017/sep/25/carpooling-mainstream-congestion-splt-gokid) 55,000 miles (90,000km) and 25 tons of carbon dioxide.

The realm of carpooling relies largely on routing algorithms, most famously used by Google Maps and Waze. Routing algorithms are foundational in navigation systems, enabling them to calculate the best paths between locations. These algorithms rely on graph theory, where road networks are represented as graphs with intersections as nodes and roads as edges. By processing vast amounts of data, including road layouts, distances, and live traffic conditions, they determine efficient routes. Continuously analyzing changes in traffic, accidents, and closures, they dynamically adapt to provide updated directions. Their goal is to balance factors like travel time, distance, and user preferences, ensuring reliable and optimized navigation experiences in real-time.

Carpool systems function as the online “middleman” between drivers and passengers. When a driver adds a ride, the system searches throughout its entire data set for a passenger match that will meet his preferences the best, factoring in his schedule and location. The same process happens vice versa, where a passenger adds a ride request.  
  
On 9.12.2024 an article about traffic jams was published on "Ynet". The main headline of the article was "You can't imagine: the traffic jams in the center have gotten worse. This is the average speed, which goes down"[[3]](#footnote-3). The average speed across the "גוש דן" area has decreased by 7.5-11% compared to earlier this year. Traffic jams are getting worse and worse.

One solution to this problem is the one for public transportation. Many think this is the best solution to the problem by getting about 50 people in a vehicle instead of a maximum of 5. But this solution has many problems and obstacles. Some of them are governmentally encouraged (budget priority), recruiting drivers for the buses, adding new lines to get to as many main places as possible.   
  
Our solution can overcome most of the obstacles. It doesn't require building new roads, buying new buses, recruiting new drivers, government intervention (that entails a lot of time and money, involving all the bureaucracy). The carpool solution doesn't require any new physical intervention.

**It is the solution with minimum changes and maximum effect!**

Competitor review

Our project has several competitors out there in the market, but each competitor differs in different aspects. The main innovation in our project is not the carpool system itself, but the idea of keeping the carpool system intra-organization for each company.

**BlaBlaCar:**

An international platform connecting drivers and passengers, primarily designed for long-distance rides. The focus of this company is on long-distance rides. Our solution (WePool) is focusing on short daily rides within a workplace area and is intra-organizational in contrast to BlaBlaCar.

**Waze Carpool:**

A carpooling service that connects drivers and passengers based on matching routes and schedules. Most of the car drivers use Waze but their carpool didn't work well and one of the reasons is riding with strangers.

**Moovit:**

A platform for planning and managing trips. Mainly used for planning trips via public transport. In the carpool area they collaborate with Waze Carpool and their problems are shared.

Although Waze Carpool and Moovit were the most popular and famous carpool platforms, they discontinued in 2023 because of changing market demands.

Our app (WePool) provides a solution to the problem of privacy and the need to coordinate times by being an intra-organizational system (most of the workers start and finish at the same time). We provide more security and privacy in contrast to wide open carpool systems.

**Gett:**

A platform offering taxis and ride-sharing solutions, including business-oriented services, but lacks the focus on intra-organizational carpooling. Their focus is on taxis and paid carpool systems.  
Our app (WePool) is independent of taxi services, making it more affordable and accessible for employees using private vehicles.

**Via:**

Via is one of the leading ride-sharing platforms in Israel. It offers shared rides in real-time, connecting passengers going in the same direction. Via also features one noticeable setback compared to traditional carpooling services that operated in Israel and dismantled later, which is it's relatively high pricing. While Via can customize solutions for specific organizations, its primary focus is on improving public transit and shared mobility across broader, often public-facing communities.

Our app (WePool) doesn’t require any additional resources from the company, or the government like via does (they must hire a driver or a shuttle service).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Waze Carpool | Moovit | BlaBlaCar | Gett | Via |
| Scope | wide | wide | wide | Wide/  limited | Wide/limited |
| Security (derived from the scope) | low | low | low | high | medium |
| Popularity | closed | closed | Popular among long rides | high | Rapidly growing |
| Pricing | low | low | low | high | medium |

As we can see our platform (WePool) is being innovative in the carpool market by giving companies the opportunity to have a private carpool system for their employees. By that, they can save money for themselves and for their employees, encourage social connections and familiarity among employees, reduce the stress of their employees (because of the unnecessary time they spend in traffic) and optimize their focus.

Stakeholders

**Primary Stakeholders**

* Employees (Workers):
  + Drivers: Offering rides and ensuring availability for passengers.
  + Passengers: Booking rides to and from work and relying on the system for safe and efficient transportation.
* HR Managers:
  + Managing employee accounts, usage policies and monitoring statistics.
* System Admin:
  + Designing, developing, and maintaining the system.

**Secondary Stakeholders**

* Organizations:
  + Supporting the adoption of the carpooling system to promote safety, reduce commuting issues, and enhance employee interaction.
  + Benefiting from improved employee satisfaction and punctuality.
* IT and Support Teams:
* Providing technical support and ensuring smooth system operations.
* Addressing issues related to system maintenance and user queries.
* API Providers:
* Services like Waze and Google Maps that provide real-time traffic data, route optimization, and address validation.
* Their performance and availability directly affect the system's reliability.

Use cases for example:

**Use Case Name: Adding a ride**

**Primary Actor**: Driver (Worker)

**Level**: User-goal

**Preconditions**:

* The driver must be logged into the app with an active account.
* The driver’s account must not be banned.

**Success Guarantee (Postconditions):**

* The ride is added to the system and visible to passengers based on the matching criteria.

**Main Success Scenario (Basic Flow):**

* The driver logs into the system and navigates to the "Add a Ride" option.
* The system displays a form for creating a new ride.
* The driver enters the following details:
* Departure time (if heading home) or arrival time (if heading to work).
* Starting location and destination.
* Number of available seats.
* Maximum detour time/distance they are willing to allow.
* The system validates the details entered:
* Ensures the locations are valid (via Waze or Google Maps).
* Upon validation, the system saves the ride details.
* The ride is made available in the system for passengers to view and book.
* The driver receives confirmation that the ride has been successfully added.

**Extensions (Alternate Flows):**

* Invalid Location:
* The system detects that the entered address is invalid (via Waze/Google Maps).
* It displays an error message, and the driver must correct the address.
* Conflict with Existing Ride:
* If the driver has an overlapping ride, the system notifies them and prevents duplicate rides.
* Insufficient Details:
  + The system prompts the driver to fill in all mandatory fields if they attempt to submit incomplete information.

**Use Case Name: Passenger books a ride**

**Primary Actor:** Passenger (Worker)

**Level:** User-goal

**Preconditions:**

* The passenger must be logged into the app with an active account.
* The passenger’s account must not be banned.
* The system must have at least one active ride created by a driver to match against the request.

**Success Guarantee (Postconditions):**

* The ride is successfully booked, and the passenger receives confirmation with trip details.
* Notifications (if enabled) are sent to the driver about the new booking.

**Main Success Scenario (Basic Flow):**

* The passenger logs into the system and navigates to the "Search for Rides" option.
* The system displays a form for entering ride search criteria.
* The passenger specifies the following details:
* Desired departure time (if heading home) or arrival time (if heading to work).
* Starting location and destination.
* The system processes the input and displays a list of available rides that match the criteria.
* The passenger selects a preferred ride from the list.
* The system confirms seat availability for the selected ride.
* Upon confirmation, the system books the ride for the passenger:
* Deducts one seat from the driver's available seats.
* Notifies the driver of the new booking.
* The passenger receives a booking confirmation with ride details, including:
* Driver's name and contact information (if applicable).
* Pickup time and location.
* Estimated arrival time.
* The booking is recorded in the passenger’s ride history.

