# Software Requirements Specification

# PRJ566 – Winter 2025

# PRJ566 – Team No: 06 (SUBTEAM-01)

# Name of Project: GO Transit App

# Project Leader: Paras Singh

**Last updated: 2025/04/13**

**Team Members:**

* **Paras Singh**
* **Khwahish Vaid**

# TABLE OF CONTENTS

1. **Introduction/Overview - Document Information**
   1. Document Authors
   2. Revision History
   3. Document Conventions
   4. Document Purpose
   5. Intended Audience
   6. Group Agreement
2. **Project Overview**
   1. Project Proposal
   2. Stakeholders and Users
   3. Project Scope
   4. System risks
   5. Operating Environment
   6. Functional Requirements
   7. Nonfunctional Requirements
   8. UI/UXD Interface Mockups
3. **Process & Data Modeling**
   1. UML Modeling: DFDs & Activity Diagrams
   2. Use Case Specification
      1. Business Rules
      2. System Use Case Diagrams
      3. Use Case Description Tables
4. **Domain Class Diagram**
5. **Database (Select either 5.1 or 5.2)**
   1. RDBMS Artifacts
      1. Scripts to create, populate, delete tables
      2. Data Dictionary
   2. NoSQL Artifacts
6. **Work breakdown Structure (WBS)**
7. **Milestones & Acceptance Criteria**
8. **Implementation Schedule (Agile/Waterfall)**
9. **Client / Faculty Sign-off**

# 1 - Introduction/Overview - Document Information

## Document Authors

* The Vinh Tran
* Amiel Thompson

## 1.2 Revision History

|  |  |
| --- | --- |
| **Week 03** | **Sections of this document that were completed/updated this week**  **"Software Requirements Specification”**  **Section 1 (1.1 to 1.6)**  **Section 2 (2.1 only)** |
| **Week 04** | **Stakeholders' List**  **Project Scope**  **System Risks**  **Operating Environment** |
| **Week 05** | **Functional Requirements Nonfunctional Requirements Data Flow Diagrams (DFDs)** |
| **Week 06** | **Activity Diagram**  **Prototype: UI/UXD - Wireframes/Mock-ups** |
| **Week 07** | **User Stories and related use case specifications with corresponding interface mock-ups including:**  **Business Rules.**  **System Use Case Diagrams**  **Use Case Descriptions** |
| **Week 08** | **Mock-ups/wireframes** |
| **Week 09** | **Domain Class Diagram**  **Team Prototype Presentations** |
| **Week 10** | **NoSQL Data Models and corresponding Documents** |
| **Week 11** | **Work Breakdown Structure (WBS)  Project Milestones & Acceptance Criteria** |
| **Week 12** | **Product Backlog (GitHub)** |
| **Week 13** | **Develop the final video presentation and completed SRS** |
| **Week 14** | **Teams’ presentation day in class** |
| END |  |

## 1.3 Document Conventions

This section outlines the conventions used throughout the Software Requirements Specification (SRS) document to ensure consistency and clarity:

* **Text Color**:
  + **Red**: Indicates an exception or error that requires immediate attention.
  + **Orange**: Marks content that is currently in progress or under review.
  + **Green**: Represents recently added text or updates made during the latest revision.
  + **Yellow Highlight**: Draws attention to important points or key takeaways.
* **Text Style**:
  + *Italicized Text*: Denotes definitions or terminology that require emphasis.
  + ~~Strikethrough~~: Represents deleted or obsolete text that is no longer valid.
* **Annotations**:
  + Comments or notes added by reviewers are enclosed within square brackets [].

## 1.4 Document Purpose

The purpose of this Software Requirements Specification (SRS) is to outline the requirements and development process for the **GO Transit App**, a mobile solution aimed at enhancing the public transit experience of commuters and travelers in the Greater Toronto Area. This document serves to:

1. Specify objectives of the app (e.g. travel planning optimization, and real-time updates).
2. Specify the technical and functional requirements in detail to direct development.
3. Document the project’s ongoing progress and encourage team members and stakeholders to collaborate.
4. Describe the project's concept, development, and testing in a two-semester roadmap.
5. By putting deliverables and milestones in writing, accountability is encouraged and documented.

The project's goal of enhancing GO Transit's user experience is in line with this SRS document.

## 1.5 Intended Audience

The intended audience of this document includes but is not limited to the following:

* Stakeholders
* Sponsors
* Project manager
* Business analysts
* Developers
* Designers
* Testers
* QA team
* Support team
* Maintenance team
* End users

## 1.6 Group Agreement

**TEAM AGREEMENT**

**Team #:** 6

**Project Title**: GO Transit App

**Project Time Frame:** Winter 2025 – Summer 2025

**Team Members:**

* The Vinh Tran
* Amiel Thompson

**Team Leadership:** The Vinh Tran

**Team Functions:**

* *We will share information through MS Teams, OneDrive, and meetings.*
* *Project milestones are to be managed via the team’s internal GitHub repo*

**Team Meetings:** Weekly meetings via MS Teams, with additional meetings when necessary.

**Team Problems**: All conflicts will be addressed constructively through open discussion during meetings.

If unresolved, issues will be escalated to the project leader for a decision,

**Team Commitment**

The undersigned members agree to work together on the project until the end of the PRJ666 next Semester. They recognize that as a team and individually they are responsible for the quality of all deliverables.

**Name Date**

|  |  |
| --- | --- |
| The Vinh Tran | 26 jan 2025 |
| Amiel Thompson | 26 jan 2025 |
|  |  |
|  |  |

ShapeShapeShapeShapeShapeShapeShapeShape

# 2 - Project Overview

## 2.1 Project Proposal

**Project Background**

GO Transit is an essential public transportation service in the Greater Toronto Area. The current in-browser applications for GO Transit are inefficient in aligning with real-time needs of daily commuters. This project aims to develop a dedicated GO Transit mobile app to help & enhance travel experience, streamline transit planning in real- time, and improve accessibility for all riders.

