



A SUMMARY REPORT ON TRANSPORT4YOU

An Intelligent Transportation System

SUBMITTED BY

KUNAL SANGWAN

SHASHANK SRIKANT

SOHIL ARORA

UNDER THE GUIDANCE OF

DR. JITENDER KUMAR CHHABRA

ASSOCIATE PROFESSOR

jitenderchhabra@nitkkr.ac.in



**DEPARTMENT OF COMPUTER ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY KURUKSHETRA
INDIA**

Table of Contents

1. Executive Summary.....	3
2. Introduction	4
3. Software Requirements Specification	5
3.1. Definitions, Acronyms and Abbreviations.....	5
3.2. Functional Requirements.....	5
3.3. Non-Functional Requirements	6
3.4. Constraints.....	6
3.5. Assumptions and Dependencies.....	7
4. Management Plan	8
4.1. Team Introduction.....	8
4.2. Work Division.....	8
4.3. Communication	8
4.4. Managing Project Components	8
5. Project Plan.....	9
5.1. Proposed Plan for the Execution of Project	9
5.2. Realized Plan for the Execution of the Project	9
6. Overall Design.....	10
6.1. Design Guidelines	10
6.2. Design Architecture.....	11
6.3. Component Design.....	11
6.4. Database Design.....	16
6.5. User Interface Design.....	18
7. Development Cycle	19
8. Implementation.....	20
8.1. Device Detection Implementation.....	20
8.2. Server Modules Implementation	Error! Bookmark not defined.
8.3. User Interface	20
9. Testing	22
10. Outcome and Lessons Learnt.....	23
10.1. Key Features Implemented.....	23
10.2. Features Not Implemented.....	23
11. Appendix.....	24
12. References.....	28

1. Executive Summary

The goal of Transport4You is to develop some of the core features of a system that can help a metropolitan transportation authority in improving the services offered to citizens. Existing transportation systems contain lots of manually controlled operations and provide almost no emphasis on commuter-centric services. The proposed system differs from such systems in terms of use of technology to identify and detect commuter (referred henceforth as a user) movement, keep him/her updated about the latest status of buses on the lines he/she travels frequently and assist him/her in identifying shortest/fastest path to travel from his/her source to the destination.

There were many aspects to this project which had to be carefully studied and analyzed during the initial stages. In order to do the feasibility study and planning for the project, a detailed discussion was held with the stakeholders and some domain experts. Based on this study, we collected information regarding various alternates of cost effective communication systems on the buses, possible algorithms related to users' travel-path learning, payment and SMS gateways. In order to ensure the timely completion of the project within our resources and strengths, the alternates, best suitable to us, for each of the above mentioned issues were shortlisted initially.

The functional requirements of the system included designing a web-based portal wherein a user could register and pre-pay for a certain number of trips; a central server system to compute and calculate fares of users; an identification/storage system on-board a bus which would store the Bluetooth/Wi-Fi details of users boarding it; a communication system between buses and a central server to relay information on the identified users; a communication system to inform the users on the status of the lines, shortest/fastest path details etc. Non-functional requirements included minimum interaction with users, minimum bandwidth usage by various communication systems, scalability and security of the system.

Major design decisions taken included design of data-structures to efficiently model user movement; a strategy to selectively identify only those users boarding/de-boarding a bus and discarding false identification of users in its vicinity; a strategy to minimize bandwidth used by the buses by locally storing information of users boarding/de-boarding at every bus-stop and consequently relaying it to the central-server; a path-learning algorithm to learn the movement of users.

The nature of the requirements led to the decomposition of the entire system into loosely coupled components. The development strategy followed was a combination of both the agile software development model and the iterative and incremental development model. These models best reflected the execution of the project as they

suited the need to quickly study a technology, implement ideas in it and allow improvements over short intervals of time.

Key technologies used to implement the project include PHP to design the web-portal for user registration; MySQL to maintain user-related data and C to implement the Bluetooth detection module. Java has been used as a supporting language to the database queries to perform algorithm-intensive tasks such as path-calculation, message generation etc. PHP has been used within CodeIgniter, an open-source web application framework for dynamic websites, coupled with libraries to implement safe data transfer. BlueZ, a popular open-source Bluetooth stack has been used to detect and connect to Bluetooth radios.

The components developed have been tested for correctness of functionality at the unit level. The integrated system has already been tested for cause-effects, with the core-functioning of the system having been verified. Further system testing and stress testing is in progress.

This project was developed under the guidance of Dr. Jitender Kumar Chhabra, Associate Professor, Department of Computer Engineering, National Institute of Technology Kurukshetra, India as part of B.Tech Minor Project, one of the two main projects to be carried out in the final year of our Bachelor's Degree in Engineering.

2. Introduction

Public transportation is one of the main services offered by municipalities, sometimes with the help of private companies. From the end user point of view, the service is perceived favorably when it is reliable and on schedule, when it covers the entire city, possibly with direct and fast connections that fulfill the needs of the various users, and when it is easy to use, e.g., to buy valid tickets and access the service.

Thus, a key challenge is to address all above needs and to offer personalized services to citizens. This requires the new Information and Communication Technologies to be massively exploited.

The goal of this project is to develop some of the core features of a system that can help a Metropolitan Transportation Authority improve the services offered to citizens. At its core, the system should offer the following basic functionalities:

- i. Allow a citizen to register into the system and pre-pay for a certain number of trips.
- ii. Recognize when a registered citizen gets on a bus and determine the journey he/she performs, calculating the fare he/she has to pay, and deducting it from his/hers credit.
- iii. When the citizen's credit is finished, allow him/her to pay the bus fare through cell phone/internet.
- iv. Provide registered citizens with information about changes in the lines they use most frequently.

Possibly, the system should be able to offer suggestions to registered citizens for alternative paths, for example because it determines that there are routes that are more optimized than the ones they usually take, or because there is a problem on some line that affects their intended path.

3. Software Requirements Specification

3.1. Definitions, Acronyms and Abbreviations

- i. '*the system*' unless specified otherwise, refers to the Transport4You system,. Any sub-system of Transport4You is specified as a *system* in a given context.
- ii. *User* and *Passenger* are interchangeably used to describe a person who travels on the MTA controlled buses.
- iii. *On-board system* refers to a system present on every bus which can identify Bluetooth and Wi-Fi devices in its vicinity.
- iv. *Administrator* refers to designated officials of the MTA who are provided with the privilege to act as the system's administrators, responsible for managing and maintaining the system.
- v. *Device* refers to a mobile phone or any similar device which contains a Bluetooth/Wi-Fi radio which may be powered on to connect to other similar devices.
- vi. *MTA* - Metropolitan Transportation Authority
- vii. *ICT* - Information and Communication Technologies
- viii. *SMS* - Short Messaging Service (available in cell phones)
- ix. *Device ID* refers to the physical address of the Bluetooth/Wi-Fi device of a User.
- x. *T_VAL* refers to the time-account provided to every user, which basically is the maximum allowable time a user can travel for a given charge.
- xi. *T_COST* refers to the cost charged for travelling 1 *T_VAL* unit of time.
- xii. The text consistently uses *she* and *her* as the pronoun for a *User*. This is to generalize the text and does not imply at any constraint on the gender of a user.

3.2. Functional Requirements

An overview of the requirements, as analyzed through the study of the problem statement resulted in the identification of key modules specifying various interactions and sub-processes taking place in the system. They are listed in brief below -

- i. **Registration Facility:** Responsible for the interaction of a customer with the Transport4You system, wherein the customer registers herself with the system and adds sufficient information to identify her on a bus.

- ii. **On-board System:** Responsible for the interaction of a registered user with the buses of the system.
- iii. **Messaging Services:** Responsible for the interaction of Tranport4You system and a user. It is implemented through instant messaging services, which informs the users of the results of the various calculations performed. These results include fare charged for travel and alternate route suggestions. It also enables a user to pay for a journey.

The functional requirements listed above are the four key requirements provided in the problem statement of Transport4You.

3.3. Non-Functional Requirements

3.3.1. Usage and Ease-of-Use

Information regarding a path that shall not be available

on a given day can be entered into the system by the administrators. All users that are likely to be affected by the unavailability of such a path are identified and sent an SMS notifying them of an alternate path and schedule of buses on it.

To enable the user more comfort and not to overload information pertaining to an unavailable route, only select information needs to be messaged to the user in the case where the change shall directly affect the user's movement.

3.3.2. Scalability

The system should support a large number of users. Additionally, the system needs to be scalable enough to add any new databases and technologies to support future requirements, if any.

