LocateMyBus: IoT-Driven Smart Bus Transit

Karthik Desingu*, Daniel Mark Isaac†, Mirunalini P.‡, Bharathi B§ and Cherry Mathew Philipose¶
*‡§Dept. of CSE, Sri Sivasubramaniya Nadar College of Engineering, India
†Dept. of ECE, Sri Sivasubramaniya Nadar College of Engineering, India
¶Dept. of English, Shiv Nadar University, Chennai, India
*karthik19047@cse.ssn.edu.in, †daniel19019@ece.ssn.edu.in, ‡miruna@ssn.edu.in, §bharathib@ssn.edu.in,
¶cherrymp@snuchennai.edu.in

***Abstract*—Buses are a common mode of commute in metropolitan cities. Uncertainty of traffic makes it difficult for buses to adhere to schedules, despite predetermining them, making it strenuous for commuters to plan local travel reliably. The proposed LocateMyBus system leverages Internet of Things(IoT) set-ups at bus stops and buses, and Machine Learning(ML) to assuage this uncertainty. The IoT network allows commuters to track live-running-status of buses, and disseminate tentative and live-status to commuters through Public Announcement(PA) systems at bus-stops, and a web-application interface. By eliminating the use of GPSs, it avoids the need for system database updations when bus-stops are relocated and eradicates the computation requirements of GPS-based location estimation, whilst providing stop-level granularity in bus-tracking. The proposed schedule prediction provides a tentative schedule of buses with stop-wise arrival times estimated using ML based on historic and real-time route data. To build and test the proposed prediction algorithm, arrival times of two bus-routes in the Massachusetts Bay Area were collected for a period of four months by periodically querying its real-time General Transit Feed Systems(GTFS). Furthermore, the proposed system's ability to log bus arrival times at each stop paves the way to building real-time GTFSs.**

***Index Terms*—Internet of Things, Machine Learning, Smart Bus Transit, Transit Feed Systems, Deep Learning, Schedule Prediction.**

I. Introduction

Commuting plays a key role in everyday life. The factors that prevent hassle-free commute include indefinite waiting, impractical signage, and inaccurate schedule, among others. Commuters are usually unaware of the arrival time of their buses. New commuters seldom have a reference to the public transport schedule. The present scenario forces commuters to be on the look out for buses and struggle to read signs on buses, amidst the waiting crowd and road traffic, making it uncomfortable and unsafe. Such issues often encourage people to resort to private transport, and cab services. Reduced use of mass transports increases traffic congestion, and vehicle emission. Furthermore, it increases travel time, fuel usage, and decreases accessibility and mobility. To alleviate traffic congestion, although different techniques have been suggested: demand-side measures such as congestion pricing and traffic management; and supply-side measures such as constructing more roads and adding lanes; expanding public transportation services is reportedly considered as one of the most promising technique [1] [2].

Providing reliable and accurate arrival information is key to improving the service of bus transit systems [3]. It will attract people to the use of public transport, and increase their contentment [4]. Travelers can be provided with reliable travel information using Advanced Public Transport Systems (APTSs), a key component of Intelligent Transportation Systems (ITSs) [5]. By using automatic vehicle location and identification, traditional bus transit service can be improved manifold. In practice, it is common to have several bus routes sharing the same road segments and bus stops. Passengers can choose different bus routes to reach their destinations. Hence, knowing when the next buses of multiple routes will arrive is useful in saving passenger waiting times and in curbing anxiety [6] [7]. However, real-time travel time information cannot be easily measured and made available directly. Therefore, mathematical models that can track and predict arrival time with reasonable accuracy, and notify the commuters in an organized manner are key to solving the aforementioned problems.

A variety of models for predicting traffic states such as travel time and traffic flow have been developed over the years. The LocateMyBus system proposed in this paper addresses the aforementioned issues in city bus transit using IoT and ML. The IoT modules enable bus arrival time logging for live bus tracking, as well as to collect data for subsequent schedule prediction. Notably, the IoT network eliminates the use of GPSs in providing bus stop level granularity in location tracking: this overcomes the need to update the database when bus stops are relocated, and the need for powerful edge devices, among other caveats of a GPS-based system. The easy-to-deploy set of IoT modules facilitate interconnection of buses and bus stops across a city, maintain a central log of bus arrival times, and pave the way to the development of a real-time GTFS for cities. The ML-based prediction algorithm integrates historic data and real-time arrival information to provide realistic estimates of bus schedules for commuters based on time-log patterns. A web-interface is proposed, in addition to the PA systems at the stops, to provide a convenient means for commuters to access information about running status and expected schedules of buses. A prototype of the proposed IoT system was built using Arduino and Raspberry-Pi boards as proof-of-concept. This work also collected arrival times of two bus routes in the Massachusetts Bay Area for a period of four months by regularly querying its real-time GTFS: this data was used to build and test the ML-based schedule prediction model, and quantitatively evaluate and compare its performance.

