Software Requirements Specification Document
Software Engineering

Team #16, The Chill Guys
Hamzah Rawasia
Sarim Zia
Aidan Froggatt
Swesan Pathmanathan
Burhanuddin Kharodawala

Table 1: Revision History

Date	**Developer(s)**	**Change**
October 9th	Swesan Pathmanathan	Added content for Goals section (G.1-G.7)
October 9th	Aidan Froggatt	Completed sections S.3, S.4, S.6
October 10th	Burhanuddin Kharodawala	Added Environment Sections (E.1 - E.6)
October 10th	Sarim Zia	Completed Project section (P.1 - P.7)
October 10th	Hamzah Rawasia	Added sections S.1, S.2, S.5
October 10th	All	Added Reflections, Finalized Document

1 (G) Goals

1.1 (G.1) Context and Overall Objectives

Commuting to McMaster University presents significant financial and environmental challenges. Students, staff, and faculty spend between 150 and 250 dollars monthly on fuel, parking, or transit, which places an economic burden on them. At the same time, single-occupancy vehicles contribute to 36 percent of greenhouse gas emissions for Canadian universities, harming both local air quality and broader sustainability goals. Existing ridesharing solutions such as Facebook Marketplace, Kijiji, or Poparide do not provide sufficient verification or security, leaving users at risk of riding with unaffiliated strangers. The objective of Hitchly is to provide a safe, affordable, and reliable ridesharing platform tailored for the McMaster community. By verifying users through McMaster email accounts, Hitchly ensures trust, reduces costs, supports sustainability, and helps the university meet its long-term community and environmental goals.

1.2 (G.2) Current Situation

Currently, McMaster commuters lack a centralized, trusted platform to connect drivers and riders within the university community. Those who rely on driving face steep monthly costs for gas and parking, while those taking transit encounter expensive or unreliable services. Informal ridesharing options exist, but they provide limited safety features and lack institutional alignment, making them unsuitable for building long-term trust. Students and staff often feel that available solutions are unsafe, inefficient, or financially burdensome. Hitchly seeks to fill this gap by providing a McMaster-specific application that facilitates secure ridesharing, reduces commuting costs, and lowers emissions.

1.3 (G.3) Expected Benefits

New processes, or improvements to existing processes, made possible by the project's results. It presents the business benefits expected from successful execution of the project. This chapter is the core of the Goals book, describing what the organization expects from the system [Meyer, 2022]. The pilot delivers clear, stakeholder-visible outcomes. Riders should see more affordable commutes (targeting approximately 20% savings) while drivers receive meaningful cost offsets (around 40%). We aim for reliable availability ($\geq$70% ride-fill and $\geq$95% on-time) so trips feel predictable from day one, with most new users completing a first ride within a week. The pilot targets a reduction in single-occupancy trips on campus and nearby (aiming for at least 10%), easing parking pressure and congestion and reducing emissions. The service will be inclusive by default (targeting about 90% match on stated accessibility preferences) and trusted through university-verified identities with a goal of very low incident rates. The university benefits from higher student satisfaction and sense of belonging, improved commuter equity, and better alignment with sustainability reporting—while maintaining privacy safeguards (goal: zero critical incidents). Broader community outcomes include cleaner air, calmer peak traffic, and greater participation in transit and shared mobility.

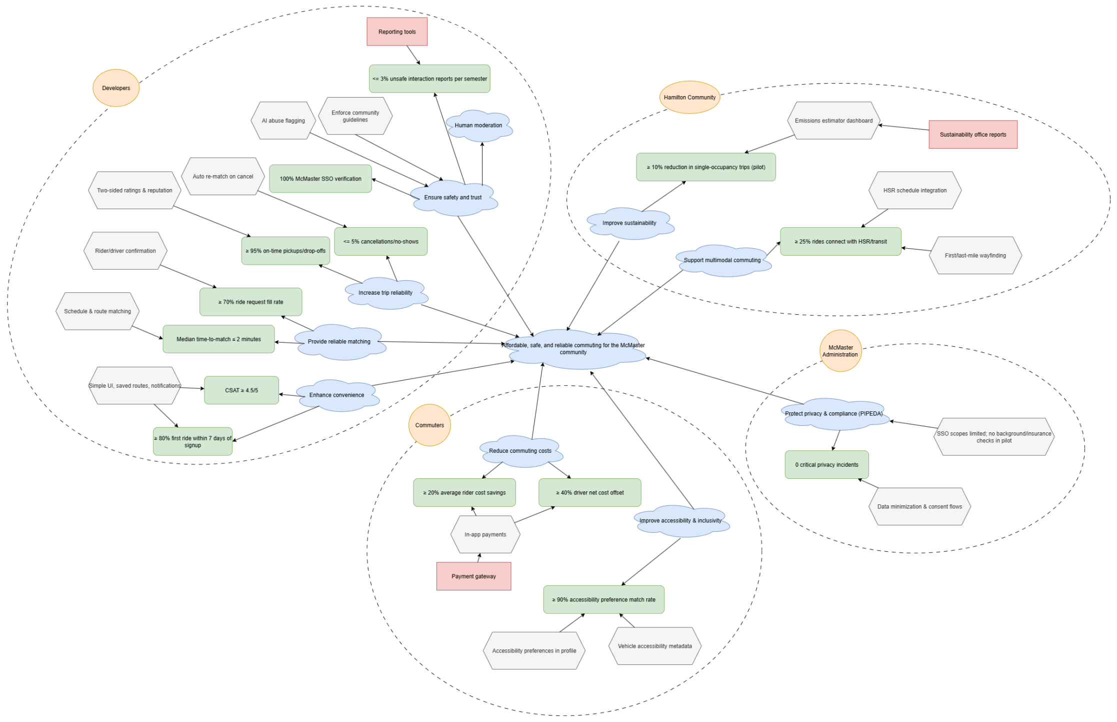

Reporting tools
<= 3% unsafe interaction reports per semester
Developers
AI abuse flagging
Enforce community guidelines
Human moderation
Auto re-match on cancel
100% McMaster SSO verification
Two-sided ratings & reputation
Ensure safety and trust
≥ 95% on-time pickups/drop-offs
<= 5% cancellations/no-shows
Rider/driver confirmation
Increase trip reliability
≥ 70% ride request fill rate
Schedule & route matching
Median time-to-match ≤ 2 minutes
Provide reliable matching
Affordable, safe, and reliable commuting for the McMaster community
Simple UI, saved routes, notifications
CSAT ≥ 4.5/5
Enhance convenience
≥ 80% first ride within 7 days of signup
Hamilton Community
Emissions estimator dashboard
Sustainability office reports
≥ 10% reduction in single-occupancy trips (pilot)
HSR schedule integration
Improve sustainability
Support multimodal commuting
≥ 25% rides connect with HSR/transit
First/last-mile wayfinding
McMaster Administration
Protect privacy & compliance (PIPEDA)
SSO scopes limited; no background/insurance checks in pilot
0 critical privacy incidents
Data minimization & consent flows
Commuters
Reduce commuting costs
≥ 20% average rider cost savings
≥ 40% driver net cost offset
In-app payments
Improve accessibility & inclusivity
Payment gateway
≥ 90% accessibility preference match rate
Accessibility preferences in profile
Vehicle accessibility metadata

Figure 1: Hitchly Goal Modelling Diagram.

1.4 (G.4) Functionality Overview

Overview. Hitchly is a campus-focused ride sharing service for the McMaster community. It verifies users by McMaster email, collects commute and vehicle data, matches riders and drivers by schedule and location, estimates cost sharing, and supports post-trip ratings.

Fundamental usage paths through the system, stated in user terms and independent of the system's structure. Detailed usage scenarios appear in the System book (S.4) [Meyer, 2022].

