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

[Not required for CAS 741 —SS]

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?

2. In your opinion, what are the advantages and disadvantages of using CI/CD?

3. What disagreements did your group have in this deliverable, if any, and how did you resolve them?

Appendix — Team Charter

[borrows from University of Portland Team Charter —SS]

External Goals

[What are your team's external goals for this project? These are not the goals related to the functionality or quality fo the project. These are the goals on what the team wishes to achieve with the project. Potential goals are to win a prize at the Capstone EXPO, or to have something to talk about in interviews, or to get an A+, etc. —SS]

Attendance

Expectations

[What are your team's expectations regarding meeting attendance (being on time, leaving early, missing meetings, etc.)? —SS]

Acceptable Excuse

[What constitutes an acceptable excuse for missing a meeting or a deadline? What types of excuses will not be considered acceptable? —SS]

In Case of Emergency

[What process will team members follow if they have an emergency and cannot attend a team meeting or complete their individual work promised for a team deliverable? —SS]

Accountability and Teamwork

Quality

[What are your team's expectations regarding the quality of team members' preparation for team meetings and the quality of the deliverables that members bring to the team? —SS]

Attitude

[What are your team's expectations regarding team members' ideas, interactions with the team, cooperation, attitudes, and anything else regarding team member contributions? Do you want to introduce a code of conduct? Do you want a conflict resolution plan? Can adopt existing codes of conduct. —SS]

Stay on Track

[What methods will be used to keep the team on track? How will your team ensure that members contribute as expected to the team and that the team performs as expected? How will your team reward members who do well and manage members whose performance is below expectations? What are the consequences for someone not contributing their fair share? —SS]

[You may wish to use the project management metrics collected for the TA and instructor for this. —SS]

[You can set target metrics for attendance, commits, etc. What are the consequences if someone doesn't hit their targets? Do they need to bring the coffee to the next team meeting? Does the team need to make an appointment with their TA, or the instructor? Are there incentives for reaching targets early? —SS]

Team Building

[How will you build team cohesion (fun time, group rituals, etc.)? —SS]

Decision Making

[How will you make decisions in your group? Consensus? Vote? How will you handle disagreements? —SS]