**Extensions (Alternate Flows):**

* No Matching Rides:
* If no rides match the passenger's criteria, the system displays a message suggesting adjustments to time or location.
* Overbooked Ride:
* If the selected ride becomes fully booked before confirmation, the system notifies the passenger and prompts them to select another ride.
* Invalid Location:
* The system detects that the entered address is invalid (via Waze or Google Maps).
* It displays an error message, and the passenger must correct the address.

Functional Requirements

**User Authentication**

Employees log in securely using organization-specific identifiers, ensuring that only authorized personnel access the system.

* Username: Company’s ID (assigned by the admin).
* Password: Worker’s ID number in the company.
* The system checks if the account is banned or not.

**User Roles and Permissions**

* **System Admin (Main User):**
* Full control over all companies and users.
* Adding companies.
* Removing companies.
* Add, remove, and manage HR managers for each company.
* Monitor system usage statistics across all companies.
* Manage ride’s cut-off times prior to the ride:
* Work-bound rides: 1 hour.
* Home-bound rides: 10 minutes.
* Manage all time frames described in the requirements.
* **HR Manager (Each company has one/several):**
* Add, update, and delete workers in their company.
* Monitor ride usage and statistics for their organization.
* Determine how many “bad points” a worker can “score” before banning.
* Determine the duration of the ban.
* Ban/unban a worker.
* **Worker:**
* **Drivers:** Offer rides based on several criteria.
* **Passengers:** Search for and choose rides.
* Receive and respond to notifications about rides.

**Address creation**

* Passengers can:
* Add a permanent address.
* Enter a new address for each ride.
* The system checks on Waze if the entered address is valid. If not, it displays an error.

**History**

Workers can view their past rides and participation statistics.

**Ride Creation and Booking**

* **Drivers:** Can create ride offers by the following criteria:
* Start time (for home-bound trips).
* Arrival time (for work-bound trips).
* Route.
* Available seats in the vehicle.
* Maximal deviation time they are willing to allow from their shortest path to the destination.
* **Passengers:** Can search for and book available rides based on:
  + Start time (for home-bound trips).
  + Arrival time (for work-bound trips).

**Ride Matching**

The system matches passengers with drivers based on the following criteria:

* For work-bound trips: No more than a 30-minute difference between the driver’s arrival time and the passenger’s desired arrival time.
* For home-bound trips: No more than a 30-minute difference between the driver’s start time and the passenger’s desired start time.
* The system ensures the specific passenger’s inclusion does not exceed the deviation threshold set by the driver.
* If the number of assigned seats reaches the limit, the trip will no longer be visible for booking.

**Scheduling and Notifications**

The system sends notifications:

* To passengers when their request is approved by the driver.
* To passengers with the initial ETA upon ride selection, clarifying it is subject to changes.
* To the driver when a trip that he created was added to the search panel successfully.
* To the driver when a new passenger requests to join a ride, allowing the driver to accept or reject the request.
* When a driver starts a trip:
* Sends a link to all passengers with ride details, ETAs, and live updates via Waze.
* Displays the route to the driver.
* ETA updates for all passengers.
* The system checks 30 minutes before the scheduled departure time to update arrival time (via Waze/Google Maps). If the new ETA exceeds the original by 5 minutes or more, notifications are sent to the driver and passengers to adjust readiness.

**Cancelation policy**

If a driver or passenger cancels a trip X time before the trip starts (time determined by the HR manager):

* They receive a "bad point."
* After X bad points (threshold determined by the HR manager), the worker is banned from the app for a defined duration.
* Users can view their accumulated bad points and understand the consequences.

Non-Functional Requirements

**Performance:**

* The system should handle up to 10,000 users across multiple companies without latency.
* Ride matching should be complete within 2 seconds after query submission in 95% of cases.

**Availability:**

* The system must be available 99.9% of the time, with scheduled maintenance during off-peak hours.

**Scalability:**

* The system should support the addition of new organizations without significant reconfiguration.
* It must be scaled to accommodate increasing users within an organization.

**Usability:**

The Android app should be intuitive, with a clean and responsive interface.

**Reliability:**

* The system must consistently provide accurate ride matches and notifications based on the filters the driver/passenger entered.
* Backup and recovery mechanisms should be in place to prevent data loss.

**Integration:**

* The app must be integrated with external APIs like Waze or Google Maps for real-time traffic data and route optimization.

**Security:**

* All data transmission must be used by HTTPS.
* Role-based access control (RBAC) ensures users can only access features relevant to their role.

**Portability:**

* The app should support Android 8.0 and above.

1. **Introduction**

**Purpose:** The purpose of our "WePool" system is to offer shared rides to workers within the same organization. This way, workers can consolidate existing friendships and create new ones, discuss matters that relate to work or common interests and above all cut significantly the money spent on commuting.

**System overview:** "WePool" strives to function as a comprehensive intra – organizational platform to connect commuting workers. The system integrates an effective and reliable routing algorithm by Waze and a self – made matching algorithm to connect drivers and passengers based on constraints applied beforehand. Our system aims to be a user-friendly mobile Android app which features 3 main user types (Admin, HR manager and worker) with varying degrees of authorization.

1. **Key Components**

* **Mobile App**: The app serves as the primary interface for the users (passengers, drivers, HR managers, and admins).
* **Server-Side Backend:** The backend processes all requests, manages role-based access, interfaces with the Waze/Google Maps API for route optimization, retrieves and updates data from the database, and handles notifications. The notification system within the backend triggers alerts to users based on events such as ride updates, schedule changes, or requests.
* **Ride Matching Service**: This service, powered by the Waze/Google Maps API, evaluates and optimizes driver routes. It calculates deviations caused by adding passengers and ensures the updated route adheres to predefined thresholds by the driver (e.g., a 10-minute maximum detour).
* **Database**: The database stores all user accounts, ride details, company settings, and usage statistics. It ensures data persistence and supports analytics for admins and HR managers.
* **Admin Panel**: Provides admins with tools to configure global settings, manage companies, and oversee system-wide analytics.
* **HR Manager Dashboard**: Enables HR managers to manage workers, monitor ride statistics, and enforce company-specific policies
* **Driver Features**: Drivers can create and manage ride offers, accept passenger requests. Routes are optimized using Waze/Google Maps for efficiency.
* **Passenger Features**: Passengers can search for and book rides, manage their ride history, and receive real-time updates.

**How They Interact**

* **Mobile App ↔ Backend**: The mobile app sends user requests to the backend, which processes the logic and returns results.
* **Backend ↔ Waze API**: The backend integrates with Waze/Google Maps to fetch route data and evaluate deviations for ride matching.
* **Backend ↔ Database**: All user, ride, and system data are stored and retrieved from the database.
* **Admin Panel & HR Manager Dashboard ↔ Backend**: These interfaces allow admins and HR managers to configure settings and retrieve analytics via backend interactions.
* **Passenger ↔ Backend:** Passengers interact with the backend to search for available rides, book rides and receive notifications about ride updates or changes. The backend processes these requests, retrieves relevant ride details from the database, and provides real-time updates.
* **Driver ↔ Backend**: Drivers communicate with the backend to create and manage ride offers, accept or reject passenger requests, and optimize routes through the Waze/Google Maps API. The backend ensures all ride-related data is updated in the database and notifies drivers of new requests or changes.

1. **Design Considerations**

Assumptions:

* Users have stable internet access.
* Users will access the app via Android devices running version 8.0 or higher.
* Users will turn on the location service before opening the app.

