Hindawi
Journal of Advanced Transportation
Volume 2020, Article ID 8895927, 18 pages
https://doi.org/10.1155/2020/8895927

Research Article

Design, Validation, and Comparative Analysis of a Private Bus Location Tracking Information System

Feras Al-Hawari, Mohammed Al-Sammarraie, and Taha Al-Khaffaf

Computer Engineering Department, German Jordanian University, Amman 11180, Jordan

Correspondence should be addressed to Feras Al-Hawari; firas.alhawari@gju.edu.jo

Received 26 April 2020; Revised 28 October 2020; Accepted 21 November 2020; Published 28 November 2020

Academic Editor: Tomio Miwa

This paper addresses various aspects related to the design, development, and validation of a web-based information system that is intended to facilitate the management of a bus transportation service offered by a Jordanian university to its staff and students. Passengers can use this system to track bus trips to find out how far a desired bus is from a specific location. Also, they can know about arrivals and departures of buses managed using this system. Specifically, this work explores UI design, data structures, database design, system architecture, and development methods to realize the required features (e.g., user roles, bus setup, driver assignment, bus routes, bus schedules, and trip monitoring) in the proposed bus location tracking system. It also suggests using the free open-source API, rather than the proprietary Google Maps API, to develop the interactive maps. The system also records trip information and solicits passenger feedback to allow reviewing and analyzing that data to enhance the quality of service, reduce operation cost, and improve passenger satisfaction. The conducted comparative analysis results illustrate that the open-source API is accurate, fast, and responsive similar to the proprietary API. Furthermore, the user survey output confirms that the deployed system is easy to use, helpful, fast, responsive, and accurate.

1. Introduction

Some universities in Jordan provide a bus transportation service to students and employees. Hence, that necessitates having a computerized system to manage features such as user accounts (e.g., passenger, driver, and administrator), user subscriptions, fee payments, bus setup (e.g., model, license plate, and capacity), driver assignment to buses, bus stops, bus routes, bus schedules, and bus trips. Moreover, the system should help passengers track any ongoing trips to check the possibility of catching a bus or finding out how far is the bus from a certain bus stop. Furthermore, it must allow university management to generate related reports (e.g., late trips report, punctual trips report, and unpaid fees report) and solicit user feedback (e.g., regarding the drivers' behavior and buses' comfortability) to evaluate service quality and passenger satisfaction. Accordingly, management can take proper actions (e.g., renew a bus and reduce fees) to improve the offered services as necessary.

In this context, the main aim of this paper is to tackle the challenges pertaining to the design and development of web-based bus management and tracking system that fulfills the needs of the German Jordanian University (GJU) (or any other institution with similar demands). Note that the system has been developed in-house to keep customizing it to meet continual university requirements such as the following:

(i) Track and record all bus trips in an efficient and accurate manner using a free open-source API.
(ii) Provide all basic system setup functionality such as managing drivers, buses, stops, routes, schedule, and trips.
(iii) Assign drivers to buses and assign buses to schedules without any conflicts (i.e., assigning two drivers to the same bus or assigning a bus to two overlapping schedules).
(iv) Integrate the system with the Student Information System (SIS) [1, 2], Human Resources (HR), and

Accounting Information System (AIS) [3, 4] to gain direct access to passenger (i.e., students or employees) profiles and financial data needed to retrieve the user account information (e.g., name, national number, and gender) and to check the payment of bus transportation fees (e.g., for login authorization purposes).

(v) Support single-sign-on- (SSO-) based authentication to allow users to log in to the system using the same credentials (i.e., a single username and password) that they use to log in to other university portals.

(vi) Allow passengers to evaluate the provided services (e.g., bus quality and driver behavior).

(vii) Offer reporting capabilities with flexible filtering criteria to enable management to assess the service quality and user satisfaction and then take suitable actions accordingly.

The rest of the paper is organized as follows. In Section 2, the literature review and main contributions of this work are presented. In Section 3, project management and software development processes as well as system design issues (e.g., UI, database, and architecture design) are addressed. In Section 4, the development details for the record trip, replay trip, detect schedules conflict, and authenticate user features are explained. The validation methods and comparative analysis results for the bus management system are illustrated in Section 5. Finally, the summary and limitations of this work as well as the planned future work are explored in Section 6.

2. Literature Review

The study in [5] proposed an ad hoc bus propagation model, taking into account bus overtaking and distributed passenger boarding behavior, to reduce bus bunching using holding control strategies for schedule and headway. Accordingly, bus bunching happens when two or more buses along the same route arrive at a designated stop simultaneously, and it is mainly related to the high uncertainty of bus systems running times, bus capacity, driving maneuvers (e.g., bus overtaking), and passenger boarding behavior. Holding control works by delaying buses at stops to regularize bus headway and reduce the overall passenger waiting time, possibly at the expense of extending on-board waiting time and total riding time. In the study in [6], a multiagent deep reinforcement learning framework to develop dynamic and flexible holding control strategies for a bus route is introduced to efficiently incorporate global coordination and long-term operation in bus holding. The work in [7] also investigated the effect of bus driver behavior on bus holding control strategies and more specifically their effort in catching up with a schedule in case of delay (i.e., schedule recovery). The research in [8] suggested a first schedule-following model where buses try to adhere to their schedule in a typical schedule-based public transport system. The model considers schedule following, bus bunching, and leapfrogging and uses observed automatic bus location and smart card data.

A flexible bus route optimization model for efficient public transportation systems based on multitarget stations was proposed in [9]. The model considers passenger demands, vehicle capacities, and transportation network and aims to find the optimal route while minimizing the vehicles' travel time. The optimization problem is solved using a heuristic algorithm based on a gravity model. The study in [10] introduced a dynamic bus scheduling model to find an optimal schedule that can adapt to the variations in passenger demands and traffic conditions and hence increase financial benefit and social satisfaction. The model aims to minimize passenger waiting time and maximize bus capacity utilization while taking into account the constraints due to bus capacity and the number of available buses, as well as ensuring that the service does not exceed the maximum anticipated headway. The study in [11] introduced a robust optimization model for limited-stop bus service with vehicle overtaking and demand dynamics, with an objective to minimize user cost and operation cost. A simulation-based optimization framework integrating the response surface methodology, which only needs to fit an initial input-output dataset before solving to optimality, is used to solve the stop-skipping optimization problem. The work in [12] discussed a heuristic framework to solve the bus routing and scheduling problem with transfer. The framework deals with the transportation of students from home to their school assuming that students may change buses on their way. The results show that allowing transfers reduces total operating costs significantly with user ride times comparable to solutions without transfers.

The study in [13] investigated a statistical method for correcting the systematic errors that occur in the estimation of bus arrival times. The errors are related to the cycle time to record each bus location and arrival information, as well as the information processing time. The method was verified by applying it to the bus information system of a city in Korea. The work in [14] resulted in a technique to predict bus arrival times based on the collected data from buses and the analysis of road conditions. The prediction is achieved using two algorithms that take advantage of historical data with current data. In the work in [15], a hybrid approach for predicting bus trajectories by integrating multiple predictors is proposed. The method minimizes the prediction error using a linear regression heuristic, and it was applied to five bus routes in different cities. In the study in [16], a methodology to predict bus arrival time using several artificial neural network models was developed. The models were built based on real-world automatic passenger counter data such as bus travel time and bus passenger load. The predicted arrival times were compared to the actual arrival times to evaluate the models' accuracy. A WiFi-sensing-based real-time bus tracking and arrival time prediction approach was discussed in [17]. The work in [18] suggested a system that utilizes the bus passengers' surrounding environmental context to estimate the bus traveling routes and predict bus arrival time at various bus stops. The study in [19] provided a two-step trip purpose labeling process for alighting station estimation based on transit smart card data. The process considers the station category, trip time, trip sequence, and alighting station frequency during five weekdays.

The work in [20] utilized massive historical coach trajectory data to extract coach operation information such as station location, station name, coach schedule, and driving route. Such data is important as it facilitates the realization of intelligent traffic information services. The main goal of the work in [21] is to test point data visualization possibilities of selected JavaScript Mapping Libraries to measure their performance and ability to cope with a big amount of data. Five libraries for marker clustering and two libraries for heat map visualization were analyzed. Loading time and the ability to visualize large datasets were compared for each dataset and each library. The study in [22] focused on the design of a bus ticketing system. The studies in [23, 24] discussed the design of RFID hardware systems to monitor buses.

The aforementioned literature review showed that the main challenges in this field are related to finding methods to collect accurate bus propagation and passenger demands information and then to use such data in the development of intelligent bus management applications that aim to increase bus capacity utilization, reduce bus operation cost, and improve passenger satisfaction. Specifically, the studies in [5–24] addressed particular issues related to bus management applications (e.g., reducing bus bunching, finding optimal schedules, predicting bus arrival time, collecting coach operation information, and designing bus tracking hardware) but did not discuss design aspects pertaining to the software development of bus management information systems as a whole. On the other hand, the software engineering aspects to develop bus management information systems that focus on bus trip tracking were introduced in [25–37], but only the systems in [25–27] discussed driver, bus, and route setup steps besides trip tracking.

Correspondingly, this work has three main contributions. First, it proposes design and development methods to realize all the needed features (e.g., account management, bus setup, route schedules, driver assignment, and trip monitoring) in a bus location tracking information system. Second, the proposed system utilizes the free open-source Leaflet library API [38] to present interactive maps, whereas the majority of the related systems used the no longer free (i.e., proprietary) Google Maps API [39, 40] for that purpose. In that regard, a comparative analysis is also conducted to evaluate the aforementioned APIs and illustrate that the open-source API is accurate, efficient, and responsive like its counterpart, in addition to being more adaptable due to its open-source nature. Third, unlike most similar work, this system records the trip information and solicits the passenger feedback to allow reviewing and analyzing that data to enhance the provided services and hence reduce operation cost as well as improving passenger satisfaction.

3. Project Management and Software Design Methods

The adopted research methodology steps are summarized in Figure 1. The project management and development processes, as well as the software analysis and design methods to realize the bus location tracking system, are described in this section, whereas some of the technology development techniques and system validation measures are discussed in Sections 4 and 5, respectively. Specifically, this section covers four main topics. First, suitable project management and software development processes to deliver this project are selected. Second, the users of the system are identified and the services offered to them are specified. Third, the database tables and data structures needed to implement the various features are proposed. Finally, the software modules needed in the adopted three-tier web-based system architecture are discussed.

3.1. Project Management and Software Development Processes.

The project management process in the systems engineering basic profile [41] was adopted to manage this project. This process is applicable in this context as it addresses important project management aspects like project planning, progress, review, versioning, and risks, whereas the iterative and incremental software development process is suitable to develop the various system modules. Based on that, several repeated waterfall cycles (i.e., iterations) have been used to implement the bus management modules (i.e., increments).

3.2. User Role Identification and System View Specification.

The system needs to support three user roles: admin, driver, and passenger. Accordingly, it must offer three views to provide related services to each user role. The main features in the admin view are summarized in the use case diagram in Figure 2, whereas the services offered to the driver and passenger roles are shown in the use case diagram in Figure 3. The services offered to each of the three user roles are discussed next.

3.2.1. Admin View.

The admin view supports management features such as system setup (e.g., buses, drivers, stops, routes, and schedules), trips monitoring (i.e., tracking and replay), and feedback evaluation (i.e., user trip reviews). The offered services in the admin view are as follows:

(1) *Manage Drivers*. This feature allows managing (i.e., adding, editing, deleting, and viewing): driver information (e.g., name, national number, address, and phone number) in the system. Such data can be used for management and reporting purposes. For example, knowing the address of a driver helps the admin in assigning the driver to a bus route that is closer to his/her place of residence.

(2) *Manage Buses*. This capability is needed to set up the buses' information (e.g., manufacturer, model, license plate, chassis number, and capacity) in the system.

(3) *Assign Drivers to Buses*. The system supports assigning one driver per bus. Hence, to assign a new driver to a bus, the original driver has to be unassigned first.