Functional requirements

1. **User verification.** At sign-up, users must verify a McMaster-affiliated email address. Drivers additionally upload a valid driver's license for verification before being allowed to post rides.This two-level process ensures that all users are affiliated with McMaster and that drivers are properly credentialed.
2. **Profile & schedule input.** Riders and drivers provide basic profile details plus commute timetable, typical routes/areas, and preferences. Drivers additionally record vehicle and license information to enable appropriate matching and accountability.
3. **Ride matching.** The system proposes compatible rider–driver pairings based on timetable overlap and approximate location. A match only becomes a trip after both sides confirm.
4. **Cost estimate.** For each proposed trip, Hitchly computes a per-rider cost share that helps drivers offset expenses and helps riders save compared to solo travel. Estimates use driver-provided cost inputs at a simple, transparent level.
5. **Ratings & reviews (extension).** After trips, users can leave ratings and short reviews to inform future choices and encourage good behavior. These signals improve decision-making over time while remaining out of scope for the initial PoC.

Most important two

User verification and **Ride matching** are most important. Verification directly addresses safety/trust, a primary adoption risk; matching is the core value creation that reduces search time and enables cost sharing. Together they validate the product's viability and de-risk the project early.

Non-functional requirements

1. **Reliability (matching correctness).** Under typical peak usage, the matching function should consistently produce correct, reproducible pairings for the same inputs. Incorrect or inconsistent matches are treated as defects and prioritized in testing.
2. **Usability (low-friction flows).** Common tasks (sign-up/verification, offering a ride, requesting a ride, confirming a match) should be understandable without training and completable in a small number of obvious steps. Clear labels and uncluttered presentation minimize onboarding time for first-time users.

RACI for requirements

Requirement	Dev Team	Drivers	Riders	McMaster Univ. / Hamilton Community
User verification	**R/A**	C	C	I
Profile & schedule input	**R/A**	C	C	I
Ride matching	**R/A**	C	C	I
Cost estimate	**R/A**	C	C	I
Ratings & reviews (extension)	**R/A**	C	C	I
Reliability (NFR)	**R/A**	C	C	I
Usability (NFR)	**R/A**	C	C	I

Legend: R = Responsible (executes the work); A = Accountable (final sign-off); C = Consulted (provides input); I = Informed (kept up to date).

1.5 (G.5) High-level Usage Scenarios

Use Case 1: Driver Offers Ride

1. A driver registers with their McMaster email and drivers license.
2. They input their schedule and vehicle details.
3. Hitchly suggests potential riders with overlapping times.
4. The driver confirms the riders they want to take.
5. The system calculates cost-sharing estimates.
6. After the trip, the driver receives ratings and reviews.

Use Case 2: Rider Finds Ride

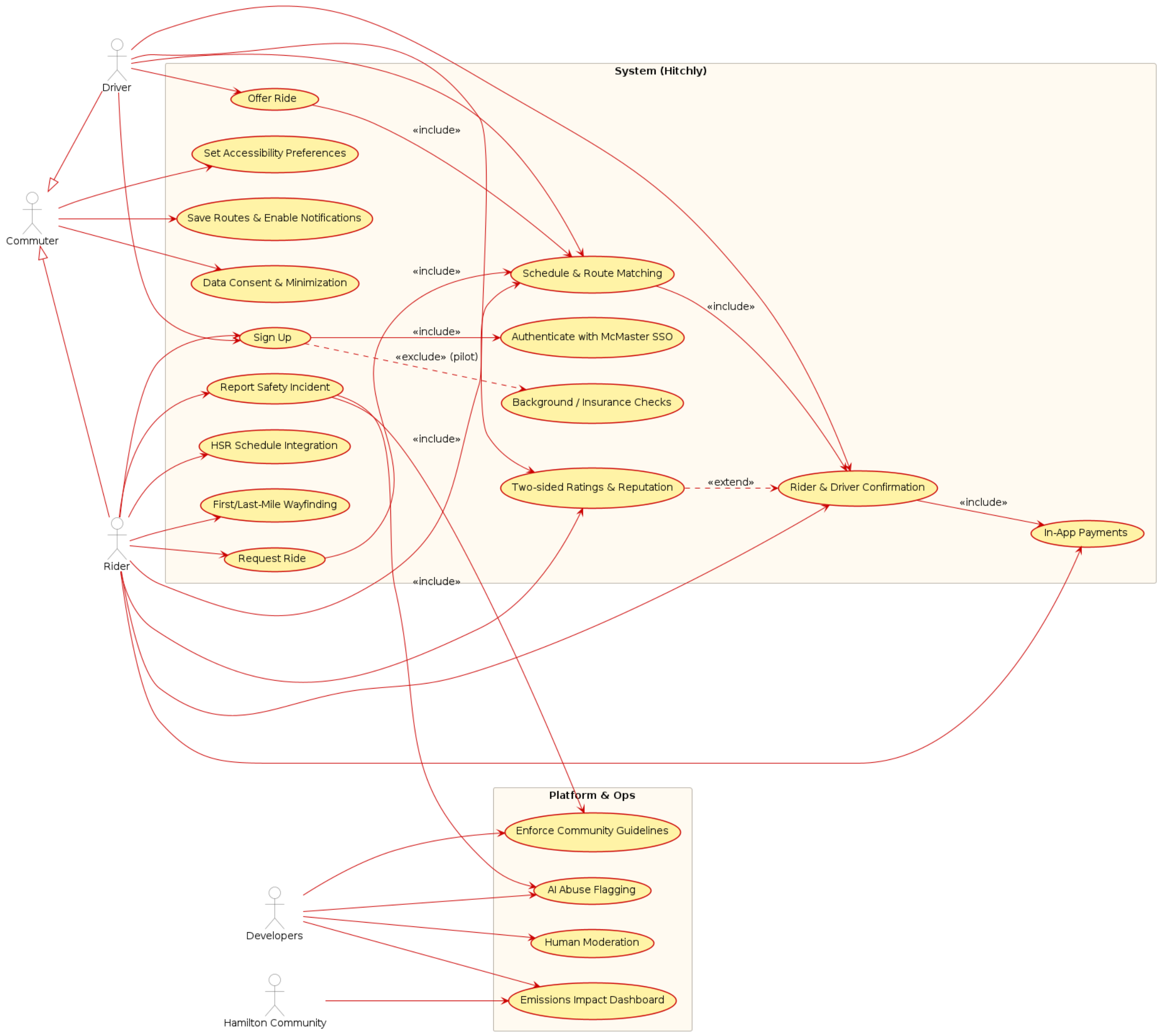
1. A rider signs up with a McMaster email.
2. They enter their daily commute times and location.
3. Hitchly presents a list of verified drivers with compatible schedules.
4. The rider selects a driver and confirms.
5. The system provides trip details, including cost share.
6. After the ride, the rider leaves a review.

Use Case 3: Report Safety Issue

1. A rider feels unsafe during a trip.
2. They submit a report through the in-app reporting tool.
3. Hitchly forwards the report to moderators.
4. The moderators review and investigate the case.
5. The driver's or rider's account is flagged if necessary.
6. The system tracks the resolution and stores it for accountability.

Use Case 4: Sustainability Tracking

1. Hitchly logs each completed trip.
2. The system calculates estimated emissions avoided.
3. Data is aggregated in dashboards.
4. McMaster administration reviews statistics to measure impact.
5. The results inform sustainability planning.
6. Reports can be shared with the broader community.



Driver
Commuter
Rider
Developers
Hamilton Community
System (Hitchly)
Offer Ride
Set Accessibility Preferences
Save Routes & Enable Notifications
Data Consent & Minimization
Sign Up
Report Safety Incident
HSR Schedule Integration
First/Last-Mile Wayfinding
Request Ride
Schedule & Route Matching
Authenticate with McMaster SSO
Background / Insurance Checks
Two-sided Ratings & Reputation
Rider & Driver Confirmation
In-App Payments
«include»
«include»
«include»
«exclude» (pilot)
«include»
«include»
«include»
«extend»
«include»
Platform & Ops
Enforce Community Guidelines
AI Abuse Flagging
Human Moderation
Emissions Impact Dashboard

Figure 2: Hitchly Use Case Diagram.