Constraints:

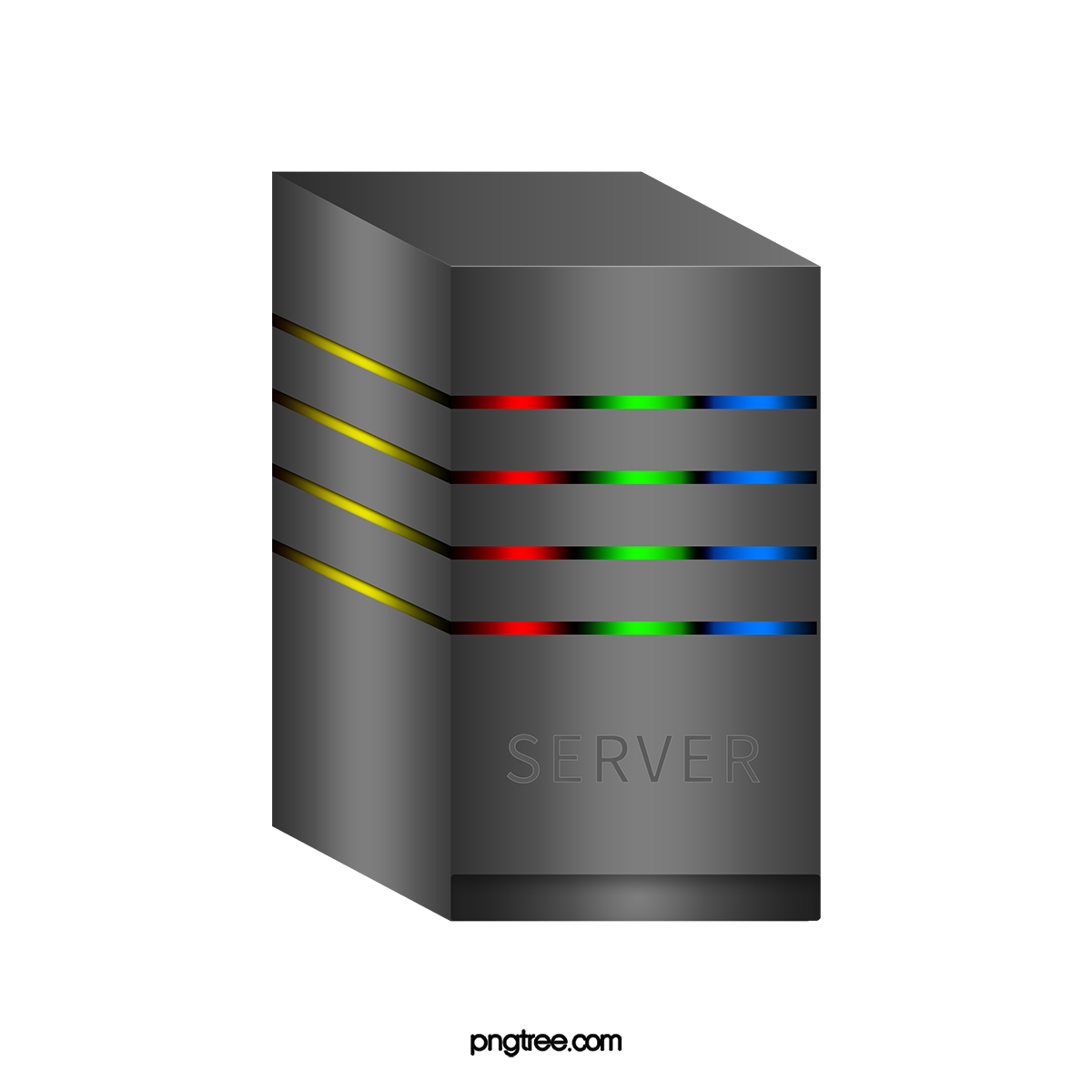
* Initial scalability is limited to 100 users per company.
* The backend can handle up to 100 concurrent requests in the initial phase.
* Daily API requests to Waze are limited (e.g., 1,000 calls/day during initial deployment).
* The database can initially store up to 100 GB of data, including user accounts, ride history, and analytics.

1. **System Architecture**

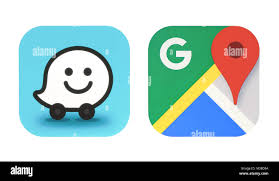
user



Mobile app



Server-Side Backend



Waze/Google Maps API



Data Base

1. **Users:**
   * Includes all roles: passengers, drivers, HR managers, and admins.
   * Initiates interactions with the system via the mobile app.
2. **Mobile App**:
   * Acts as the client-side interface for all users (passengers, drivers, HR managers, and admins).
   * Sends user requests (e.g., ride searches, ride creation, or settings updates) to the backend.
   * Receives responses from the backend, such as ride-matching results, notifications, and updated data.
3. **Server-Side Backend**:
   * **Core Logic**: Processes all user requests from the mobile app, manages role-based access, and enforces permissions.
   * **Ride Matching Service**: It integrates with the Waze/Google Maps API to calculate optimized routes, evaluate deviations, and match passengers with drivers while adhering to driver-defined thresholds (e.g., a 10-minute maximum detour).
   * **Notification System**: It triggers alerts and updates for users based on events like ride confirmations, cancellations, or schedule changes.
   * **Database Operations**: The backend retrieves and updates data from the database to ensure consistent user, ride, and system data management.
4. **Waze/Google Maps API**:
   * Provides real-time traffic and route optimization data.
   * The backend queries Waze/Google Maps through the Ride Matching Service to fetch updated route calculations for drivers and ensure optimized ride assignments.
5. **Database**:
   * Stores all system data, including user accounts, ride details, company-specific settings, and logs for analytics.
   * Supports persistent data storage for backend operations and ensures seamless access for admins, HR managers, and regular users.

**Technological Requirements:**

* Development Environment: Android Studio.
* Programming Languages: java for app development and backend algorithms.
* Database Technology: PostgreSQL.

1. **Data Design**

Key entities include user, passenger, driver, ride, HR manager, admin, company, admin – company relationship, HR manager – company relationship.

The database schema:

* User table:
* user\_id INT PRIMARY KEY AUTO\_INCREMENT
* name VARCHAR(100) NOT NULL
* email VARCHAR(150) UNIQUE NOT NULL
* password VARCHAR(255) NOT NULL
* role ENUM('Admin', 'HR Manager', 'Driver', 'Passenger') NOT NULL
* company\_id INT
* is\_banned BOOLEAN DEFAULT FALSE
* created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP
* FOREIGN KEY (company\_id) REFERENCES Company(company\_id)
* Passenger table:
* passenger\_id INT PRIMARY KEY
* preferred\_pickup\_location VARCHAR(255)
* ride\_preferences VARCHAR(255)
* FOREIGN KEY (passenger\_id) REFERENCES User(user\_id)
* Driver table:
* driver\_id INT PRIMARY KEY
* vehicle\_details VARCHAR(255)
* available\_seats INT
* FOREIGN KEY (driver\_id) REFERENCES User(user\_id)
* Ride table:
* ride\_id INT PRIMARY KEY AUTO\_INCREMENT
* driver\_id INT NOT NULL
* start\_location VARCHAR(255) NOT NULL
* end\_location VARCHAR(255) NOT NULL
* start\_time DATETIME NOT NULL
* available\_seats INT NOT NULL
* company\_id INT
* created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP
* FOREIGN KEY (driver\_id) REFERENCES Driver(driver\_id)
* FOREIGN KEY (company\_id) REFERENCES Company(company\_id)
* Ride Passenger table:
* ride\_id INT NOT NULL
* passenger\_id INT NOT NULL
* joined\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP
* PRIMARY KEY (ride\_id, passenger\_id)
* FOREIGN KEY (ride\_id) REFERENCES Ride(ride\_id)
* FOREIGN KEY (passenger\_id) REFERENCES Passenger(passenger\_id)
* HR manager table:
* hr\_manager\_id INT PRIMARY KEY
* FOREIGN KEY (hr\_manager\_id) REFERENCES User(user\_id)
* Admin table:
* admin\_id INT PRIMARY KEY
* FOREIGN KEY (admin\_id) REFERENCES User(user\_id)
* Company table:
* company\_id INT PRIMARY KEY AUTO\_INCREMENT
* company\_name VARCHAR(255) UNIQUE NOT NULL
* created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP
* Admin\_Company table:
* admin\_id INT NOT NULL
* company\_id INT NOT NULL
* PRIMARY KEY (admin\_id, company\_id)
* FOREIGN KEY (admin\_id) REFERENCES Admin(admin\_id)
* FOREIGN KEY (company\_id) REFERENCES Company(company\_id)
* HR\_Manager\_Company table:
* hr\_manager\_id INT NOT NULL
* company\_id INT NOT NULL
* PRIMARY KEY (hr\_manager\_id, company\_id)
* FOREIGN KEY (hr\_manager\_id) REFERENCES HR\_Manager(hr\_manager\_id)
* FOREIGN KEY (company\_id) REFERENCES Company(company\_id)

**Data flow:**

User logs in to the system and identifies his role (admin, HR manager or worker). The backend verifies the user's credentials and role. If the user is a worker who's a passenger in the current session, then he chooses his desired departure time and location. Following this step, he will be displayed with all relevant rides. He chooses the ride most suitable for him.