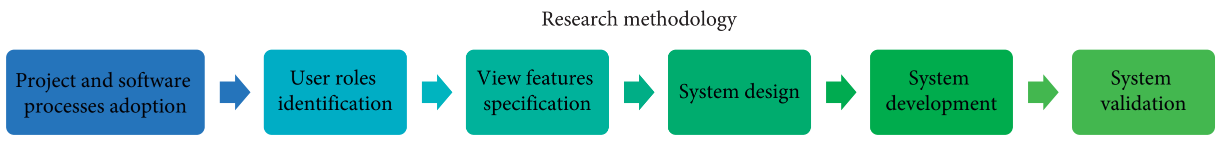

Research methodology
Project and software processes adoption
User roles identification
View features specification
System design
System development
System validation

Figure 1: The adopted research methodology steps.

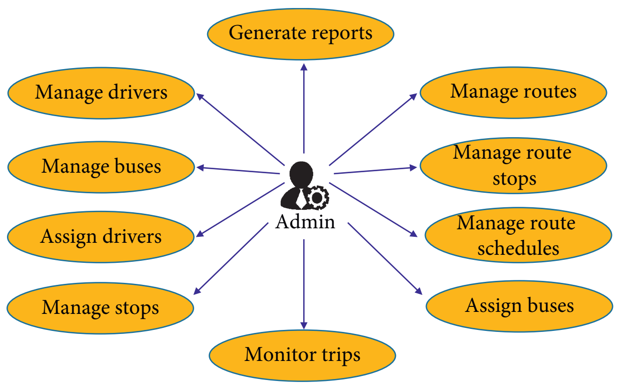


Figure 2: The use case diagram for the main features in the admin view.

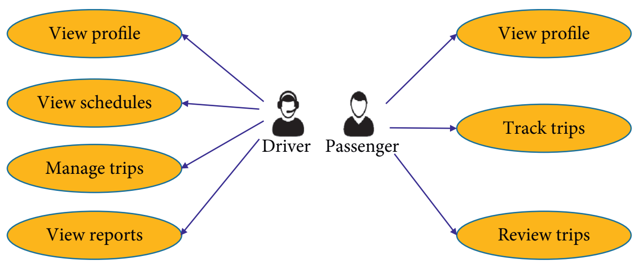


Figure 3: The use case diagram for the main features for the driver and passenger user roles.

(4) *Manage Stops*. This service is used to define the bus stops that can be used to specify the different bus routes. Each bus stop is identified by the name of the place where the buses stop to let passengers get on and off.

(5) *Manage Routes*. The admin view permits managing bus routes using the screen shown in Figure 4. A bus route consists of two (i.e., the source and destination) or more bus stops (see Figure 5) and has a unique code. The bus route can be associated with several schedules. The route schedule specifies the expected arrival time of the bus to each stop along the route as shown in Figure 5. Each route schedule has a unique identifier. Also, a bus must be assigned to each schedule. Further, a route can be associated with several identical schedules as long as they are assigned different buses. Accordingly, the assigned buses take trips on a daily basis on the designated schedule times. Based on the example in Figure 5, *Bus 1*, *Bus 2*, and *Bus 3* take the trips with *Id 1* and *Id 2*, the trip with *Id 3*, and the trip with *Id 4*, respectively, on a daily basis as shown in Figure 6. Note that the bus trips will continue daily as long as their route schedules remain active.

(6) *Manage Route Stops*. The system allows managing the bus stops for each route. Each stop has a different order than its counterparts to enable the system to display the stops based on their defined order.

(7) *Manage Route Schedules*. The system allows managing the bus route schedules via the screen shown in Figure 7. Note that it checks whether the arrival time to a certain bus stop is greater than or equal to the arrival time to the previous bus stop in a given bus route schedule.

(8) *Assign Buses to Route Schedules*. A specific bus should be assigned to each bus route schedule via the screen shown in Figure 7. The same bus may be assigned to multiple bus route schedules as long as there are no time overlaps between those schedules. When a bus is being assigned to a schedule, the system automatically checks whether or not that bus is assigned to an overlapping schedule to avoid conflicts or proceed with the operation (see subsection 4.3).

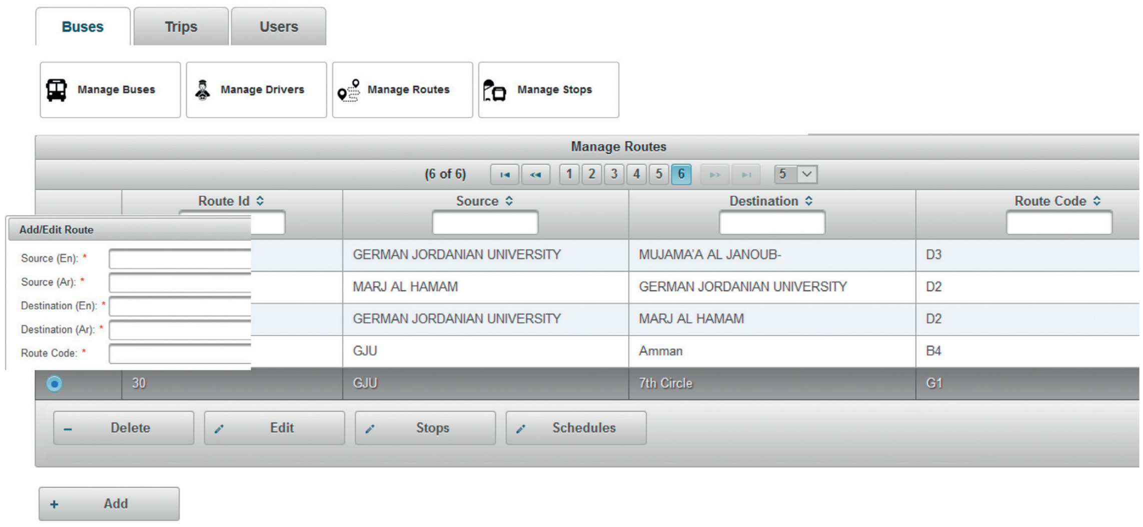
(9) *Monitor Trips*. The admin can also track an ongoing bus trip or replay a past bus trip, if needed. For that purpose, the system displays the selected trip map and keeps updating the marker position to point at the current or recorded bus position (i.e., longitude and latitude values) at a given time (see Figure 8).

(10) *Generate Reports*. The admin may generate reports with flexible filtering criteria (see Figure 9) to display trip details such as driver name, bus information, departure time, arrival time, route stops, trip status, and trip reviews. Accordingly, the admin can evaluate the bus punctuality and the passenger satisfaction. Then, the evaluation reports can be shared with university management for needed action (e.g., keep the bus transportation company, replace a bus, or reward a driver).

3.2.2. Driver View. The driver view offers the following features:

(1) *View Profile*. A driver can view his/her information (e.g., picture, name, national number, and phone number).

(2) *View Schedules*. This feature displays the bus route schedules that are assigned to a driver as shown in Figure 10. A driver can start an on-time trip (i.e., highlighted in green) or a late trip (i.e., highlighted in



Manage Routes

(6 of 6) 1 2 3 4 5 6 5

Add/Edit Route

Source (En): *

Source (Ar): *

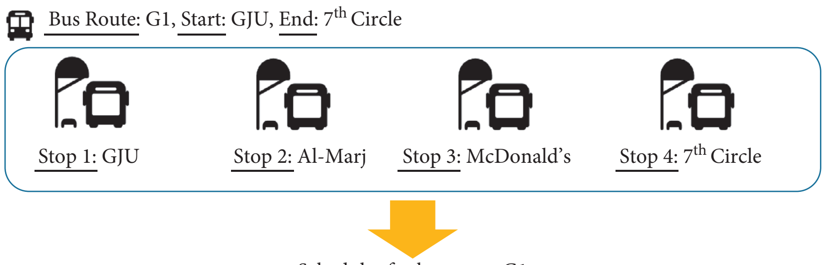
Destination (En): *

Destination (Ar): *

Route Code: *

	Route Id	Source	Destination	Route Code
		GERMAN JORDANIAN UNIVERSITY	MUJAMA'A AL JANOUB-	D3
		MARJ AL HAMAM	GERMAN JORDANIAN UNIVERSITY	D2
		GERMAN JORDANIAN UNIVERSITY	MARJ AL HAMAM	D2
		GJU	Amman	B4
●	30	GJU	7th Circle	G1

Figure 4: The manage routes screen in the admin view.



Schedules for bus route G1

Schedule Id	Assigned bus	Stops			
		GJU	Al-Marj	McDonald's	7th Circle
1	Bus1	8:00 AM	8:15 AM	8:45 AM	9:00 AM
2	Bus1	11:00 AM	11:15 AM	11:45 AM	12:00 PM
3	Bus2	8:00 AM	8:15 AM	8:45 AM	9:00 AM
4	Bus3	1:00 PM	1:15 PM	1:45 PM	2:00 PM

Figure 5: An example showing a bus route with its stops and schedules.

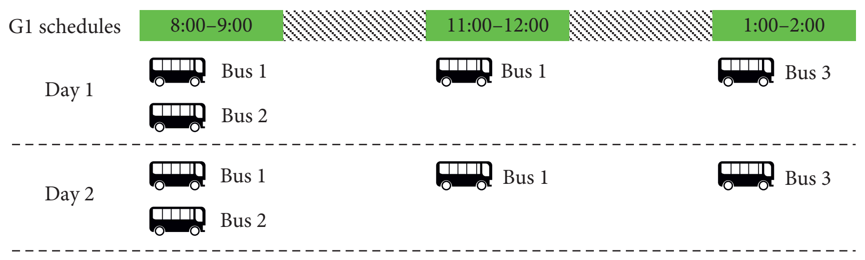
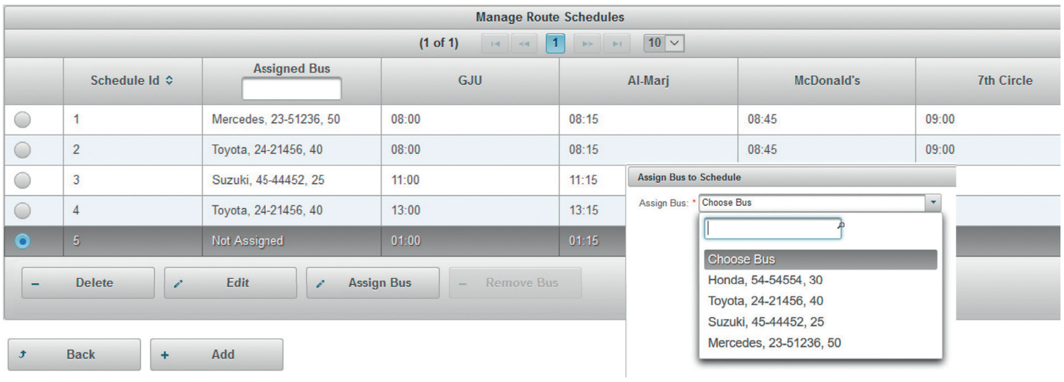


Figure 6: The different bus trips on day 1 and day 2 based on the bus route schedules in Figure 5.

orange) but cannot initiate a too early trip (i.e., highlighted in red). Note that this page (and the other pages in the driver view) is designed to be responsive such that it is navigable and viewable in screens with different sizes.

(3) *Manage Trips*. Once a trip is started, the driver can see a trip map with a marker that keeps changing place to point at the current bus location as shown in Figure 8. In the background, the system periodically saves the current bus location (i.e., current longitude and latitude) for the trip in the database table shown in Figure 11 to allow users (e.g., admin and passengers) to track or replay the trip. The driver must end the trip (to stop the data recording) when the bus arrives at its final destination. Further, a trip must be canceled in case it was started by mistake.