The remainder of this paper is organized as follows: Section II presents a summary of related work for city bus tracking and schedule prediction, and highlights the key contributions of this work; Section III describes the proposed system, detailing its individual components namely, the IoT modules, web-server and interface, and the ML-based schedule prediction module; Section IV presents a discussion of the proposed system, comparing it with related work, and describing the modeled proof-of-concept prototype. It also describes the data collection methods, presents a quantitative analysis of the ML-based schedule prediction system evaluated on data collected from the MBTA transit system for the purposes of this study, and presents a comparison with other existing prediction systems; Section V concludes the paper, summarizing the presented system and schedule prediction method, and suggesting future directions.

II. Related Work

A literature survey was started to understand the need for smart bus commute systems. The General Transit Feed Specification (GTFS) defines a common format for public transportation schedules. GTFS "feeds" let transit agencies publish their transit data and developers write applications that use that data in an inter-operable way. The iBart application [8] is one such application that provides scheduled arrivals, service advisories and a trip planner using GTFS. Other such apps exist for transit systems in Boston, Chicago, and London. [9]. Although very useful, only few cities provide a real-time GTFS at present.

In [10], the application of IoT in transportation is highlighted. The usage of GPS and RFID tags are primary mechanisms for location tracking. Authors in [11] have presented a survey on usage of smart phone based sensing for intelligent transportation. Embedded sensors in smart phone such as accelerometer, gyroscope, and global positioning system (GPS) have been applied for obtaining traffic information and vehicle information. The solution in [12] involves using Representational State Transfer (REST) APIs which users can access through an android application, SMS or web-portals. Authors in [13] proposed a flexible system for tracking student buses. It provides real-time information about aspects like location, speed, and number of students. Authors in [14] use the cellular network and GPS for public bus transportation and management system. They successfully implement an integrated system with RFID tags, GPS and GSM for accurate location. In [15], the proposed system aims to provide real-time bus tracking and ETA display at all stops. This allows commuters to have an idea about which bus to expect, when waiting at a stop.

A variety of prediction models have been developed, in literature, for bus arrival times. The most common ones can be classified into: *machine learning*, *historical average*, *regression* including support vector machines (SVMs) and artificial neural networks (ANNs), *Kalman filtering-based*, and *dynamic* moddels. Regression models predict and explain a dependent variable with a mathematical function formed by a set of independent variables [16]. Patnaik et al. proposed a set of multilinear regression models to estimate bus arrival times using the data collected by automatic passenger counter (APC) [17]. Distance, boarding, dwell times, number of stops and alighting passengers and weather descriptors were used as features. Jeong [4] and Ramakrishna et al. [18] also developed multilinear regression models using different sets of inputs. Multilinear regression models are useful in identifying the features important for prediction. Ramakrishna et al found out that bus stop dwell times from the origin of the route to the current bus stop in minutes and intersection delays from the origin of the route to the current bus stop in minutes are less important inputs [18]. Machine learning algorithms can screen a large number of variables to find combinations that can reliably predict results [19]. A prediction model based on SVM was proposed in [20], which used time, weather, road segment, running time of the current road segment and running time of the next link as the input parameters to predict the running time of the bus. An artificial neural network model based on site was used to predict bus arrival times in [16]. A real-time bus arrival prediction system is presented in [21], where a model-based algorithm is developed. It uses real-time data to improve the model with the help of passenger feedback. The Easy Come Easy Go (ECEG) [6] study presents an algorithm that uses real-time GPS data from the field, and takes delays into account for the prediction of bus arrival times. This system relies on powerful smartphones installed on a bus, and is independent of the bus operating companies.