If the user is a worker who's a driver, then he adds a ride at specific time and to specific place. In addition, he sets the maximal time which he is willing to add to his commute to pick others up. Later that day, he will receive a notification telling him there's a passenger willing to join his ride. The driver accepts the request.

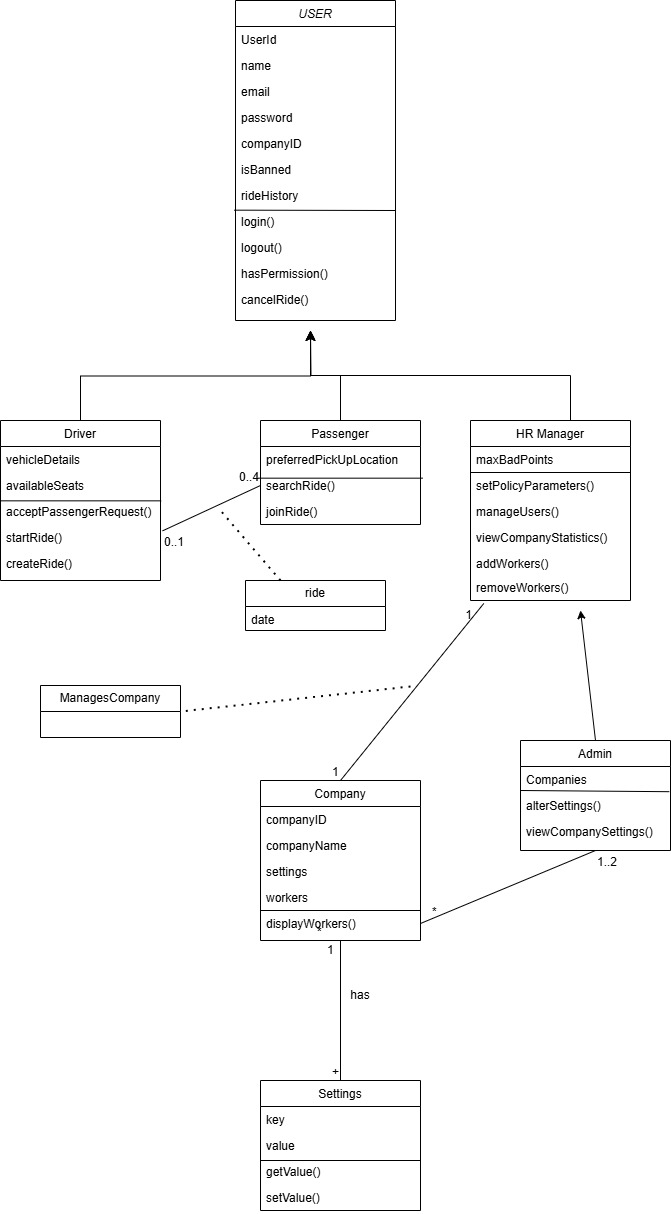
If the user is an HR manager, he navigates to the dashboard on the app where he can view and analyze statistics such as ride history for the company and pending requests.

In the case of an administrator, the user accesses the dashboard where he can add or remove a company from the service. In addition, the user views cross-company statistics of app usage to make business – oriented insights.

1. **Detailed Class Design**

* User:
* Attributes: User\_ID, name, email, password, companyID, isBanned, rideHistory
* Methods: login (), logout (), hasPermission(), cancelRide()
* Driver:
* Attributes: vehicle\_details, available\_seats
* Methods: acceptPassengerRequest (), startRide(), createRide()
* Passenger:
* Attributes: prefferedPIckUpLocation
* Methods: searchRide(), joinRide()
* HR Manager:
* Attributes: maxBadPoints
* Methods: setPolicyParameters(), manageUsers(), viewCompanyStatistics()
* Ride:
* Attributes: date
* Methods:
* Admin:
  + Attributes: Companies
  + Methods: alterSettings(), viewCompanySettings()
* Company:
  + Attributes: company\_ID, company\_Name, settings, workers
  + Methods: addWorkers(), removeWorkers(), displayWorkers()
* Settings:
  + Attributes: key, value
  + Methods: getValue(), setValue()

The classes above are due to changes both with their attributes and methods but those are our main classes



STP

|  |  |
| --- | --- |
| Application Testing | Project |
| Aviel Smolanski & Elior Uzan | Written by |
| 20/05/2025 | Last edit |
| Aviel Smolanski & Elior Uzan | Submitted by |

Distribution control

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Approved by | Written by | Date |
| 20.5.2025 |  | Aviel Smolanski & Elior Uzan | PM |

**System description**

The WePool Carpool System is a platform designed to facilitate organized carpooling within organizations and workplaces. The system enables employees from the same company to plan and join shared rides to and from work, reducing travel costs and environmental impact.

The system was developed in response to the growing need for secure, efficient, and eco-friendly commuting solutions. It supports both drivers and passengers and differentiates between roles to provide role-based functionality.

WePool is accessible via a dedicated Android mobile application and is supported on Android – powered devices.

The application allows:

* Drivers create rides, define available seats, preferred arrival times, and planned routes.
* Passengers search for suitable rides based on destination, time preferences, and minimal route deviation.
* HR managers activate and deactivate users based on their own discretion, as well as having full access to driver and passenger's features.

Additional system capabilities include:

* Real-time ride search and request system for passengers.
* Ride planning and route optimization for drivers.
* Role-based access control and dynamic UI rendering.
* Notifications sent via Firebase Cloud Messaging when rides are updated or confirmed.
* Live navigation integration using Google Maps when a ride begins.
* Secure storage of user data, favorite locations, and ride history in Firebase Firestore.

**Test objectives:**

* To ensure that the system meets user and stakeholder requirements, displays relevant content, and performs the necessary actions.
* To validate that the application adheres to high standards of **quality**, **usability**, and **performance**.
* To verify consistent behavior and compatibility across multiple devices and Android OS versions.
* To confirm that all business logic and user flows are properly covered by tests.
* To detect and address any critical bugs before release.
* To guarantee that the application meets security requirements, including authentication and data protection.

**STP Document Goals:**

The purpose of the STP (Software Test Plan) is to provide a comprehensive outline of the test strategy, from planning to execution, for approval and reference.

The document serves as a guiding framework for the testing process and includes:

* Definitions and explanations of key terms and concepts used throughout the project.
* A detailed description of the test cases to be executed and their mapping to specific system requirements.
* A full explanation of the bug reporting lifecycle, from identification to resolution and verification.
* A test planning section including timeline, milestones, and team role assignments.
* A risk assessment identifying potential issues that may impact the testing or development process.