Manage Route Schedules

(1 of 1)

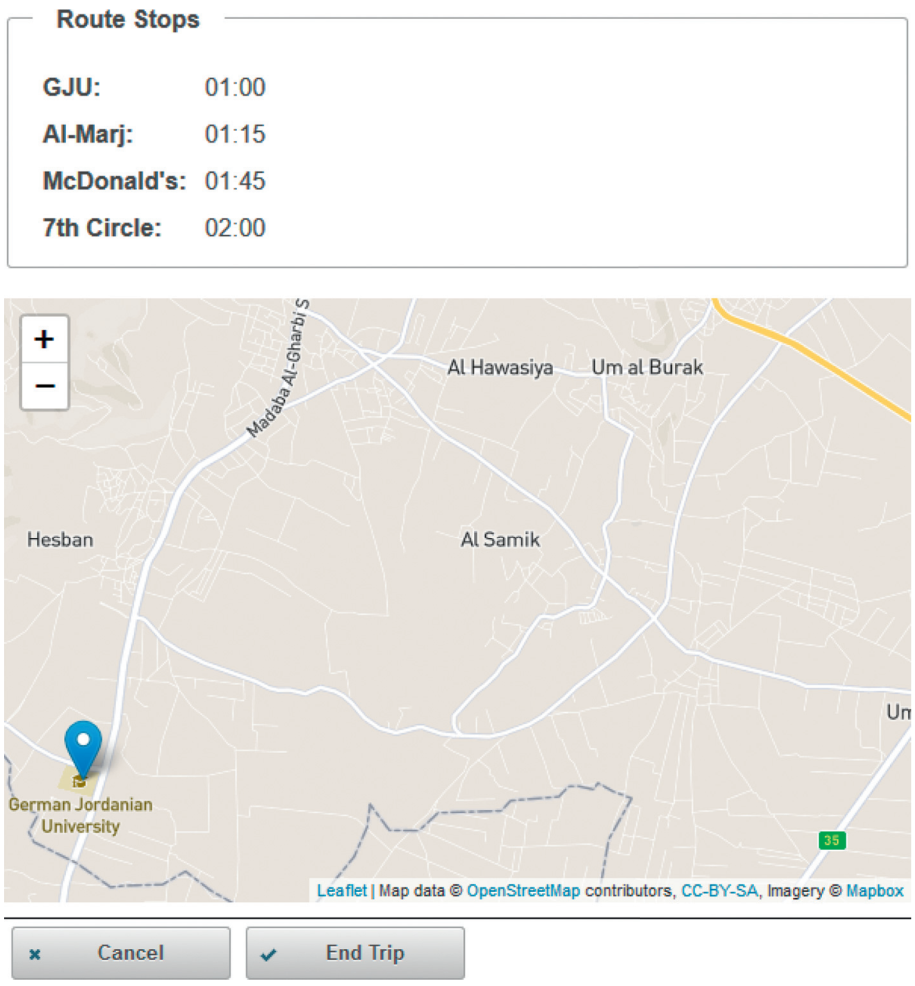
	Schedule Id	Assigned Bus	GJU	Al-Marj	McDonald's	7th Circle
○	1	Mercedes, 23-51236, 50	08:00	08:15	08:45	09:00
○	2	Toyota, 24-21456, 40	08:00	08:15	08:45	09:00
○	3	Suzuki, 45-44452, 25	11:00	11:15		
○	4	Toyota, 24-21456, 40	13:00	13:15		
●	5	Not Assigned	01:00	01:15		

Assign Bus to Schedule

Assign Bus: * Choose Bus

Choose Bus
Honda, 54-54554, 30
Toyota, 24-21456, 40
Suzuki, 45-44452, 25
Mercedes, 23-51236, 50

Figure 7: The manage route schedule screen in the admin view.



Route Stops

GJU:	01:00
Al-Marj:	01:15
McDonald's:	01:45
7th Circle:	02:00

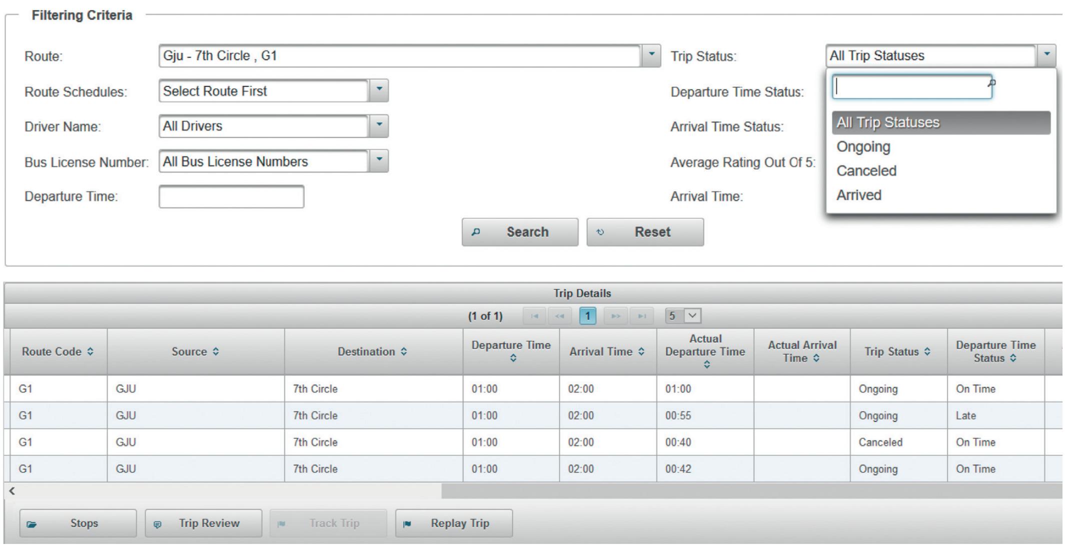
Al Hawasiya
Um al Burak
Madaba Al-Gharbi S
Hesban
Al Samik
German Jordanian University
35
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Imagery © Mapbox
Cancel
End Trip

Figure 8: The manage trip screen in the driver view.

(4) *View Reports*. This feature allows displaying the details (e.g., bus information, departure time, arrival time, route stops, and trip status) of the trips that were initiated by the driver.

3.2.3. Passenger View. The passenger view supports the following three services:

(1) *View Profile*. A passenger can view his/her profile. In case the passenger is a student, the profile would contain information retrieved from the SIS database such as name, student ID, degree, and major. On the other hand, when the passenger is an employee, the profile would contain data obtained from the HR database such as name, employee ID, job title, and department.



Filtering Criteria

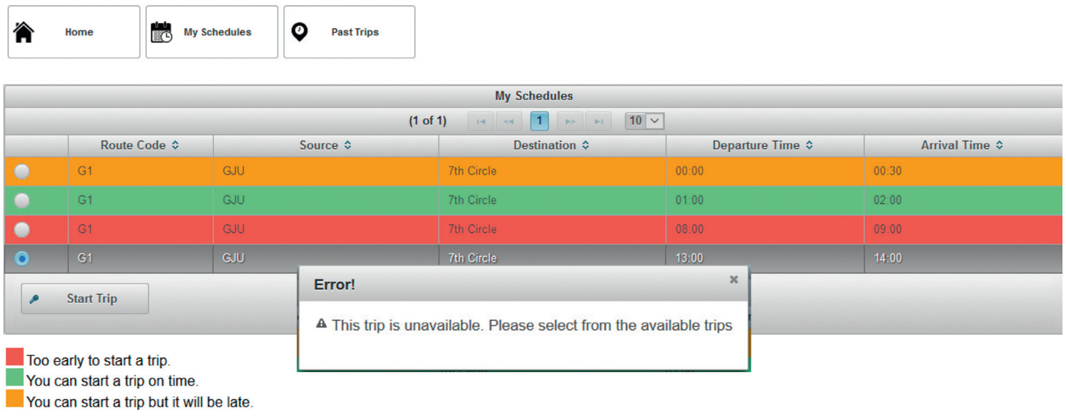
Route:	Gju - 7th Circle , G1	Trip Status:	All Trip Statuses
Route Schedules:	Select Route First	Departure Time Status:	
Driver Name:	All Drivers	Arrival Time Status:	
Bus License Number:	All Bus License Numbers	Average Rating Out Of 5:	
Departure Time:		Arrival Time:	

Trip Details

(1 of 1)

Route Code	Source	Destination	Departure Time	Arrival Time	Actual Departure Time	Actual Arrival Time	Trip Status	Departure Time Status
G1	GJU	7th Circle	01:00	02:00	01:00		Ongoing	On Time
G1	GJU	7th Circle	01:00	02:00	00:55		Ongoing	Late
G1	GJU	7th Circle	01:00	02:00	00:40		Canceled	On Time
G1	GJU	7th Circle	01:00	02:00	00:42		Ongoing	On Time

Figure 9: The trips report in the admin view.



My Schedules

(1 of 1)

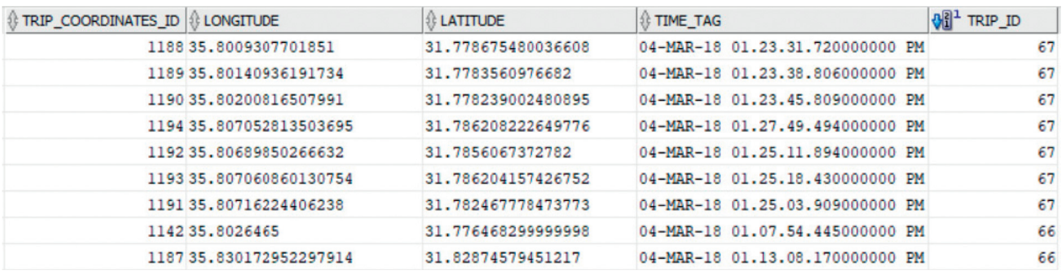
	Route Code	Source	Destination	Departure Time	Arrival Time
○	G1	GJU	7th Circle	00:00	00:30
○	G1	GJU	7th Circle	01:00	02:00
○	G1	GJU	7th Circle	08:00	09:00
●	G1	GJU	7th Circle	13:00	14:00

Error!

This trip is unavailable. Please select from the available trips

🟥 Too early to start a trip.
🟩 You can start a trip on time.
🟧 You can start a trip but it will be late.

Figure 10: The schedule screen in the driver view.

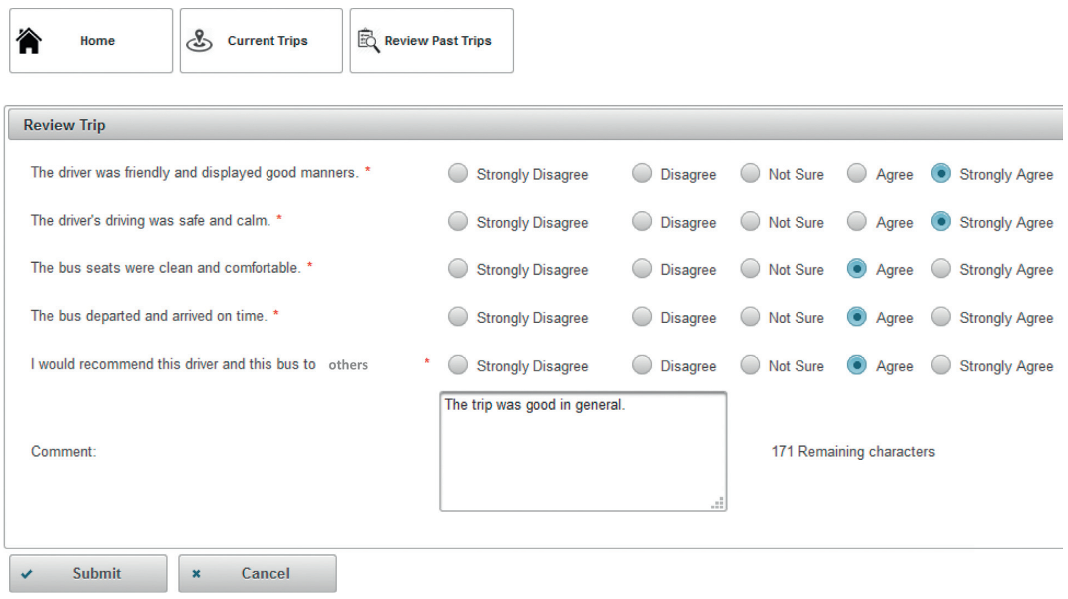


TRIP_COORDINATES_ID	LONGITUDE	LATITUDE	TIME_TAG	TRIP_ID
1188	35.8009307701851	31.778675480036608	04-MAR-18 01.23.31.720000000 PM	67
1189	35.80140936191734	31.7783560976682	04-MAR-18 01.23.38.806000000 PM	67
1190	35.80200816507991	31.778239002480895	04-MAR-18 01.23.45.809000000 PM	67
1194	35.807052813503695	31.786208222649776	04-MAR-18 01.27.49.494000000 PM	67
1192	35.80689850266632	31.7856067372782	04-MAR-18 01.25.11.894000000 PM	67
1193	35.807060860130754	31.786204157426752	04-MAR-18 01.25.18.430000000 PM	67
1191	35.80716224406238	31.782467778473773	04-MAR-18 01.25.03.909000000 PM	67
1142	35.8026465	31.776468299999998	04-MAR-18 01.07.54.445000000 PM	66
1187	35.830172952297914	31.82874579451217	04-MAR-18 01.13.08.170000000 PM	66

Figure 11: The trip coordinates database table.