1.6 (G.6) Limitations and Exclusions

- McMaster-Only Access: Hitchly will only be available to McMaster students, staff, and faculty during the pilot phase. Users must verify their identity through a McMaster email. This restriction ensures trust and control but limits the platform's reach to a single institution.
- Background and Insurance Checks: For the pilot phase, Hitchly will not perform in-depth background checks or validate driver insurance. Verification is limited to McMaster email zn d driver license authentication. This keeps onboarding simple but places responsibility on users to ensure their own compliance with driving regulations.
- Limited Route Flexibility: Hitchly will not provide dynamic routing or live GPS tracking during the pilot. Riders and drivers must coordinate based on general pickup and drop-off locations. This simplifies the system but may limit convenience compared to fully featured ridesharing platforms.

1.7 (G.7) Stakeholders and Requirements Sources

Direct Stakeholders

Students They are frequent commuters with tight budgets who will use Hitchly to save money and connect with peers. Relevant because they face high monthly commuting costs.
Persona: "Sarah, a second-year student living off-campus, spends over $200 monthly on commuting. She uses Hitchly to reduce expenses and meet fellow students traveling the same route."

Staff Staff members travel daily to campus and benefit from both cost savings and reduced parking stress. Relevant because they often drive alone and bear significant expenses.
Persona: "James, a staff member living in Burlington, drives to McMaster five days a week. Hitchly allows him to share costs and avoid the hassle of finding parking."

Faculty Professors and teaching assistants need reliable commuting options. Relevant because faculty members often have predictable schedules that align well with ridesharing.
Persona: "Dr. Patel, a professor commuting from Toronto twice a week, finds Hitchly helpful to split costs and reduce environmental impact."

Drivers Any McMaster-affiliated individual who owns a car and can offer rides. Relevant because they reduce their commuting costs by sharing expenses.Drivers must verify their identity by uploading a valid license before offering rides.
Persona: "Ali, a graduate student with a car, uses Hitchly to cover his parking and gas by picking up two riders daily."

Riders Any McMaster-affiliated individual who does not own a vehicle. Relevant because they rely on others to get to campus affordably.
Persona: "Maria, an international student living in residence, relies on Hitchly to find rides for weekend trips to Hamilton downtown."

Indirect Stakeholders

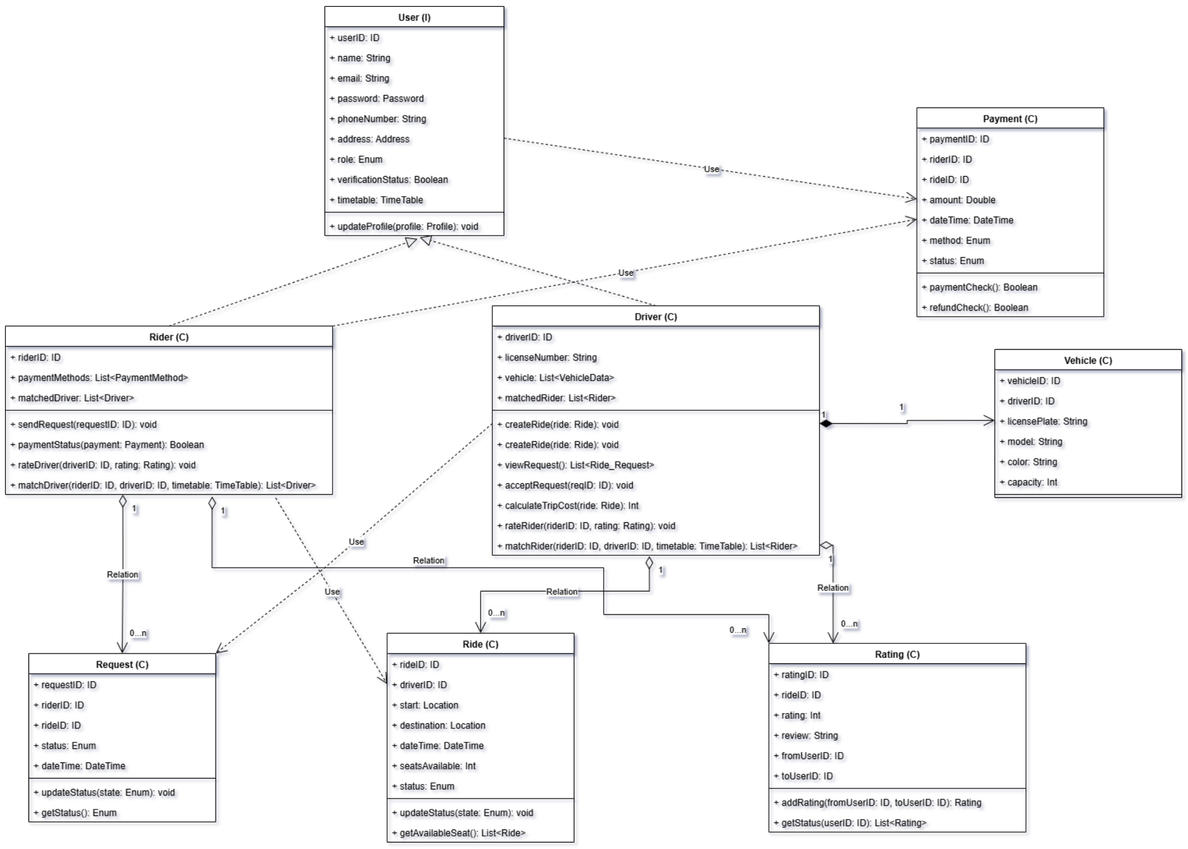
McMaster University Administration Gains reduced congestion and measurable sustainability improvements, aligning with institutional goals.

Hamilton Community Benefits from fewer cars on the road, reduced emissions, and improved air quality.

2 (E) Environment

2.1 (E.1) Glossary

- **Rideshare:** A service that allows riders to carpool with a single driver to reach a common destination.
- **Drivers:** They are users that will provide rideshare services with their personal vehicles.
- **Riders:** They are users that will carpool to and from McMaster University.
- **Working Condition:** A vehicle is in working condition if it has all necessary functions in the vehicle working. For instance, it must have functional head/tail lights, tires, engine, brakes, brake lights etc.



User (I)
+ userID: ID
+ name: String
+ email: String
+ password: Password
+ phoneNumber: String
+ address: Address
+ role: Enum
+ verificationStatus: Boolean
+ timetable: TimeTable
+ updateProfile(profile: Profile): void
Payment (C)
+ paymentID: ID
+ riderID: ID
+ rideID: ID
+ amount: Double
+ dateTime: DateTime
+ method: Enum
+ status: Enum
+ paymentCheck(): Boolean
+ refundCheck(): Boolean
Use
Use
Rider (C)
+ riderID: ID
+ paymentMethods: List<PaymentMethod>
+ matchedDriver: List<Driver>
+ sendRequest(requestID: ID): void
+ paymentStatus(payment: Payment): Boolean
+ rateDriver(driverID: ID, rating: Rating): void
+ matchDriver(riderID: ID, driverID: ID, timetable: TimeTable): List<Driver>
Driver (C)
+ driverID: ID
+ licenseNumber: String
+ vehicle: List<VehicleData>
+ matchedRider: List<Rider>
+ createRide(ride: Ride): void
+ createRide(ride: Ride): void
+ viewRequest(): List<Ride_Request>
+ acceptRequest(reqID: ID): void
+ calculateTripCost(ride: Ride): Int
+ rateRider(riderID: ID, rating: Rating): void
+ matchRider(riderID: ID, driverID: ID, timetable: TimeTable): List<Rider>
Vehicle (C)
+ vehicleID: ID
+ driverID: ID
+ licensePlate: String
+ model: String
+ color: String
+ capacity: Int
1
1
1
1
Relation
Use
Relation
Use
Relation
Relation
1
1
0...n
0...n
0...n
0...n
Request (C)
+ requestID: ID
+ riderID: ID
+ rideID: ID
+ status: Enum
+ dateTime: DateTime
+ updateStatus(state: Enum): void
+ getStatus(): Enum
Ride (C)
+ rideID: ID
+ driverID: ID
+ start: Location
+ destination: Location
+ dateTime: DateTime
+ seatsAvailable: Int
+ status: Enum
+ updateStatus(state: Enum): void
+ getAvailableSeat(): List<Ride>
Rating (C)
+ ratingID: ID
+ rideID: ID
+ rating: Int
+ review: String
+ fromUserID: ID
+ toUserID: ID
+ addRating(fromUserID: ID, toUserID: ID): Rating
+ getStatus(userID: ID): List<Rating>

Figure 3: Domain Model

2.2 (E.2) Components

1. **MFA (Multi-Factored Authentication) Services:** The application will be interacting with components that will allow the application to verify users with their McMaster University emails with SMS/Phone verification. This is to ensure reliability and user trust.