**Reference:**

|  |  |
| --- | --- |
| **Description** | **Name** |
| Software requirements specification document | SRS |

**Glossary:**

The following is a list of terms and concepts that will help in navigating various test documents, some of which will be used throughout this document.

|  |  |
| --- | --- |
| **Description** | **Term** |
| Product requirements document | PRD |
| Software Requirements Specification Document | SRS |
| Software test plan | STP |
| Software test description | STD |
| Team foundation server | TFS |
| Microsoft test manager | MTM |
| Quality center | QC |
| Application lifecycle management | ALM |
| Software development lifecycle | SDLC |
| the ability to track each test case back to its corresponding requirement | Traceability |
| Mobile device platform | Mobile |
| Google's mobile operating system | Android |
| Internet data transfer protocol | HTTP |
| an organized collection of data that allows efficient storage, retrieval, and management of information. | Database |
| A concise test scenario that addresses a specific requirement | Test case |
| User level – the smallest building blocks of the system | User story |
| Smooth flow throughout all stages of the process from start to finish without any faults | Happy flow |
| Code testing under the responsibility of the development team | Unit testing |
| testing the combination of capabilities, which may be reflected in structural, functional, and non-functional tests, across different levels | Integration testing |
| testing that examines the behavior of the system as a whole | System testing |
| the final testing stage, usually performed by the end user or client. | Acceptance testing |
| checks what the system does | Functional testing |
| checks how the system performs its tasks | Nonfunctional testing |
| Tests that cover scenarios where incorrect input is entered into the system, expecting it to remain stable and continue functioning with an appropriate error message | (N) |
| Sanity tests – the most critical and important tests in the product, representing the core functionality of the system | Smoke tests |
| tests after a system fix to ensure that the bug has been resolved | Confirmation tests |
| tests after changes are made to the system. These are usually repeated tests that previously passed successfully | Regression tests |
| Exploratory testing performed without predefined test scripts | Monkey testing |
| Test was executed and passed | Passed |
| Test was executed and failed | Failed |
| Test in blocked status – the test cannot be executed due to various reasons | Blocked (test) |
| Test not executed for various reasons despite being written | Not run test |
| A term for halting development work until the next iteration. | Code freeze |
| the visual part of a software application that allows users to interact with the system through graphical elements like buttons, icons, and menus. | GUI |
| bug that prevents the user from using the system and causes it to crash. This bug must be resolved before releasing the system version. | Critical bug |
| Major bug – a high-severity issue that disrupts the normal operation of the system but does not cause it to crash and can be worked around. It is important to address this type of bug, and it is not recommended to release a system version containing it. | High level bug |
| Medium-severity bug – allows the user to continue working but may occasionally cause delays in the release of the system version. | Medium bug |
| Low-severity bug – usually related to the user interface and does not affect system functionality. These bugs are typically addressed with low priority and do not delay the version release. | Low level bug |
| It is an iterative, cycle-based model that allows for responsiveness to changes and involves the client throughout the development process. | Agile model |
| Development cycle in the Agile model – typically lasts 2–4 weeks | Sprint |
| Development environment – minimally controlled, with little to no change reporting. The main concern is ensuring the environment functions properly. | DEV |

**Test levels:**

**Sanity Testing**

The first tests to be performed – conducted by the QA testers.  
These are basic and critical tests that allow for quick and efficient identification of whether the core/main functionality of the product works as expected.

Sanity tests serve as "entry scenarios"; if an issue is found at any of these stages, there is no point in continuing to test the product.  
The existing bugs must first be fixed, and the product should be returned to the development team.

**System Testing**

System testing examines the behavior of the system.  
These are large-scale tests, primarily functional and non-functional, aimed at ensuring that the system performs what it is required to do — and does not perform what it is not supposed to do.

The testing will include, among other things: Database tests, Interaction between internal processes, Integration with external systems, Performance tests, Load and stress testing, Resilience tests, Usability testing, Information security tests and Recovery testing from failure scenarios.

**Regression Testing**

After any change is made to the system, whether due to a version update or a bug fix—besides verification tests (which ensure the bug was resolved), regression testing is also necessary.

This involves testing parts of the system that functioned correctly before the change and were not directly modified but might have been affected by it. The goal is to ensure they were not unintentionally broken.  
These are typically repeated tests that had previously passed successfully.

**Integration Testing**

These tests connect two or more components of the system.  
They help ensure the quality and correctness of the system’s partial process model by verifying that the integrated units work together as expected.

**Monkey Testing**

This is a technique where the user enters the application and inputs random, often invalid, values to observe the system’s behavior or to try to cause it to crash.  
There are no strict guidelines in these tests; the tester performs them based on experience and intuition.

**Test planning:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Responsibility** | **End date** | **Start date** | **Products** |
| QA team lead | 20.5.25 | 30.4.25 | STP |
| QA team lead | 25.5.25 | 20.5.25 | STD |
| QA team lead | 30.5.25 | 25.5.25 | STR |

**Test Execution:**

The testing process will be carried out in two cycles (each lasting 6 days)

In the first testing cycle, we will perform sanity tests, GUI tests, integration tests, system tests, and additional technical tests to verify system integrity.

The second testing cycle will include regression tests for verification and monkey tests to rule out the presence of hidden bugs in the system.

**Test tree:**

Cycle 1:

**1. Android (Mobile App)**

1.1 Authentication & User Roles

1.1.1 Sign up with email and password

1.1.2 Login with valid credentials

1.1.3 Invalid login attempt

1.1.4 Role selection: Driver / Passenger

1.1.5 Logout and session clearing

1.1.6 Auto-login on app restart

1.2 Driver Functionality

1.2.1 Create a new ride (TO\_WORK / TO\_HOME)

1.2.2 Add pickup points using Autocomplete

1.2.3 View active and past rides

1.2.4 Cancel a planned ride

1.2.5 Approve passenger requests

1.2.6 Start navigation using Google Maps

1.2.7 Receive ride start notification

1.2.8 Filter rides by date/time/direction

1.2.9 View static map preview of ride

1.3 Passenger Functionality

1.3.1 Search for available rides

1.3.2 Request to join a ride

1.3.3 View active ride status

1.3.4 Cancel ride request (with time constraints)

1.3.5 View request history (approved, pending, declined)

1.3.6 Receive notifications for ride updates

1.3.7 Filter rides by destination, time, and direction

1.4 Shared Features

1.4.1 View and select favorite locations

1.4.2 Add new favorite location (with placeId)

1.4.3 View ride details and passenger list

1.4.4 Bottom navigation visibility

1.4.5 Role-based screen navigation

1.4.6 View static map image from Firebase

Cycle 2: Android App Testing

* Full Regression
* Sanity (Critical Flow Testing)
* Exploratory Monkey Testing

2.1 Admin / HR Manager Functionality (Android only)

2.1.2 Add a new company  
 2.1.3 Assign an HR Manager to a company  
 2.1.4 View all registered users  
 2.1.5 View all rides associated with the company  
 2.1.6 Block or activate user accounts  
 2.1.7 Manage company locations (add/edit/remove)