3.3.3. Other Requirements

3.3.3.1. Security

- i. The interaction between the central database and the on-board database should be minimal and secure.
- ii. A clear distinction should exist between administrator privileges and user privileges while accessing the registration facility.

3.4. Constraints

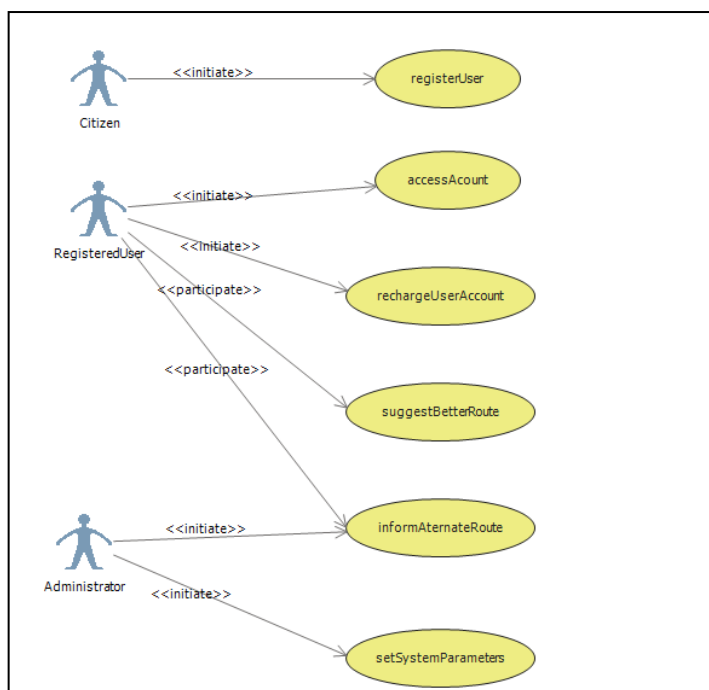


Figure 1– Use-Case representing Requirements

3.4.1. Reliability Requirement

A passenger has to be mentioned of an alternate route before her expected boarding of a bus on the route.

3.4.2. Memory Constraints

No explicit memory constraints have been mentioned in the requirements documentation of the problem statement. However, we assume that the on-board system does not have the computational prowess and the memory capabilities of conventional computers per se, but are rather crude hardware devices with functionality to the tune of a micro-controller with a few KB of memory for storage. This assumption has been made with the consent of the problem setters of Transport4You.

3.4.3. Other General Constraints

- i. The pro-active involvement of a passenger should be minimal.
- ii. Minimum communication with a passenger needs to be done. Moreover, only those passengers should be intimated through messages that have opted for it.

3.5. Assumptions and Dependencies

- i. Every passenger keeps his Bluetooth/Wi-Fi radio on at all the times.
- ii. There exists a system on each bus which can connect to Bluetooth and Wi-Fi devices in its vicinity. This system has a high quality of service and is assumed not to fail.
- iii. Only registered users use the bus service.
- iv. The interaction between a bus-stop and a bus is implemented through wireless communication, which lets a bus know the identification of a stop. The customer's boarding and de-boarding at a stop are thus referenced against the information provided by the bus-stop, which in turn facilitates successful fare calculation for a user. This assumption was decided in concurrence with the stakeholders.
- v. A passenger gets on to the bus and leaves a bus only at bus stops.
- vi. There exist a good number of buses on any given route such that, at any given point of time, a passenger may expect a bus to arrive at a bus-stop 'shortly', wherein shortly maybe defined by a few minutes.
- vii. A fully implemented design of the on-board computer's connectivity with the server through GPRS exists.

4. Management Plan

4.1. Team Introduction

Our team consists of three members, out of which only one of us had a prior experience of web development and none of us had any previous experience with Bluetooth technology. All of us had a working knowledge of database management systems.

4.2. Work Division

We assigned tasks and responsibilities based on skills and experience of each team member. Below are the individual responsibilities:

- 1) Kunal was responsible for the design of Database and implementing its functions; Java-based programming
- 2) Shashank was responsible for the Web Designing aspect.
- 3) Sohil was responsible for the design of the Database and implementing its functions

Each was responsible for documenting and testing modules developed by him. The general design decisions regarding the algorithms used, strategy followed, choice of data-structures, choice of schema design of the databases was done in consensus of each of the three members' opinions and was concurred by our team guide.

4.3. Communication

As the team size was small, communicating the various aspects of the development was not hard. Moreover, since the three of us lived in the same university dormitory, informal and constant communication about the project took place throughout the development time.

Since this project was taken up as part of our undergraduate course curriculum, we had a faculty guiding and monitoring our progress. We would meet our guide, Dr. Chhabra, at 10:30 AM on every Wednesday of the week, when he would review our work done in that week. The night before the meeting (Tuesday) was devoted to discussing amongst ourselves the plan for the week ahead. On discussing various design related issues with our guide, we would fix a target for the next week and begin work on it. If our guide suggested revisions/corrections to the work done in the week, we would spend the first couple of days working on it and then try to accommodate the original target for the week.

Communication with our stakeholders and other domain experts took place frequently via E-Mail and telephone.

4.4. Managing Project Components

The project components were carefully selected and designed so as to accommodate both individual domain expertise of the team members and functional independence. Each member was responsible for the total

development of the module chosen by him. Once a component was fully developed, its documentation took a day to complete. The various modules were integrated as and when they were developed and tested at the unit level. Any bugs or issues in the integration would be communicated over telephone or through a small discussion between the team members. The fix, followed by its reflection on the documentation would immediately follow.

5. Project Plan

As *Transport4You* is composed of different modules and were developed by different people, we made a project plan in the first week to have good organization in both timing and tasks. Following our knowledge in software engineering, we created milestones for each phase. As we had no experience in developing such a large system and were not aware of most of the associated technologies, we decided to spend the month before the beginning of the semester on doing a thorough feasibility study and requirement analysis.

It was decided from the beginning of the project that we would follow an agile model for system development [Refer section 7]. A constraint on time and the need to quickly learn new technologies and implement ideas in them suited such a proposal.

5.1. Proposed Plan for the Execution of Project

The blue section of the Gantt chart described in Figure 2 represents the split-up of our planned work schedule of our project. Following the design phase, the major part of our work was planned to be spent on implementation, as our team was inexperienced in the technologies required by the problem statement.

5.2. Realized Plan for the Execution of the Project

The red portion of the Gantt diagram [Figure 2] represents the actual split-up of time spent in execution. Most of the split-up in the proposed plan was followed, barring the extra weeks spent in the continuous revision and development of the database related design issues. Relative to the scope of our minor project, the entire project's execution was two weeks behind the proposed schedule. We were particularly off-schedule in the design of the database, as we better understood the requirements and nature of implementation following each iteration of our database design.

Activity	June- July	Wk 31	Wk 32	Wk 33	Wk 34	Wk 35	Wk 36	Wk 37	Wk 38	Wk 39	Wk 40	Wk 41	Wk 42	Wk 43	Wk 44	Wk 45	Wk 46
Project Preparations																	
Feasibility																	