(2) *Track Trips*. A passenger may track any ongoing trip to check the possibility of catching a bus or finding out how far is the bus from a bus stop. The system displays the selected trip map and keeps updating the marker position to point at the bus position at the current time (see Figure 8). Noting that this page (and the other pages in the passenger view) is designed to be responsive such that it is navigable and viewable in screens with different sizes.

(3) *Review Trips*. A passenger can provide feedback regarding any trip using the trip review form shown in Figure 12. The feedback is related to issues such as driver's behavior, driver's driving, bus quality (e.g., comfortability and cleanliness), trip punctuality, and general comments. The system admin can evaluate the user feedback to make sure passengers are satisfied with the provided bus transportation service.



Review Trip

The driver was friendly and displayed good manners. *	Strongly Disagree	Disagree	Not Sure	Agree	[✓ Strongly Agree]
The driver's driving was safe and calm. *	Strongly Disagree	Disagree	Not Sure	Agree	[✓ Strongly Agree]
The bus seats were clean and comfortable. *	Strongly Disagree	Disagree	Not Sure	[✓ Agree]	Strongly Agree
The bus departed and arrived on time. *	Strongly Disagree	Disagree	Not Sure	[✓ Agree]	Strongly Agree
I would recommend this driver and this bus to others *	Strongly Disagree	Disagree	Not Sure	[✓ Agree]	Strongly Agree

Comment: The trip was good in general. 171 Remaining characters

Figure 12: The review trip form in the passenger view.

3.3. Database Design. For example, the ER diagram for the major tables that are utilized by the bus management system is shown in Figure 13. Accordingly, the drivers, buses, and stops information are stored in the DRIVERS, BUSES, and STOPS tables, respectively. The DRIVER_ID and BUS_ID are saved in the BUSES_DRIVERS table when a driver is assigned to a bus. The bus routes information (e.g., code, source, and destination) are saved in the ROUTES table. A bus route is associated with several bus stops in the ROUTES_STOPS table. The bus routes' schedules are stored in the ROUTES_SCHEDULES table. The time a bus arrives at a bus schedule stop is saved in the ROUTES_-STOPS_SCHEDULES table. A bus is assigned to a schedule in the BUSES_SCHEDULES table. The trip summary is saved in the TRIPS table. The latitude and longitude values of each recorded position in a bus trip are stored in the TRIPS_COORDINATES table (also see Figure 11).

3.4. System Architecture. The bus management system is a web-based Java EE application that is separated into client, web, and data tiers as shown in Figure 14. In the client tier, users (i.e., clients) would access the system via web browsers running on computers or phones. In the web tier, the system modules are deployed and executed in a web container that resides within the Java EE server (running on a remote host in a data center). In the data tier, the buses, SIS, HR, and AIS databases are managed by the Relational Database Management System (RDBMS). A software module (e.g., manage buses, manage routes, and manage stops) consists of Java-Server Faces (JSF) [42] pages, managed beans, and Data Access Objects (DAOs). A JSF file includes HTML and JSF elements to represent the UI components (e.g., tables, text fields, checkboxes, and menus) in the related module. A managed bean is a Java object that is automatically managed (i.e., instantiated, executed, updated, and destroyed) by the web container. It saves the state of the corresponding JSF page and implements the business logic of the page. The JSF framework offers more services to support data validation and conversion, event handling, page templates, and language translation. It also allows the use of custom UI component libraries like PrimeFaces [43] within the JSF pages. A DAO is used by a bean to manage the application information in the data tier. For that purpose, it utilizes the Java Database Connectivity (JDBC) API to interact with the RDBMS to retrieve data from, or store data in, the desired database tables.

4. Development Methods

The development details for some of the main features within the bus management system are discussed next.

4.1. Trip Tracking and Recording. The sequence diagram in Figure 15 illustrates the developed methods to display and record a bus trip map. In that regard, when a driver starts a trip, the trip map page will be rendered every 15 seconds until the trip is ended. Given that the latitude and longitude values of the bus location are collected via JavaScript functions in the client tier, therefore the *inputHidden* elements have been used in the map's JSF page to allow storing these values in a managed bean named *mapBean* whenever the page is requested by the browser. The JSF framework invokes the *setLat* and *setLng* methods on the *mapBean* object to store the received latitude and longitude values, respectively. It also records the captured coordinates in the database by first invoking the *saveLatLng* method on the *mapBean* object. In turn, the *mapBean* object invokes the *insertLatLng* method on the related DAO, which then uses the Java JDBC API to communicate with the RDBMS to insert the coordinates of the current bus location in the TRIPS_COORDINATES table. Consequently, the JSF framework generates the HTML tags and JavaScript code for

