Improving Public Transit Applications by Utilizing Location Data

Specification

Author
Tim Vahlbrock
Westphalian University of Applied Sciences
Gelsenkirchen, Germany
tim@vahlbrock.de

Supervisor
Prof. Dr. Wolfram Conen
Westphalian University of Applied Sciences
Gelsenkirchen, Germany
wolfram.conen@w-hs.de

I. Introduction

One of the major challenges for the upcoming years is bringing more people from individual transit options to public ones. A big factor in wether people are willing to use public transit options is the convenience of usage. Therefore accurate and reliable information systems are required to keep passengers updated on delays, changes and arrival times. The variety of mobile applications (see examples [1, 2]) available from different public transit offer many different features, but often target users on occasional, long distance trips, rather than commuters [1]. This specification describes location based concepts targeted at commuters along with the according development process and evaluation. The concepts will be implemented as a proof of concept in form of a headless client server pair, both might be integrated into an existing application at a later point of time.

II. Nearby Platforms Detection

In contrast to occasional, long trip travelers, commuters and other users of long-time tickets often have a confident knowledge about the options they need to take and therefore will not enter the trip in the application. Nevertheless they may be interested in the departure times when waiting at a platform, especially when delays and disruptions are expected. The goal is to be able to automatically determine nearby platforms, which would allow to show according departure and disruption information. This eliminates the need to manually enter the station or platform and reduces the amount of information shown to the relevant ones. Listing 1 specifies the feature in Gherkin [3].

```gherkin
Feature: Nearby Platforms Detection
  In order to be able to see nearby transit options without manually entering the station
  As a commuter
  I want to be able to detect nearby platforms

  Scenario: Detects the current platform
    Given I am on a platform
    When I check for my current status
    Then the current platform is detected

  Scenario: Detects multiple nearby platforms
    Given I am near multiple platforms
    When I check for my current status
    Then the current platform is detected
    And nearby platforms are detected

  Scenario: Detects no platform
    Given I am on a platform
    When I check for my current status
    Then no nearby platforms are detected
```

Listing 1: Nearby Platforms Detection in Gherkin

[1]For the sake of readability, all passengers frequently using similar routes will be referred to as “commuters”.

III. Onboard Train Detection

Just like departure information, information about the current train is not easily accessible to commuters, who usually do not plan the trip in the application. Usually, they will need to manually enter the departure point from which they already left and set a planned departure time in the past. Alternatively an upcoming station needs to be entered as the departure point from memory. This is especially cumbersome if it needs the be repeated on a daily basis. The proposed solution is to automatically detect the train the user is traveling on by combining the current location and direction of travel with the scheduled timetable, as well as current delay data. This would allow to present information about the train the user is currently on, without requiring additionally inputting information. Listing 2 specifies the feature in Gherkin.

```
Feature: Onboard Train Detection
  In order to be able to see information about the train I am currently on
  As a commuter
  I want to be able to detect the train I am currently on

  Scenario: Detect the train I am currently on
    Given I am on a train
    When I check for my current status
    Then the train I am on is detected

  Scenario: Detects the train I am currently on when it is at a station
    Given I am on a train at a station
    When I check for my current status
    Then the train I am on is detected

  Scenario: Detects the platform I am at when I am next to a train on a platform
    Given I am on a train at a station
    When I check for my current status
    Then the train I am on is detected

  Scenario: Detects the train I am currently on when another train is passing
    Given I am on a train
    And another train is passing in the opposite direction
    When I check for my current status
    Then the train I am on is detected

  Scenario: Detects I am on a train when location glitched
    Given my previous location was nowhere near
    And I am on a train
    When I check for my current status
    Then the train I am on is detected

  Scenario: Detects no train when I am not on a train
    Given I am not on a train
    When I check for my current status
    Then no train is detected

  Scenario: Detects I have left the train when I did
    Given I was on a train
    And I left the train
    When I check for my current status
    Then no train is detected
```