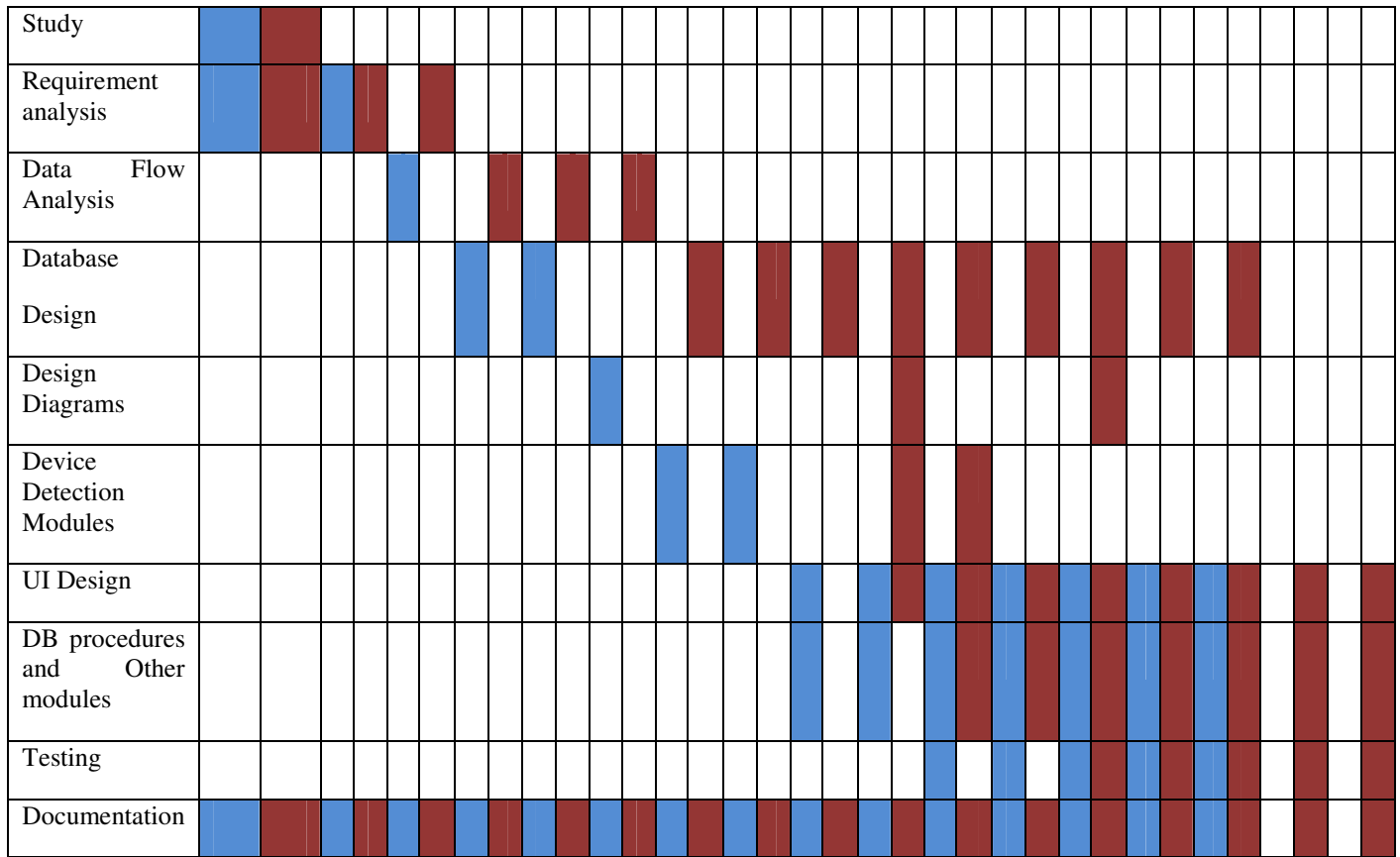


Figure 2 – Gantt chart for project plan
The proposed plan is marked in blue and the realized plan in red.

6. Overall Design

This section contains various aspects of the design of the Transport4You system.

6.1. Design Guidelines

There were some general guidelines which were followed in the entire design process of the system. Some of these guidelines were –

- Open Source Technologies** – The technologies used to implement various aspects of the system had to preferably be open source and free. These technologies generally are supported by strong community-based development and maintenance teams.
- Scalability** – Any component developed would have to contribute to the general scalability of the entire system. This would mean gracefully supporting growing numbers of users of the system and introduction of newer requirements not fully identified during development.
- Software Engineering Principles** – The principles of software engineering were to be made use of extensively. This included use of design aids such as Use Cases, Sequence Diagrams etc. to better

understand and model the requirements and use of a suitable development model for the entire execution of the project.

Having these as our guidelines, we went onto prioritize the development of a fully functional system over the development of a fully functional *scalable* system. This priority was a consequence of the sheer complexity in the working of the system. We wanted to ensure core functionality before we moved onto ensuring that the system would scale to extensions required in the system.

6.2. Design Architecture

The initial decomposition of the system was a result of the boundaries of the system. The different parts that form the system, namely *Device Detection Mechanism*, *Database and Server Management* and *User Interface*, are connected only by a communication medium. Device detection mechanism and the database and server management system are connected together by a GPRS connection, which helps in transfer of the log file from the bus to server. The web based user interface connects to the server through HTTP.

Any further decomposition of any of these components into sub-subsystems is on the basis of functionality. For example, the Database and server management is divided into three components, namely *log processing/learning path pattern*, *finding alternative path* (in case of path unavailability) and *suggesting optimal path*.

This clear way of decomposing of the system led to the design and development of functionally independent components which were loosely coupled. This added greatly to the scalability of the entire system.

Figure 3 depicts a level one data flow diagram of the system. It identifies various modular sections of the system discussed and the nature of interaction between them.

6.2.1. Design Issues

Some issues in the design that were inherent to the problem statement included –

- i. Successfully managing to detect and track large crowds travelling in buses, with a very low expectation of user-initiated interaction with the bus-systems when travelling.
- ii. Keeping the instant messaging between the users and the system to a bare minimum.
- iii. Identifying and notifying only those users who board a bus and not other users in the detected in the vicinity of the bus who do not board or intend to board the bus.

6.3. Component Design

Key strategies used in the design of components in an attempt to tackle the challenges listed include –

- i. **Detection of Users Boarding a Bus**

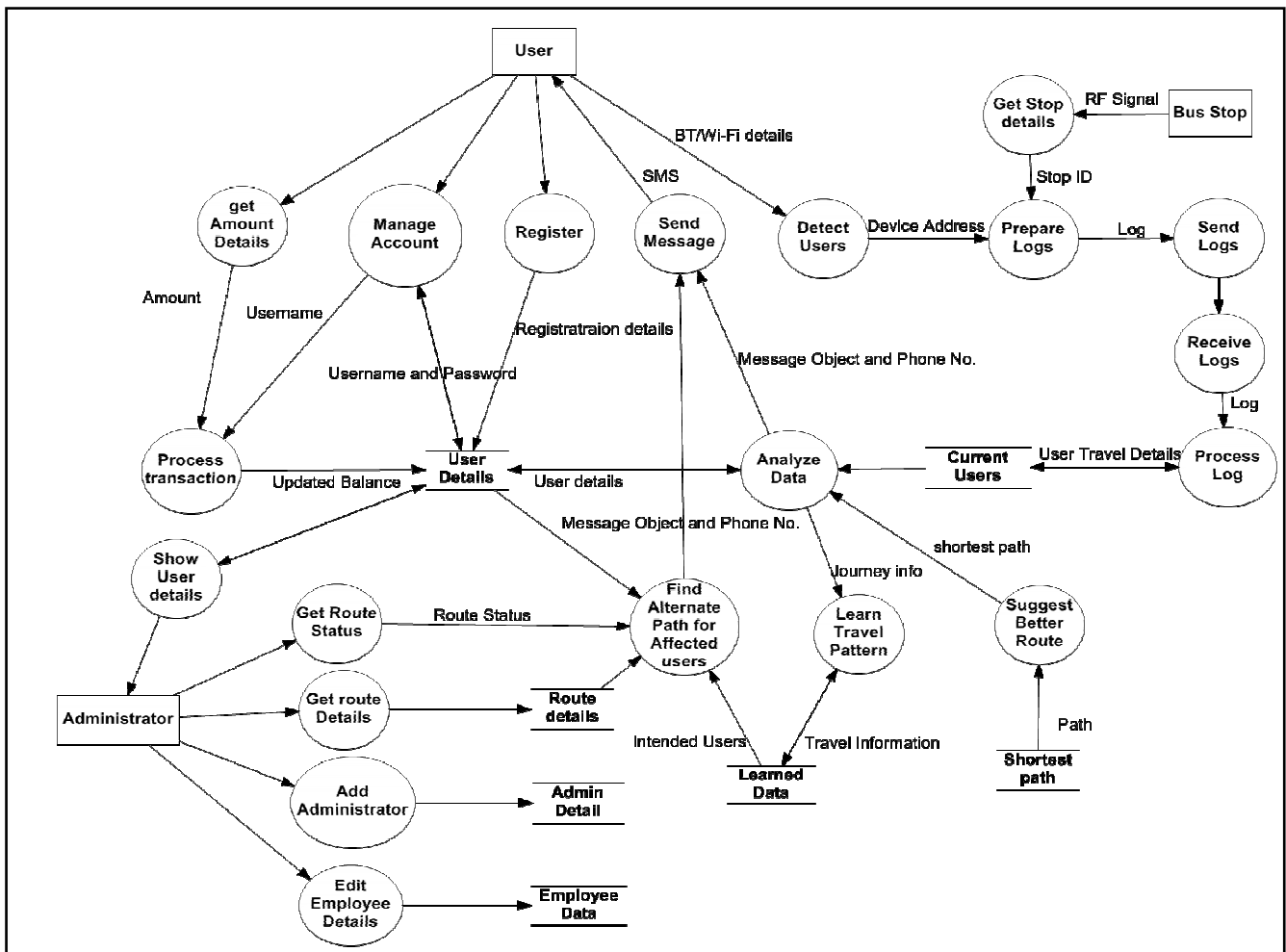


Figure 3- Level One Data Flow Diagram for the System

In order to recognize users using the transportation system, it was needed to first identify them on the basis of the address of the device carried by them. This identification is being done by scans performed by the on-board system to obtain Bluetooth/Wi-Fi networks in its vicinity. For the purposes of this project, we have focussed on identifying only Bluetooth connections and not Wi-Fi connections as their nature of working is quite similar. The implementation of a Wi-Fi connection detection system shall be quite an effortless extension to our existing implementation of the system.

