Development Plan
ProgName

Team #, Team Name
Student 1 name
Student 2 name
Student 3 name
Student 4 name

Table 1: Revision History

Date	**Developer(s)**	**Change**
Date1	Name(s)	Description of changes
Date2	Name(s)	Description of changes
...	...	...

[Put your introductory blurb here. Often the blurb is a brief roadmap of what is contained in the report. —SS]

[Additional information on the development plan can be found in the lecture slides. —SS]

1 Confidential Information?

[State whether your project has confidential information from industry, or not. If there is confidential information, point to the agreement you have in place. —SS]

[For most teams this section will just state that there is no confidential information to protect. —SS]

2 IP to Protect

[State whether there is IP to protect. If there is, point to the agreement. All students who are working on a project that requires an IP agreement are also required to sign the "Intellectual Property Guide Acknowledgement." —SS]

3 Copyright License

[What copyright license is your team adopting. Point to the license in your repo. —SS]

4 Team Meeting Plan

[How often will you meet? where? —SS]

[If the meeting is a physical location (not virtual), out of an abundance of caution for safety reasons you shouldn't put the location online —SS]

[How often will you meet with your industry advisor? when? where? —SS]

[Will meetings be virtual? At least some meetings should likely be in-person. —SS]

[How will the meetings be structured? There should be a chair for all meetings. There should be an agenda for all meetings. —SS]

5 Team Communication Plan

[Issues on GitHub should be part of your communication plan. —SS]

6 Team Member Roles

[You should identify the types of roles you anticipate, like notetaker, leader, meeting chair, reviewer. Assigning specific people to those roles is not necessary at this stage. In a student team the role of the individuals will likely change throughout the year. —SS]

7 Workflow Plan

- How will you be using git, including branches, pull request, etc.?
- How will you be managing issues, including template issues, issue classification, etc.?
- Use of CI/CD

8 Project Decomposition and Scheduling

- How will you be using GitHub projects?
- Include a link to your GitHub project

[How will the project be scheduled? This is the big picture schedule, not details. You will need to reproduce information that is in the course outline for deadlines. —SS]

9 Proof of Concept Demonstration Plan

The primary risks for this project are user trust and safety and the reliability of the ride-matching system. Without a secure verification process tied to McMaster email accounts, users may hesitate to adopt the platform. Similarly, if the matching algorithm fails to correctly pair riders and drivers based on schedules and locations, the application will not provide value. The proof of concept will focus on validating these two core capabilities:

- **McMaster Email Verification**
  - Users must sign up using a McMaster-affiliated email.
  - Verification ensures that only students, staff, and faculty can access the platform.
  - Demonstrates credibility and mitigates safety concerns for early users.
- **Ride Matching Prototype**
  - A minimal matching engine takes sample schedules and trip data from riders and drivers.

  - Produces correct pairings with cost-sharing estimates.
  - Confirms feasibility of integrating timetable and location-based logic.

By validating these areas first, the team ensures the platform addresses its two biggest risks of safety and usability before expanding into additional features such as live tracking, queuing, or ratings.

10 Expected Technology

The project will be built as a **cross-platform mobile application** for iOS and Android. The technology stack covers the full lifecycle: ideation, design, development, testing, deployment, and app store release.

a. Concept Ideation & Design

- **Wireframing & Prototyping:** Figma for user flows, wireframes, and visual design.
- **Collaboration & Task Management:** GitHub Projects for task tracking, milestones, and agile planning.

b. Frontend (Mobile App)

- **Framework:** React Native with Expo (cross-platform).
- **Language:** TypeScript for static typing and type safety.
- **UI & Styling:** Tailwind CSS via NativeWind and shadcn/ui for consistent styling.
- **Navigation:** React Navigation for multi-screen flows.
- **State Management & Data Fetching:** React Query integrated with tRPC client for type-safe server communication.

