Software Documentation for WhenBus

Venketep Prasad M C,Rachit Garg,Vineeth J,Aravind B,Vamshi Krishna J

***Abstract*—The WhenBus application presented in this paper is a real time scheduling application that can be used by travellers to effectively transport in a city. In essence, WhenBus collects data from users on the real time location of the buses and uses that data to answer the bus scheduling queries of other users. This paper presents the main aspects of the WhenBus app gives details of the description, requirements and the use case modelling scenarios.**

I. Introduction

The WhenBus application is a software product that is used by users to get real time data on the location of the buses in a city transportation system. Our application asks users to tell if they have currently boarded the bus and uses that data to answer queries on the real time availability of the buses. In some cases when enough real time information is not available, it queries the city transportation database and uses that to answer user queries.

This paper, in its remaining part, is structured as follows: Section 2 presents a brief description of the WhenBus application, Section 3 describes the main functional and non-functional requirements of the application, Section 4 provides details on use cases and scenarios.

II. General Description

The WhenBus app requires users to confirm if they are already present inside a bus. This information is then sent to a server where its used to update the bus location based on the gps coordinates of the user. Users also query the app for finding an appropriate bus schedule and bus stop. This is obtained from a database that is stored in a server.

From an organisational point of view, there are two main components, first being asking users their requirements and getting the static schedule from a database. Second being real time updating of the schedule. Both components are used by the end user, where the second component is for users who are currently inside the bus. Both components require user to send his location. The first component requires user to manually enter the source, destination and the bus to be used. The second component requires user to mention if there location is synchronous with the bus location. As soon as all the information is entered, given that there are enough users using our app. We get a real time mapping of most of the buses in our city transportation system.

III. Requirements Specification

The requirements for the When bus have been specified as detailed below.

A. Functional Requirements

1) The Whenbus shall provide the means of entering and storing:
   - User location.
   - User Destination.
   - Details of the bus that the user has boarded in.
   - Time when the user leaves the bus stop.
2) The WhenBus shall take into consideration for generating the schedule using the following parameters: Current user location, Destination of the user, and Distance of the user from the nearest bus stop, Details of the bus entered by the person who is already travelling in the bus in which our user will board on later.
3) WhenBus will then suggest the approximate time the user has to wait in the bus station to board a bus based on the questions that the user answered.
4) WhenBus will suggest the most efficient route constrained to time when there are multiple buses needed to reach the destination.

B. Non-Functional Requirements

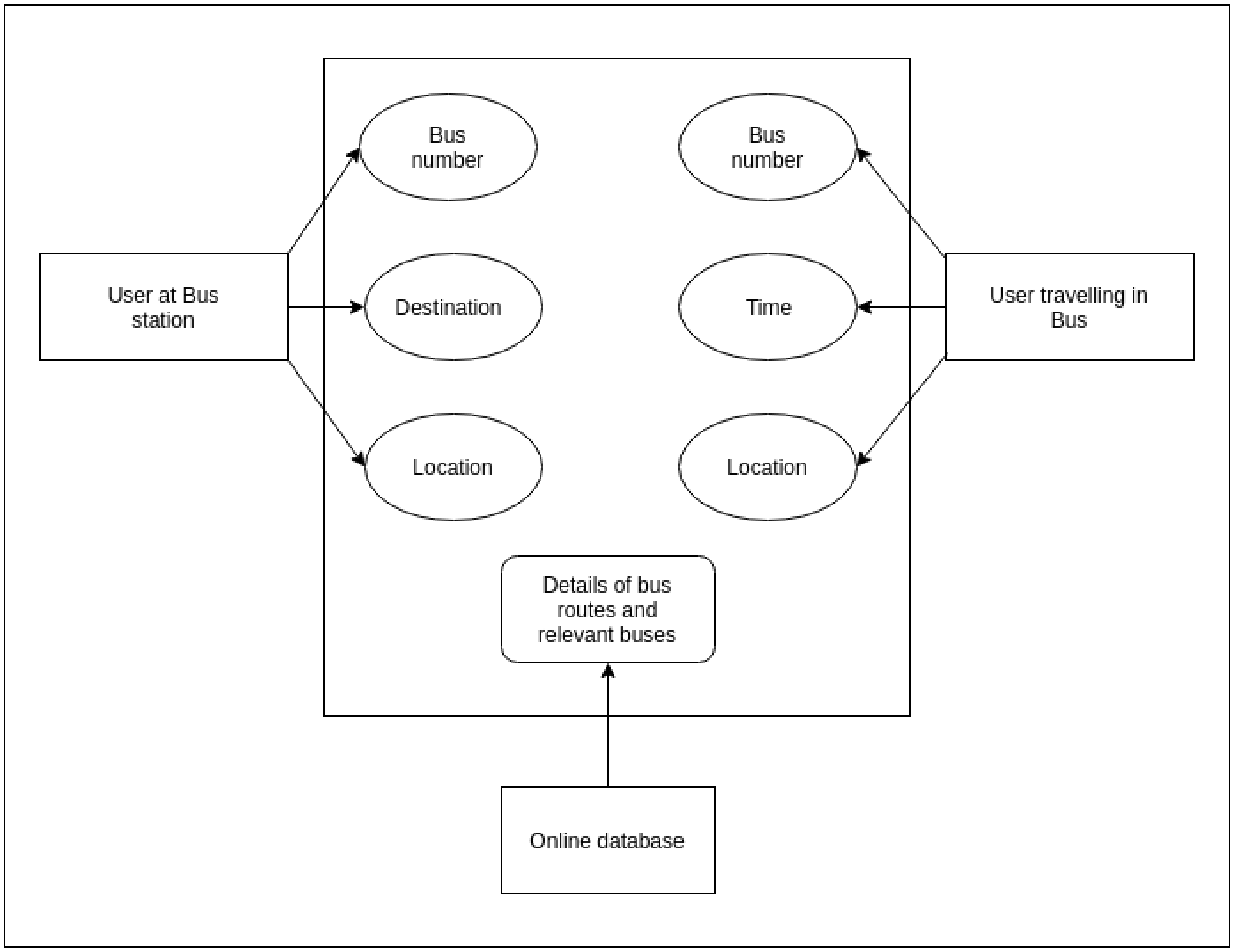
1) The WhenBus shall be written in Java.
2) The interfaces shall be implemented using Android studio.
3) The WhenBus shall store its data using firebase.