A. Key Contributions

This paper makes the following key contributions:

- An ML-based schedule prediction model that integrates historic patterns and real-time statistics in bus arrival times to estimate bus schedules is proposed.
- Bus arrival times of two bus routes in the Massachusetts Bay Area is collected for a period of four months by regularly querying its real-time GTFS. This data is used to build and test the schedule prediction model.
- An IoT framework that allows live bus tracking with a bus stop level granularity in bus location. The framework overcomes the need to update the database when bus stops are relocated, and the need for powerful edge devices, among other caveats of a GPS-based system.
- An easy-to-deploy set of IoT modules is proposed to facilitate interconnection of buses and bus stops across a city, maintain a central log of bus arrival times, and pave the way to the development of a real-time GTFS for cities.

III. Proposed System

The proposed system models a framework that comprises four broad elements, namely the *IoT system*, *Machine Learning (ML) based schedule prediction*, *the web server*, and *the web interface for commuters*. The IoT network interconnects sensors, controllers and actuators across bus stops and buses through a web server. The ML element performs real-time prediction of buses' segment travel times to estimate ETAs. The web interface allows commuters access to running status

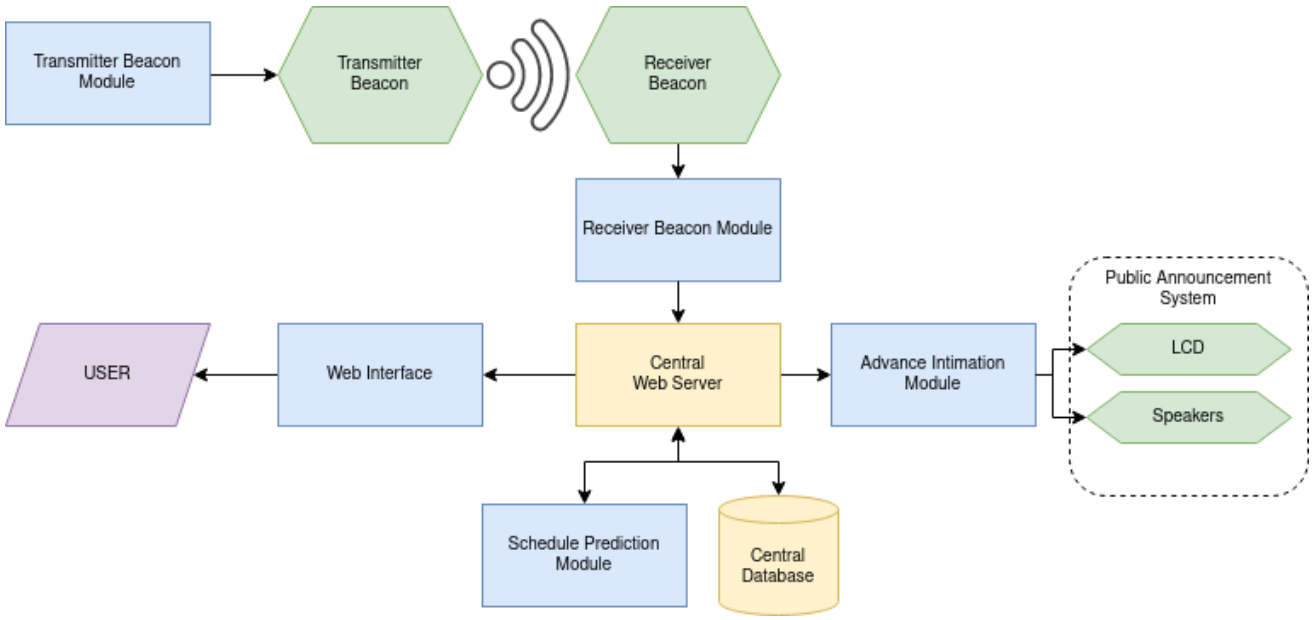

Fig. 1. Overview of the architecture of LocateMyBus system

of buses in transit, and to estimated schedules of buses that will commute in future. The implementation code developed for a prototype of the proposed system will be made available at: https://github.com/karthik-d/LocateMyBus.

A. Overall Architecture

The overall set up of the four elements is depicted in Figure 1. The *web server* forms the central coordinating entity for managing communication across the deployed IoT devices. The *receiver beacons* ping the server to log arrivals. The *transmitter beacons* inform the server about associations between buses and trips. The *advance intimation module* draws imminent arrivals and ETAs for specific bus stops from the server, and the *web application module* provides an interface that presents a real-time user-view to commuters.