**Problem Statement**

|  |  |
| --- | --- |
| The Problem of: | Lack of user-friendly mobile apps with essential features for GO Transit services like real-time updates, smart route planning and other integrated features in one place |
| Affects: | Metrolinx, Transit Operators and Staff, Local Government & Urban Planners (Stakeholders) Commuters (End Users) |
| The impact of which is: | Delays & difficulty in trip planning, navigating multiple and complex routes, managing time-sensitive trips, frustrations in trip decisions |
| A successful solution would: | Provide seamless, user-friendly real-time schedule, navigation and updates,  Optimized route suggestions,  Personalized usability and habits,  Improved accessibility |

**Product Vision**

|  |  |
| --- | --- |
| For | Daily commuters, occasional riders, first time riders, tourists |
| Who | Need a reliable, efficient, and user-friendly way to navigate GO Transit’s extensive network |
| The Product Name | GO Transit (Is a Mobile App) |
| That | Provides real-time updates, optimized route planning & schedule, quick trip tracking, ticketing, and accessibility features in one place |
| Unlike | GO Transit’s official in-browser apps, Google Maps, Transit app, whose features and integration with GO’s services can be confusing, lacking or unfriendly to riders |
| Our product | Delivers a tailored, easy-to-use transit experience by integrating accessible, real-time trip planning and guidance, ensuring a more reliable and convenient journey for all riders using the GO system |

## 2.2 Stakeholders and Users

|  |  |
| --- | --- |
| **Stakeholder Name/Identifier** | **Category** |
| Metrolinx | Primary Sponsor: Owns GO Transit services; provides data, funding, and strategic guidelines |
| Commuters (End Users) | End User: Daily riders relying on the app |
| Government Transport Agencies | Regulatory/External: Ensure compliance with accessibility standards, data privacy laws, and transit policies. |
| GO Transit Operations Team | Internal User: Manages transit schedules, delays, and service updates |
| Third-Party Payment Providers | External Partner: Fare payment integration (e.g., Presto, credit/debit cards). |
| Q/A Testing team | Developers, Internal Users: Validates app performance, security, and usability before release |
| Developers | Developer/Technical: Designs, develops, and maintains the app’s functionality and infrastructure |

## 2.3 Functional Requirements

#### **Trip Planning** Users can enter their starting point and destination, and the app will suggest the best routes based on travel time, number of transfers, and other preferences. It’ll also show estimated travel times, waiting times, time travelling on foot and provide step-by-step directions, making it easy to navigate even for new riders. Users can also specify their own schedule (go now, arrive by or start at)

* User enters and edits destination location
* User edits origin location
* User chooses their schedule timing
* System shows suggested trips
* System updates suggested trips’ time-sensitive information in real-time (animated and colour-coded based on urgency)
* User inspects individual suggested trips for detailed routes, stops, transfers and timing
* User starts a trip that is best suited to their travel needs

#### **Real-Time Transit & Trip Tracking**

The app will provide live updates on train and bus locations, estimated arrival times, and any delays. Users will be able to see exactly where their vehicle and on-foot location are on a map, get real-time notifications for destination stops or issues, and plan accordingly.

Users will be able to select a trip based on the app’s suggested routes and “Start” the trip to begin trip tracking in a higher priority – the app will send notifications and announcements regarding the trip as user journeys to their destination.

* User “starts” a trip to begin tracking their journey
* System displays trip timeline that tracks and updates trip progression
* System updates time-sensitive information, and either displays or issues notifications about warnings, delays or changes to a trip based on a priority system
* System announces notifications when user approaches a route’s destination or final stop by default

The app should aim to have intuitive navigation between trip inspections, trip selection and trip searching with seamless interactions.

#### **Trip QuickStart**

* User selects quick destinations from the home screen (E.g. home, work, school)
* User customizes quick destinations on the home screen via the settings menu
* System finds and suggests trips for quick-start destinations
* User selects nearby transit routes from the home screen
* System shows all the stops from the route selected
* User quick-selects a stop from the selected route to start their trip

#### **Offline Mode**

For times when users don’t have internet access, the app will allow them to save their favorite routes and schedules. This way, they can still check trip details without needing to be online. The app will update saved information automatically whenever an internet connection is available.

#### **Voice Commands**

To improve accessibility and usability, the app will support voice commands. Users will be able to perform route and trip finders using voice commands.

#### **Fare Calculator & Payment Integration**

The app will include a fare calculator so users can quickly check how much their trip will cost. In future updates, it will also support payment integration, allowing users to link their Presto cards or other digital payment options directly in the app.

#### **Alerts & Notifications**

The app will send real-time service alerts, reminders for upcoming departures, and notifications about any delays. Users can customize which types of alerts they want to receive.

#### **Accessibility Features**

To ensure that everyone can use the app, it will include accessibility features like voice-over support for visually impaired users, high-contrast mode for better readability, and text-to-speech options for reading schedules aloud.

* User accesses and turns on accessibility modes that are not supported by default in the settings menu

#### **Multi-Modal Transit Support**

Sometimes, one form of transit isn’t enough. The app will help users combine different types of transportation—buses, trains, and subways—into a single, seamless trip plan. It’ll show estimated wait times, transfer points, and total travel time for a hassle-free commute.

#### **User Profiles & Trip History**

Users will have the option to create an account where they can save frequently used routes, view past trips, and customize their preferences. This will make it easier to quickly check schedules and plan future trips.

#### **Customer Support & Feedback**

If users run into issues or have suggestions, they’ll be able to contact customer support directly through the app. There will also be a feedback feature where they can report details and issues during a trip to help other riders.