2. **Payments Interface (i.e, Credit Cards, PayPal, Apple Pay etc.):** The application will also interact with a payment interface to allow for a smooth and streamlined payment process with Stripe.
3. **Location API:** Google Maps will be used in several components of the application. This API will be used for location tracking, distance calculations, and for routing.

2.3 (E.3) Constraints

These are a few limitations and restrictions coming from the environment onto the system:

1. **Usage Constraint:** The project is currently limited and available to the McMaster community only. This would only include users that are currently students, staff members, or faculty members of the university. This application is currently limited to the Android and IOS operation systems. They will be compatible with any device with this operating system.
2. **Regulatory Constraint:** The project is constrained to comply with the PIPEDA laws and standards. This ensures that the user's data is handled carefully and with significant attention to maintaining data integrity and security. Each driver must have a Candian issued G2/G license, and the vehicle must be registered and insured. Each driver must comply with Ontario traffic rules and regulations. They must also not exceed the maximum passenger limit for their vehicle.
3. **Technical Constraints and Limitations:** A limitation of the Google Maps API is that it will be limited to providing minimal data for locations in rural areas. All functionalities in the applications must be completed in a limited time. For instance, the processing time for payments must be minimal.

2.4 (E.4) Assumptions

There are a few important assumptions that will be made to simplify the system. These will be assumed to be held, however, are not explicit constraints in the system.

1. **User Behavior:** All users of the rideshare application will not intentionally cause any harm to each other during a trip. They will not misbehave or litter in the car. This is an important assumption as it ensures the safety of users and simplifies the process of carpooling.
2. **Vehicle Condition:** Each driver has a vehicle that is in proper working condition. They are assumed to keep their vehicles clean. It will also be assumed that drivers will regularly maintain their vehicles.
3. **Language:** All users of the rideshare application must have a basic understanding of the English language, and they are able to communicate their thoughts with each other.
4. **User Base Demographic:** The application will assume a user base consisting of members of the McMaster University community. This may include students, staff, and faculty. This user group encompasses a diverse range of ages, genders, and ethnicities.
5. **Technical Compatibility and Reliability:** All third-party applications and API are assumed to be reliable. For instance, the Maps API will provide accurate data for all users in the GTA region. The API's rate limits will not exceed the expected user load. They are also assumed to have minimal downtime.

2.5 (E.5) Effects

Effects of the application include:

1. **Improved Sustainability and Reduced Traffic Congestion:** The system will positively affect the environment by lowering the GHG emissions caused by vehicles of individuals commuting to and from McMaster University. Secondly, as a result of carpooling, there will be a reduction in the use of single-occupancy vehicles which will cause a reduction in traffic congestion on roads and highways.
2. **Usage of Public Transit:** There will be a significant reduction in the usage of buses and trains to commute to McMaster University because of the transition of people moving towards using the rideshare application, Hitchly. It may become convenient and affordable for commuters to use the rideshare services. Thus, resulting in a potential reduction in the usage of public transit.
3. **Increase in Parking Spaces:** There can be an increase in available parking spaces due to the overall reduction of vehicles arriving at McMaster University. As a result of using the rideshare platform Hitchly, the number of single-occupancy vehicles will decrease. This would mean that fewer cars would arrive at McMaster University, reducing the overall demand for parking spaces.
4. **Increased Savings:** There can be an overall increase in the amount each commuter saves as a result of shifting to carpooling from other modes of transportation. Carpooling enables users to divide and share their cost, resulting in leaving them with more money in hand.

2.6 (E.6) Invariants

- Regular traffic will continue to move in Hamilton and surrounding areas where riders and drivers live. There can be traffic congestion on the roads due to car crashes and unusual weather conditions. There will also be times when an emergency vehicle may be on the roads occasionally, causing traffic congestion. These conditions will be present and will not change after the implementation of the application.
- The city of Hamilton will have and maintain their city infrastructure. There will be traffic lights, roads highways, pedestrian sidewalks, that will function normally. The traffic rules and signs will be followed and maintained regardless of the implantation of the application.
- McMaster Parking systems will function as usual. There will not be any changes made to the parking rules or guidelines to specifically cater to the rideshare application users.
- Wi-Fi networks will operate independently and can become slow due to torrential weather conditions. Users may be affected by this as it can affect their accessibility to a Wi-Fi network during their ride. Lastly, the Wi-Fi networks function independently of the implementation of the application.

3 (S) System

The System book refines the Goals book by focusing on more detailed requirements.

3.1 (S.1) Components

Hitchly is composed of 6 main components that collectively provide the necessary functionality required for the application.

Matchmaking Engine: The matchmaking engine is responsible for matching riders with drivers based on schedules/timetables, departure times, routes and locations, and other user preferences. It interfaces with the scheduling system to ensure rides can be matched, whether it be one-time or recurring. It will optimize for cost, convenience, and schedule alignment.

Scheduling and Booking Manager: Allows users to upload their schedules to inform about their availability to facilitate matchmaking. It also ensures booked rides are stored, updated, or cancelled as per defined policies.

Safety and Verification Module: This module provides secure login through McMaster email authentication. This ensures that only verified students and staff at McMaster may use the system, providing reassurance and safety for users and a sense of community.

Payment and Cost Sharing System: This system automatically calculates the cost of the ride for each rider based on the distance, number of passengers, gas, and parking costs. It manages in-app payment transactions and a transparent fee breakdown.

User Profile Manager: Stores user details such as a profile picture, bio, ratings, preferences, and more. Allows other users to view a profile when matching with each other to see information about them, as well as reliability metrics such as completed rides, any strikes, and more.

Rating and Reputation System: Tracks the attendance, punctuality, and safety related behaviour of drivers and riders. Issues strikes and penalties as needed, and integrates with the user profile manager to display user ratings and history.

The following diagram shows how these components interact and communicate when the application is being used.