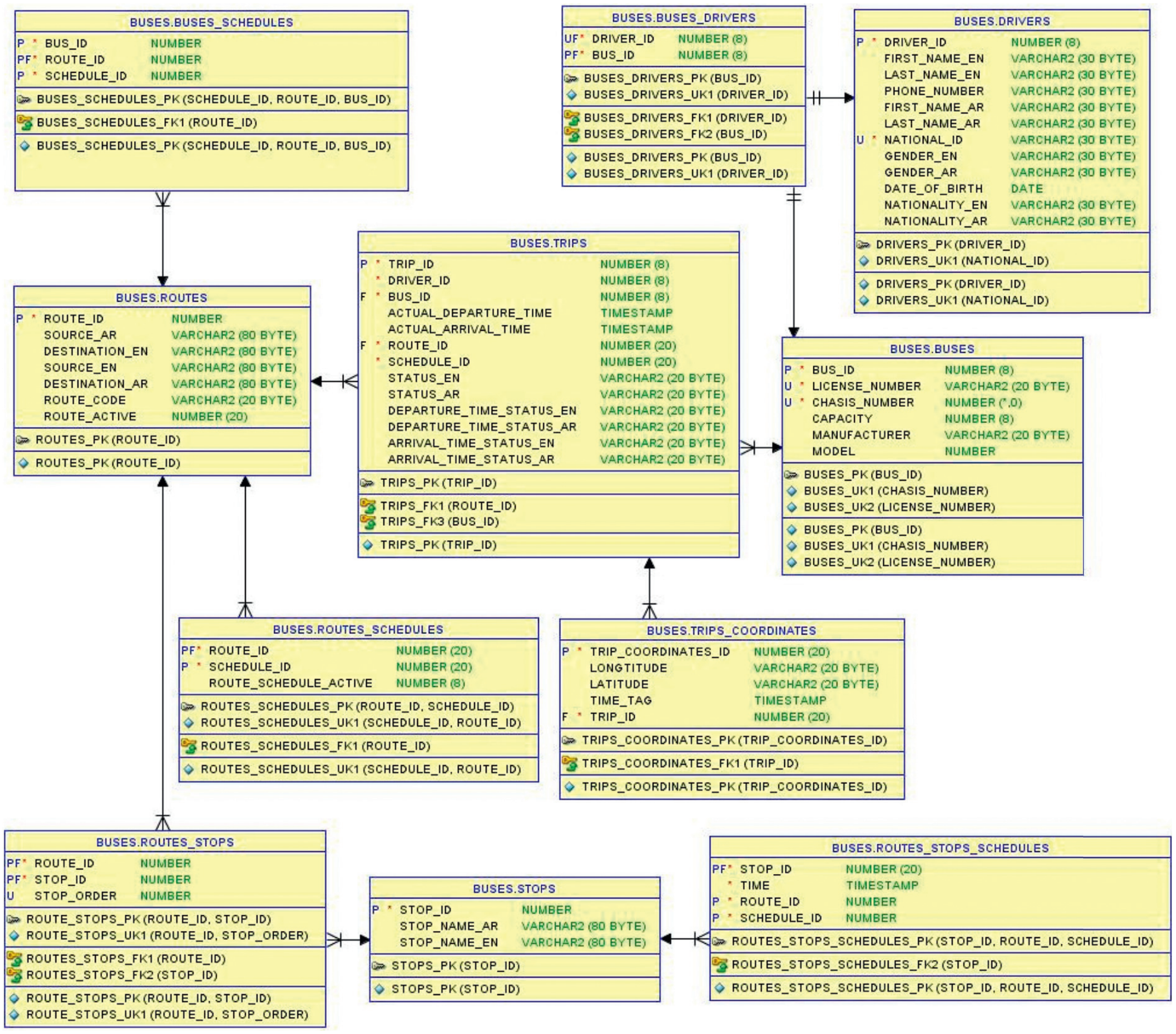

BUSES.BUSES_SCHEDULES
P * BUS_ID NUMBER
PF* ROUTE_ID NUMBER
P * SCHEDULE_ID NUMBER
BUSES_SCHEDULES_PK (SCHEDULE_ID, ROUTE_ID, BUS_ID)
BUSES_SCHEDULES_FK1 (ROUTE_ID)
BUSES_SCHEDULES_PK (SCHEDULE_ID, ROUTE_ID, BUS_ID)
BUSES.BUSES_DRIVERS
UF* DRIVER_ID NUMBER (8)
PF* BUS_ID NUMBER (8)
BUSES_DRIVERS_PK (BUS_ID)
BUSES_DRIVERS_UK1 (DRIVER_ID)
BUSES_DRIVERS_FK1 (DRIVER_ID)
BUSES_DRIVERS_FK2 (BUS_ID)
BUSES_DRIVERS_PK (BUS_ID)
BUSES_DRIVERS_UK1 (DRIVER_ID)
BUSES.DRIVERS
P * DRIVER_ID NUMBER (8)
FIRST_NAME_EN VARCHAR2 (30 BYTE)
LAST_NAME_EN VARCHAR2 (30 BYTE)
PHONE_NUMBER VARCHAR2 (30 BYTE)
FIRST_NAME_AR VARCHAR2 (30 BYTE)
LAST_NAME_AR VARCHAR2 (30 BYTE)
U * NATIONAL_ID VARCHAR2 (30 BYTE)
GENDER_EN VARCHAR2 (30 BYTE)
GENDER_AR VARCHAR2 (30 BYTE)
DATE_OF_BIRTH DATE
NATIONALITY_EN VARCHAR2 (30 BYTE)
NATIONALITY_AR VARCHAR2 (30 BYTE)
DRIVERS_PK (DRIVER_ID)
DRIVERS_UK1 (NATIONAL_ID)
DRIVERS_PK (DRIVER_ID)
DRIVERS_UK1 (NATIONAL_ID)
BUSES.TRIPS
P * TRIP_ID NUMBER (8)
* DRIVER_ID NUMBER (8)
F * BUS_ID NUMBER (8)
ACTUAL_DEPARTURE_TIME TIMESTAMP
ACTUAL_ARRIVAL_TIME TIMESTAMP
F * ROUTE_ID NUMBER (20)
* SCHEDULE_ID NUMBER (20)
STATUS_EN VARCHAR2 (20 BYTE)
STATUS_AR VARCHAR2 (20 BYTE)
DEPARTURE_TIME_STATUS_EN VARCHAR2 (20 BYTE)
DEPARTURE_TIME_STATUS_AR VARCHAR2 (20 BYTE)
ARRIVAL_TIME_STATUS_EN VARCHAR2 (20 BYTE)
ARRIVAL_TIME_STATUS_AR VARCHAR2 (20 BYTE)
TRIPS_PK (TRIP_ID)
TRIPS_FK1 (ROUTE_ID)
TRIPS_FK3 (BUS_ID)
TRIPS_PK (TRIP_ID)
BUSES.ROUTES
P * ROUTE_ID NUMBER
SOURCE_AR VARCHAR2 (80 BYTE)
DESTINATION_EN VARCHAR2 (80 BYTE)
SOURCE_EN VARCHAR2 (80 BYTE)
DESTINATION_AR VARCHAR2 (80 BYTE)
ROUTE_CODE VARCHAR2 (20 BYTE)
ROUTE_ACTIVE NUMBER (20)
ROUTES_PK (ROUTE_ID)
ROUTES_PK (ROUTE_ID)
BUSES.BUSES
P * BUS_ID NUMBER (8)
U * LICENSE_NUMBER VARCHAR2 (20 BYTE)
U * CHASIS_NUMBER NUMBER (*,0)
CAPACITY NUMBER (8)
MANUFACTURER VARCHAR2 (20 BYTE)
MODEL NUMBER
BUSES_PK (BUS_ID)
BUSES_UK1 (CHASIS_NUMBER)
BUSES_UK2 (LICENSE_NUMBER)
BUSES_PK (BUS_ID)
BUSES_UK1 (CHASIS_NUMBER)
BUSES_UK2 (LICENSE_NUMBER)
BUSES.ROUTES_SCHEDULES
PF* ROUTE_ID NUMBER (20)
P * SCHEDULE_ID NUMBER (20)
ROUTE_SCHEDULE_ACTIVE NUMBER (8)
ROUTES_SCHEDULES_PK (ROUTE_ID, SCHEDULE_ID)
ROUTES_SCHEDULES_UK1 (SCHEDULE_ID, ROUTE_ID)
ROUTES_SCHEDULES_FK1 (ROUTE_ID)
ROUTES_SCHEDULES_UK1 (SCHEDULE_ID, ROUTE_ID)
BUSES.TRIPS_COORDINATES
P * TRIP_COORDINATES_ID NUMBER (20)
LONGTITUDE VARCHAR2 (20 BYTE)
LATITUDE VARCHAR2 (20 BYTE)
TIME_TAG TIMESTAMP
F * TRIP_ID NUMBER (20)
TRIPS_COORDINATES_PK (TRIP_COORDINATES_ID)
TRIPS_COORDINATES_FK1 (TRIP_ID)
TRIPS_COORDINATES_PK (TRIP_COORDINATES_ID)
BUSES.ROUTES_STOPS
PF* ROUTE_ID NUMBER
PF* STOP_ID NUMBER
U STOP_ORDER NUMBER
ROUTE_STOPS_PK (ROUTE_ID, STOP_ID)
ROUTE_STOPS_UK1 (ROUTE_ID, STOP_ORDER)
ROUTES_STOPS_FK1 (ROUTE_ID)
ROUTES_STOPS_FK2 (STOP_ID)
ROUTE_STOPS_PK (ROUTE_ID, STOP_ID)
ROUTE_STOPS_UK1 (ROUTE_ID, STOP_ORDER)
BUSES.STOPS
P * STOP_ID NUMBER
STOP_NAME_AR VARCHAR2 (80 BYTE)
STOP_NAME_EN VARCHAR2 (80 BYTE)
STOPS_PK (STOP_ID)
STOPS_PK (STOP_ID)
BUSES.ROUTES_STOPS_SCHEDULES
PF* STOP_ID NUMBER (20)
* TIME TIMESTAMP
P * ROUTE_ID NUMBER
P * SCHEDULE_ID NUMBER
ROUTES_STOPS_SCHEDULES_PK (STOP_ID, ROUTE_ID, SCHEDULE_ID)
ROUTES_STOPS_SCHEDULES_FK2 (STOP_ID)
ROUTES_STOPS_SCHEDULES_PK (STOP_ID, ROUTE_ID, SCHEDULE_ID)

FIGURE 13: The ER diagram for the major tables used by the bus management system.

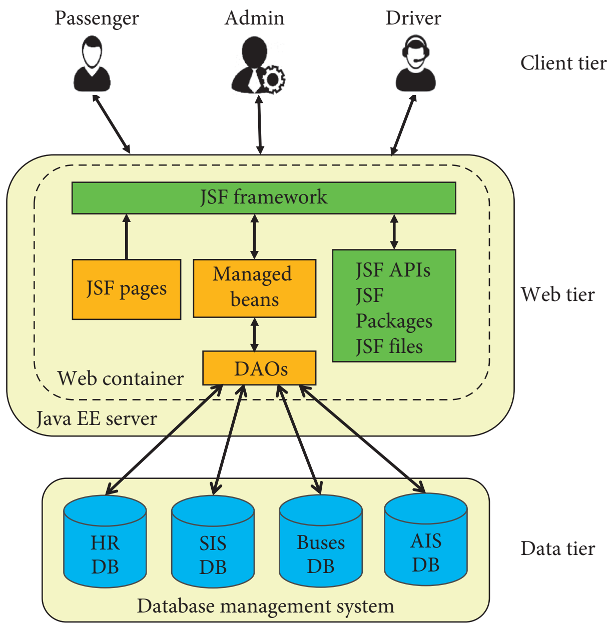


FIGURE 14: The three-tier architecture of the bus management system.

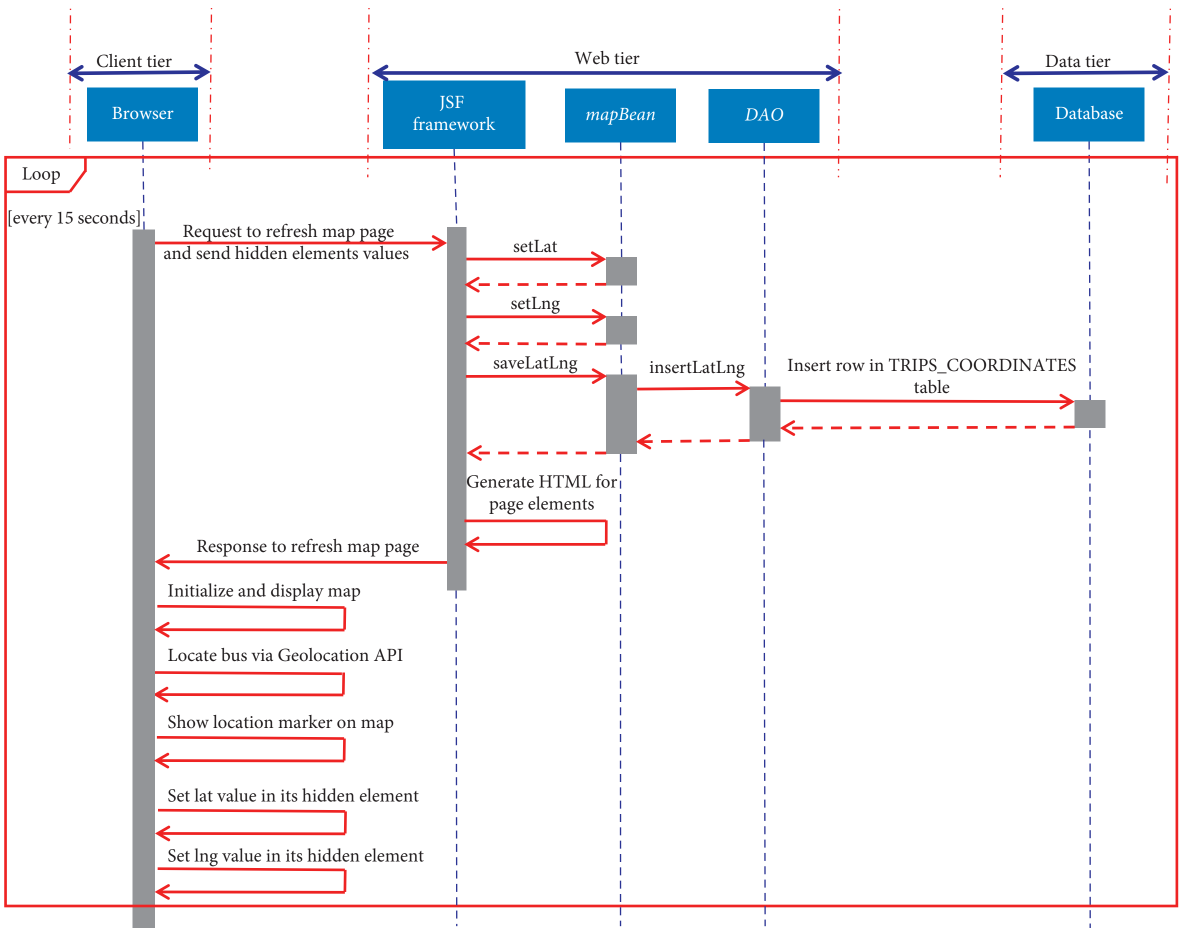

Client tier
Web tier
Data tier
Browser
JSF framework
mapBean
DAO
Database
Loop
[every 15 seconds]
Request to refresh map page and send hidden elements values
setLat
setLng
saveLatLng
insertLatLng
Insert row in TRIPS_COORDINATES table
Generate HTML for page elements
Response to refresh map page
Initialize and display map
Locate bus via Geolocation API
Show location marker on map
Set lat value in its hidden element
Set lng value in its hidden element

Figure 15: The sequence diagram for the bus trip tracking and recording code interactions.

the elements in the map page and then sends the page within an HTTP response message to the browser. The Leaflet library API [38] is utilized by the JavaScript code that is executed by the browser to initialize and display the interactive map. Next, the Geolocation API is used by the Leaflet *locate* function to capture the coordinates of the current bus location. Then, a marker for the current bus coordinates is added and displayed on the map. Finally, the latitude and longitude values of the current bus location are saved in the values of the *inputHidden* elements for processing upon submitting the form data of the map page to the web application server.

4.2. Trip Replay. The sequence diagram in Figure 16 illustrates the developed methods to replay a saved bus trip. When the trip page is requested by the browser, the JSF framework invokes the *getTripCoords* method on the *mapBean* object to retrieve the markers information of the desired trip for replay. In turn, the *mapBean* object invokes the *buildTripCoords* method on the related DAO, which then uses the Java JDBC API to communicate with the RDBMS to retrieve the coordinates' rows of the desired trip from the TRIPS_COORDINATES table. Consequently, the JSF framework generates JavaScript code to display the trip's recorded markers on the map. Next, it generates HTML tags and JavaScript code for the rest of the elements on the page. Finally, it encapsulates the generated page within an HTTP response message that is sent to the browser. The Leaflet library API is utilized by the JavaScript code that is executed by the browser to initialize and display the interactive map. Next, all the markers for the replayed trip are added and displayed on the map.