c. Backend & API

- **Framework:** tRPC running on Node.js (standalone server serving the mobile app).
- **Language:** TypeScript (shared between frontend and backend).
- **Hosting:** Railway for backend server and PostgreSQL database (low cost, minimal maintenance).
- **Authentication:**
  - McMaster email domain verification at signup.
  - JWT for secure session handling.
- **Core Services:** Matching engine, ride scheduling, and trip storage.

d. Database & Persistence

- **Database:** PostgreSQL for structured relational data.
- **ORM:** Drizzle ORM for schema management and migrations.
- **Hosting:** Managed PostgreSQL via Railway.

e. Testing & Quality Assurance

- **Unit Testing:** Jest for frontend and backend logic.
- **Integration Testing:** Supertest for backend endpoint verification.
- **Static Analysis:** ESLint and Prettier for linting and formatting.

f. Development Workflow & CI/CD

- **Version Control:** Git + GitHub repository.
- **Branching Strategy:** Feature branches with pull requests.
- **CI/CD:** GitHub Actions for automated testing, linting, type-checking, and deployments.
- **Environments:**
  - Development: Local Expo Go + Railway database.
  - Staging: Railway preview deployment + Expo EAS preview builds.
  - Production: Stable backend deployments + app store builds.

g. Deployment & Hosting

- **Backend Deployment:** Railway containerized Node.js hosting.
- **Database Deployment:** Railway managed PostgreSQL.
- **Frontend Builds:** Expo EAS Build for iOS and Android binaries.
- **App Store Distribution:**
  - Apple App Store via Expo EAS.
  - Google Play Store via Expo EAS.

11 Coding Standard

To maintain **clarity, maintainability, and collaboration**, the team will adopt the following coding standards:

Style & Formatting

- TypeScript with consistent naming conventions and modular architecture.
- ESLint and Prettier enforce linting and formatting across the codebase.

Documentation

- JSDoc comments for functions, APIs, and modules.
- Inline comments for complex logic (e.g., ride-matching algorithm).

Version Control Practices

- Git with feature branches for all new work.
- Pull requests with mandatory peer review before merging.
- All commits to main branch must be squashed and merged (enforced via GitHub rulesets).
- Descriptive commit messages using Conventional Commits format.
- GitHub Projects for issue tracking, milestones, and labeling (bug, feature, documentation, etc.).

Testing Practices

- Unit tests for core features (authentication, ride matching).
- Integration tests for backend endpoints.

These standards ensure the codebase is **professional, maintainable, and scalable**, meeting rubric expectations for a senior design project.

Appendix — Reflection

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. **Why is it important to create a development plan prior to starting the project?**
   Creating a development plan is important because it establishes structure, expectations, and direction for the team before work begins. It ensures that roles, responsibilities, communication methods, and goals are clearly defined, which reduces confusion later in the project. It also helps identify risks early and provides a roadmap that keeps the team accountable and aligned.

2. **In your opinion, what are the advantages and disadvantages of using CI/CD?**
   The main advantages of CI/CD are faster development cycles, early detection of bugs, and consistent integration of code, which reduces the likelihood of major conflicts at later stages. It also enforces good coding practices and increases overall software reliability. However, disadvantages include the setup time and learning curve required, especially for a student project, as well as the overhead of maintaining pipelines and infrastructure when the team may already be managing heavy coursework.

3. **What disagreements did your group have in this deliverable, if any, and how did you resolve them?**
   Not in this deliverable in particular, but earlier we had a disagreement regarding the proposal idea for our capstone project. After debating with facts and reasoning, we collectively concluded that pursuing the rideshare project would be the most impactful and feasible direction for our team.

Appendix — Team Charter

External Goals

Our external goals for this project are to develop an application that delivers meaningful real-world impact that also extends beyond the scope of capstone. We aim to achieve a high grade in this course, higher than a 10, while also competing at the Capstone EXPO and presenting our work to the McMaster community. In addition, we intend for this project to serve as a strong portfolio piece, highlighting our technical, design, and collaborative skills to future employers. Finally, we recognize the potential for this project to evolve into a startup, enabling it to create value and impact beyond the university setting—potentially the next big ride app competing against Uber and Lyft.