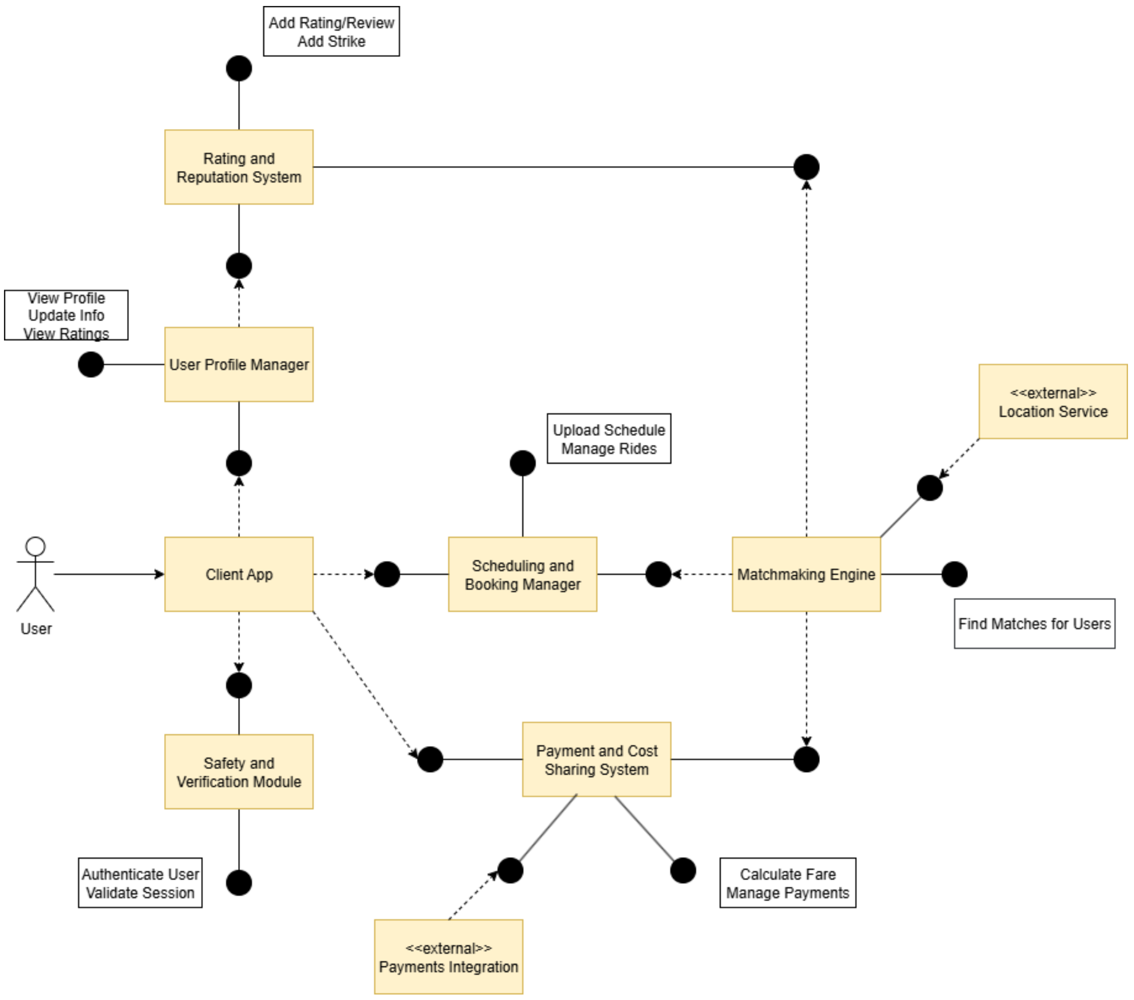


Figure 4: Hitchly Component Diagram.

3.2 (S.2) Functionality

Matchmaking engine

Functional Requirements:

1. Route and Schedule matching: The system shall compare driver routes and passenger locations to provide optimal ride matches.
2. Availability Updates: The engine shall update available rides in real time as users create, edit, or cancel rides.
3. Preference filtering: The engine shall allow users to filter matches by preferences, such as gender, music, vehicle type, etc.
4. Multi-passenger support: The system shall allow for multiple passengers to match to the same driver, based on driver preference and vehicle capacity.

Non-functional requirements:

1. Matching performance: The engine shall generate ride matches within 10 seconds of a request.
2. Location security: The engine shall ensure that user's information used for matchmaking is

not accessible outside of the development environment.

Scheduling and Booking Manager
Functional Requirements:

1. Ride scheduling: Users shall be able to schedule rides in advance based on the results of the matchmaking engine.
2. Booking confirmation: The system shall generate confirmations for bookings in the applications, and notify passenger(s) and the driver.
3. Cancellation and Rescheduling: Users shall be able to cancel or reschedule rides within the specified timeframe and rules.
4. Reminder notifications: The system shall send reminders to all passengers and the driver before any scheduled rides.

Non-functional requirements:

1. Data Integrity: All booking data shall be securely stored and updated as required.
2. Confirmation speed: Booking confirmations and changes shall be finalized and delivered to users within 5 seconds.

Safety and Verification Module
Functional Requirements:

1. User Identity Verification: The system shall validate users through university email authentication in order to use the application.
2. Driver Verification: The system shall store and verify driver license and vehicle registration information.
3. Ride Check-in: The system shall prompt passengers and drivers to check in at the start and end of the ride.
4. Safety check: The system shall keep track of any misconduct reported by users or any strikes on a user's account.

Non-functional Requirements:

1. Data Security: Data used for verification shall be encrypted to ensure security.
2. Auditability: The system shall maintain a log of all verification actions for at least 1 year.

Payment and Cost Sharing System
Functional Requirements:

1. Fare Calculation: The system shall automatically calculate the costs for the ride for each passenger based on distance, number of passengers, parking costs, etc.
2. Secure Transactions: The system shall support payment in the form of credit card, debit card, and other mobile wallet payment methods.
3. Hold on Funds: The system shall put funds on hold until after a ride has been completed, in order to deal with cancellations or refunds.
4. Expense Summaries: The system shall provide ride expense summaries and earnings reports for drivers.

Non-functional Requirements:

1. Transaction Security: Payments must be processed following security standards.
2. Processing Speed: Transactions shall complete within 10 seconds of initiation.

User Profile Manager
Functional Requirements:

1. Profile Creation: The system shall allow users to create profiles when signing up, including personal details and ride preferences.
2. Profile Updates: The system shall allow users to update their profile information at any time.
3. Ride History: The system shall store and display ride history and ratings for drivers and passengers.
4. View Profiles: The system shall allow users to view profiles of other users when matched with them.

Non-functional Requirements:

1. Privacy: Any personal information used when signing up shall not be visible to other users.
2. Performance: Profile data shall load within 2 seconds.

Rating and Reputation System
Functional Requirements:

1. Ride Ratings: The system shall allow users to rate each other after each ride.
2. Issue Reporting: The system shall allow users to report any issues or misconduct that occurs during a ride.
3. Minimum Reputation Thresholds: The system shall enforce a minimum rating for certain actions (e.g: Offering rides, hosting multiple passengers, joining scheduled rides, etc.).
4. Positive Reinforcement: The system shall reward users who consistently obtain high ratings by giving out badges, titles, and more.

Non-functional Requirements:

1. Rating Integrity: The system shall detect and flag fradulent and spam reviews.

3.3 (S.3) Interfaces

Hitchly exposes its functionality through both user-facing interfaces and programmatic interfaces (APIs). These define how students, staff, and faculty interact with the application and how Hitchly integrates with external systems.

3.3.1 (S.3.1) User Interfaces

(S.3.1.1) Mobile App (Primary Interface) Hitchly is delivered as a cross-platform mobile app (iOS and Android via React Native/Expo). Users can:

- Sign up and log in using McMaster email verification.
- Input personal data, commuting details, timetable, and location.
- Drivers can register their vehicles, upload license details, and set estimated trip costs.

- Riders can browse and request available rides that match their timetable and location.
- Both roles can view trip summaries, receive notifications, and use ratings/reviews.
- A clean, accessible UI built with NativeWind + shadcn/ui ensures usability.

(S.3.1.2) Notifications Push notifications (Expo notifications service) alert users when:

- A match is found.
- A ride is confirmed or updated.
- A trip is completed with a generated summary.

3.3.2 (S.3.2) Program Interfaces (APIs)

(S.3.2.1) Hitchly Public API

- Authentication API: Validates McMaster-affiliated emails and issues JWT tokens.
- User API: Manages user accounts, profiles, schedules, and roles (driver/rider).
- Matching API: Exposes endpoints for submitting commute data, retrieving matches, and confirming trips.
- Trip API: Provides trip summaries, cost breakdowns, and trip history.
- Ratings API: Allows posting and retrieving ratings/reviews.