On identification, it was decided to store the collected data as a flat file database [1] on the detection hardware's memory. A log of the device addresses detected is stored against the time of the detection and the stop details in the flat file. The bus-stop's details such as name and ID of the bus-stop are obtained from the system available at every bus-stop [Refer to Section 3.5, Assumptions and Dependencies sub-section of this report].

The overall strategy adopted to detect users is –

The scan operation of the on-board system is triggered as the bus leaves a bus-stop. The detected users from the scan are stored in the flat-file. Comparison of the logs of users detected at the present stop and at a previous stop reveals those users who are still on-board, those who've just boarded and those who de-boarded.

The scan operation occurs twice, with a time-gap of 30-40 seconds. This is done assuming that the bus would have moved a considerable distance away from the bus-stop in the span of those 30 seconds, reducing the chances of detecting a user who does

not intend to board the bus but was detected at the bus stop anyway. Comparison of the files generated by the two scans identifies such users. Their details are then discarded from the logs, resulting in one final log of users for that bus-stop.

This strategy thus keeps track of every user, thereby making fare calculation and path learning/suggestions for each such user possible.

ii. Relay of Log-Files on Buses

Transmission of the final flat-file containing details of users on-board and users de-boarded at a particular bus-stop follows the end of the two scans performed by the detection system after moving away from that bus-stop. This final file is transmitted to the central server through GPRS. Refer to Section 3.4.2 for a justification of such a choice.

iii. Model of the City

The map of the city is being stored as a directed graph [2], with bus-stops acting as nodes in it and the roads connecting two such stops representing the edge between two nodes. The system infers a journey performed by a user by keeping track of the stops at which she gets in and out of buses. The information received from the log-files is translated as movement on the graph of the city. A journey is thus realized as the combination of buses (therefore, a combination of the bus-stops) the user takes. Modelling the city

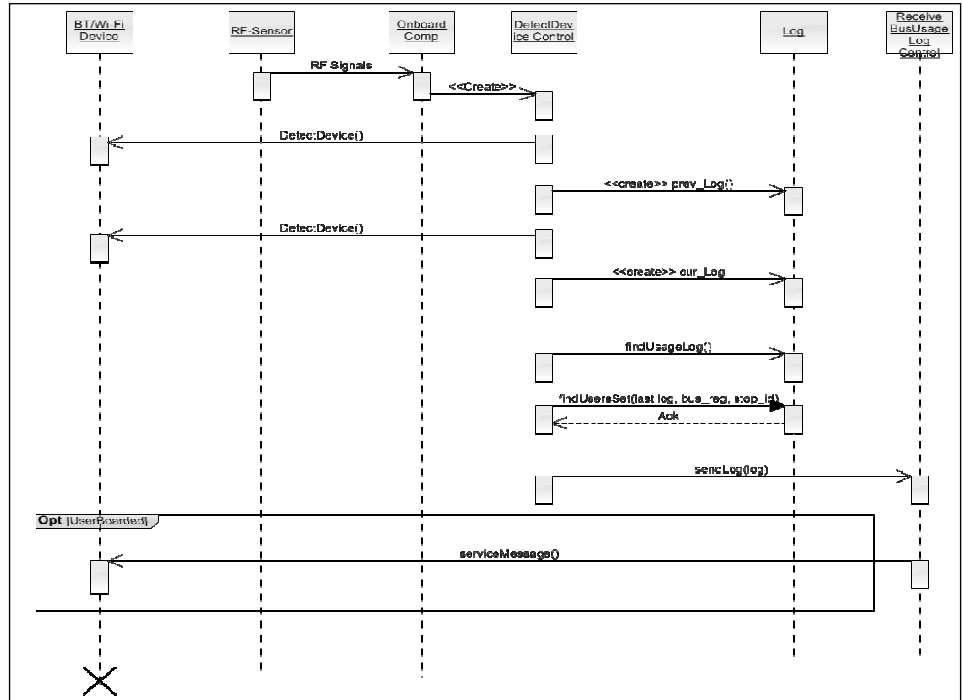


Figure 4– Sequence Diagram for Device Detection

as a graph also enables use of graph-theory based algorithms to calculate shortest/fastest paths and alternate paths [3].

It is required that the shortest path be calculated for every user as part of the user-centric services Transport4You offers. However, performing such an operation for every user would be a time-consuming process, given that the operation is of the order $O(n^3)$ [3]. Hence, the shortest path between every two bus-stops on the city's graph is pre-calculated, so as to quickly dispense suggestions to users. The graph is also open to editing by the Administrator of the system. The pre-calculations for shortest paths are re-done whenever the graph is edited.

iv. **Fare Calculation**

The fare calculation is done based on time coupons (referred henceforth as T_VAL) provided to every user. Each T_VAL costs T_COST units of currency (INR, dollars, euros). T_VAL is deducted for every second it is detected that a user is travelling in the system. This is applicable even if the user de-boards a bus and boards another bus several times to reach an intended destination. A user is charged another T_COST if she continues to travel after the expiration of T_VAL units of time. In such a case, T_VAL of a user is reset and the deduction continues.

Our strategy to keep track of users and calculate fare for each is as follows –

T_VAL of a user is updated after every log from the bus she is travelling in reaches the central server. This log, as discussed above, provides stop-by-stop information of user movement. If a user de-boards a bus and then her T_VAL expires, a half an hour of buffer time (this is a configurable parameter of the system. Refer to *Appendix B, ER Diagram*, for a list of configurable parameters) is allowed to pass after which the system infers that the user has ended her journey at the stop she was last detected to de-board. This follows with the resetting of T_VAL for that user, calculation of total cost for that journey, messaging that cost to the user and saving the path details of the journey for that user in the system database.

However, if the user boards another bus within the buffer-time (which is realized from the log-file received from that bus), it is assumed that the user is still continuing the same journey she was making when she got off the bus last detected at.

v. **Path Learning**

The path learning of a given user is being performed by defining a threshold for the number of times a given user uses a given path. Every journey of a user is stored in the system database. If the same journey is performed a number of times by a user, it is inferred that the journey's path is frequently used

by her. It is this information that the path suggestion routines look up to identify specific users who would be affected by the unavailability of a path.

Our team was not able to design and implement a full-fledged machine learning algorithm due to a time constraint. However, we did analyze a few algorithms and after discussing it with experts in the industry (one engineer from Yahoo!, Bangalore [4] and one from Alcatel Lucent Bell Labs, Bangalore [5]), the following designs to learn the paths of users seemed efficient–

- a. Markov Decision Processes [7]
- b. Data mining through frequent item set calculation [8]

vi. User Notification

The user is notified of her journey details as and when the system infers that the user has completed one journey. In real time, this message would arrive typically after half an hour the user has made the journey. This is so because the system waits some units of buffer time before concluding that the user has terminated her journey and is not waiting to board a connecting bus to elsewhere. Since there is no explicit user-acceptance requirement, a message being delivered within half an hour of the journey should be acceptable.

As a result, all balance details are provided at the end of a user’s journey. Since there is no explicit requirement for the user not to receive messages at the end of a journey, we went ahead with the design. However, if there is a requirement to notify the user before the commencement of a journey, the system is scalable enough to incorporate it.

The extent to which a user wishes to be notified about the information being generated by the system is controlled through a set of choices being explicitly provided in the web-based registration portal. These choices are looked up before messaging a user. If a user does not wish to be disturbed through messages, it’s followed so.