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Type of Test** | **Test Environment** |
| Login with email and password | 1. UI Testing 2. Functional Testing 3. Input validation 4. Integration Testing | Android 13+ device with Firebase Authentication configured |
| Role selection screen loads correctly | 1. Functional Testing 2. UI Testing | Logged-in user with multiple roles |
| Create new ride as driver | 1. UI Testing 2. Functional Testing 3. Data Validation 4. Integration with Google Maps | Android device with GPS + Internet access |
| Add pickup points using Autocomplete | 1. Functional Testing 2. Integration Testing 3. UI Testing | Android with Google Places API key |
| Search for rides as passenger | 1. Functional Testing 2. Edge Case Handling 3. UI Testing | Android with passenger role, pre-existing rides in Firestore |
| Join a ride | 1. Functional Testing 2. Integration with Firestore 3. Input validation | Logged-in passenger, eligible ride exists |
| Receive notification on ride approval | 1. Notification Testing 2. Integration Testing 3. Background behavior | Android with FCM token registered |
| View active ride with passenger list | 1. UI Testing 2. Functional Testing 3. Real-time data display | Driver user with active ride and confirmed passengers |
| Cancel ride as driver | 1. Functional Testing 2. Data Update Verification 3. Validation checks | Driver logged in, active ride present |
| Add/remove companies | 1. Functional Testing 2. Permission validation 3. UI & backend logic testing | Admin role, backend Firestore role management |
| Store and load favorite locations | |  | | --- | | 1. Functional Testing 2. UI Testing 3. Integration with Place API |  |  | | --- | |  | | Android with location permission and saved favorites |
| Navigation via Google Maps | 1. Intent launch test 2. Real-time GPS behavior 3. UI state during navigation | Android 11+ device, Google Maps installed |
| Handle offline scenarios | 1. Functional Testing 2. Network error handling 3. UI fallback verification | Internet manually disabled during usage |
| Push notification when driver starts navigation | 1. FCM Testing 2. Functional Verification 3. Real-time messaging | Passenger with registered FCM token |
| Any Android device with a production-like environment | 1. Exploratory Testing 2. Random interaction 3. Crash detection | Monkey test for unexpected input |

**Environmental Requirements for System Development:**

**1. Android Development Setup** \* Android Studio with Kotlin & Jetpack Compose

\* Android SDK 30+ with emulator or physical device

\* Firebase and Google Maps SDKs integrated

**2. Backend Integration**

\* Firebase project with Auth, Firestore, FCM, and Functions

\* Internet access for API calls

\* Git for version control

**Work Environments for System Testing:**Microsoft Excel for managing STD test cases and the carpool app system.

**Configuration and Version Management:**

**There are three separate work environments that must not be mixed:**

1) Development Environment -

This environment is designated solely for developers and is managed by the development manager.  
It is where the development team works continuously on the full version of the system, including changes and updates. Therefore, this is considered a “non-clean” environment.  
Once development is complete, the version is passed by the development manager to the QA manager for testing.

2) Testing Environment –

Only QA personnel are authorized to work in this environment, which is managed by the QA manager.  
All system and regression tests are executed here.  
The goal is to maintain a “clean” environment, dedicated exclusively to testing, simulating the client environment as accurately as possible, and allowing tests across multiple operating systems.  
Once testing is complete, the version is handed over to the PM by the QA manager.

3) Production Environment –

This is the final environment before releasing the version to the client, and only the Project Manager (PM) is authorized to work here.  
It is used strictly for system maintenance and must never be used for testing or development.

**Success metrics:**

**Tests:**

1) 100% of sanity tests must pass successfully

2) At least 85% of planned test cases must be executed

3) At least 81% of executed tests must pass

4) A maximum of 4% of tests may be inexecutable

**Bugs:**

1) No Critical severity defects

2) No High severity defects

3) A maximum of 5% of tests may include medium severity defects

4) A maximum of 10% of tests may include Low severity defects

**Technology and resources:**

|  |  |
| --- | --- |
| 4 computer workstations to serve the QA team's testing needs | Workstations |
| Android powered WePool app | Software and operating systems |

**Other means to carry out the tests:**

|  |  |
| --- | --- |
| None | Automated tools for test execution |
| Mtm, Jira, Excel | Tools for test management |
| Network printer, minimum 100MB broadband internet | Other equipment |
| Setting up system tests on clones with different configurations, according to the requirements | Tools for simulating the real environment |

**Authority:**

|  |  |
| --- | --- |
| The Project Manager (PM) will present the system requirements at the start of the project and will oversee the project throughout. The R&D Manager will coordinate with the QA Manager to ensure complete documentation of the testing process. | Definition of roles and responsibilities |
| The QA team is authorized to open and close bugs with Low and Medium severity. High and Critical severity bugs require approval from the QA Manager before they can be closed. | Jurisdiction |

**Bug report:**

Submitted bug

Reject

Duplicate

Bug meeting

Bug is closed

Bug verification

Reopen test meeting

Bug resolved

Freeze

Reproduce

Postponed to next release

Monitor

Open bug

Explanation of the steps:

|  |  |  |
| --- | --- | --- |
| **Responsibility** | **Description** | **Step name in the diagram** |
| QA | Discovery of a new bug | Submitted bug |
| QA, PM and DEV | Bug meeting – during which the team determines whether an issue is indeed a bug, prioritizes the open bugs, and decides whether a change in the system specification is required due to the discovered bugs. | Bug meeting |
| QA, DEV | When there is no basis for the bug, it is usually due to a misunderstanding. | Reject |
| QA | Duplicate bugs usually closed before the meeting. | Duplicated |
|  | Opening a new bug in the system | Open bug |
| QA | After the bug has been fixed by the DEV team, it is returned to the QA team for regression testing and verification. | Bug resolved |
| QA, DEV | The QA team determined that the bug was not resolved and returned it to the DEV team. | Reopen test meeting |
| QA | Verification by the QA team that the bug has indeed been fixed and that no new bugs were introduced because of the fix (through the performed verification and regression tests). | Bug verification |
| QA |  | Bug is closed |
| QA, DEV | The bug is returned to the DEV team, but they are unable to reproduce it. Therefore, they send it back to the QA team with the status Reproduce. The QA team must attempt to reproduce the bug and reopen it. If they are unable to reproduce it, the bug is closed. | Reproduce |
| QA, DEV, PM | The bug fix has been postponed to the next version due to lack of urgency. | Postponed to next release |
| Dev | Fixing the bug requires in-depth investigation – it is a complex bug. Tracking and reproduction are needed, and the process may take longer than usual – approximately one to two weeks. | Monitor |

**Severity level definitions for bug defects:**

|  |  |  |
| --- | --- | --- |
| **Priority** | **Description** | **Bug severity** |
| Top priority, immediate handling of the defect | A very high severity defect that prevents the functioning of core components and the execution of fundamental processes. Failure to address this defect will result in a delay in the version release to the client. | Critical bug |
| Fix the defect before the start of the next testing cycle | A high severity bug is causing malfunction of the system. It is a noticeable issue from the client's perspective, such as an error message when launching the application. Even if usage can continue, a bug of this severity may lead to a delay or postponement of the version release. | High level bug |
| Fix the defect before the end of the next testing cycle | A medium severity defect causing partial malfunction of the system, while still allowing continued use. Such a defect may pose a risk to the version release. | Medium bug |
| Fix the bug in the future | A low severity bug – minor issues, usually in the graphical interface (or others that do not affect system functionality), but preferably should be fixed. The version will still be released even if the bug is not resolved before the end of the release process. | Low level bug |

**Tools for bug reporting and test management:**Version control, requirement entry, test writing, bug tracking, and report generation will be carried out using Microsoft Excel.