A rating system should be incorporated into the app to collect user’s feedback after each trip

* User rates their experience using the app after a trip
* User asks to not show the trip rating menu again

## 2.4 Nonfunctional Requirements

|  |  |
| --- | --- |
| **Requirement type** | **Requirement** |
| Operational | * The app should be compatible for both iOS (15.0+) and Android (11+) * The app should be written in React Native (and/or SwiftUI for native iOS deployment) * UI design’s visual hierarchy should account for the available screen real estate across different popular mobile device sizes, preventing and minimizing overflowing for critical content, and allow users to expand content panels when intuitive and necessary * History and saved information functionalities to improve usability on subsequent and frequent uses of the core features of the app * GPS access must be authorized for the core features to work correctly * Default origin location is always user’s current location * If GPS’s access is not granted or a user experiences weak GPS signal, the core features should fall back on network-based location (or last known GPS location) and the app should display the issue on the UI |
| Performance | * The app should load and update real-time transit data every ten seconds or less under normal conditions. * Even in high-traffic situations, it should never take more than 45 seconds to refresh route information on the trip finding results screen. * The app should feel smooth and responsive between navigations and interactions * If the app includes payment options, security will be a top priority. Any transaction must be verified within a safe window |
| Security | * Stores users’ data and user history locally where applicable * Users’ sensitive information (locations, payment information, travelling habits) must be encrypted and handled legally and ethically |
| Cultural | * English and French language support * Chinese in a future update |

## 2.5 Project Scope

**Project Objective**

Create a user-friendly GO Transit mobile app that provides real-time transit updates, trip finding & planning, trip quick-start, accessibility support, and payment integration.

**Project High-Level Requirements**

App deployment on mobile devices (iOS and Android) with all planned core features ready and optimized, delivered at the end of PRJ666.

React Native based (SwiftUI for native support on iOS)

Consistent user experience across devices and platforms.

Reliable real-time data synchronization for accurate transit updates.

Regular updates and maintenance for bug fixes, and new features.

**In Scope**

* Real-Time Transit Tracking – Live updates on train and bus locations, delays, departure and arrival times, transfer wait times, foot traffic times.
* Trip Planning – Route recommendations based on the user’s location, destination, and schedule preferences. Trip tracking mode that updates user’s progression of the selected trip
* Trip QuickStart – Start a trip based on saved favourite locations or choose a destination stop from nearby available transit routes.
* Offline Mode – Users can save their favorite routes and schedules to access them even without an internet connection.
* Voice Commands – Hands-free trip planning and schedule checking.
* Fare Calculator – Estimating fare costs before travel.
* Alerts & Notifications – Service alerts, upcoming departures, and delay notifications.
* Accessibility Features – High-contrast mode, voice-over support, and other inclusive design elements.
* Multi-Modal Transit Support – Integration with other public transport (subway, buses, streetcars).
* User Profiles & Trip History – Save frequent routes, view past trips, and customize preferences.
* Customer Support & Feedback – Direct access to report issues and submit feedback.

**Out of Scope (Phase 2)**

* In-app fare payments (future updates may include Presto integration).
* Ride-sharing or private transportation services integration (e.g., Uber, Lyft).
* Advanced travel predictions beyond real-time updates.
* Multilingual support (only English & French at launch, future expansion possible).
* Third-party advertising or promotions within the app.

## 2.6 System Risks

|  |  |
| --- | --- |
| **Risk** | **Response** |
| **Inconsistent GPS accuracy may cause incorrect real-time transit tracking.** | We’ll use multiple data sources (like Wi-Fi positioning and cell towers) alongside GPS to improve accuracy. The app will also notify users when location data is unreliable. |
| **High server load during peak hours could slow down app performance.** | We’ll optimize backend infrastructure, implement load balancing, and use caching to reduce server strain and keep response times fast. |
| **Confusion with navigation and UI** | We’ll carefully analyze our design decisions, conduct short and iterative user testing rounds in our design & develop process  Design should follow standardized UX practices and affordances |
| **Real-time data inaccuracy** | Display last update for transparency, use predictive measure when necessary, implement fallbacks |
| **Integration issues with tracking APIs, API outage** | Implement fallback logic, issues warning messages during outages |
| **Data breaches** | Use https encryption for all data in transit, encrypt all sensitive data |
| **Payment data leaks** | Ensure all compliances are met for payment processing |
| **Accessibility gaps** | Test app features with assistive tech, follow accessibility guidelines |
| **Privacy compliance** | Be transparent about data collection and regularly review privacy policies to ensure compliance  Let user opt in and out of data collection  Store only necessary data and anonymize them when possible |
| **App crashes resulting in losing currently tracked trip** | Implement local caching and try to recover last trip on relaunch |

## 2.7 Operating Environment

The GO Transit App is designed to work across different mobile devices, networks, and systems, making it easy for users to plan trips, buy tickets, and get real-time transit updates. Here are the operating environments the app will need to function properly:

#### 2.7.1 Supported Devices

* The app will support most modern smartphones and tablets, running **Android 10+** and **iOS 15+**.

#### 2.7.2 Software & Backend

* The app will use cloud-based servers (like AWS, Google Cloud, or Azure) to handle data and ensure everything runs smoothly.
* A secure database (such as MongoDB, PostgreSQL, or Firebase) will store user accounts, tickets, and travel history.
* Payment processing will be handled through trusted services like **Apple Pay, Google Pay, PayPal, and credit/debit cards**.

#### 2.7.3 Internet & Network

* Most features require an **active internet connection**
* Some offline support will be available, allowing users to access recent schedules and purchase tickets without Wi-Fi or data.
* All sensitive data transmissions will be **encrypted** to keep user information secure.

#### 2.7.4 Security & Privacy

* Sensitive information will be **encrypted using AES-256 and TLS protocols**, making it unreadable to hackers.
* The app will follow privacy regulations like **PIPEDA (Canada), GDPR (EU), and PCI-DSS** for payment security.
* If synchronized user accounts are implemented (instead of utilizing local profiles), Users can **log in securely** with email/password, Google/Apple accounts, or even biometric authentication (Face ID, fingerprint).

#### 2.7.5 Third-Party Services & Integrations

* **Google Maps Directions API** (Transit Mode) for route finder
* **Metrolinx’s GO Transit API** for static schedules, real time feeds for delays, and service updates.
* **Google Maps, Apple Maps, or OpenStreetMap** for navigation and route planning.
* **Push notifications** via Firebase and Apple’s notification service to send travel alerts and reminders.