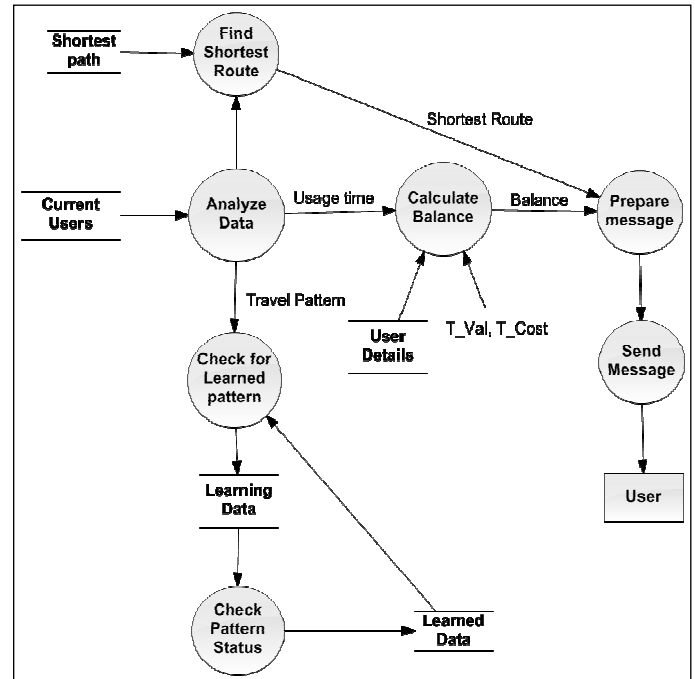


Figure 5– DFD for Path learning functionality

6.4. Database Design

The design choices made resulted in the ER Diagram described in Appendix B. The current design is normalized to the best extent possible. The tables allow excess data to be stored in it, which is obtained as a result of some pre-computation operations. This helps save user-response time in notifying shortest paths, alternate paths and other details.

6.4.1. Key Design Features

i. Inform Alternative Routes

To achieve this functionality the system needs to find all the intended users that normally uses the path which is currently unavailable. This information is available from the *learnedroutes* table.

To suggest alternative paths in case the actual path used by the user is not available, we will use the *tempshortestpaths* table, which contains the alternative shortest paths for each pair of reachable nodes.

ii. Suggest Optimum Routes

To suggest optimum routes, the system stores all possible sets of optimum routes. For a given start-stop and end-stop in a user's journey, we can compare the route used by the user and determine if it is optimum or not.

iii. Travelling

The system needs to track the stops where the user boards the bus and the stop where the user de-boards the bus in order to form a path of travel. The *currentusers* table and the tables *boardingusers* and *deboardingusers* are used to maintain this information.

6.4.2. Performance and Maintenance Design Features

- i. The Start Stop and the End Stop of a user's journey are being stored along with the path travelled in all the tables even though it's possible to determine them, given the path. This has been done to minimize complexity as calculating them is an expensive process. Also, indexing these fields is easier compared to indexing the path.
- ii. The *routes* table will contain all possible routes on which buses can ply on rather than just the routes possible in the system. This is a one-time process which saves processing time when calculating the path used by a user and the shortest paths between any given pair of stops.
- iii. The shortest route between any two bus-stops can be calculated any time from the information available from the routes and the *busrouteinfo* table. But to optimize the process, the shortest paths between ever two stops reachable from each other has been pre-computed and stored in the *shortestpaths* table.

6.4.3. Other Features

- i. Everyone who interacts with the system, including the general user, the administrators, the drivers of the buses etc. will have a *user_id* which will be unique. The *user_id* for each system user will be defined in the *Users* table in the database.
- ii. A user can have multiple devices registered. These addresses will all be stored in the *user_devices* table in the database.
- iii. Irrespective of the number of devices a user has registered with the system, the messages sent to that user will all be to a single phone number which will be given by the user while registering and stored in the *system_users* table in the database.
- iv. The maximum number of alternative/optimal paths the system finds for a user can be specified by an administrator as a system parameter. This has been limited to 5.
- v. The frequency of usage of a route by a user after which it is considered to be learnt is also a system parameter and can be configured by an administrator. Also, the maximum period of inactivity after which a learnt route is discarded is a system parameter, also configurable by an administrator.
- vi. The System Parameters that are being used are all stored in the *SystemParameters* table rather than being passed as arguments to procedures to make the system more configurable. These parameters can be modified by an administrator at any point in time.

6.4.4. Robustness of Design

The design we have for the various components of the system successfully manages corner cases of internal calculation and user services. Some cases identified that the design gracefully manages are –

- i. **Users Detected in the Vicinity of a Bus** : Discussed above
- ii. **Failure in Communication of the Logs Generated by a Bus**: This case is a possibility, even though we assume that the connection is as reliable as possible. Barring the case that the system fails to detect both, the boarding and de-boarding of a user, a partial failure of the communication system is handled well by the design. A partial failure may result in the following scenarios -
 - a. A user may have been detected to have boarded a bus but the log containing his de-boarding information may have failed to transmit.
 - b. The detection system may have failed to detect the boarding of a user but may detect the de-boarding of the same user at a different bus-stop.

Given the thorough data for each user movement being stored in the database, it is easy to provide the logic to identify occurrences of such scenarios. The current handling of such cases ensures that the user is not billed extra and any discrepancy found in the cost is borne by the system.

- iii. **Path Learning:** Although the current path learning algorithm follows a naïve statistical inference model, it ensures, within the limits of its applicability, that the users identified as intended users of an unavailable route. This includes identifying the intention of travel based on the time of the day and day of the week. As an illustration, if a user uses an early morning line only on Mondays and the same is unavailable on a Wednesday, he shall not be notified of any alternate routes.

6.5. User Interface Design

The user interface in our project is presented as a web-based portal which enables both, a User of the system and its Administrator to carry out functionality attached to each role.

A User may make use of this portal to interact with the Transport4You system, wherein, she provides choices in preferences, credit to deduct from, device details which he shall use so as to enable his detection on-board a bus etc. The data entered is the basis of actions performed by the system concerning the notification of system parameters such as route availability status, user balance information etc. The data entered is also a functional requirement for the working of the on-board detection systems, which facilitate the identification of a User.

For an Administrator of the system, the portal acts as means to enter information concerning the buses and routes in the systems. She is given the privilege to enter details of the various bus-stops of the city, the routes connecting these stops, the Bus IDs plying on these routes and the actual buses assigned to each of these IDs. A natural constraint exists in both, the addition and deletion of these details.

This core functionality aside, an Administrator is also privileged to view details of the various registered Users of the system and assign corresponding designation (‘roles’) to Users if they belong to the Transport4You system.

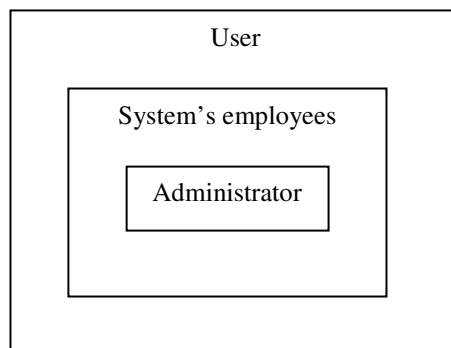


Figure 6- Hierarchy in the User Interface usage

7. Development Cycle

The development strategy used was basically a combination of both the **agile software development model** and the **iterative and incremental software development model**.

The whole system was complex and included many functional parts to perform together to provide the system's required functionality. There were components on the bus side that interacted with the server, there was a UI through which the user could interact with the system and the administrator could administer the whole system through a dedicated portal. Additionally, there were functionalities like learning from user activities, providing users with updates through messaging, alternative path suggestions and more. Therefore, the functionalities were developed through various iterations, each iteration adding a new functionality to the system.

A component based development approach was employed and the project was broken down into smaller components, wherein each component provided exclusive functionality or supported other components to help them deliver their functionality. These components were developed as a part of various iterations.

The team was small, comprising of three members only. Each team member was, as a result, responsible for developing a complete component and followed it up with testing and integrating the functionality with the overall system.

As a part of the first iteration, the device detection mechanism was implemented and an initial instance of the supporting database was put into place. This database instance witnessed many changes during other iterations, based on project requirements realized. The major components developed by distinguishable iterations are mentioned in Figure 7.