IV. Use Case Modeling

As part of the modeling process, the functionality of the WhenBus has been defined using use cases and scenarios. At a high level of abstraction the entire functionality of the class scheduler is captured in the use case diagram which is shown below.

A. Use Case Diagram

The use case diagram shown in Figure below depicts the interactions between the user waiting at the bus station,the user traveling in the bus, the and the online database system system. First user waiting at the bus station will enter his destination, preferred bus number. Then we will calculate the approximate time he should wait at the bus station using the data retrieved from the user traveling in the bus and the online database.


Bus number
Bus number
User at Bus station
Destination
Time
User travelling in Bus
Location
Location
Details of bus routes and relevant buses
Online database

V. State Transition Table

State	Enter	Change fields
At bus stop	Query page	
In bus	Real time info	
Real time info	Thank you page	In bus
Query page	Result page	At bus stop
Result page	In bus	At bus stop
Thank you page		

VI. Z Notation for the state transition

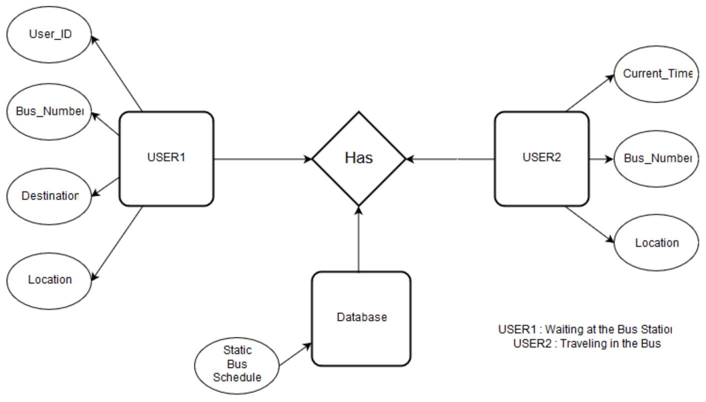
The Z notation for the WhenBus app is as follows:

STATE ::== At bus stop | In bus | Real time info | Query page | Result page | Thank you page
EVENT ::== Enter | Change fields
FS == (STATE X EVENT) -> STATE

N, T, C:FSM

C = N $\oplus$ T
N = {s: STATE; e:EVENT • (s,e) ->s}
T = {
(At bus stop, Enter) -> Query page ,
(In bus, Enter) -> Real time info,
(Real time info, Enter) -> Thank you page,
(Query page, Enter) -> Result page,
(Result page, Enter) -> In bus,
(Real time info, Change fields) -> In bus,
(Query page, Change fields) -> At bus stop,
(Result page, Change fields) -> At bus stop }

VII. ER Diagram



VIII. API calls and Software tools

Google Distance Matrix API:
It is used to retrieve duration and distance values based on the recommended route between start and end points. We used this API in our application to find the nearest busstop from our current location.

Firebase:
The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client.We used this as our cloud Database for our application. For querying and updating the database we use firebase API calls. These calls are being made in the transition between first and second window and the last window. In the transition between the first and second window we make a firebase API call to access the database and fetch the necessary results. In the last window we make the API cal to update the database.

IX. Individual Contributions:

We had different modules:

1) Front-end (android,UI design)
2) Backend (Creating database, API calls and fetching query results)

We created database using PHP-SQL and we tried to do entire backend with PHP, then as we have mentioned firebase as our storage database in requirement specification, we changed the querying part from PHP to firebase.
Individual contributions are as follows:
Android Studio - Vamshi krishna, Vineeth, Rachit
PHP backend - Venketep prasad , Aravind
Firebase - Rachit,Vamshi
Design - Venketep prasad, Vineeth ,Aravind