4.3. Schedule Conflict Detection. The system automatically detects schedule conflict when attempting to assign a bus to an overlapping schedule as shown in Figure 17. The pseudocode for the schedule conflict detection algorithm is shown in Figure 18. The algorithm is based on the assumption that each driver is assigned to only one bus as stated at line 1. The departure time and arrival time of the route schedule (i.e., *selSched*) to which the bus is being assigned are computed at line 2 and line 3, respectively. The schedules to which the selected bus is already assigned are retrieved from the BUSES_SCHEDULES table and then saved in the *busSchedulesArray* object at line 4 and line 5, respectively. The *foreach* block that starts at line 6 and ends at line 21 checks each bus schedule (i.e., *busSched*) in the *busSchedulesArray* object for conflicts. The stop schedules for the current *busSched* are retrieved from the ROUTES_STOPS_SCHEDULES table and then saved in the *busStopSchedulesArray* object at line 7 and

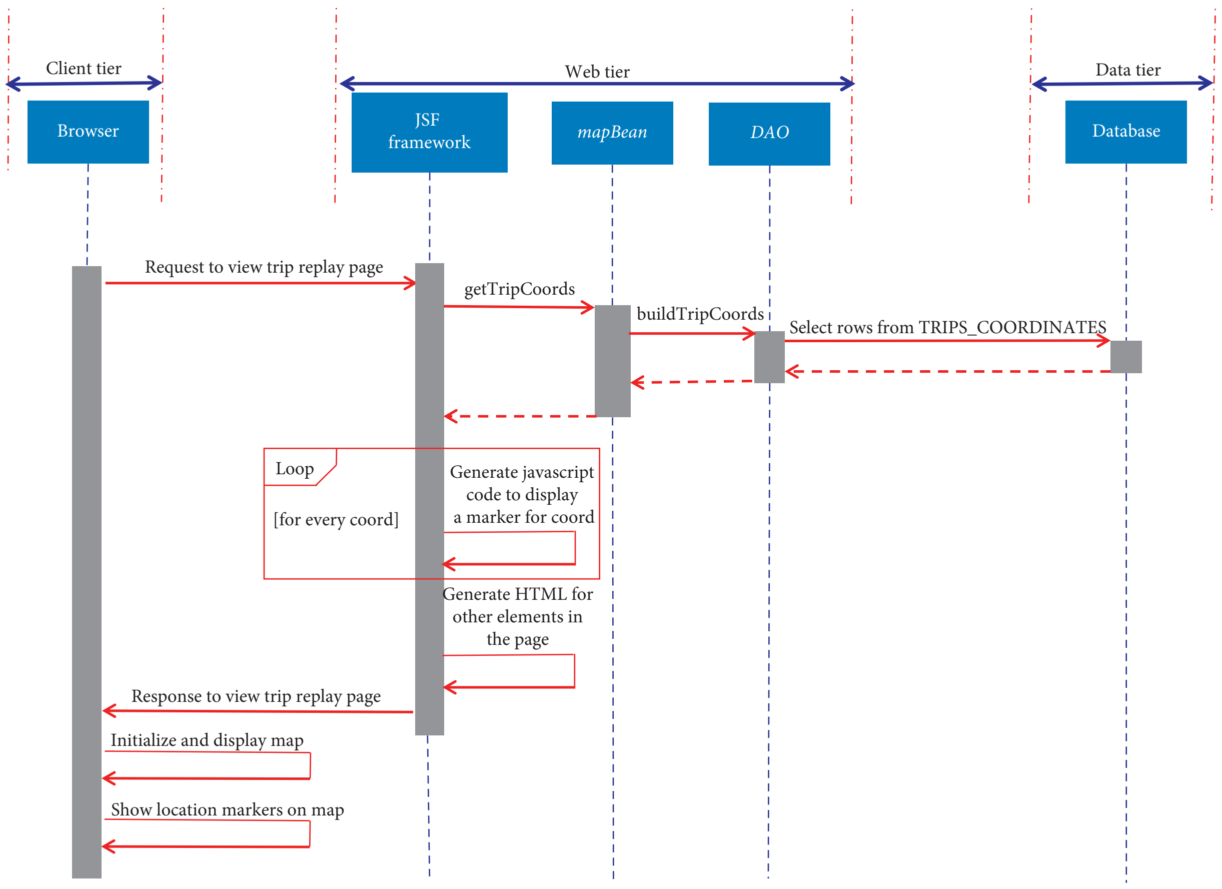
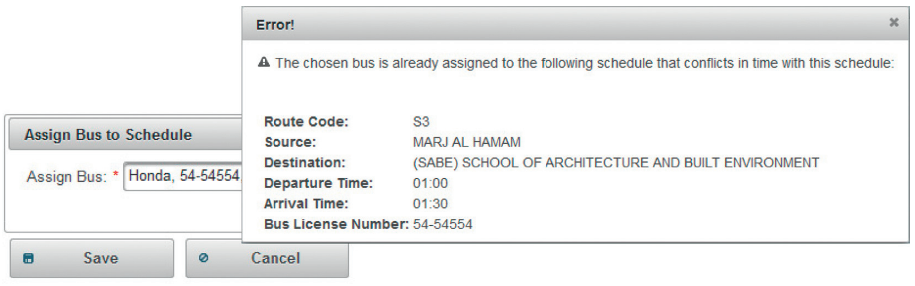

Client tier
Web tier
Data tier
Browser
JSF framework
mapBean
DAO
Database
Request to view trip replay page
getTripCoords
buildTripCoords
Select rows from TRIPS_COORDINATES
Loop
[for every coord]
Generate javascript code to display a marker for coord
Generate HTML for other elements in the page
Response to view trip replay page
Initialize and display map
Show location markers on map

Figure 16: The sequence diagram for the replay bus trip code interactions.



The chosen bus is already assigned to the following schedule that conflicts in time with this schedule:

Route Code:	S3
Source:	MARJ AL HAMAM
Destination:	(SABE) SCHOOL OF ARCHITECTURE AND BUILT ENVIRONMENT
Departure Time:	01:00
Arrival Time:	01:30
Bus License Number:	54-54554

Figure 17: The conflict detection message when attempting to assign a bus to an overlapping schedule.

line 8, respectively. Accordingly, the departure time and arrival time of the current *busSched* are computed at line 9 and line 10, respectively. The arrival and departure times of the current *busSched* are compared to their counterparts of the *selSched* to check for any schedule conflict within the conditional *if* statement block that starts at line 12 and ends at line 20. In case a schedule conflict is detected, an informative error message (see Figure 17) is constructed at line 18 for display, and a *break* statement is issued at line 19 to exit the *foreach* block.

4.4. User Authentication and Authorization. GJU users (e.g., students and employees) log in to all university portals using SSO credentials (i.e., with a single username and password) that are managed and saved in the Microsoft Active Directory (AD) [44]. Accordingly, the bus management system uses the Java Naming and Directory Interface (JNDI) as well as the Lightweight Directory Access Protocol (LDAP) to communicate with, and authenticate users in, the AD. Once a passenger is authenticated, the system also verifies whether that user had subscribed to the bus transportation service and paid the required fees (based on the related data in the buses, SIS, HR, and AIS databases shown in Figure 14). Consequently, a passenger is denied access to the system in case authentication or authorization fails.

4.5. Application Security and Data Protection. The following security measures are implemented to protect the application and database servers from any unauthorized access: the

```
- Given that a driver is assigned to only one bus
- selSchedDepartureTime ← the time of the selSched stop with the lowest order
- selSchedArrivalTime ← the time of the selSched stop with the highest order
- Retrieve the schedules for the bus to be assigned to the selSched from the BUSES_SCHEDULES table
- Save the retrieved schedules in the busSchedulesArray

foreach busSched in busSchedulesArraydo
    - Retrieve the busSched stop schedules from the ROUTES_STOPS_SCHEDULEStable
    - Save the retrieved stop schedules in the busStopSchedulesArray
    - busSchedDepartureTime ← the time of the bus stop schedule with the lowest order
    - busSchedArrivalTime ← the time of the bus stop schedule with the highest order

    if ((busSchedDepartureTime BETWEEN (selSchedDepartureTime AND selSchedArrivalTime)) OR
        (busSchedArrivalTime BETWEEN (selSchedDepartureTime AND selSchedArrivalTime)) OR
        (selSchedDepartureTime BETWEEN (busSchedDepartureTime AND busSchedArrivalTime)) OR
        (selSchedArrivalTime BETWEEN (busSchedDepartureTime AND busSchedArrivalTime)))
    then
        - Flag the schedule conflict
        - Construct an informative error message for display
        - break // Exit foreach loop
    end // if statement
end // foreach loop
```

Figure 18: The pseudocode for the schedule conflict detection algorithm.

security-centric Linux operating system (OS) is utilized to run all servers; a strong password policy is in place to access the required servers and applications by authorized users only; the OS and all used software are regularly updated to close any vulnerabilities; antivirus tools are deployed to protect all servers from malware; the network security access list is defined to block any access to the admin view and database from outside the GJU intranet; besides, the HTTPS protocol is enabled in the application server for user authentication, data confidentiality, and data integrity purposes.

5. Validation and Results

The system testing results, user survey outcomes, and the feature comparison to related work are presented in this section.

5.1. Trip Tracking Accuracy and Speed.

Three samples from the many trips that were tracked during the system testing phase are shown in Figures 19–21. The majority of the recorded bus locations during those trips were very accurate as illustrated in the three figures. In Figure 19, all the recorded points were accurate and none of them deviated from the taken path. On the other hand, only one point in Figure 20 was outside the taken path (i.e., an outlier point), and hence, it has a negligible effect on the path tracking accuracy. In Figure 21, a short gap (due to a possibly missing marker on that part of the path) was noticed on the tracked path, and that can be related to an encountered delay in the Geolocation API while attempting to locate the bus location. In some cases, the existence of some short gaps in the path is not an issue as they can be attributed to a faster movement of the bus in between the poll operations. As far as the value of the poll interval, it was found that using a value of about 10 seconds is suitable to efficiently record and accurately reproduce the taken trip paths. Noting that the longest traveled

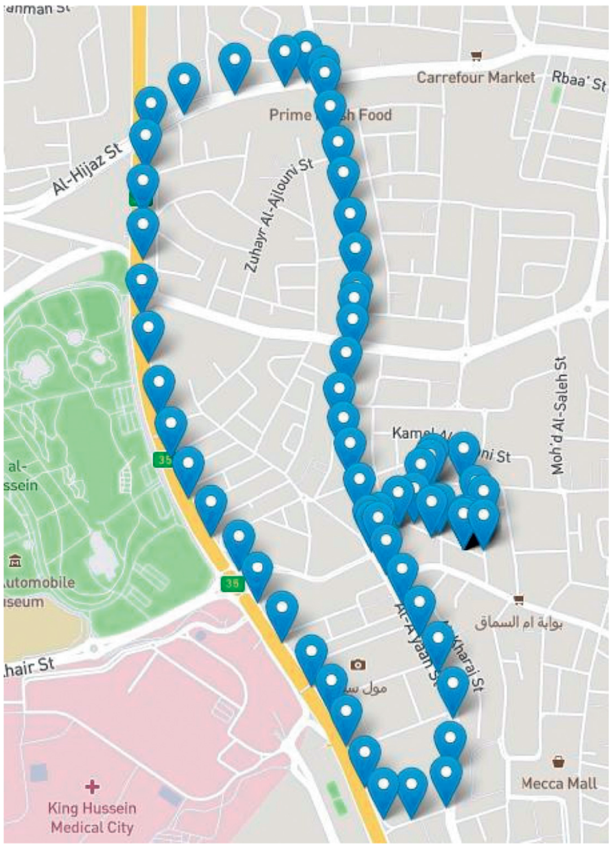


Figure 19: A very accurate trip path.

distance in the shown figures was about 5 kilometers during the trip shown in Figure 19.

Another experiment was conducted to compare the speed of the Leaflet API with that of its Google Maps counterpart. The results of this investigation are illustrated in Figure 22 in which the measured elapsed times to locate and display 70 markers using both packages are shown. Based on that, both APIs are fast and responsive, with

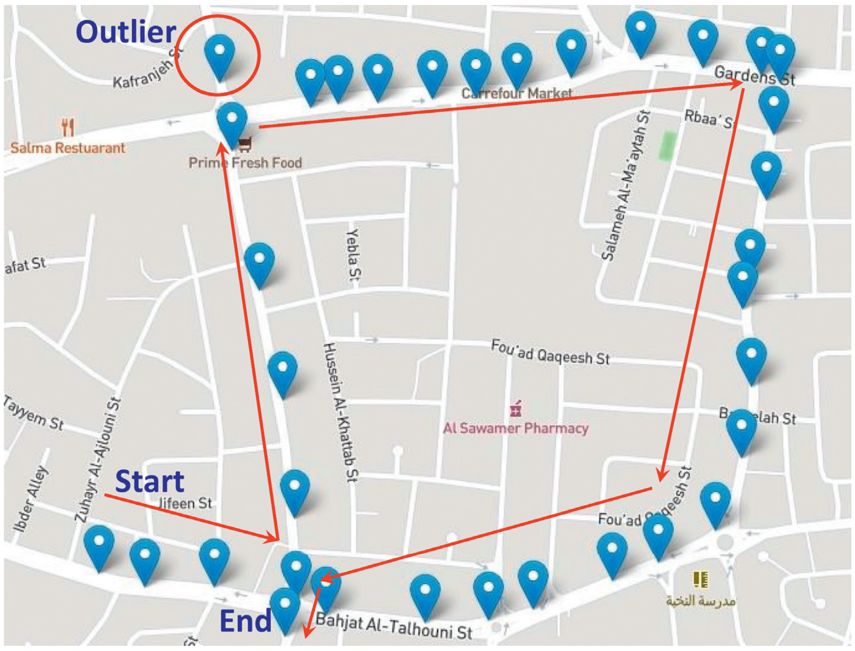


Figure 20: A trip path in which there is one outlier marker.