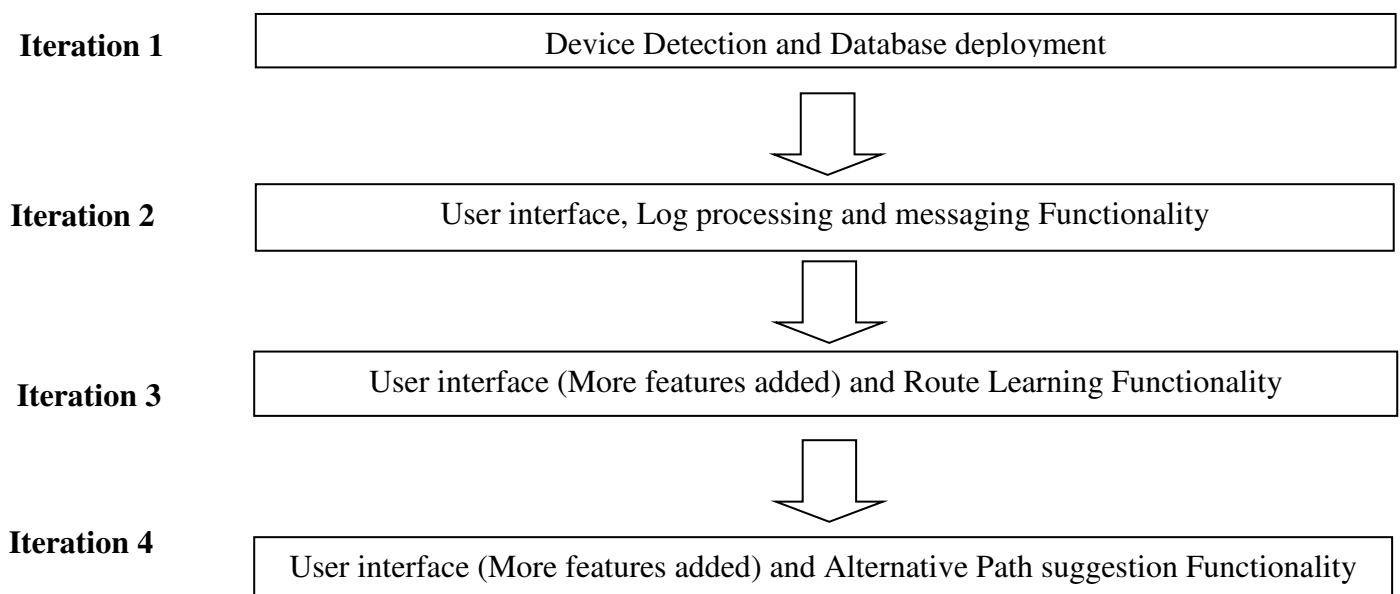


Figure 7 - Development Cycle of the Project

The complete User Interface was developed over iterations as various components representing core functionality requiring a user interface were added iteratively.

8. Implementation

The implementation aspects of the various modules involved are discussed in this section -

8.1. User Interface to Allow Citizens to Register and Pre-Pay

A web-based User Interface was developed for citizens to register into the system. The entire user interface was developed within CodeIgniter [10], a framework facilitating secure, rapid development of web content through PHP. The framework followed a loosely implemented Model-View-Controller [12] model. The implementation of the User Interface worked closely with the implementation of the Database operations. Hence, a constant communication between developers of these two modules was required. The website produced offers rich functionality for a registered user. One may view past journeys performed, register multiple devices with the system and pay through cash cards. The payment gateway in the current implementation is only being simulated through a simple form. This is so because the payment option was not one of the core functionalities we focused on. Moreover, acquiring such a gateway from a commercial vendor required an initial capital. PHP-based APIs from any online vendors such as PayPal [13] can be easily integrated in our current implementation to provide this functionality. The same applies for the messaging gateway.

The framework provided a convenient model to connect to a MySQL [11] database and also provided rich libraries for securing and validating form data.

8.2. Recognition of Registered Users On-Board

The modules for device detection are implemented in C keeping in mind that C is the choice in embedded systems. The main modules implemented include a device detecting program and a module to prepare final logs. The device detection program is implemented for Bluetooth with the help of BlueZ [9], which is the canonical Bluetooth stack for Linux. The device detection on the bus is designed to perform device detection twice at an interval of X seconds; this interval is dependent on the speed of the bus so it has to be configured before the system goes online. The functionality of finding the users from the log who boarded or de-boarded the bus at a bus-stop is performed on the bus' hardware in order to minimize bandwidth usage. As a consequence, the final preparation of the log requires the on-board system to keep a copy of previous logs to find the boarding and de-boarding users. The information about the stop (available from the RF Sensors on the stop), the time and the bus information is also sent with the log to the server.

8.3. Fare Calculation and Route Suggestion

8.3.1. Database

Due to lack of practical experience in implementing scalable databases, we faced a big challenge to implement an efficient database that supports all the functionalities of the system and is scalable as well. After discussing the issues regarding overall design with a domain expert [6], we decided to come up with a basic design that supports all the core functionalities of the system. Then, after a number of discussions within the team, we came up with the current design over a number of iterations.

The major challenges we faced while designing the database were regarding storage of routes, their corresponding buses and supporting the functionality of finding alternative routes. The problem we faced while storing the routes and corresponding buses was- how to store which bus is related to which route in order to efficiently support the queries to find the path and distance travelled by a user, given the *bus_id*, *start_stop* and *end_stop*. These queries are used while processing the logs from the buses.

Major functionalities implemented through the database procedures include log processing, finding intended users, processing completed journey and classifying regular journeys of users. While implementing the procedures, many hidden requirements emerged when trying to support an efficient design of the procedures. Loading of the *current_users* table with the boarding and de-boarding information about the users, a module which was planned to be implemented fully in Java, was implemented completely through database procedures to increase its efficiency. As a result, two tables - *boarding_users* and *deboarding_users* tables were added to the system. The messages formatting functionality for alternate path suggestion was also separated from the database to remove an unnecessary load on the database.

Various parameters of the system like TCOST, TVAL, BUFFER_TIME (time), etc are made configurable and are kept in the *system_parameters* table. These parameters can be edited through administrator's privileges in the web portal.

8.3.2. Java Modules

Java modules are implemented for two main purposes - firstly, to share the load of the database and secondly, to form an interface between two components. The *loadTable.java* module that loads the *boarding_users* and *deboarding_users* table basically forms a connection between the database and the program on the server side that receives logs from buses.

There are separate java modules to prepare messages for the alternate path and the optimum path suggestions. These modules are implemented separately because they require almost no support from the database. The final outcome of these messaging modules is text files that contain the messages that are to be forwarded to the users

concerned. The content in these files could be directly used in any messaging API that shall be used in the system.

In the implementation of the shortest/fastest path calculation, an important observation was that the shortest and the fastest path calculation between two stops in the system did not necessarily yield the same result. Such a scenario was observed where a shortest path calculated between two stops required changing a lot of buses on intermediate stops, which rendered such a path to be a ‘slower path’ when compared to another path which was not necessarily the shortest (distance-wise) between the two stops. So, to take into account the delays while changing buses at the stops, a new overhead was calculated at each stop. This overhead is directly proportional to the frequency of the buses arriving at the bus stop that are headed towards the user’s next destination.

The modules for path calculation of optimum path and alternative path are similar. The input to these modules is the differing factor. The optimum path calculation takes as input the graph of the city whereas the alternative path calculation module takes in the graph of the city with the current unavailable routes removed. The algorithm followed to calculate the path is the same.

9. Testing

The system has been thoroughly tested at the unit level. The integration of each of the modules has been tested for cause-effects.

The user interface had close to 20 registered users by the first day it went online on the dormitory LAN. All the basic functionality was tested by them, invoking a positive feedback of the interface.

A thorough testing of the system involving the functional testing (equivalence class testing, boundary-value testing) and structural testing is being carried out and documented at the time of submission of this report.

The following table summarizes our test plan -

Functionality	Test performed	Expected Result	Result
Onboard user detection	Some Bluetooth devices were kept in range of the system, while the device detection module was executed. Also some of the devices were kept temporarily in the range of the system Bluetooth to simulate users in the vicinity while the bus is moving.	<p>The final log generated by the device detection system should have two lists of users, those who have boarded and those de-boarded, in the required format.</p> <p>The temporary user’s devices should not be included in the final list.</p>	The device detection module, modules to compare logs and the driver programs are working as expected.