(S.3.2.2) Third-Party Integrations

- Email Verification: Integrates with McMaster's email domain for signup restrictions.
- Mapping/Location Services: Uses a mapping provider (e.g., Mapbox or Google Maps API) to process geolocation and routing.
- Payment (future stretch): Possible integration with Stripe/PayPal for cost-sharing transactions.

Through these interfaces, Hitchly ensures safety (via controlled access), usability (via a student-focused app), and extensibility (via type-safe APIs).

Nothing available at this point.

3.4 (S.4) Detailed Usage Scenarios

This section provides example interactions between users and Hitchly. Scenarios are written as user stories and include normal, special, and erroneous cases.

3.4.1 (S.4.1) Rider Requests a Ride

1. Sarah, a second-year student, opens the Hitchly app and logs in with her McMaster email.
2. She inputs her class timetable and home address.
3. Hitchly suggests matching drivers commuting at similar times.
4. Sarah requests a ride with a compatible driver.
5. The driver accepts, and both receive a confirmation notification.
6. After the ride, Hitchly records the trip and generates a summary.

3.4.2 (S.4.2) Driver Posts Availability

1. Omar, a graduate student, registers as a driver by submitting his license and car details.
2. He enters his weekly commute schedule and estimated fuel/parking costs.
3. Hitchly calculates a per-rider cost and displays suggested prices.
4. Riders with compatible schedules can now request seats in Omar's car.
5. Omar accepts two riders and confirms the trip.

3.4.3 (S.4.3) Error Case — Invalid Email

1. A user attempts to register using a non-McMaster email.
2. The system rejects the signup and prompts them to use a valid @mcmaster.ca email.
3. The attempt is logged for auditing.

3.4.4 (S.4.4) Safety Case — Mismatched Data

1. A rider reports a driver whose profile information does not match their car/license.
2. Hitchly flags the account and prevents further matches until the issue is resolved.
3. Admins can review and verify the documentation.

3.4.5 (S.4.5) Ratings and Reviews

1. After a successful trip, riders are prompted to rate their driver on punctuality and safety.
2. Omar receives a 5-star rating and positive feedback.
3. His credibility score increases, making him more likely to be selected in the future.

These scenarios highlight both standard use and exception handling, supporting the goals of affordability, safety, and sustainability.

Nothing available at this point.

3.5 (S.5) Prioritization

Priority Table for Matchmaking Engine

Requirement ID	Requirement Name	Priority Level	Reasoning
F211	Route and Schedule Matching	Must Have	Core feature of the app's functionality.
F212	Availability Updates	Must Have	Essential for reliability and real-time operation.
F213	Preference Filtering	Should Have	Improves the user experience but not essential to core functionality.
F214	Multi-passenger Support	Must Have	Having multiple passengers per driver is essential.
NF211	Matching Performance	Should Have	Efficient algorithm and system is important for attracting users.
NF212	Location Security	Must Have	It is essential that the user's location information is kept secure.

Priority Table for Scheduling and Booking Manager

Requirement ID	Requirement Name	Priority Level	Reasoning
F221	Ride Scheduling	Must Have	Core functionality of the app.
F222	Booking Confirmation	Must Have	Required for user trust.
F223	Cancellation and Rescheduling	Must Have	Essential for smooth operation for all users.
F224	Reminder Notifications	Nice to Have	Improves reliability but not critical for functionality.
NF221	Data Integrity	Must Have	Needed for user trust and safety.
NF222	Confirmation Speed	Should Have	Users will rely on quick confirmation to assess functionality.

Priority Table for Safety and Verification Module

Requirement ID	Requirement Name	Priority Level	Reasoning
F231	User Identity Verification	Must Have	Required for the functionality and access of the app.
F232	Driver Verification	Must Have	Mandatory for trust and liability.
F233	Ride Check-in	Should Have	Ensures rides run smoothly but not essential.
F234	Safety Check	Nice to Have	Improves user safety and trust, but not critical for functionality.
NF231	Data Security	Must Have	Ensures that personal information is encrypted and securely stored.
NF232	Auditability	Nice to Have	Important for keeping track of user activity over time.

Priority Table for Payment and Cost Sharing System

Requirement ID	Requirement Name	Priority Level	Reasoning
F241	Fare Calculation	Must Have	Core functionality of cost-sharing.
F242	Secure Transactions	Must Have	Essential for user trust and legal compliance.
F243	Hold on Funds	Must Have	Needed for smooth functionality for all users.
F244	Expense Summaries	Nice to Have	Helpful for users to track budgets but not core functionality.
NF241	Transaction Security	Must Have	All payments must be handled securely for safety and legality.
NF242	Processing Speed	Should Have	Users will want a smooth process when completing payments.

Priority Table for User Profile Manager

Requirement ID	Requirement Name	Priority Level	Reasoning
F251	Profile Creation	Must Have	Needed when signing up to use the application.
F252	Profile Updates	Must Have	Important for accuracy of information and preferences.
F253	Ride History	Should Have	Builds trust and data availability but does not directly improve functionality.
F254	View Profiles	Should Have	Important for user safety and trust when booking rides.
NF251	Privacy	Must Have	Protecting personal information is necessary for user confidence.
NF252	Performance	Nice to Have	Users will want fast navigation through information.

Priority Table for Rating and Reputation System

Requirement ID	Requirement Name	Priority Level	Reasoning
F261	Ride Ratings	Must Have	Critical component of the rating system.
F262	Issue Reporting	Must Have	Important for evaluating and ensuring the safety of all users.
F263	Minimum Reputation Thresholds	Should Have	Enforces repurcussions for behaviour that goes against set rules, but not critical for functionality.
F264	Positive Reinforcement	Nice to Have	Provides benefits to users who consistently use the app as intended, but also not essential.
NF261	Rating Integrity	Should Have	Important to protect users reputations when possible.

The prioritization of requirements mainly focused on their importance to the app's core functionality, along with the safety and security of users and their data. The core functionality refers to features that directly affect how users engage with Hitchly on a regular basis, such as finding and booking rides, managing payments, creating an account, etc. Requirements that are central to these experience are classified as "Must Have". Other features that provide significant improvements to convenience, trust, or improved user experience are categotized as "Should Have". Finally, features that offer additional ease of use or extra value but are not critical to the app's purpose are placed in the "Nice to Have" category.

3.6 (S.6) Verification and Acceptance Criteria

Verification and validation (V&V) ensure Hitchly meets stakeholder expectations for usability, safety, and functionality. The following criteria define when the system will be considered satisfactory.

3.6.1 (S.6.1) Module Testing

(S.6.1.1) Authentication Module

- Accepts only @mcmaster.ca emails.
- Issues secure JWT tokens for valid accounts.
- Rejects invalid domains with appropriate error messages.

(S.6.1.2) Matching Engine

- Correctly pairs riders and drivers based on timetable and location inputs.
- Handles edge cases (no matches, multiple matches) gracefully.

(S.6.1.3) Trip Summary

- Generates accurate summaries with cost estimates.

- Stores records in PostgreSQL without data loss.

3.6.2 (S.6.2) Integration Testing

- End-to-end tests ensure the mobile frontend communicates correctly with backend APIs.
- McMaster email verification integrates seamlessly with signup flow.
- Map/location services provide accurate routing data.

3.6.3 (S.6.3) System Testing

- Hitchly runs on iOS and Android devices without critical crashes.
- UI components are accessible (WCAG AA compliance).
- Notification services deliver timely alerts for matches and trip updates.

3.6.4 (S.6.4) User Acceptance Testing (UAT)

- Conducted with a group of McMaster students and staff.
- Success criteria:
  - Users can sign up, request rides, and complete a trip without assistance.
  - At least 80% of participants report satisfaction with ease of use and reliability.
  - No major safety concerns raised during trial.