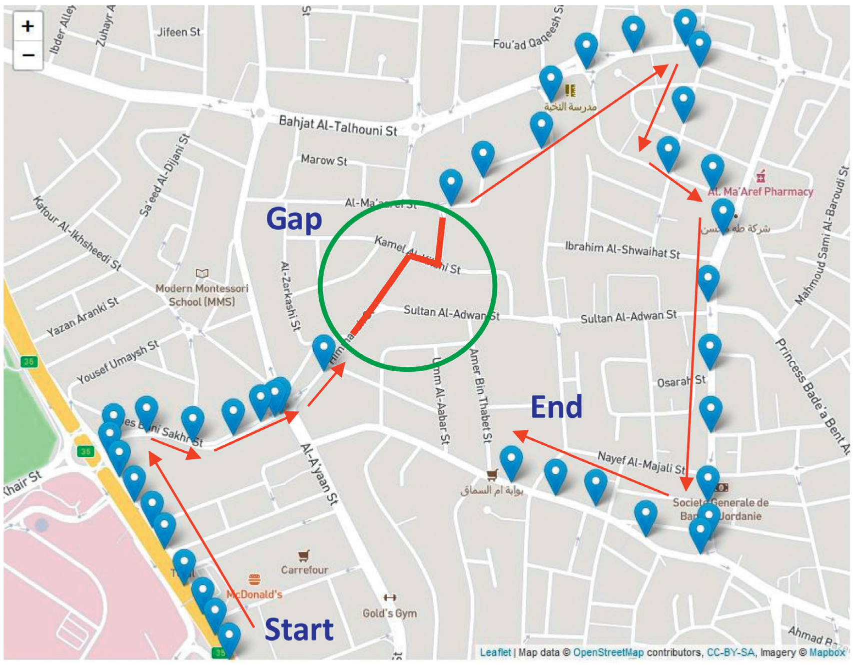


Figure 21: A trip path in which there is a short gap.

negligible speed differences. Specifically, the minimum, average, and maximum measured times using Leaflet were equal to 21 ms, 52.64 ms, and 152 ms, respectively, whereas the minimum, average, and maximum recorded times using Google Maps were equal to 16 ms, 57.05 ms, and 192 ms, respectively. Therefore, this experiment asserts that using the Leaflet API to develop interactive maps in this system is advantageous because it is free and adaptable, with almost no speed differences relative to the no longer free Google Maps API.

The average time to add the markers on a map for a recorded trip was also evaluated as shown in Figure 23. Given that the travel time for most trips to/from GJU spans from 30 minutes to 90 minutes, hence the needed number of points to record a trip in the database ranges from 180 points to 540 points based on a 10-second poll interval. Therefore, measuring the average time to display the markers for trips with a number of points ranging from 100 points to 600 points is sufficient to evaluate the speed of the trip replay feature. Accordingly, the measured average time values were

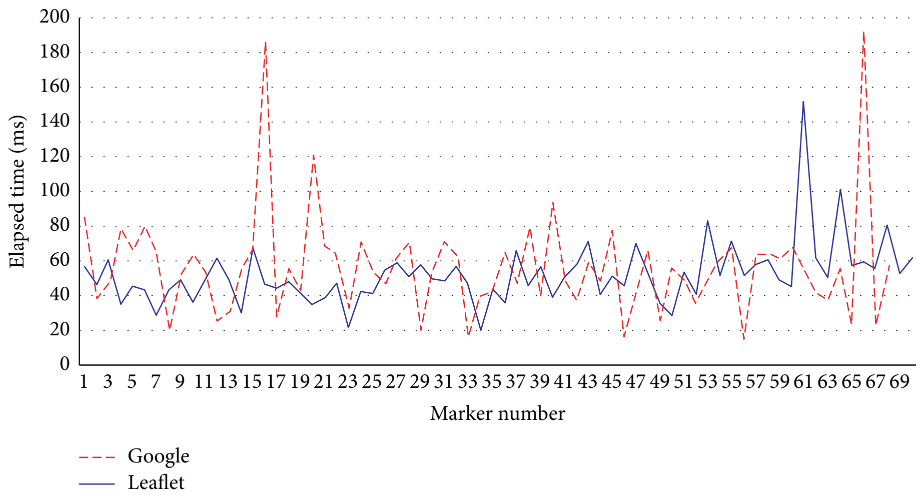


FIGURE 22: The elapsed times to locate and display 70 markers using Leaflet and Google Maps.

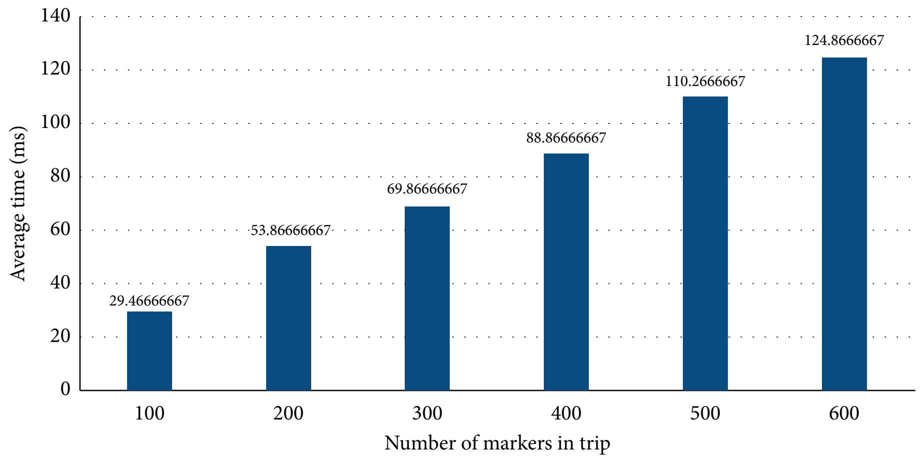


FIGURE 23: The average times to display the markers on a map for different recorded trips.

reasonably fast as the maximum value did not exceed 125 ms for a trip comprised of 600 markers.

5.2. Application Assessment and Trip Satisfaction Surveys. The employees and students (i.e., about 58 users) traveling in three buses were asked to try the application and then complete a system assessment survey and a trip satisfaction review. A total of 45 participants completed the aforementioned surveys as shown in Tables 1 and 2, respectively. The answers to each question in the surveys were based on a five-point Likert scale. Hence, each question had five answers as follows: strongly agree (5 points), agree (4 points), not sure (3 points), disagree (2 points), and strongly disagree (1 point).

Based on Table 1, 93% of the participants strongly agreed or agreed that they have an acceptable Internet service. Also, 98%, 87%, 80%, 87%, and 91% of the users strongly agreed or agreed that the system is easy to use (i.e., question 2), accurate (i.e., question 3), fast (i.e., question 4), responsive (i.e., question 5), and helpful (i.e., question 6), respectively.

The travelers used the trip review form shown in Figure 12 to provide their feedback regarding the trip. Accordingly, 96% and 100% of the participants strongly agreed or agreed that the drivers were friendly (i.e., question 1) and drove safely (i.e., question 2), respectively, as shown in Table 2. Also, 93% of them would recommend the drivers to others (i.e., question 3). On the other hand, 87% and 89% of the users strongly agreed or agreed that the bus was comfortable (i.e., question 4) and arrived on time (i.e., question 5), respectively.

5.3. Feature Comparison to Related Work. A feature comparison to the related systems discussed in [25–37] is shown in Table 3. In that regard, this work is centered on designing a complete bus management portal, and its novelty can be summarized as follows:

(i) The free open-source Leaflet library has been used to develop interactive maps in this system, while the majority of the other systems relied on the no longer free Google Maps API for that purpose. Not to mention, Leaflet is similarly fast and responsive,

Table 1: The system assessment survey results.

#	Question	Answers					Total answers	Average score
		5	4	3	2	1		
1	My Internet service is fast and reliable	23	19	3	0	0	45	4.444
2	The bus tracking system is easy to use	26	18	0	1	0	45	4.533
3	The reported bus locations are accurate	18	21	6	0	0	45	4.267
4	The map refresh rate is acceptable	17	19	7	2	0	45	4.133
5	The UI fits nicely within the phone screen	18	21	4	2	0	45	4.222
6	The provided service is helpful	16	25	3	1	0	45	4.244

Table 2: The trip satisfaction survey results.

#	Question	Answers					Total answers	Average score
		5	4	3	2	1		
1	The driver was friendly and displayed good manners	25	18	2	0	0	45	4.511
2	The driver's driving was safe and calm	28	17	0	0	0	45	4.622
3	I would recommend the driver to others	26	16	2	1	0	45	4.489
4	The bus seats were clean and comfortable	17	22	4	2	0	45	4.200
5	The bus departed and arrived on time	24	16	5	0	0	45	4.422

Table 3: A feature comparison to related work.

Reference	Leaflet maps API	SSO	Integration with other systems	Manage buses	Manage and assign drivers	Manage stops, routes, schedules, and trips	Detect schedules conflict	Track trips	Replay trips	Review trips	Reports
GJU buses system	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[25]	✗	✗	✗	✓	✓	✓	✗	✓	✗	✗	✗
[26]	✗	✗	✗	✓	✓	✓	✗	✓	✗	✗	✗
[27]	✗	✗	✗	✓	✗	✓	✗	✓	✗	✗	✗
[28]	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
[29]	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
[30]	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
[31]	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
[32]	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
[33]	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
[34]	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
[35]	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
[36]	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
[37]	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗

but it is also more adaptable due to being open source.

(ii) Unlike most papers, this paper discussed all setup aspects related to the management of buses, drivers, stops, routes, schedules, and trips.

(iii) Unlike the rest of the systems, this system implements an algorithm to detect bus schedule conflict that is needed to prevent the assignment of two drivers to the same bus or the assignment of a bus to overlapping schedules.

(iv) None of the other systems developed a trip replay feature as in this work despite the importance of that to review and archive previous trips.

(v) Unlike all related work, this system uses SSO for user authentication due to its importance in preventing users from memorizing several passwords and relieving administrators from assigning credentials to every new user.

(vi) Unlike all other systems, the bus management system is integrated with the SIS, HR, and AIS databases. Accordingly, the system setup is easier as the administrator does not have to manage students, employees, fees, or payments as all that data are directly accessible from the aforementioned databases for authorization or display purposes.

(vii) Only this system gave the passengers the chance to evaluate the offered services (e.g., bus quality and driver behavior).

(viii) This system is the only one that supports reporting capabilities with flexible filtering criteria as needed to enable management to assess the service quality and user satisfaction for further improvements.

6. Summary and Future Work

A flexible, easy to use, fast, accurate, and helpful web-based bus management and tracking system that is designed and developed in-house and supports the following user roles has been introduced in this paper:

(i) Admin user role is responsible for tasks such as system setup, trip scheduling, report generation, and service evaluation.

(ii) Driver user role takes care of starting, monitoring, canceling, and stopping the assigned trips.

(iii) Passenger user role monitors any ongoing trip to check the possibility of catching a bus or finding out how far is the bus from a bus stop. In addition, this user can provide feedback regarding the offered services for improvement purposes.