Log processing and optimum path generation.	<p>A sample log sequence from a bus was manually generated and fed to the loadTables.jar modules, to load the boarding_users and deboarding_users module.</p> <p>System was also tested for special cases like intermediate logs are missing.</p>	<p>The boarding_users and deboarding_users are loaded properly.</p> <p>The journey info of a user is recorded in the current_users table.</p> <p>The learning_routes or learned_routes tables are updated with the info available</p> <p>The messages regarding the completed journeys and optimum path as required are being generated.</p>	<p>LoadTables.java module is working properly.</p> <p>The BoardedBus, and DeboardedBus procedures working properly.</p> <p>processCompletedJourney and learnRoute procedures are working properly.</p> <p>prepareMessage.java and optimumPath procedures are working properly.</p>
Alternative path	<p>The unavailable path is informed through admin portal and message to be sent to the intended users are created.</p>	<p>The users selected to be informed satisfies the criteria of intended users.</p> <p>The final messages are generated.</p>	<p>alternativePath.java and prepareMessageType2.java modules are working properly.</p>
User interface	<p>Admin and user functionalities of the UI are tested.</p> <p>When unavailable path is submitted to the system the message regarding the alternative path are being generated.</p>	<p>The database is updated with the new information available from the admin or users.</p> <p>The alternative path functionality works as expected.</p>	<p>UI is working properly with the rest of the system</p>

10. Outcome and Lessons Learnt

The developed model is able to demonstrate all the core requirements of the system. The following list describes the salient features the system has and some features which have not been implemented but are easily to extend to the existing implementation.

10.1. Key Features Implemented

- User registration through a web-based portal. The portal can be accessed at www.shashank.pciot.com. Refer to Appendix A for screenshots of the portal.
- User detection on-board a bus.
- Finding alternative path, given the movement of a user.
- Finding optimum path, given the movement of a user.
- Fare calculation for a user, given the movement of a user.

10.2. Features Not Implemented

Based on the complexity of the certain functionality identified and due to the time constraint that came with the execution of our project, we decided to leave out the following functionality out of the scope of this project.

- i. Instant Short Message Servicing (SMS) between the system and a user - due to finances involved in buying a dedicated messaging server.
- ii. The implementation of a Wi-Fi detection module - due to functional similarity with the working of the implemented Bluetooth detection module.
- iii. GPRS Communication between a bus and the system - due to finances involved in buying a dedicated messaging server.
- iv. Recharging a user's credit account for the system through a payment gateway - due to finances involved in buying a dedicated merchant.

These four functionalities are being simulated through appropriate stubs for the purposes of our project.

- v. A thorough path learning algorithm, implementing advanced concepts in the theory of probability and statistics – due to a time constraint and complexity involved in such modeling methods. As of now, we follow a naïve statistical inference model.

The development of the project introduced us to the distributed development of a system. None of us had any experience in designing and developing from scratch such a complex system. It was a good learning curve for all us.

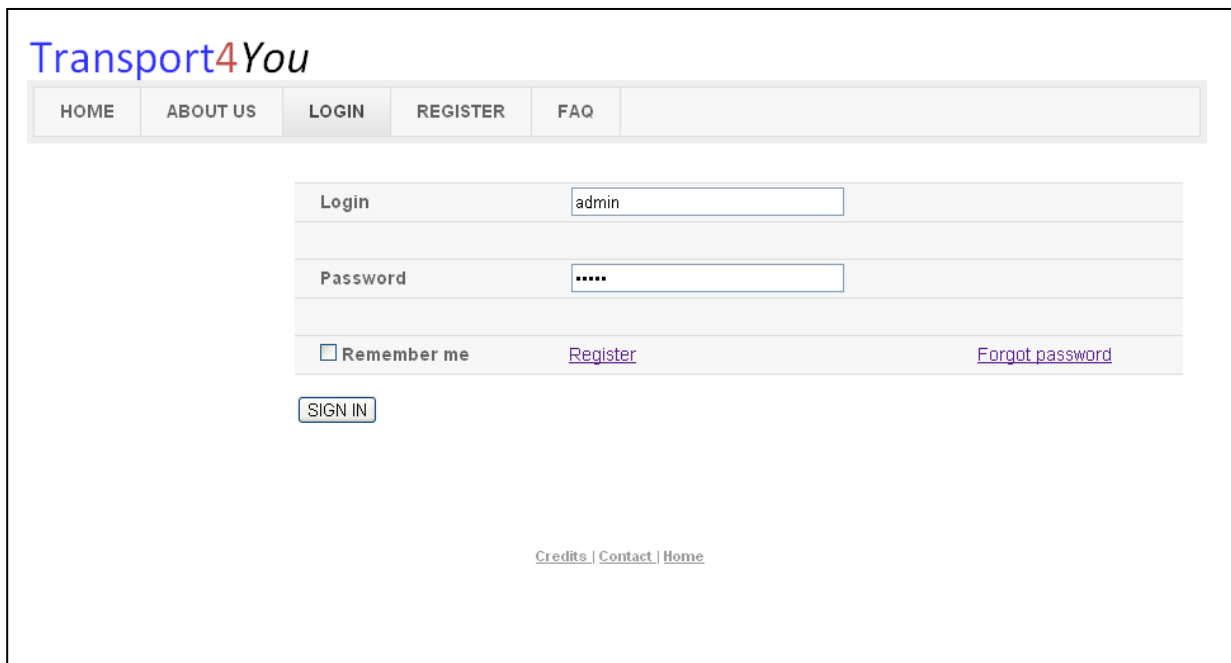
An important aspect which the project taught us was the importance of software engineering principles. Until now, we just had a theoretical exposure to the subject and had no perception of the impact of such principles in the design of such systems. In foresight, it now makes us realize how important an organized, well tested approach to the design of the system was in effectively meeting goals within the decided time frame. This project also gave us opportunity to code and gain practice in some very useful technologies like PHP, Java etc. Also, a study of the working of payment gateways and messaging gateways was an educative experience. The documentation that we had to prepare for both SCORE and our course curriculum has definitely improved our ways of gathering and expressing our ideas.

11. Appendix

A. Screenshots

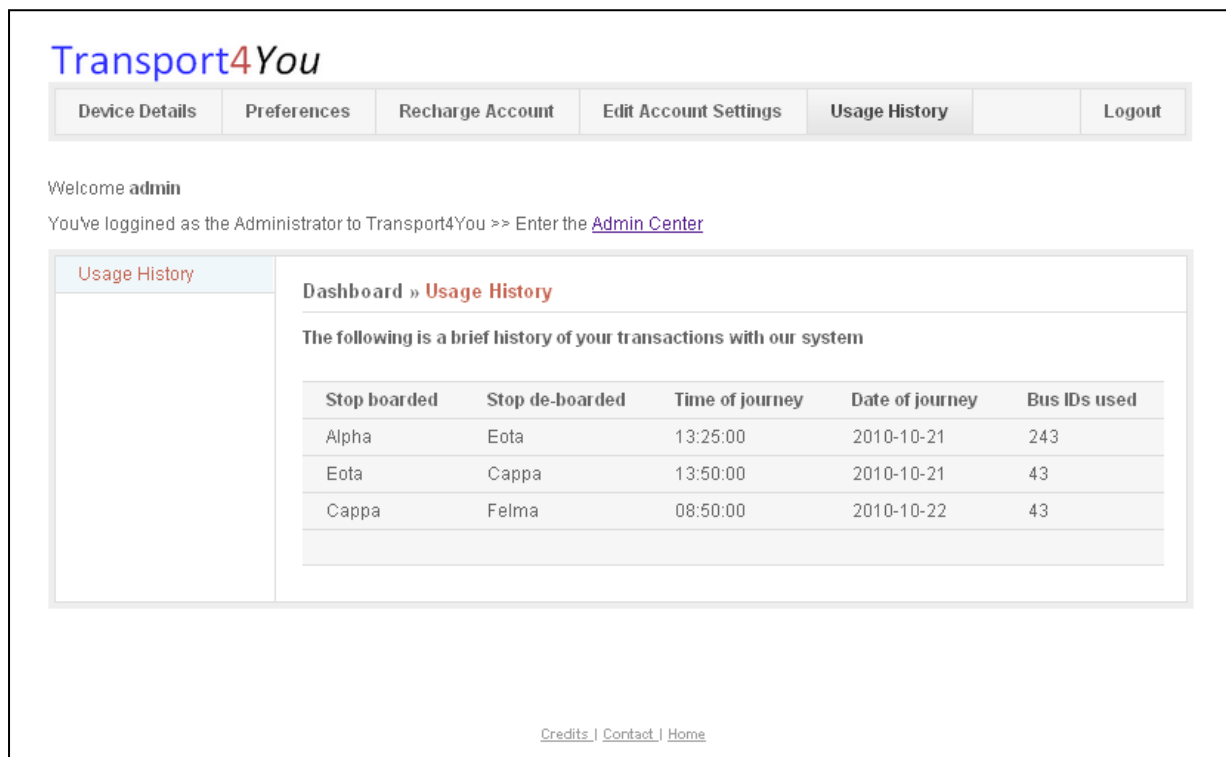
The following are screenshots of the implemented registration portal for Transport4You, available at www.shashank.pcriot.com

1. Login Page – Transport4You



The screenshot shows the Transport4You login page. At the top is the logo "Transport4You". Below it is a navigation bar with links: HOME, ABOUT US, LOGIN, REGISTER, and FAQ. The main content area contains a login form with fields for "Login" (containing "admin") and "Password" (containing "*****"). There is a checkbox for "Remember me", a "SIGN IN" button, and links for "Register" and "Forgot password". At the bottom, there are links for "Credits", "Contact", and "Home".