3.6.5 (S.6.5) Static Analysis and Code Quality

- ESLint and Prettier pass on all commits.
- TypeScript compiler passes with no errors.
- Code coverage of unit tests $> 70\%$.

3.6.6 (S.6.6) Acceptance Criteria Summary

Hitchly will be deemed acceptable if:

- All core modules pass automated and manual tests.
- Integration tests confirm reliable operation across components.
- UAT demonstrates usability and trustworthiness.
- The system supports the project's goals of affordability, safety, and sustainability.

4 (P) Project

4.1 (P.1) Roles and Personnel

The team will consist of the following roles which have been pre-assigned based on qualifications:

- **Project Manager (Swesan):**
  - Oversee overall progress and deadlines
  - Coordinate with TA and professor if needed
  - Handle document submission
- **Frontend Developer(s) (Burhan, Sarim):**
  - Create mobile interfaces using design tools (Figma, Adobe XD)
  - Develop user interface for the application using chosen development tools
  - Integrate frontend UI with backend APIs
- **Backend Developer(s) (Aidan, Hamzah, Sarim):**
  - Develop server-side logic, APIs, and database connections
- **Database Engineer (Burhan, Swesan):**
  - Design database schema
  - Optimize database queries
- **Tester(s) (All team members):**
  - Write and conduct test cases
  - Document bugs and performance issues
  - Collect user feedback

*****All team members will be working on documentation*****

4.2 (P.2) Imposed Technical Choices

The following imposed technical choices will apply:

- **Programming Language & Framework:**
  - Must use languages/frameworks that all team members are familiar with.
  - Mobile framework should be cross-platform to allow for both iOS and Android usage (React Native).
  - Development machines will primarily run macOS and Windows; compatibility testing will be conducted for both.
- **APIs / External Services:**
  - Must use an external Maps API for routing and geolocation.
  - Must use McMaster servers for hosting and storage (if permitted).
- **Collaboration:**
  - All code will be managed through GitLab repositories with protected main branches.

  - Each feature will be developed in separate branches and merged via pull requests after peer review.

4.3 (P.3) Schedule and Milestones

The project development will be divided into multiple sprints, each focusing on specific objectives and deliverables. This sprint-based approach ensures iterative progress and early testing of key functionalities.

Sprint 1: Project Foundations (October)

- Finalize the technology stack for the application.
- Set up the development environment and ensure the mobile application environment runs on all developers' hardware.
- Create Figma wireframes to map out the application flow.
- Design the initial database architecture for how information will be stored.

Sprint 2: Create MVP (November)

- **Epic 1:** Implement fundamental features of the carpool application.
  As a user, I want to be able to connect with another user who has a similar schedule and a relevant commute route.
- **Epic 2:** Implement basic payment features.
  As a user, I want to be able to complete a carpool session and split parking or fuel costs through the application.
- **Minimal Viable Product (MVP):**
  The proposed Sprint 2 focuses on foundational features crucial for delivering a functional beta version of the application. The epics in this sprint implement Hitchly's core functionality — carpool matching and basic payment — allowing for an initial user testing phase to validate the core system logic.

Sprint 3: Core System Expansion (December)

- **Epic 3:** Add user verification.
  As a user, I want to ensure all application users are verified through their university email.
- **Epic 4:** Add multi-passenger support.
  As a driver, I want to be able to connect to multiple passengers based on my vehicle capacity.
- **Epic 5:** Add complex fare calculation.
  As a user, I want the system to calculate the fare for each passenger based on gas cost, number of passengers, and parking.
- **Epic 6:** Add ride check-in.
  As a user, I want drivers and passengers to be prompted to check in at the start and end of each ride.

Sprint 4: Profile Addition (January)

- **Epic 7:** Add profile system.
  As a user, I want to create a personal profile to store my details and ride preferences.
- **Epic 8:** Add profile updates.

As a user, I want to be able to update my personal information at any time.

- **Epic 9:** Add filtering to matching.
  As a user, I want to filter matches by preferences such as gender, music, or vehicle type.
- **Epic 10:** Add rating system.
  As a user, I want to rate and review other users that I carpool with.

Sprint 5: Final Feature Additions (February)

- **Epic 11:** Add ride scheduling.
  As a user, I want to schedule my ride in advance.
- **Epic 12:** Add booking confirmation.
  As a user, I want to receive a confirmation for my ride booking and notify other users part of the same ride.
- **Epic 13:** Add reminder notifications.
  As a user, I want to receive reminder notifications for my scheduled ride as the time approaches.
- **Epic 14:** Improve rating and review system.
  As a user, I want the review system to influence user trust scores and matching priority.

Sprint 6: Testing and Finalization (March – April)

- Conduct unit, integration, and system-wide testing across all modules to ensure stability and reliability.
- Optimize application performance and fix any remaining bugs identified during testing.
- Conduct user acceptance testing (UAT) with a pilot group and incorporate feedback.
- Prepare final presentation materials, poster, and demonstration for the April capstone showcase.

4.4 (P.4) Required Technology Elements

- **Compliance with Payment API:**
  - The application requires a secure payment gateway API that will be integrated with Hitchly to allow passengers to pay drivers for their carpool sessions.
  - This technology will enable the secure processing of payments, transactions, and the generation of billing information.
- **Compliance with Maps API:**
  - The application relies on navigation and location data to match users for carpooling based on their current location and routes to the university.
  - This will require full integration with a Maps API that allows precise user tracking, location comparison, and real-time route navigation across the campus area.
- **Authentication API:**
  - The application requires a secure authentication system that can reliably handle user login details.
  - This API will be integrated with Hitchly to manage user access, verification, and login sessions.

- **Database Management System:**
  - A reliable DBMS is essential for storing and managing user accounts, trip history, and audit logs for safety purposes.
  - This system will comply with the Hitchly application requirements and will be responsible for secure data storage and retrieval.

4.5 (P.5) Risk and Mitigation Analysis

Some of the risks present during development are listed below:

- **Low User Adoption:** The concept of carpooling with other students may not initially be popular, as many students may prefer driving alone or using public transportation. This could lead to low user adoption, especially during the testing phase when beta testers are needed to identify bugs and provide feedback. Since Hitchly relies on a large user base to strengthen its matchmaking algorithm, a limited number of users could reduce match quality. One mitigation strategy for this risk is to begin recruiting beta testers early and develop promotional material highlighting the benefits of carpooling. Starting outreach early would help increase awareness and build a stronger user base to test the application effectively.
- **Inaccurate Campus Data:** The application utilizes on-campus information such as parking costs and parking lot locations. Any changes to this information during the course of development could lead to inconsistencies or inaccurate data within the application. A mitigation strategy for this risk is to periodically check in with McMaster Parking Services to ensure that all campus-related data remains accurate and up to date.
- **Lack of User Trust with Data:** Some users may hesitate to trust a student-developed application with their personal and payment information. This lack of trust could negatively affect user adoption and engagement. A mitigation strategy for this risk is to integrate reliable third-party payment APIs, such as Stripe, to handle transactions securely. Additionally, maintaining transparency with users about which API services and security measures are implemented will help build trust and confidence in the system.

4.6 (P.7) Requirements Process and Report

The requirements process will involve interaction with direct stakeholders identified in Section G.7. For each stakeholder, the type of interview, its objective, and the most important question to be asked are listed below.