By ensuring the app runs efficiently in this environment, GO Transit users will get a smooth, secure, and reliable experience every time they travel.

## 2.8 UI/UXD Interface Mock-ups

## Screens screenshot of a phone AI-generated content may be incorrect.

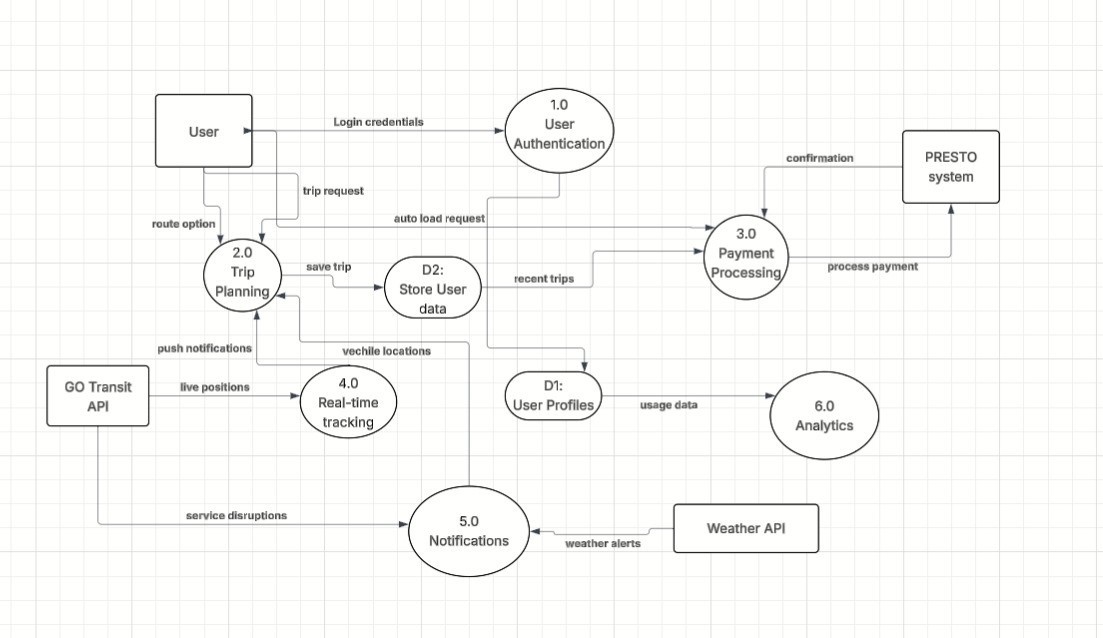
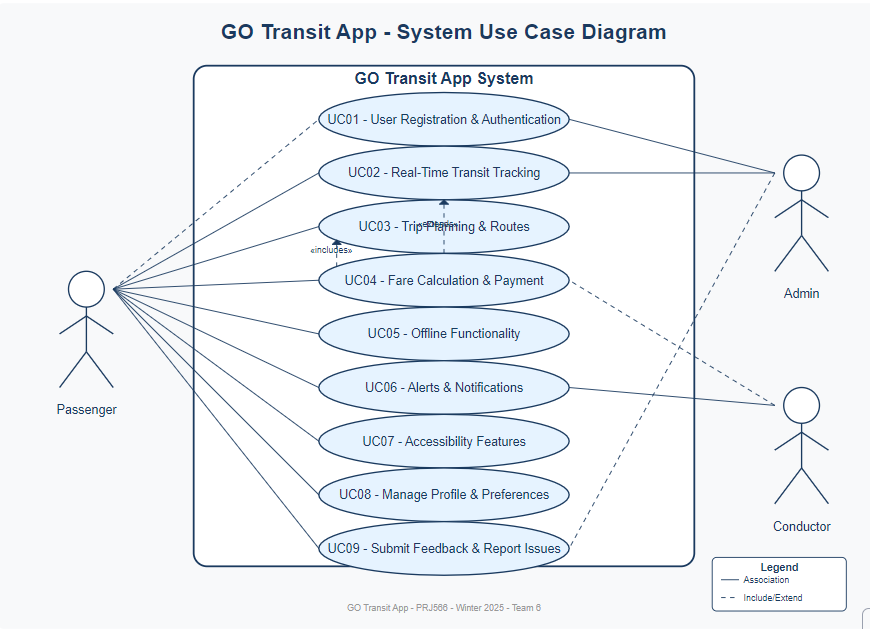
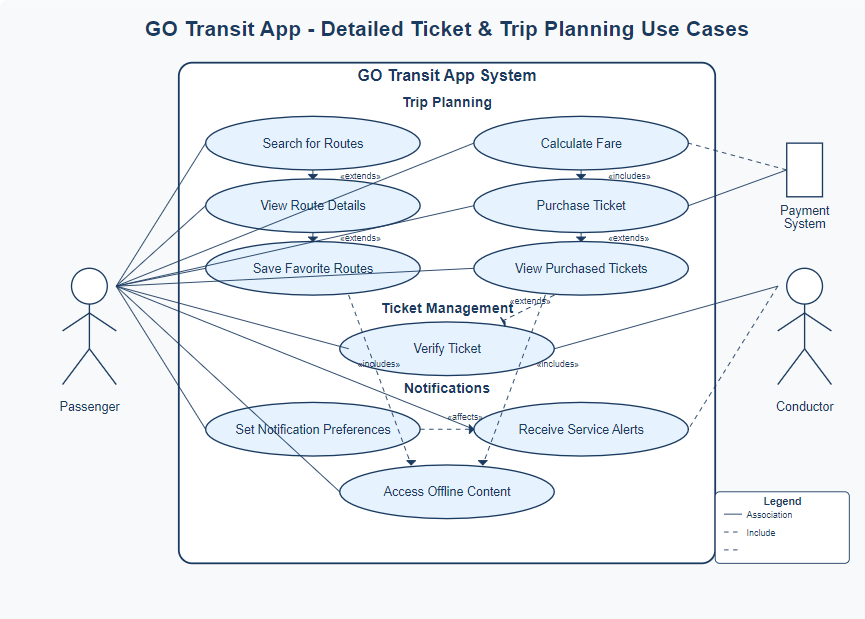
## Screens screenshot of a phone AI-generated content may be incorrect. Screens screenshot of a phone AI-generated content may be incorrect. Screens screenshot of a phone AI-generated content may be incorrect.