Listing 2: Onboard Train Detection in Gherkin

IV. Learning User Behavior

Most commuters will take the same routes on regular basis. Additionally, even when non commuting users will often get on or off at similar stations, regardless of the day and time. By learning those important trips and stops, the previously described features may be enhanced and extended. When it is detected, that the user has boarded a train, the exit station may be inferred or at least suggested from regular trips. This also applies to the final destination of the trip. Learning a regular trip of the would additionally allow to inform the user about upcoming blockings and impairments of trips not yet started. Listing 3 specifies the feature in Gherkin.

```
Feature: Learning User Behavior
  In order to have my actions predicted
  As a commuter
  I want my behavior to be learned

  Scenario: Predicting the exit station
    Given I frequently exit at a specific station
    And I am on a train that includes that station
    When I check for predictions
    Then the station is predicted as my exit station

  Scenario: Predicting the final destination
    Given I frequently travel to a location
    And I am on a train traveling in the direction of the location
    When I check for predictions
    Then the location is predicted as my final destination

  Scenario: Predicting a future trip
    Given I travel to a location at regular intervals
    And an interval is upcoming
    When I check for predictions
    Then the location is predicted as my final destination
```

Listing 3: Learning User Behavior in Gherkin

V. Non functional Requirements

Location data is a highly critical class of data in terms of privacy. It can not only be used to determine the current position of a user, but also to determine the users home and work address, their daily commute and other patterns, which is highly useful, but also very sensitive information. Therefore the protection of the location data should have a high priority. Wherever viable, calculations should be done on the client and knowledge of the users location on the server side should be sufficiently vague. The increased level of privacy will likely introduce unintended side effects regarding storage, network and battery usage. Nevertheless those effects should be minimized, for example by regarding reasonable compromises. Due to the high battery consumption, usage of location services should be limited to a reasonable degree in terms of precision and update interval.
The software should be well-tested and documented. The tests should cover vital implementations, however a specific test coverage is not set. As this software will not be deployed productively, it's security has a lower priority. Nevertheless the software should be designed with security in mind, be developed with proper isolation and follow best practices.

VI. Development Process

The development will be done based on the Behavior Driven Development (BDD) approach, using the Gherkin Specifications in listings 1, 2 and 3. One of the main motivators for agile development methods is goal for more, but also more efficient communication. Nevertheless, the application of agile methods can also provide other benefits software development projects with a single developer like this one. The concept of Personal Extreme Programming (PXP) introduces strategies for this [4]. Besides BDD, the following concepts of PXP are applied in this project additionally:

- **Continuous Integration:** Source Code is versioned, automatically build and tested on every commit.
- **Test Driven Development:** Tests are written before the implementation.
- **Refactoring:** The code is continuously refactored to improve readability and maintainability.
- **Spike Solutions:** When encountering problems, the implementation of new features is preceded by a spike solution to evaluate the feasibility and complexity.
- **Time Recording:** The time spent on each task is recorded to evaluate the efficiency of the development process.

The concept of a development iteration is maintained for organizational purposes and set to a length of two weeks. The planning is performed using a separate project in the "Oeffis" namespace on GitHub, the code is submitted to a newly created repository.

VII. Evaluation

The evaluation of the implemented module will be based on the following criteria.

- **Completeness:** The module implemented all the features specified.
- **Viability:** The way of implementation is viable for production usage, especially on mobile devices.
- **Privacy:** The way of implementation respects the sensibility of the location data and minimizes the risk of it being misused.

References

[1] Deutsche Bahn AG. “Der DB Navigator.” (2023), [Online]. Available: https://www.bahn.de/service/mobile/db-navigator (visited on 09/27/2023).

[2] Verkehrsverbund Rhein-Ruhr AöR. “VRR App.” (2023), [Online]. Available: https://www.vrr.de/de/fahrplan-mobilitaet/vrr-app/ (visited on 09/27/2023).

[3] SmartBear Software, *Gherkin Syntax*. [Online]. Available: https://cucumber.io/docs/gherkin/ (visited on 10/17/2023).

[4] Y. Dzhurov, I. Krasteva, and S. Ilieva, “Personal extreme programming - an agile process for autonomous developers,” *Faculty of Mathematics and Informatics, Sofia University*, 2009.