- **Student (Sarah):**
  - **Type of interview:** Closed. The objective is to understand the willingness of students to participate in carpooling.
  - **Question:** *"What factors would motivate you to use a student-based carpooling service instead of public transportation or driving alone?"* We aim to identify which features would most effectively encourage user adoption.
- **Staff (James):**

  - **Type of interview:** Open. The objective is to understand how staff members currently commute to work.
  - **Question:** *"What is your current mode of transportation to campus?"* This question will help determine how large the potential staff user base is and identify commuting challenges the app could address.
- **Faculty (Dr. Patel):**
  - **Type of interview:** Closed. The objective is to understand what benefits faculty members might expect from using a carpooling application, as financial incentives may not be their primary motivation.
  - **Question:** *"What benefits would you expect from using a university-based carpooling application compared to your current commute method?"* This will help gauge which features are most valued by faculty members.
- **Driver (Ali):**
  - **Type of interview:** Open. The objective is to understand what would motivate drivers to participate as drivers rather than passengers.
  - **Question:** *"What incentives would encourage you to become a driver in a carpooling application?"* This question will help determine which incentives—such as cost-sharing or community benefits—are most appealing to drivers.
- **Rider (Maria):**
  - **Type of interview:** Open. The objective is to understand any concerns passengers may have when using a carpooling application.
  - **Question:** *"What are some concerns you may have about using a carpooling application that matches you with other users?"* This will help identify safety measures and system features that can address potential user concerns.

5 Reflection

1. What went well while writing this deliverable?

   Burhan:
   The team had a clear idea of what we were trying to achieve with this document. The template is organized very well. It covers all of the concepts that need to be discussed for requirements. Moreover, the team was very active in brainstorming ideas for certain sections (i.e, function requirements, invariants, etc.). Lastly the instructions provided for most sections were very clear and detailed, which helped me understand and write the document easily. Sarim:

   Our team used the last document as a learning process to build off on what we did wrong. We made sure to hold frequent group sessions to ensure we were all on the same page. This was extremely important for a document like the SRS as there are many different sections that require team members to stay on the same page and work towards one specific goal. Hamzah:

   I believe our team did a good job of dividing the work evenly so we are each doing sections that relate to each other. Also, we got started early on this deliverable so everyone had adequate time to complete their sections. The meyer template was easy to follow once it was set up, allowing our work to be very organized. Swesan:

   Since I am currently taking 3RA3 (Requirements), much of the content felt familiar and directly applicable. This helped me understand the structure of the deliverable and made writing easier, since I could connect course concepts with what we were expected to produce. It also gave me confidence that our sections were aligned with best practices. Aidan:

   What went well for me was how well our group collaborated and communicated throughout the process. Everyone contributed meaningfully and was responsive to feedback, which made integrating sections much smoother. I also found that using the Meyer template made it easier to stay consistent and structured, since it clearly outlined what was expected for each section. Writing my portion went smoothly because of the clear division of work and early start—we avoided last-minute stress and had time to review everything together before submission.

2. What pain points did you experience during this deliverable, and how did you resolve them?

   Burhan:
   For some sections, the instructions were a little ambiguous. For example, the environment section, the instructions for assumptions and constraints were very similar, making it a bit unclear on how to differentiate between those two sections. Moreover, the SRS and the Hazard Analysis document both had a section for listing assumptions. This caused some confusion while writing the deliverable as I was unsure of which section to provide more detail. I was able to resolve this by asking detailed and specific questions about the ambiguity and confusion with the sections and instructions to the TA during our informal discussions.
   Sarim:

Since there was not one speciic SRS document to pick from, there was alot of decision making involved in terms of which sections were important for the rubric, and which information needed to be on the Hazard Analysis document rather than the SRS or vice versa. It was extremely useful for us to utilize our TA meeting and ask questions regarding any ambiguity we faced, and to contiously review the rubric. Hamzah:

The initial set up of the document was confusing at first since there are several SRS template options to choose from, all of which have different formats and content. Additionally, there seemed to be many sections that were redundant and/or repetitive between sections and documents, which required clarification from the TA during our meeting. This includes questions such as what diagrams are required, how many requirements are needed, and more. Swesan:

One of the main challenges was making the diagrams and ensuring everyone's parts of the document were consistent in style and content. At first, this caused some confusion and extra editing, but I resolved it by refining the diagrams carefully and coordinating with the team to keep formatting and tone aligned across sections. Aidan:

A challenge I faced was ensuring consistency between sections, especially when multiple people were editing the same parts or referencing the same requirements differently. Early drafts had minor overlap or conflicting terminology, which took some coordination to resolve. I also found it difficult to interpret some of the expectations for specific subsections at first, since the template can be dense. I overcame this by meeting with teammates frequently to align on definitions and by reviewing past examples and TA feedback to make sure our document followed the expected structure.

3. How many of your requirements were inspired by speaking to your client(s) or their proxies (e.g. your peers, stakeholders, potential users)?

   The stakeholders for our project are McMaster students and faculty that commute to campus, so this would include our team members and many of our peers. By discussing our pain points and desires for this application with other students who are commuters, we were able to get a better understanding of what our application needs to include to meet their standards and expectations. This includes many of the features mentioned in section S.2, since this covers what is expected for each major component in the system.

4. Which of the courses you have taken, or are currently taking, will help your team to be successful with your capstone project.

   One of the most helpful courses for the capstone project so far has been SFWRENG 3RA3 (Software Requirements and Security Concerns), as the main project in this course was following the Meyer Tenplate to write a requirements document. This was extremely useful for this deliverable in particular, but also for all documentation deliverables in general since we got experience and learned key concepts which can be carried over. Other helpful courses include SFWRENG 2AA4, SFWRENG 3BB4 (Software Design I and II) as they gave experience working in a group on a development project, and learning key software engineering principles which will be helpful for the development of our application.

5. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

   Burhan:
   I have minimal experience working on building a full stack application. There are a few things that I would need to acquire knowledge about. Our application will be based on a stack including React Native, TypeScript, and PostgreSQL. I have a lot of experience working with SQL; however, I haven't worked with PostgreSQL before. This is an area in which I need to do some research and practice. Moreover, as this course involves a lot of documentation, I also need to acquire some knowledge within this area to strengthen my grammar and formal writing skills.

   Sarim:
   Since my experience has mainly been in web development, I will need to focus my learing on mobile development. We chose React Native based on our skillset, since alot of us had experience with React.JS in web development so the skill would be somewhat familiar to us. This choice still requires me to learn a new skill but since both are comparitively similar, it makes for an easy transition. Hamzah:

   One of the main skills I will need to learn is TypeScript, as we plan to use it to develop the application. I have experience with languages like JavaScript and other typed languages like Java, however I have not used TypeScript before, so this will be a new experience for me which I will need to take time to learn and get familiar with to be able to contribute better to the development of the application. Swesan:

   For me, the most important skills are collaboration, coding, and motivation. Since this is a team-based project, being able to work effectively with others and keep everyone on track will be essential. I also want to improve my technical coding contributions especially with type script while maintaining the drive to push the project forward.

6. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

   Burhan:
   For PostgreSQL, I will read the documentation provided by the official website and then

proceed to watch YouTube tutorials on the basics of how the technology works. The next step would be to learn how to use it efficiently in a full stack environment. For this as well, I will watch YouTube tutorials. For my writing skills, I will go over some formal grammar documentations online. Moreover, I will also practice what I have learned with grammatical exercises found online. Sarim:

To learn mobile development in React Native, my initial plan is to follow a youtube tutorial that goes through the fundamentals of framework. Once I learn the fundamentals, I will take some time to create simple mobile applications and utilize documentation to support my development. I believe that to apply any skill you need to be fully understand the fundamentals. By watching an initial tutorial, I would truly understand what I need. Being able to work on simple applications will help reinforce the fundamentals and learn through making mistakes. Hamzah:

In order to familiarize myself with TypeScript, I plan to read documentation and go through video tutorials to get a good understanding of the fundamentals and how it can be used. Additionally, once I have a grasp of the basics, I will attempt practice questions and/or small applications to apply my understanding and improve. Being able to follow along is useful, but actually applying the knowledge is the best way to learn. Swesan:

For collaboration and motivation, two approaches are: (1) actively communicating with teammates and holding regular check-ins, and (2) staying accountable by being on top of my own tasks and deadlines. For coding, I will continue practicing through team coding sessions and peer reviews. I plan to pursue these approaches because working closely with my team and staying consistent with my responsibilities will naturally build both my technical and teamwork skills.