# Process and Data Modeling

## 

## **3.1 UML/DFD Modeling and Activity Diagram**

#### 3.1.1 Data Flow Diagram Modeling

**  
  
**

#### 3.1.2 Activity Diagram

## **A screenshot of a computer AI-generated content may be incorrect.**

## **3.2 Business Rules**

|  |  |  |
| --- | --- | --- |
| Business Rule Number | Business Rule Description | Related UC |
| BR01 | **User Registration & Authentication**   * User does not need to register for an account to access the app’s core functionalities | UC01 |
| BR02 | **Real-Time Transit Tracking**   * The app must display live updates for train and bus locations * GPS accuracy is validated, and users should be notified of unreliable location data. | UC02 |
| BR03 | **Trip Planning & Route Suggestions**   * Users must input destination location to receive route suggestions. * The app provides multiple route options based on travel time, cost, and convenience.   Users can save frequently used destinations for quick access | UC03 |
| BR04 | **Trip Quickstart**   * Users must be able to select favourite destinations from the home screen to quickly find trips to the said destinations from their current location * Favourite destinations must be fully customizable in settings menu * Users must be able to inspect nearby transit routes and quick-start a trip to one of the stops on the route | UC04 |
| BR04 | **Fare Calculation & Payment Integration**   * Users can estimate the fare costs when inspecting a trip * Future updates may include Presto card linking or direct payments through third-party services. * Transactions must be processed securely, and any unverified payments will be rejected. | UC05 |
| BR05 | **Offline Functionality**   * Users should be able to access saved destinations and schedules without an internet connection. * Cached data must be updated automatically when the user regains connectivity. | UC06 |
| BR06 | **Alerts & Notifications**   * Users must receive push notifications for service delays, route changes, and upcoming departures. * Notifications must be customizable, allowing users to enable/disable specific alerts. | UC07 |
| BR07 | **Accessibility Compliance**   * The app must support voice-over functionality for visually impaired users. * High-contrast mode and text resizing must be available for better readability. * Voice commands must be enabled for hands-free trip planning. | UC08 |
| BR08 | **User Profile & Preferences**   * Users can access and clear trip history, favorite destinations, and custom settings. * Data must be stored securely and comply with privacy regulations (PIPEDA, GDPR). | UC09 |
| BR09 | **Customer Support & Feedback**   * Users must be able to report issues and submit feedback directly through the app. | UC10 |
| BR10 | **System Performance & Security**   * The app must fetch transit data in under 10 seconds under normal conditions. * High server loads must be managed using cloud infrastructure and caching techniques. * All sensitive user data must be encrypted using AES-256 and TLS protocols. | UC11 |

## **3.3 Use Case Specifications:**

## 

**3.3.1**

**Use Case 1: Purchase Ticket**

|  |  |
| --- | --- |
| **Component** | **Details** |
| Actors | Passenger |
| Preconditions | 1. User must have a valid account 2. User must be logged in. 3. Users must have a valid payment method linked to the app. |
| Post Conditions | 1. Digital ticket is generated and stored in the user's account.  2. Payment is processed successfully. 3. Trip details are saved in user's history. |
| Description | Allows a user to buy a GO Transit ticket through the app. |
| Exception Conditions | • Payment failure due to insufficient funds → Show error message. • Network connectivity issues → Store transaction attempt and retry when connection is restored. • Invalid route selection → Prompt user to select valid stations. Service unavailable → Display message with alternative purchase options. |
| Business Rule | BR04: Fare Calculation & Payment Integration |

## **Corresponding Mockup: Use Case 1: Purchase Ticket**

A screenshot of a computer

AI-generated content may be incorrect.

**Use Case 2:** **View Real-Time Transit Updates**

|  |  |
| --- | --- |
| **Component** | **Details** |
| Actors | Passenger |
| Preconditions | 1. User must have an active internet connection. 2. Users must have location services enabled. |
| Post Conditions | 1. User receives accurate real-time information about train/bus locations 2. Users can view estimated arrival times and any service delays. |
| Description | Provides users with live updates on train and bus locations and schedules. |
| Exception Conditions | • Internet connectivity issues → Show cached data with timestamp. • GPS accuracy problems → Notify user of potential location inaccuracies. • API failure → Display message and estimated schedules from stored data • Location services disabled → Prompt user to enable for better experience. |
| Business Rule | BR02: Real-Time Transit Tracking |

## **Corresponding Mockup: Use Case 2: View Real-Time Transit Updates**

A screenshot of a phone

AI-generated content may be incorrect.

**Use Case 3: Plan Trip**

|  |  |
| --- | --- |
| Component | Details |
| Actors | Passenger |
| Preconditions | 1. User must have destination information. 2. Users must have access to route data. |
| Post Conditions | 1. User receives optimized route suggestions. 2. Trip details can be saved for future reference. |
| Description | Allow users to plan their journey by entering the starting point and destination |
| Exception Conditions | • Route not available → Show alternative options or next available time. • Invalid locations → Prompt user to enter valid transit stops. • Schedule changes → Alert user if selected route is affected by delays. • No routes found → Suggest nearest possible stations or alternative travel dates. |
| Business Rule | BR03: Trip Planning & Route Suggestions |

**Corresponding Mockup: Use Case 3: Plan Trip**

A screenshot of a computer

AI-generated content may be incorrect.

