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 of 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 the minimum changes and maximum effect!**

Competitorreview

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 have to hire a driver or a shuttle services).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 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 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 entered details:
* 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 a 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.
* Updates ETA 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 scale 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 integrate with external APIs like Waze or Google Maps for real-time traffic data and route optimization.

**Security:**

* All data transmission must use 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 system "WePool" 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 an 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:

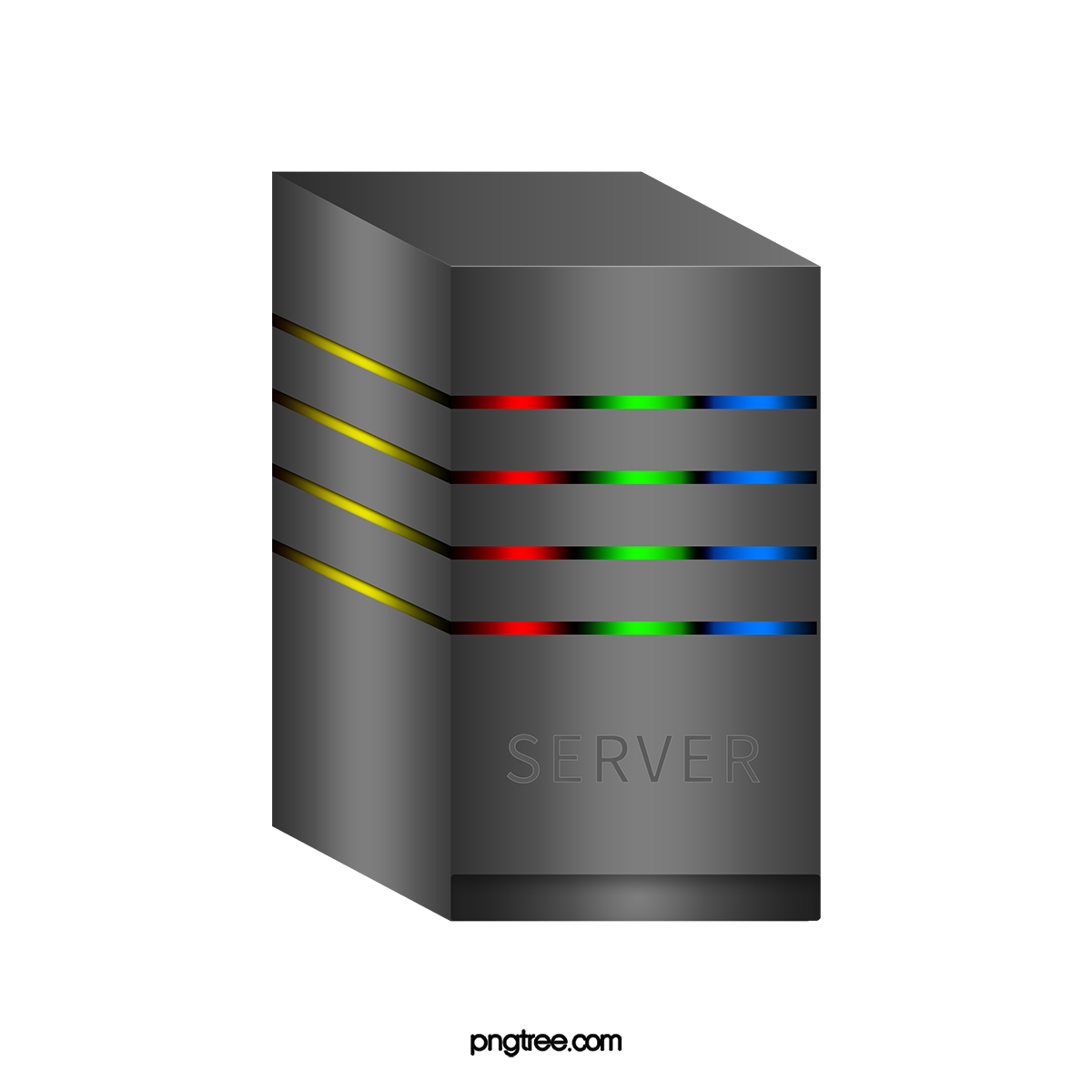
* Initial scalability is limited to 100 users per company.
* The backend can handle up to 100 concurrent requests in the initial phase.
* Daily API requests to Waze are limited (e.g., 1,000 calls/day during initial deployment).
* The database can initially store up to 100 GB of data, including user accounts,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

1. **System Architecture**

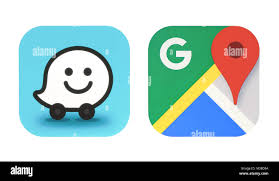
user



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.

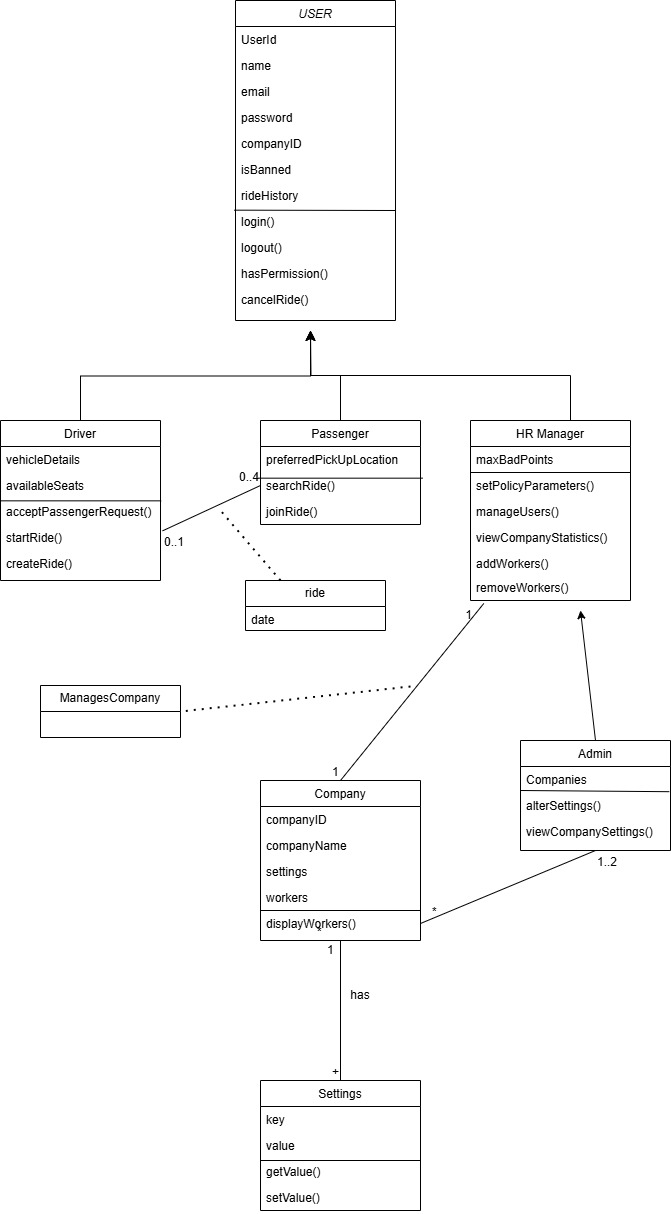
If the user is an HR manager, he navigates to the dashboard on the app where he can view and analyze statistics such as ride history for the company and pending requests.

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.



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)