CPEN 321 Requirements Document

Group: Time Your Trip

Members: James Zhou, Johannes Gallmann,
Sirine Trigui, Yixin Zhao, Navjashan Singh

Requirements, Vision, and Scope

Product Vision Statement

Our product is a mobile application that sends alerts to the users when their bus is arriving at the user's chosen stop, to prevent them from missing it when doing other tasks on bus. The purpose of this app is to allow commuters to be more productive with their time during long commutes by allowing them to spend time doing useful tasks (ie. studying, sleeping) without having to worry about missing their stop. There are similar products which take users' destination, and use GPS to get the location then alert users on designated stops. Our app will similar to these apps, with the difference being that our app will offer a non real-time version of the app that uses GTSF data stored in a custom database in case of failure from the Translink Open API or TransLink Database. Besides, if the user needs to transfer to other different buses, it will be troublesome to enter all destinations for each bus; yet our application only requires information which is easy to access for the user.

Problem statement:

The problem of	Making long commute trips more productive for users
affects	TransLink bus users
The impact of which is	Instead of doing useful tasks, users will have to divert their attention to making sure that he/she gets off at the correct stop.
A successful solution would be	To come up with an app that will notify users when he/she reaches the chosen stop.

Product Position Statement:

for	Bus passengers
who	Provide us with some parameters as bus-stop-# , and destination stop.
Our system	Is a mobile application that utilizes the Android OS and the internet access capabilities of mobile phones.

that	prompts the user to enter the bus stop and bus number to return all the bus stops ahead from which the user has to select their final destination. By providing so, an estimated time will be returned and the user alarm will be turned on.
unlike	TravAlert; TranSit
Our product	Takes advantage of both real-time and static transit information and will switch between the two according to the status of the Translink Database.

End users and feature list:

Translink bus passengers will be our end users. They will input the details which the app will ask for such as bus stop number, bus number, bus ID and then their destination on. Once they have entered all this information, the app will display the time it takes to reach the destination and will set an alarm for the user to get alert before his/her destination arrives. The user is just expected to have some basic knowledge about how to input information into an app.

Constraints :

Our product works with the information received from TransLink using the TransLink OPEN API and GTFS data. It's freely available and we can access by registering with them. As we are making an android app, our platform will be Android Studio.

Scope and Limitations :

Our app will only be restricted to the Lower Mainland, as TransLink's database only covers this region. We will be using GPS data in order to track the user's current position and to use this factor to make our trip duration estimation algorithm more efficient and accurate. We will also be integrating a MySQL database to store temporary information retrieved from queries, and to store the large GTFS data.

Assumptions and Dependencies:

Our product is highly dependent on the Translink's API, as it receives all the information regarding bus schedule, bus ID's and stops, as well as the GTFS data for providing static estimated times. For our app to work properly, we will be expecting that the information we are receiving from TransLink is correct and reliable.

Feature List:

1. Storing user inputs and formatting a Query for the TransLink Open API based on them.
2. Providing a helpful UI system that will guide users to input data.
3. Undo feature that allows users to cancel inputs and start over.
4. Connecting to TransLink Open API and retrieving relevant data in JSON format.
5. Parse JSON data to retrieve relevant information regarding arrival times.
6. Online database that can store temporary data retrieved from TransLink Open API and GTFS data.
7. Using GPS data to keep track of user's location relative to the location of the chosen destination stop.
8. Algorithm that calculates time of arrival based on real-time data and GPS information.
9. Limiting phone data usage through minimizing the number of requests sent.
10. Accessing GTFS data whenever the Translink Open API is down, through developing queries that can access relevant data stored in the database.
11. Set alarm/notifications based on the calculated estimated time of arrival.