**Use Case 4: Set Up Alerts & Notifications**

|  |  |
| --- | --- |
| Component | Details |
| Actors | Passenger |
| Preconditions | 1. User must have an account. 2. Users must have enabled notifications for the app. |
| Post Conditions | 1. User receives customized alerts for their selected routes. 2. Notification preferences are saved to user profile. |
| Description | Allows users to customize what transit alerts they receive. |
| Exception Conditions | • Notifications disabled on device → Prompt user to enable in device settings. • Push notification failure → Store alerts in-app for viewing later. • Too many notification selections → Warn about potential notification overload. • System outage → Notify user when service is restored. |
| Business Rule | BR06: Alerts & Notifications |

**Corresponding Mockup: Use Case 4: Set Up Alerts & Notifications**

A screenshot of a computer

AI-generated content may be incorrect.

**Use Case 5: Access Offline Functionality**

|  |  |
| --- | --- |
| Component | Details |
| Actors | Passenger |
| Preconditions | 1. User must have previously saved routes or schedules. 2. The app must have cached the necessary data during last online session. |
| Post Conditions | 1. User can view essential information without internet connection. 2. The saved tickets remain accessible offline. |
| Description | Enables users to access critical transit information without an internet connection. |
| Exception Conditions | • No previously cached data → Display message explaining limitation. • Expired cached data → Warn user about potential inaccuracies. • Essential features unavailable offline → List features that require connection. • Storage limitations → Prompt user to manage offline content. |
| Business Rule | BR05: Offline Functionality |

**Corresponding Mockup: Use Case 5: Access Offline Functionality**

A screenshot of a computer

AI-generated content may be incorrect.

**Use Case 6: Submit Feedback & Report Issues**

|  |  |
| --- | --- |
| Component | Details |
| Actors | Passenger |
| Preconditions | 1. User must be logged in. 2. User must have experienced the service or feature being reported. |
| Post Conditions | 1. Feedback is logged into the system.  2. User receives confirmation of submission. |
| Description | Allow users to report problems or provide suggestions about the app or transit service. |
| Exception Conditions | • Form submission failure → Save draft and retry. Inappropriate content detection → Prompt user to revise. • High volume of similar reports → Inform user issue is already known. • Anonymous submission attempt → Remind user of login requirement. |
| Business Rule | BR09: Customer Support & Feedback |

**Corresponding Mockup: Use Case 6: Submit Feedback & Report Issues**

A screenshot of a chat

AI-generated content may be incorrect.

**Use Case 7: Manage User Profile & Preferences**

|  |  |
| --- | --- |
| Component | Details |
| Actors | Passenger |
| Preconditions | 1. User must have an active account. 2. User must be logged in. |
| Post Conditions | 1. Profile changes are saved securely. 2. User preferences are applied across the app experience. |
| Description | Allows users to update personal information and customize app settings. |
| Exception Conditions | • Invalid input → Highlight fields needing correction. • Email already in use → Prompt for different email. • Password doesn't meet requirements → Explain security criteria • Data sync failure → Save locally and retry when possible. |
| Business Rule | BR08: User Profile & Preferences |

**Corresponding Mockup: Use Case 7: Manage User Profile & Preferences**

A screenshot of a phone

AI-generated content may be incorrect.

**Use Case 8: Use Accessibility Features**

|  |  |
| --- | --- |
| Component | Details |
| Actors | Passenger (with accessibility needs) |
| Preconditions | 1. User must have device with accessibility support 2. Accessibility features must be enabled in app settings. |
| Post Conditions | 1. App interface adapts to user's accessibility needs. 2. Users can navigate and use all essential features. |
| Description | Enables users with disabilities to use the app through specialized interface options. |
| Exception Conditions | • Device compatibility issues → Explain limitations • Voice recognition errors → Provide alternative input methods. • Screen reader conflicts → Offer troubleshooting steps. • Feature unavailable in current version → Indicate upcoming support. |
| Business Rule | BR07: Accessibility Compliance |

**Corresponding Mockup: Use Case 8: Use Accessibility Features**

A screenshot of a computer

AI-generated content may be incorrect.

**Use Case 9: Verify and Use Digital Ticket**

|  |  |
| --- | --- |
| Component | Details |
| Actors | Passenger, Conductor |
| Preconditions | 1. User must have purchased a valid ticket. 2. Tickets must not be expired or already used. |
| Post Conditions | 1. Ticket is validated for the trip. 2. Usage is recorded in the system. |
| Description | Allows users to display and validate their digital tickets for transit use. |
| Exception Conditions | • Expired ticket → Display clear message with renewal options. • Ticket validation failure → Provide error code and support contact. • Device battery critical → Offer low-power ticket display mode. • No ticket found → Guide user to purchase history or buying options. |
| Business Rule | BR04: Fare Calculation & Payment Integration |