Unlike many existing systems, this system incorporates all the user interfaces and data structures needed to manage buses, drivers, stops, routes, schedules, and trips. Also, it utilizes an algorithm to detect bus schedule conflict to prevent assigning several drivers to the same bus or assigning a bus to overlapping schedules. Moreover, it supports SSO user authentication and it is integrated with other systems (e.g., SIS, AIS, and HR) to directly access data related to students, employees, fees, and payments without the need for extra setup steps. In addition, it provides reporting capabilities with flexible filtering criteria to allow management to evaluate the service quality and user satisfaction for further action.

The sequence diagram for the code methods (in the system's client, web, and data tiers) used to display and record a bus trip has been discussed. Moreover, the sequence diagram that represents the code interactions to replay a previous bus trip has been explained.

One of the advantages of this system is that it uses the free open-source Leaflet package to develop interactive maps and track the bus trips. The comparative analysis between the open-source API and the proprietary Google API when tracking several bus trips illustrated that the benefits of using the Leaflet API are justified as they come without any effects on accuracy, performance, or responsiveness when compared to the no longer free Google Maps API. Further, the Leaflet API is more adaptable than its counterpart due to its open-source nature.

One of the limitations of this work is that it does not predict the bus arrival time to enhance customer satisfaction, but it only displays and records the actual locations of a traveling bus during a specific trip. Note that the saved historical trip data can be used to predict bus arrival times when needed. Also, this work does not attempt to avoid bus bunching and it does not regulate bus headway; however, such issues are not much of a concern in the case of the private GJU bus transportation system. Moreover, as of now, the introduced system does not collect the bus capacity during a trip for the possible use of such data in the development of scheduling algorithms to reduce, for example, bus operation cost.

As far as future work, the current plan is to enhance the system to support features such as bus seat reservation, bus capacity collection, and bus pass scanning (e.g., via a QR code reader). Such data can be utilized, for example, to reduce bus operation cost. Furthermore, the possibility to integrate accurate methods to predict bus arrival time to improve passenger satisfaction is under consideration.

Data Availability

The data used in the study will be available upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] F. Al-Hawari, "Mygju student view and its online and preventive registration flow," *International Journal of Applied Engineering Research*, vol. 12, no. 1, pp. 119–133, 2017.

[2] F. Al-Hawari, A. Alufeishat, M. Alshawabkeh, H. Barham, and M. Habahbeh, "The software engineering of a three-tier web-based student information system (MyGJU)," *Computer Applications in Engineering Education*, vol. 25, no. 2, pp. 242–263, 2017.

[3] F. Al-Hawari, "Analysis and design of an accounting information system," *International Research Journal of Electronics and Computer Engineering*, vol. 3, no. 2, pp. 16–21, 2017.

[4] F. H. Al-Hawari and M. S. Habahbeh, "Secure and robust web services for e-payment of tuition fees," *International Journal of Engineering Research and Technology*, vol. 13, no. 7, pp. 1795–1801, 2020.

[5] W. Wu, R. Liu, and W. Jin, "Modelling bus bunching and holding control with vehicle overtaking and distributed passenger boarding behaviour," *Transportation Research Part B: Methodological*, vol. 104, pp. 175–197, 2017.

[6] J. Wang and L. Sun, "Dynamic holding control to avoid bus bunching: a multi-agent deep reinforcement learning framework," *Transportation Research Part C: Emerging Technologies*, vol. 116, Article ID 102661, 2020.

[7] W. Wu, R. Liu, and W. Jin, "Integrating bus holding control strategies and schedule recovery: simulation-based comparison and recommendation," *Journal of Advanced Transportation*, vol. 2018, Article ID 9407801, 13 pages, 2018.

[8] L.-M. Kieu, D. Ngoduy, N. Malleson, and E. Chung, "A stochastic schedule-following simulation model of bus routes," *Transportmetrica B: Transport Dynamics*, vol. 7, no. 1, pp. 1588–1610, 2019.

[9] S. Ji-Yang, H. Jian-Ling, C. Yan-Yan, W. Pan-Yi, and J. Jian-Lin, "Flexible bus route optimization for multitarget stations," *Mathematical Problems in Engineering*, vol. 2020, Article ID 7183465, 8 pages, 2020.

[10] B. A. Kumar, G. H. Prasath, and L. Vanajakshi, "Dynamic bus scheduling based on real-time demand and travel time," *International Journal of Civil Engineering*, vol. 17, no. 9, pp. 1481–1489, 2019.

[11] W. Wu, R. Liu, W. Jin, and C. Ma, "Simulation-based robust optimization of limited-stop bus service with vehicle overtaking and dynamics: a response surface methodology,"

Transportation Research Part E: Logistics and Transportation Review, vol. 130, pp. 61–81, 2019.

[12] M. Bögl, K. F. Doerner, and S. N. Parragh, "The school bus routing and scheduling problem with transfers," *Networks*, vol. 65, no. 2, pp. 180–203, 2015.

[13] S. Kim, C. Lee, Y. Kim, S. Lee, and D. Park, "Error correction of arrival time prediction in real time bus information system," *Journal of Advanced Transportation*, vol. 44, no. 1, pp. 42–51, 2010.

[14] I. Ashour, M. Zorkany, and M. Shiple, "Design and implementation of transportation management system," in *Proceedings of the International Conference on Vehicle Technology and Intelligent Transport Systems*, pp. 11–18, Lisbon, Portugal, 2015.

[15] M. Fadaei, O. Cats, and A. Bhaskar, "A hybrid scheme for real time prediction of bus trajectories," *Journal of Advanced Transportation*, vol. 50, no. 8, pp. 2130–2149, 2016.

[16] M. Chen, J. Yaw, S. I. Chien, and X. Liu, "Using automatic passenger counter data in bus arrival time prediction," *Journal of Advanced Transportation*, vol. 41, no. 3, pp. 267–283, 2007.

[17] W. Liu, "Wilocator: wifi-sensing based real-time bus tracking and arrival time prediction in urban environments," in *Proceedings of the 36th International Conference on Distributed Computing Systems (ICDCS)*, pp. 529–538, IEEE, Nara, Japan, 2016.

[18] P. Zhou, Y. Zheng, and M. Li, "How long to wait? predicting bus arrival time with mobile phone based participatory sensing," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, pp. 379–392, Windermere, UK, 2012.

[19] K. Lu, A. Khani, and B. Han, "A trip purpose-based data-driven alighting station choice model using transit smart card data," *Complexity*, vol. 2018, Article ID 3412070, 14 pages, 2018.

[20] J. Li, Q. Li, Y. Zhu, Y. Ma, Y. Xu, and C. Xie, "An automatic extraction method of coach operation information from historical trajectory data," *Journal of Advanced Transportation*, vol. 2019, Article ID 3634942, 15 pages, 2019.

[21] R. Netek, J. Brus, and O. Tomecka, "Performance testing on marker clustering and heatmap visualization techniques: a comparative study on javascript mapping libraries," *ISPRS International Journal of Geo-Information*, vol. 8, no. 8, p. 348, 2019.

[22] A. Shingare, A. Pendole, N. Chaudhari, P. Deshpande, and S. Sonavane, "GPS supported city bus tracking & smart ticketing system," in *Proceedings of the International Conference on Green Computing and Internet of Things (ICGCIoT)*, pp. 93–98, IEEE, Delhi, India, 2015.

[23] K. Aggrawal, "RFID based intelligent bus management and monitoring system," *International Journal of Engineering Research & Technology*, vol. 3, no. 7, pp. 6–13, 2014.

[24] M. Malleswari, M. K. Rao, K. Supriya, K. P. Krishna, and B. R. Teja, "RFID based college bus management system," *International Research Journal of Engineering and Technology*, vol. 5, no. 5, p. 3, 2018.

[25] A. Ahmed, E. Nada, and W. Al-Mutiri, "University buses routing and tracking system," *International Journal of Computer Science and Information Technology*, vol. 9, no. 1, pp. 95–104, 2017.

[26] G. Jemilda, R. B. Krishnan, B. Johnson, and G. L. Sangeeth, "Mobile application for college bus tracking," *International Journal of Computer Science Mobile Computing*, vol. 4, no. 3, pp. 500–507, 2015.

[27] S. Eken and A. Sayar, "A smart bus tracking system based on location-aware services and QR codes," in *Proceedings of the International Symposium on Innovations in Intelligent Systems and Applications*, pp. 299–303, IEEE, Alberobello, Italy, 2014.

[28] D. B. M. Vidyavathi, A. Ahamed, H. Sultana, Y. Madhumathi, and F. Begum, "College bus tracking system (traveline)," *International Journal of Advanced Research in Computer Science*, vol. 9, no. 3, pp. 90–93, 2018.

[29] M. Kamisan, A. Aziz, W. Ahmad, and N. Khairudin, "UiTM campus bus tracking system using arduino based and smartphone application," in *Proceedings of the 15th Student Conference on Research and Development*, pp. 137–141, IEEE, Putrajaya, Malaysia, 2017.

[30] R. K. Megalingam, N. Raj, A. L. Soman, L. Prakash, N. Satheesh, and D. Vijay, "Smart, public buses information system," in *Proceedings of the International Conference on Communication and Signal Processing*, pp. 1343–1347, IEEE, Bangkok, Thailand, 2014.

[31] M. Sneha, C. N. Urs, S. Chatterji, M. Srivatsa, K. Pareekshith, and H. A. Kashyap, "Darideepa: a mobile application for bus notification system," in *Proceedings of the International Conference on Contemporary Computing and Informatics*, pp. 724–727, IEEE, Mysuru, India, 2014.

[32] R. Jisha, A. Jyothindranath, and L. S. Kumary, "Iot based school bus tracking and arrival time prediction," in *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pp. 509–514, IEEE, Udupi, India, 2017.

[33] K. Sujatha, P. N. Rao, K. Sruthi, and A. A. Rao, "Design and development of android mobile based bus tracking system," in *Proceedings of the First International Conference on Networks & Soft Computing*, pp. 231–235, IEEE, Guntur, India, 2014.

[34] P. Kaulage, A. Pingale, P. Bhoite, V. Mankar, and P. Yadav, "Bus tracking and bus arrival time, location prediction system," *International Research Journal of Engineering and Technology*, vol. 4, no. 5, pp. 806–811, 2017.

[35] S. Dhende, V. Kaotekwar, V. Kokane, and V. Karambelkar, "Intelligent bus system using rfid, zigbee and gprs," *International Research Journal of Engineering Technology*, vol. 4, no. 4, 2017.

[36] S. Sankarananrayanan and P. Hamilton, "Mobile enabled bus tracking and ticketing system," in *Proceedings of the 2nd International Conference on Information and Communication Technology*, pp. 475–480, IEEE, Bandung, Indonesia, 2014.

[37] A. A. Surve, R. P. Nahar, G. K. Somavanshi, and K. Dive, "Enhancing the functionality of bus monitoring and tracking system," *International Journal of Technology Enhancements and Emerging Engineering Research*, vol. 3, no. 4, pp. 93–97, 2015.

[38] V. Agafonkin, "Leaflet: an open-source javascript library for mobile-friendly interactive maps," 2019, https://leafletjs.com/.

[39] J. P. Ventoso, "Switching from google maps to leaflet," 2019, https://www.endpoint.com/blog/2019/03/23/switching-google-maps-leaflet.

[40] Google, "Google maps platform," 2020, https://cloud.google.com/maps-platform/maps.

[41] C. Y. Laporte, R. V. O'Connor, and L. H. G. Paucar, "The implementation of ISO/IEC 29110 software engineering standards and guides in very small entities," in *Proceedings of the International Conference on Evaluation of Novel Approaches to Software Engineering*, pp. 162–179, Barcelona, Spain, 2015.

[42] E. Burns and C. Schalk, *JavaServer Faces 2.0: The Complete Reference*p. 722, First edition, McGraw Hill, New York, NY, USA, 2010.

[43] PrimeTek, “Primefacses: leading provider of open source UI component libraries,” 2020, https://www.primefaces.org/showcase/.

[44] D. Iseminger, *Active Directory Services for Microsoft Windows 2000*, Microsoft Press, Redmond, WA, USA, 1999.