User Stories and Use Cases:

Use Case: Set a notification for when the bus reaches the chosen stop

Identification:	1.1 Sets alarm based on real-time TransLink data for the user's bus
Primary Actor	A commuter about to board a bus.
Stakeholders & Interests	Commuters would want to take advantage of this use case in order to maximize their efficiency while riding the bus. Bus drivers would want to use this app in order to make sure that they are arriving at their destination on time and to send feedback to TransLink if they feel that there are errors between real-time data and actual arrival times.
Preconditions	● TransLink Database is operational ● App is currently displaying starting menu ● No inputs have been registered ● User must have access to a bus stop number
Post-conditions	Success: ● An alarm/notification is set ● A window is displayed with a "cancel" button, navigating to the starting window app, allowing the user to start over and cancel the previously set alarm. Failure: ● An error toast is displayed

	● App resets to its starting menu with the previous parameters wiped.
Main Success Scenario	1. User inputs the bus stop number 2. System verifies and displays the buses arriving in the next 30 minutes, as well as an alternative box for inputting a specific bus ID. 3. User selects a bus from display. 4. System verifies and displays all of the stops for that bus 5. User selects stop. 6. System verifies, calculates, and sets the timer to ring when the user reaches that stop.
Extensions & Alternative Flows	1. Bus stop number cannot be identified (error): ● Error toast is displayed ● System returns to starting menu, prompting user to input bus stop number again. 2a .No busses can be detected within the time frame (error) ● System displays the first bus that will be arriving at the selected stop 2b. User inputs a select bus ID 1. Bus ID cannot be identified -System stays on current page, toasts that bus ID is invalid, and prompts user to choose a bus displayed or to input an ID again 2. Bus ID is identified. -System proceeds with from step 4 2c. User presses undo button ● System resets to main menu and the previous input for bus stop number is wiped. 4. User presses undo button ● System resets to the bus display menu and the previous input for bus selection is wiped 6. User presses cancel button ● System resets to the main menu, the set timer is canceled, and the system redirects to the starting menu with all inputs wiped.
Open Issues	● When a valid bus stop number does not return any buses (issue with Translink Database) ● When a valid bus selection does not produce any stops (issue with Translink Database) Hypothetical solutions: default to stored GTFS database.

Identification:	1.2 Sets alarm based on estimated times provided by the GTFS data
Primary Actor	A commuter about to board a bus.
Stakeholders & Interests	Commuters would want to take advantage of this use case in order to maximize their efficiency while riding the bus and when TransLink encounters errors in their system (which unfortunately happens quite often).

Preconditions	● The most recent GTFS data is downloaded and stored in an online database. ● App is currently displaying starting menu. ● No inputs have been registered. ● User must have access to a bus stop number.
Post-conditions	Success: ● An alarm/notification is set ● A window is displayed with a reset button, navigating to the starting window app, allowing the user to start over and cancel the previously set alarm. Failure: ● An error toast is displayed ● App resets to its starting menu with the previous parameters wiped.
Main Success Scenario	1. User inputs the bus stop number 2. System verifies and prompts user to enter a bus ID. 3. User selects a bus. 4. System verifies and displays all of the stops for that bus 5. User selects stop. 6. System verifies, calculates, and sets the timer to ring when the user reaches that stop.
Extensions & Alternative Flows	1a. Bus stop number cannot be identified (error): ● Error toast is displayed ● System returns to starting menu, prompting user to input bus stop number again. 2a .No busses can be detected within the time frame (error) ● System displays the first bus that will be arriving at the selected stop 2b. User inputs a select bus ID 3. Bus ID cannot be identified -System stays on current page, toasts that bus ID is invalid, and prompts user to choose a bus displayed or to input an ID again 4. Bus ID is identified. -System proceeds with from step 4 2c. User presses undo button ● System resets to main menu and the previous input for bus stop number is wiped. 4. User presses undo button ● System resets to the bus display menu and the previous input for bus selection is wiped 6. User presses cancel button ● System resets to the main menu, the set timer is canceled, and the system redirects to the starting menu with all inputs wiped.
Open issues	GTFS data only provides estimated times, does not produce real-time transit information

	What happens when TransLink Open API is back online again?