**Bug tracking and management:**

|  |  |
| --- | --- |
| **Responsibility** | **Bug severity** |
| Project team | Critical bug |
| Project team | High level bug |
| Project team | Medium bug |
| Project team | Low level bug |

**Test deliverables:**  
Test deliverables provided by the QA team to the project management, including documents, reports, and presentations.

|  |  |  |
| --- | --- | --- |
| **By** | **Delivery date** | **Document** |
| PM | 20/05/2025 | STP |
| QA manager | 25/05/2025 | STD |
| PM + R&D | 30/05/2025 | STR |

STD

User story: Admin adds a new company

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test** | **Test Description** | **Goal** | **Tester** | **Conditions** | **Flow** | **Expected Result** | **Passed/**  **Failed** |
|  |  |  |  |  |  |  |  |
| 1 | Testing that an Admin can add a new company. | Approve an Admin can add a new company. | Aviel | Admin account credentials, Android device and stable Internet connection. | 1) Log in WePool app with email = elioruzan06@gmail.com and Password = 123456.   2) In the Home page click on 'Role Selection' and then click on 'Admin'.  3) In the Admin Menu page click on 'Companies'.   4) In the page opened click the sign '+'.   5) Enter Company Name = 'Android' and Company Address = בוגרשוב 77 תל אביב and then click 'Confirm'. | The company added successfully, Firebase has been updated with the new company object, and it appears after the page had been auto - refreshed. | Passed |
| User story: Admin adds an HR manager to existing company |  |  |  |  |  |  |  |
| 2 | Testing that an Admin can add a new HR manager. | Approve an Admin can add a new HR manager. | Elior | Admin account credentials, Android device, stable Internet connection and an existing user with company code 'EBJQLC' (for which the company name = 'android'). | 1) Log in WePool app with email = elioruzan06@gmail.com and Password = 123456.   2) In the Home page click on 'Role Selection' and then click on 'Admin'.  3) In the Admin Menu page click on 'Companies'.   4) In the page opened click the admin icon.   5) In the popup window's top text field type in Shoshana Uzan and click 'Confirm'. | The HR manager added successfully, Firebase updated accordingly and after the page had been auto - refreshed and the tester clicks the admin icon - Shoshana Uzan appears under the current HR manager. | Passed |
| 3 | Testing that an Admin can filter rides by company name.  User story: Admin filters rides by company name | Approve an Admin can filter rides by company name. | Aviel | Admin account credentials, Android device, stable Internet connection and an existing company with rides. | 1) Log in WePool app with email = elioruzan06@gmail.com and Password = 123456.   2) In the Home page click on 'Role Selection' and then click on 'Admin'.  3) In the Admin Menu page click on 'Rides'.   4) expand Sort & Filter block on top and click 'Filter'. in the dropdown menu select 'Company Name'. type 'מכללת אפקה' in the box labeled Company and click 'Search'.  5) Compare the list of rides that appear with what Elior sent via WhatsApp on 9/6/25 at 9:28. | 1:1 match with the message sent via WhatsApp. | Passed |
|  |  |  | User story: Admin filters users by company name |  |  |  |  |
| 4 | Testing that an Admin can filter users by company name. | Approve an Admin can filter users by company name. | Aviel | Admin account credentials, Android device, stable Internet connection and existing company with users. | 1) Log in WePool app with email = elioruzan06@gmail.com and Password = 123456.   2) In the Home page click on 'Role Selection' and then click on 'Admin'.  3) In the Admin Menu page click on 'Users'.   4) expand Sort & Filter block on top and click 'Filter'. in the dropdown menu select 'Company Name'. type 'מכללת אפקה' in the box labeled Company and click 'Search' .  5) Compare the list of users that appear and their details with what Elior sent via WhatsApp on 9/6/25 at 9:45.  User story: HR manager activates a user | 1:1 match with the message sent via WhatsApp. | Passed |
| 5 | Testing that an HR manager can activate a new user. | Approve an HR manager can activate a new user. | Elior | HR manager accounts for credentials, Android device, stable Internet connection and a recently signed up user to the company to which the HR associated. | 1) Log in WePool app with email = uzanshoshana11@gmail.com and Password = 123456.  2) In the Home page click on 'Role Selection' and then click on 'Hr. manager'.   3) In the HR Manager Dashboard click on 'Manage Employees'.  4) expand Sort & Filter block on top and click 'Filter'. in the dropdown menu select 'Username'. type 'Sam Altman' in the box labeled Search by Name and click 'Search'.  5) Click the 'Inactive' button on the top right corner and activate the user. | The user has been activated and in his Firebase collection the field 'active' is assigned with true value. | Passed |
|  |  |  | User story: HR manager updates company's location |  |  |  |  |
| 6 | Testing that an HR manager can update the company's location. | Approve that an HR manager can update company's location | Elior | HR manager account credentials, Android device, stable Internet connection and an existing company. | 1) Log in WePool app with Email = uzanshoshana11@gmail.com and Password = 123456.  2) In the Home page click on 'Role Selection' and then click on 'Hr. manager'.   3) In the HR Manager Dashboard click on 'Manage Company'.  4) In the page opened click on 'Update Location'.  5) Type 'בני אפרים 226' in the box labeled New Location Address and click 'Save'.  User story: Driver creates a new ride | The company's location has been updated and in its Firebase collection (under ID = 1352e466-d16e-43c8-afe1-13e5165517  59) the field 'name' within location shows 'בני אפרים 226'. | Passed |
| 7 | Testing that a user in driver mode can create a new ride. | Approve that a user in driver mode can create a new ride. | Aviel | Activated user account with driver mode enabled Android device and a stable Internet connection. | 1) Log in WePool app with Email = Almoguzan07@gmail.com and Password = 123456.  2) In the Home page click on 'Role Selection' and then click on 'Driver'.   3) In the Driver Menu click on 'Create Ride'.  4) In the page opened choose direction 'To Work'.  5) Choose the Start location to be Home , Seats available = 3 and max detour = 10. set the date to be 01/07/25 and arrival time to be 10:00. | Ride was created successfully and is now accessible within Firebase ride collection where its fields match 1:1 to the user's input. | Passed |
| 8 | Testing that a user in driver mode cannot create a ride with a past departure time. | Making sure that ride creation is only for future dates. | Aviel | Activated user account with driver mode enabled Android device and a stable Internet connection. | 1) Log in WePool app with email = Almoguzan07@gmail.com and Password = 123456.  2) In the Home page click on 'Role Selection' and then click on 'Driver'.   3) In the Driver Menu click on 'Create Ride'.  4) In the page opened choose direction 'To Work'.  5) Choose the Start location to be Home, Seats available = 3 and max detour = 10. set the date to be 01/06/25 and arrival time to be 10:00. | Ride failed to create, a toast saying "Select later time" pops and Firebase ride collection remains untouched. | Passed |
|  | User story: Passenger joins a ride |  |  |  |  |  |  |
| 9 | Testing that a user in both driver and passenger modes cannot join his own ride. | Making sure join ride option is exclusively for others. | Elior | Activated user account with driver and passenger modes enabled Android device and a stable Internet connection. | 1) Log in WePool app with Email = Almoguzan07@gmail.com and Password = 123456.  2) In the Home page click on 'Role Selection' and then click on 'Driver'.   3) In the Driver Menu click on 'Create Ride'.  4) In the page opened choose direction 'To Work'.  5) Choose the Start location to be Home , Seats available = 3 and max detour = 10. set the date to be 01/07/25 and arrival time to be 10:00   6) Go back to home page, select passenger mode and click join a ride 'To Work'.  Set the start location to be Home and date and time as stated in step 5. click 'Show Available Rides'. | The ride previously proposed by the user in driver mode will not be accessible from passenger mode when wants to join a ride. | Passed |
| 10 | Testing that the ride matching algorithm finds rides within max detour deviation. | Making sure ride matching algorithm works flawlessly. | Aviel | Activated user account with passenger mode enabled Android device and a stable Internet connection. | 1) Log in WePool app with Email = test@wepool.com and Password = 12345 (same company as Almog Uzan).  2) In the Home page click on 'Role Selection' and then click on 'Passenger'.   3) In the Passenger Menu click on 'Join Ride'.  4) In the page opened choose direction 'To Work'.  5) Choose the Start location to be 'לוי אשכול 21 נתניה'. set the date to be 01/07/25 and arrival time to be 10:00 (We know that this address is within max detour deviation of the ride proposed in test 7). Click 'Show Available Rides'. | Ride created in test no. 7 appears as one of the search results for a ride. | Passed |