2. Usage History Displayed for a User



The screenshot shows the Transport4You usage history page. At the top is the logo "Transport4You". Below it is a navigation bar with links: Device Details, Preferences, Recharge Account, Edit Account Settings, Usage History, and Logout. The main content area shows a welcome message for "admin" and a link to the "Admin Center". A sidebar on the left has a link for "Usage History". The main content area displays the "Usage History" section with a table of transactions.

Dashboard » Usage History

The following is a brief history of your transactions with our system

Stop boarded	Stop de-boarded	Time of journey	Date of journey	Bus IDs used
Alpha	Eota	13:25:00	2010-10-21	243
Eota	Cappa	13:50:00	2010-10-21	43
Cappa	Felma	08:50:00	2010-10-22	43

Credits | Contact | Home

3. Add Bus ID (Provided in Administrator's Center)

Transport4You

Users

Buses

Routes

Back to Dashboard

Add Bus Stop

Delete Bus Stop

Add Bus ID

Delete Bus ID

Add a Bus

Delete a Bus

Dashboard » Admin Center » Buses

Add a Bus ID into the system

Set a Bus ID

143

Reset Bus ID

ID Journey Time

01:20:00

ID Frequency

15

ID Start Time

07:00:00

ID End Time

22:00:00

Select a stop

Alpha

Add stop to From-path

Add stop to To-path

FROM Path

TO Path

Stops added: 3

Stops added: 3

Alpha

Cappa

Eota

Beta

Cappa

Remove stop

Alpha

Remove stop

Submit stops

4. Inform Alternate Path (Provided in Administrator's Center)

Transport4You

Users

Buses

Routes

Back to Dashboard

Add Route

Delete Route

Inform Bad Route

Check Route

Dashboard » Admin Center » Routes

Inform a non-functional route on the system

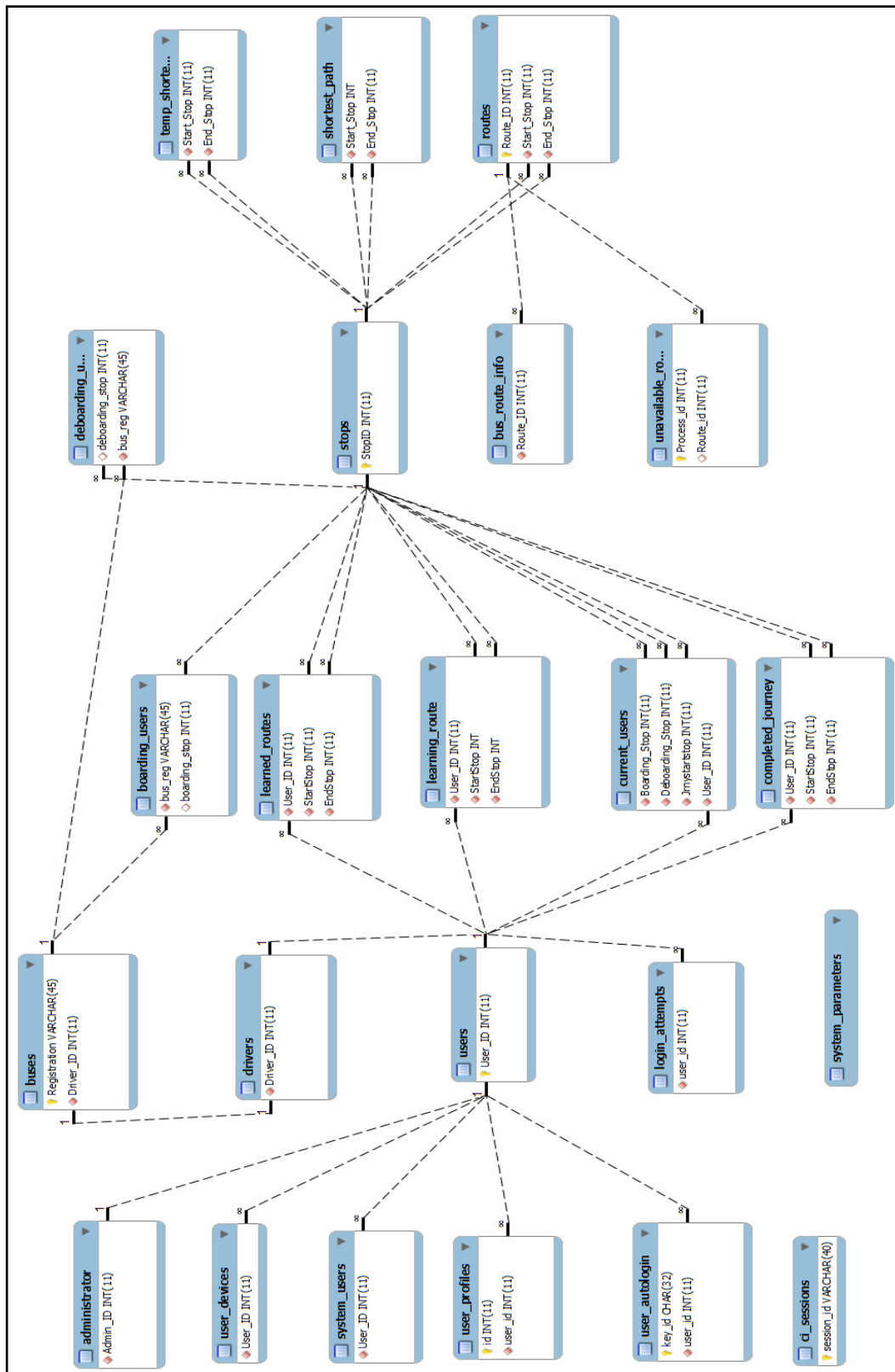
The following is a list of paths currently non-functional

Process	Unavailability		Reason	
ID	From	To		
2	15:56:00	17:45:00	major traffic congestion at Briand Square and Telephone exchange junction	Clear

Add an unavailable path

B. Entity-Relationship Diagram

The figure describes the Entity-Relationship (ER) diagram for the system. It depicts an abstraction of the comprehensive ER Diagram described in the Design Document (Section).



12. References

1. *An introduction to Flat File Databases*, http://en.wikipedia.org/wiki/Flat_file_database#Flat_files; Last visited on 20 November 2010
2. *An introduction to Directed Graphs*, http://en.wikipedia.org/wiki/Directed_graph; Last visited on 20 November 2010
3. *Floyd Warshall Algorithm*, T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Second edition, Sept. 2001.
4. *Radha Chitta*, Systems Engineer, Yahoo! Bangalore, <http://sites.google.com/site/radhacr/>
5. *Dinesh Govindaraj*, Member, Technical staff, Alcatel Lucent Bell Labs, Bangalore, <http://sites.google.com/site/gdineshcse/home>
6. *Praveen Kumar*, Project Lead, Tata Consultancy Services, Gurgaon, India, praveen.kum@tcs.co.in
7. *An introduction to Markov Decision Processes*, http://en.wikipedia.org/wiki/Markov_decision_process; Last visited on 20 November 2010.
8. *An introduction to Association Rule Learning*, http://en.wikipedia.org/wiki/Association_rule_learning; Last visited on 20 November 2010.
9. *Bluez*, A Bluetooth stack for Linux, <http://www.bluez.org>, Last visited on 10 November 2010.
10. *Codeigniter*, A PHP Framework, <http://www.codeigniter.com>, Last visited on 20 November 2010.
11. *MySQL Workbench*, A tool for generating E-R Diagrams of database schema, <http://dev.mysql.com/downloads/workbench/>, Last visited on 1 November, 2010.
12. An introduction to Model-View-Controller architecture, Last visited on 1 November 2010. <http://en.wikipedia.org/wiki/Model%E2%80%93View%E2%80%93Controller>
13. *Paypal*, An e-commerce business allowing payments and money transfers through the Internet. <http://www.paypal.com>, Last visited on 3 November 2010