**Corresponding Mockup: Use Case 9: Verify and Use Digital Ticket**

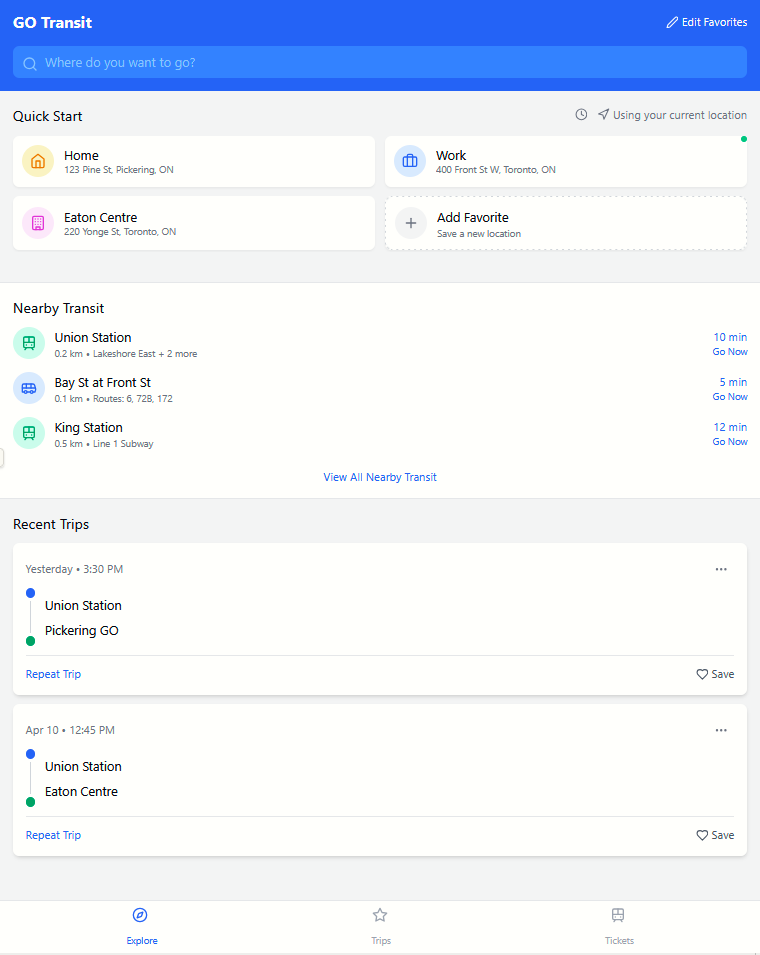
A screenshot of a phone

AI-generated content may be incorrect.

**Use Case 10: Quick-start Trip to Favorite Destinations**

|  |  |
| --- | --- |
| Component | Details |
| Actors | Passenger |
| Preconditions | 1. User must have saved at least one favorite destination. 2. Users must have location services enabled. |
| Post Conditions | 1. User receives route options from current location to selected favorite destination. 2. Users can immediately proceed to trip details or ticket purchase. |
| Description | Allows users to quickly start trip planning from the home screen to favorite destinations or nearby transit stops. |
| Exception Conditions | • Location services unavailable → Prompt users to enter current location manually. • Favorite destinations not configured → Guide user to add favorites in settings. • Nearby transit routes unavailable → Display message with alternative planning options. • Connection issues → Use cached data with timestamp indicator. |
| Business Rule | BR04: Trip QuickStart |

**Corresponding Mockup: Use Case 10: Quick-start Trip to Favorite Destinations**



**Use Case Diagram: it includes the granular actions as well as mentioned in the feedback**A diagram of a person with different colored circles

AI-generated content may be incorrect.

A diagram of a ticket management system

AI-generated content may be incorrect.

A diagram of a trip planning

AI-generated content may be incorrect.

A diagram of a system

AI-generated content may be incorrect.

A diagram of a system

AI-generated content may be incorrect.

# Domain Class Diagram

A diagram of a computer program

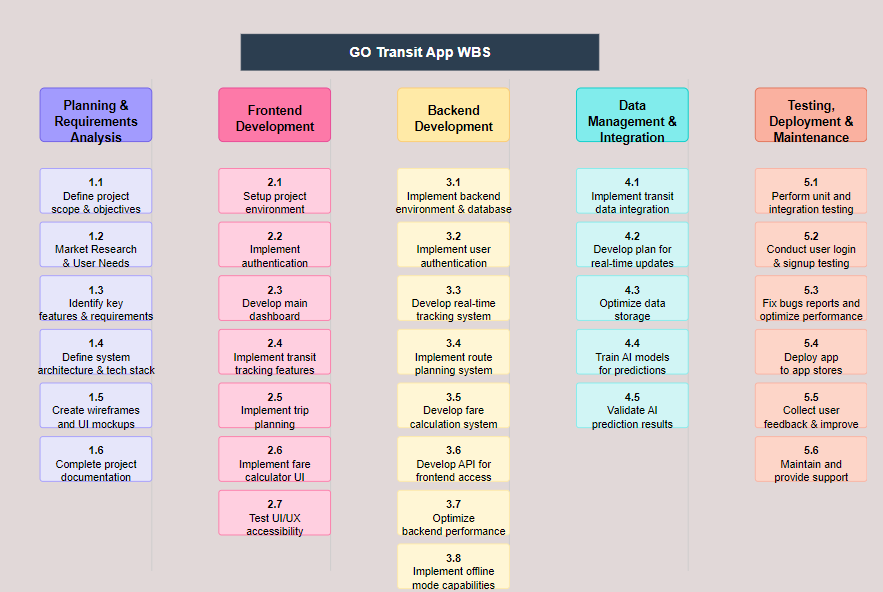
AI-generated content may be incorrect.

# Database

Please find the attached JSON documents in the Deliverables/Week 12 folder to view our conceptual JSON data schema and sample for the current class structure.

# Work Breakdown Structure (WBS)

## Work Breakdown Structure



GO Transit App Work Breakdown Structure Overview

The GO Transit App project is organized into five main phases that will be executed in parallel with dependencies between them:

**1. Planning & Requirements Analysis**

This foundational phase involves:

* Defining project scope and objectives (1.1)
* Conducting market research and analyzing user needs (1.2)
* Identifying key features and requirements (1.3)
* Defining system architecture and technology stack (1.4)
* Creating wireframes and UI mockups (1.5)
* Preparing project documentation (1.6)

**2. Frontend Development**

This phase focuses on the user interface and experience:

* Setting up the project environment (2.1)
* Implementing user authentication (2.2)
* Developing main dashboards (2.3)
* Implementing transit tracking features (2.4)
* Implementing trip planning (2.5)
* Implementing calculator UI (2.6)
* Testing UI/UX accessibility (2.7)

**3. Backend Development**

This phase handles the server-side functionality:

* Implementing backend environment and database (3.1)
* Implementing user authentication (3.2)
* Developing real-time tracking system (3.3)
* Implementing route planning system (3.4)
* Developing fare calculation system (3.5)
* Developing API for frontend access (3.6)
* Optimizing backend performance (3.7)
* Implementing offline mode capabilities (3.8)

**4. Data Management & Integration**

This phase focuses on data handling and analytics:

* Implementing transit data integration (4.1)
* Developing API for real-time updates (4.2)
* Optimizing data storage (4.3)
* Training AI models for predictions (4.4)
* Validating AI prediction results (4.5)

**5. Testing, Deployment & Maintenance**

This phase ensures quality and availability:

* Performing unit and integration testing (5.1)
* Conducting user login and signup testing (5.2)
* Fixing bugs and optimizing performance (5.3)
* Deploying app to app stores (5.4)
* Collecting user feedback and implementing improvements (5.5)
* Maintaining and providing ongoing support (5.6)