| 11 | Testing that a ride where the max available seats had been reached is no longer available. | Making sure ride matching algorithm works flawlessly. | Aviel | Activated user account with passenger mode enabled Android device, a stable Internet connection and a ride at full capacity. | 1) Log in WePool app with Email = test@wepool.com and Password = 12345 (same company as Almog Uzan).  2) In the Home page click on 'Role Selection' and then click on 'Passenger'.   3) In the Passenger Menu click on 'Join Ride'.  4) In the page opened choose direction 'To Work'.  5) Choose the Start location to be 'נחום 20 נתניה', set the date to be 01/07/25 and arrival time to be 10:00 (We know that the ride created in test 7 starts there, so it doesn't increment detour, and it's full). Click 'Show Available Rides'. | The ride created in test 7 doesn’t appear as one of the results. | Passed |
|  |  |  |  | User story: Driver cancels a home bound ride with passengers |  |  |  |
| 12 | Testing that a driver cannot cancel a homebound ride less than 10 minutes before the ride starts. | Making sure ride cancellation policy is applied properly. | Elior | Activated user account with Driver mode enabled Android device and a stable Internet connection. | 1) Log in WePool app with Email = Almoguzan07@gmail.com and Password = 123456.  2) In the Home page click on 'Role Selection' and then click on 'Driver'.   3) In the Driver Menu click on 'Create Ride'.  4) In the page opened choose direction 'To 'Home'.  5) Choose the Start location to be Home , Seats available = 3 and max detour = 10. set the date to be today and departure time 15 minutes ahead.  6) Simultaneously, log in WePool app with email = test@wepool.com and Password = 12345 (same company as Almog Uzan) and send request to that ride.   7) From Almog Uzan's account accepts that request. Wait 6 minutes! Then under Active Rides try to cancel the ride. | The ride failed to be cancelled, and an appropriate toast appears. | Passed |
|  |  |  | User story: User updates personal details |  |  |  |  |
| 13 | Testing that a user cannot update a password with less than 6 characters long. | Making sure compatibility with Firebase's authentication requirements. | Aviel | Activated user accounts, Android device and a stable Internet connection. | 1) Log in WePool app with email = Almoguzan07@gmail.com and Password = 123456.  2) In the Home page click 'Update Details'.  3) Set New Password and Confirm Password fields to '12345' and click Save and Exit. | Password fails to update; an error toast appears at the bottom. | Passed |
|  | User story: Inactive user tries to use the app |  |  |  |  |  |  |
| 14 | testing that an inactive user doesn't have access to the system's core functionalities. | Making sure only authorized users can access app services. | Aviel | Android device, existing company code and a stable Internet connection. | 1) In 'Welcome to WePool' page click Sign Up.  2) Type in your name and an arbitrary well formatted Email address in required fields.   3) Type in a password with at least 6 characters and confirm it where required. Choose your company code to be 'EBJQLC'. Select at least one role. Fill in other fields arbitrarily.  4) Click Register and accept T&C.  5) After being automatically navigated back to Login page, enter the credentials you have just created.  6) Click on 'Role Selection button'. | Navigation to Role Selection page fails and a toast saying the user must contact the HR manager to use the app is shown at the bottom. | Passed |
| User story: Driver approves a passenger's request to join a ride |  |  |  |  |  |  |  |
| 15 | Testing that a passenger receives a notification when his request being approved. | Making sure the impeccability of the notification handler. | Elior | Android device, 2 active users, existing ride and ride request and stable Internet connection. | 1) Log in WePool app with Email = Almoguzan07@gmail.com and Password = 123456.  2) In the Home page click on 'Role Selection'.  3) In the page opened click on 'Driver'.  4) In the Driver Menu click on 'Requests'.  5) Set the filter to 'Pending' and Refresh.  6) Look for a ride request made by Elon Musk on 01/07/2025, click the 'Pending' button and approve it.  7) Unlock the last phone you used to log in Elon Musk's account. | Notification from WePool app that the request had been approved appears at the status bar of the recently unlocked phone. | Passed |
| User story: Passenger cancels a work bound ride more than 60 minutes prior to departure time |  |  |  |  |  |  |  |
| 16 | Testing that a passenger can cancel a work bound ride more than 60 minutes prior to departure time. | Making sure WePool's cancellation policy applies. | Elior | Android device, 2 active users (at least 1 with passenger mode) , existing ride and approved ride request and stable Internet connection. | 1) Log in WePool app with Email = Almoguzan07@gmail.com and Password = 123456.  2) In the Home page click on 'Role Selection'.  3) In the page opened click on 'Passenger'.  4) in the Passenger Menu click on 'Active Rides'.  5) Click Search and Cancel the ride for 01/08/25 (there's only one ride and today is June 25). | The ride had been cancelled successfully and all parties involved got their relevant notifications. | Failed |
|  | User story: Driver views the planned route of his ride |  |  |  |  |  |  |
| 17 | Testing that a driver can view the planned route for his proposed rides. | Making sure the map interface works smoothly. | Aviel | Android device, active user with drive mode, existing ride and stable Internet connection. | 1) Log in WePool app with Email = president@gmail.com and Password = 123456.  2) In the Home page click on 'Role Selection'.  3) In the page opened click on 'Driver'.  4) in the Driver Menu click on 'Active Rides'.  5) Click Search and then click the Map icon in the ride card (only one exists for this user). | The planned route is displayed in a line, and Start location, Destination and Pickup stops if exist are highlighted with colors specified in the map legend. | Passed |
|  | User story: Driver calls a passenger in his ride |  |  |  |  |  |  |
| 18 | Testing that a driver can call a passenger in his ride. | Making sure the app interfaces smoothly with the phone's built in dialer app. | Aviel | Android device, 2 active users (at least one with driver mode), existing ride with passengers and stable Internet connection. | 1) Log in WePool app with Email = president@gmail.com and Password = 123456.  2) In the Home page click on 'Role Selection'.  3) In the page opened click on 'Driver'.  4) in the Driver Menu click on 'Active Rides'.  5) Click search and in the single ride card displayed click the 'Passengers' button.  6) Select Almog Uzan in the Ride Details alert dialog and click the 'Call' button. | Phone's dial app opens with the phone number displayed ready to be called. | Passed |
|  |  |  | User story: Driver starts a ride from within the app |  |  |  |  |
| 19 | Testing that a driver can start riding navigation using Google Maps. | Making sure smooth interfacing with Google Maps services. | Aviel | Android device, 2 active users (at least one with driver mode), existing ride with passengers and stable Internet connection. | 1) Log in WePool app with email = president@gmail.com and Password = 123456.  2) In the Home page click on 'Role Selection'.  3) In the page opened click on 'Driver'.  4) in the Driver Menu click on 'Active Rides'.  5) Click search and in the single ride card displayed click the 'Start' button. | Phone's local Google Maps app opens with the planned route and Relevant Pickup stops as specified in the ride card | Passed |
|  |  |  |  |  |  |  |  |
| 20 | Testing that a user can add Preferred locations. | Making sure Preferred Locations feature works smoothly. | Aviel | Android device and an active user. | 1) Log in WePool app with email = president@gmail.com and Password = 123456.  2) Click the 'Preferred Locations' button.  3) Click the '+' icon and in the 'Search Address' text fields and type in הנביאים 34 כפר סבא. | Preferred location added successfully and updated in the Firebase user's collection under that specific user ID | Passed |

User story: User adds a favorite location

1. " Carpooling in times of crisis: Organizational identification as a safety belt" - https://pmc.ncbi.nlm.nih.gov/articles/PMC9443075/#b0165 [↑](#footnote-ref-1)
2. "Carpooling could be the traffic solution we already have" - https://www.here.com/learn/blog/carpooling [↑](#footnote-ref-2)
3. " "אתם לא מדמיינים: הפקקים במרכז החמירו | זו המהירות הממוצעת, שהולכת ויורד-https://www.ynet.co.il/news/article/bybitkj4je [↑](#footnote-ref-3)