Attendance

Expectations

Our team expects all members to treat meetings as a priority and respect one another's time by being punctual and engaged. Attendance is essential to keep the project on track, and clear communication is expected whenever conflicts arise.

- Members should arrive on time; being more than 10 minutes late without notice is discouraged, as it is disruptive and shows a lack of respect for everyone else's time.
- No more than 2 unexcused absences are permitted per term.
- Leaving early or joining late is only acceptable if communicated beforehand.
- Members are expected to be fully present during meetings and avoid unrelated activities.
- Anyone who misses a meeting is responsible for reviewing notes and following up to stay up to date.

Acceptable Excuse

- Transit delays, weather issues, or unforeseen personal emergencies (as long as the team is notified).
- Illness, family emergencies, or direct academic conflicts such as exams or required presentations.
- Overlapping academic or work responsibilities, or unavoidable scheduling conflicts that are shared in advance.

- Urgent situations (e.g., emergency calls or messages) that require brief attention, but members should minimize distractions.

Note: There is no acceptable excuse for failing to review notes and catch up after missing a meeting — this is always the member's responsibility.

In Case of Emergency

If a team member experiences an emergency and cannot attend a meeting or complete their assigned work:

- They must notify the team as soon as possible through the primary communication channel (e.g., Discord/Teams).
- No personal details need to be shared beyond confirming that it is an emergency.
- The member should communicate how much of their work has been completed and what remains outstanding.
- If the task cannot be finished in time, the member should either:
  - Delay completion and commit to catching up later, or
  - Transfer responsibility to another team member, providing enough context or materials to allow for a smooth handoff.
- If responsibility is transferred, the original member should agree on how they will make up their contribution in a future task to keep workloads fair.

Accountability and Teamwork

Quality

Our team expects all members to come prepared for meetings, with assigned tasks completed to a high standard. Deliverables should be accurate, polished, and ready for integration or review. Members are encouraged to review one another's work to ensure consistency and to help catch errors early. The expectation is to go beyond simply completing tasks, aiming instead to produce work that reflects professionalism and attention to detail.

Attitude

Team members are expected to maintain a positive and professional attitude, respecting all contributions and encouraging open discussion. All ideas will be heard and considered, but discussion should remain solutions focused. Disagreements will be handled constructively, using a clear process: clarify the issue, listen to all perspectives, establish common ground, and agree on a resolution.

Stay on Track

Our team will stay accountable by setting measurable targets and monitoring progress regularly. Each member is expected to contribute fairly and consistently so that the workload is balanced, and deadlines are met.

Metrics:

- **Attendance:** Members are expected to attend at least 90% of scheduled meetings.
- **Commits:** Each member should make regular commits (minimum 2 per week), reflecting steady progress.
- **Issues:** All members must contribute to opening, discussing, and closing GitHub Issues, with at least 1–2 issues closed per week per person.
- **Documentation & Reviews:** Members are expected to contribute to documentation and review teammates' work at least once per deliverable.

Rewards:

- The top contributor for each milestone (measured by commits, issues closed, and meeting contributions) will be recognized and treated to a free lunch by the team.
- Members who consistently meet or exceed expectations may receive lighter workloads during peak deadlines as recognition.

Consequences:

- Members falling short of expectations will first receive a reminder and the chance to catch up.
- If underperformance persists, the issue will be discussed with the TA or instructor and reflected in peer evaluations.
- For minor misses (e.g., attendance or commits), fun consequences may be applied, such as bringing coffee/snacks to the next meeting.

Team Building

Cohesion will be maintained through open communication, supportive collaboration, and celebrating milestones together. Occasional informal check-ins will help strengthen relationships and ensure a positive team culture.

Decision Making

Decisions will aim for consensus whenever possible. If consensus cannot be reached, the team will rely on rational, objective, and fact-based decision making rather than subjective opinions. Evidence, data, and alignment with project goals will guide choices. As a last resort, majority voting will be used.