**The WBS shows clear dependencies between phases. For example, frontend development (2.x) relies on planning decisions (1.x), while integration testing (5.1) depends on completion of both frontend and backend components. The structure ensures all necessary components are built, tested, and deployed in a logical sequence to create a fully functional transit application.**

# Milestones and Acceptance Criteria

**Comprehensive 14-Week Timeline with Milestones and Acceptance Criteria for GO Transit App**

**Week 1-2: Planning & Requirements Analysis**

Milestone 7.1: Requirements Documentation Package

* Timeline: End of Week 2
* Deliverables: Project scope document, market research report, feature prioritization matrix, system architecture diagram
* Acceptance Criteria:
  + Stakeholder approval of project scope and objectives (Task 1.1)
  + Market research identifies at least 5 key user pain points (Task 1.2)
  + Feature list prioritized using MoSCoW method (Task 1.3)
  + Architecture diagram includes all system components and data flows (Task 1.4)
  + Project timeline and resource allocation approved by management

**Week 3-4: Design & Environment Setup**

Milestone 7.2: Design Package & Development Environment

* Timeline: End of Week 4
* Deliverables: UI/UX wireframes, style guide, functional prototypes, development environments
* Acceptance Criteria:
  + Wireframes created for all primary user journeys (Task 1.5)
  + UI prototypes tested with at least 5 potential users (Task 1.5)
  + Development environments configured for both frontend and backend (Tasks 2.1, 3.1)
  + Style guide complies with accessibility standards (WCAG 2.1 AA)
  + Design system components documented for developer reference

**Week 5-6: Core Authentication & Database Implementation**

Milestone 7.3: Authentication System

* Timeline: End of Week 6
* Deliverables: Working user authentication system, database schema, API endpoints
* Acceptance Criteria:
  + Users can register, login, and recover accounts (Tasks 2.2, 3.2)
  + Authentication system secures data with industry-standard encryption
  + Database schema handles all required data relationships
  + API endpoints for user management functional and documented
  + Authentication system passes security testing with zero critical vulnerabilities

**Week 7-8: Real-time Tracking & Core Features**

Milestone 7.4: Core Functionality Package

* Timeline: End of Week 8
* Deliverables: Real-time tracking system, dashboard UI, data integration layer
* Acceptance Criteria:
  + Real-time tracking updates vehicle positions at least every 30 seconds (Tasks 2.4, 3.3)
  + Main dashboard displays relevant transit information (Task 2.3)
  + Transit data successfully integrates with backend systems (Task 4.1)
  + API for real-time updates handles at least 100 requests per second (Task 4.2)
  + System maintains 98% accuracy in vehicle location reporting

**Week 9-10: Trip Planning & Fare Calculation**

Milestone 7.5: Planning & Calculation Systems

* Timeline: End of Week 10
* Deliverables: Route planning system, fare calculator, predictive models
* Acceptance Criteria:
  + Trip planning algorithm finds optimal routes within 3 seconds (Tasks 2.5, 3.4)
  + Fare calculation accurately reflects current transit pricing (Tasks 2.6, 3.5)
  + System handles multi-modal journeys (bus, train, etc.) seamlessly
  + Trip plans account for real-time disruptions and delays
  + AI models predict arrival times with 90% accuracy within ±2 minutes (Task 4.4)

**Week 11-12: Integration & Testing**

Milestone 7.6: Beta Release

* Timeline: End of Week 12
* Deliverables: Fully integrated application, test reports, performance optimization
* Acceptance Criteria:
  + All frontend and backend components successfully integrated (Task 3.6)
  + Unit and integration tests pass with >90% coverage (Task 5.1)
  + User journey testing validates all primary use cases (Task 5.2)
  + Performance optimization reduces load times to under 3 seconds (Tasks 3.7, 5.3)
  + Offline mode functionality preserves core features without internet (Task 3.8)
  + Data storage optimized to keep app below 100MB size (Task 4.3)

**Week 13-14: Deployment & Launch Preparation**

Milestone 7.7: Production Release

* Timeline: End of Week 14
* Deliverables: Store-ready application, support documentation, analytics dashboard
* Acceptance Criteria:
  + Application passes all store review requirements (Task 5.4)
  + User documentation completed and accessible within app
  + Feedback system implemented and tested (Task 5.5)
  + Support protocols established with response SLAs (Task 5.6)
  + Analytics in place to monitor post-launch performance
  + Final testing confirms zero critical or high-priority bugs

**Detailed Acceptance Criteria by Feature Area**

**Real-time Transit Tracking**

* Location accuracy within 50 meters of actual vehicle position
* Map visualization clearly distinguishes vehicle types and directions
* System handles at least 1,000 simultaneous vehicle tracking points
* Tracking history maintained for at least 24 hours
* Push notifications for tracked route delays function correctly

**Trip Planning System**

* Algorithm accounts for walking distance, transfers, and accessibility needs
* Results include at least 3 route options when alternatives exist
* Plans adapt to real-time conditions (weather, traffic, service disruptions)
* Saved trips receive automatic updates when conditions change
* Estimated arrival times accurate within ±3 minutes during normal conditions

**Fare Calculation**

* Calculator accounts for all fare types, zones, and discount programs
* Fare information updates automatically when transit pricing changes
* Users can compare costs between different payment methods
* Weekly/monthly pass value comparison tool functions correctly
* System accurately calculates complex multi-leg journey pricing

**Data Management & Performance**

* Database queries optimized to execute in under 100ms
* App remains responsive during peak usage periods
* Battery consumption does not exceed 5% per hour of active use
* Memory management prevents leaks during extended usage
* System gracefully handles API failures with user-friendly error messaging

# Implementation Schedule

[Product Backlog (Agile-Scrum): Github link](https://github.com/vintrn/prj566-ncc-25winter-g6/milestone/8)

# Client / Faculty Sign-off

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

X .

Name of Client/Rep/Professor