B. IoT System

The IoT system equips the bus stops and buses with sensors, actuators and network interfacing hardware necessary to track and log bus schedules. The IoT system is modeled as a set of independent pluggable modules, presented below:

a) Transmitter Beacon Module.: The transmitter module is installed on all buses. The module has two key components, an RF tag (transmitter beacon) with a unique number that identifies the bus, and a trip code entry interface. Before the trip begins, the trip code is registered using the entry interface. The module pings the server to associate this trip code with the bus identifying number. At the end of the trip, the entry interface can be used to de-register the trip code.

b) Receiver Beacon Module.: The receiver beacon module is installed in all bus stops, and features a RF reader and a networking device. It receives the unique bus identifier code stored in the RF card of the arriving bus, and pings a HTTP request to the web server along with the code. The remote server, upon receiving the bus identifier number, recognizes the current trip the bus is servicing and the client stop, and logs the arrival time in the database. It is worth noting, that the edge devices at the bus stops do not invest compute power in decoding the current trip of the arriving bus. This responsibility is left to the remote server.

c) Advance Intimation Module.: This module is also a part of the bus stops' IoT system. It pings the remote server at regular intervals, requesting for updates in transit status of buses scheduled to arrive at the corresponding bus stop. Received transit updates are presented in real-time on Public Announcement Systems (PASs) at the bus stop. Any form of announcement system can be interfaced with this module. The ETA estimation using ML and schedule changes are handled in entirety by the central server.

C. Machine Learning for Schedule Prediction

Most metropolitan transit systems are characterized by several bus routes sharing the same *road segments* and bus stops (*segment* refers to the road between two consecutive bus stops). For instance, Figure 3 contains ten such common segments for SL4 and SL5 between the stops *Nubian SQ* and *Chinatown*. The parameters affecting the segment travel times in such transit systems can be constant like segment length, or be transient in time like traffic conditions on road and queuing up of buses at stops among others. The transient parameters are characterized by inherent patterns that recur over time, as well as by dynamic variations that cause them to deviate from these patterns. The proposed schedule prediction model predicts baseline segment travel times by training an ML regression model that deduces these patterns from historic data of travel times. The baseline prediction is improved in real-time by applying a Kalman filter that corrects the prediction based on most recent travel time observations for the particular segment. The following sections present formal definitions

of the learning techniques, model inputs, rationales and the overall learning framework.

1) Support Vector Regression: Support Vector Machines are capable of mapping the features from the input space to a higher dimensions space to model complex relations between the features. Suppose that the training data comprises x_n as a multivariate set of N observations with corresponding target values y_n. Support Vector Regression (SVR) aims to find a smooth function $f(x_n)$ that deviates from y_n by a value no greater than ϵ (equation 1). This is solved by optimizing the objective function J (equation 2).

$$f(x_n) = \omega\phi(x_n) + b \tag{1}$$

$$J = \frac{1}{2}\|\omega\|^2 + C\sum_{n=1}^{N} L_\epsilon(y_n, f(x_n)) \tag{2}$$

where $\phi(x)$ translates input to high dimensional space, ω is the parameter vector, L_ϵ is the ϵ-insensitive loss function, and C is the box constraint to control the trade-off between the smoothness of $f(x)$ and toleration of errors larger than ϵ.

SVR solves equation 1, subject to minimizing equation 2, for parameters to obtain the target regression line (equation 3). a_n and a_n^* are non-negative Lagrangian multipliers for the data point x_n in the training data.

$$f(x) = \sum_{n=1}^{N}(a_i - a_i^*)\phi(x_i)\phi(x) + b \tag{3}$$

$$f(x) = \sum_{n=1}^{N}(a_i - a_i^*)G(x_i, x_j) + b \tag{4}$$

Non-linearity is introduced into the SVR regression line by incorporating a kernel function. The kernel functions are parameterized by x_i and x_j, as shown in equation 3, to produce equation 4. In this work, the Gaussian and polynomial kernel functions (with degree d) are employed (equations 5 and 6 respectively).

$$G(x_i, x_j) = exp(-\|x_i - x_k\|^2) \tag{5}$$

$$G(x_i, x_j) = (x_i.x_j)^d \tag{6}$$

2) Neural Networks: Neural Networks (NNs) are powerful learning architectures that can model highly complex relationships between their inputs and outputs. The input variables are presented to the NN such that the function signal appearing at the output of neuron j is computed according to equation 7.

$$Y_j = \Psi_j(\sum_{i=1}^{m} w_{ji}X_i + b_j) \tag{7}$$

where m is number of inputs applied to neuron j, $\{X_i\}$ is set of inputs of neuron j, Y_j is output of the jth neuron, w_{ji} is the synaptic weight connecting the ith input to jth neuron, b_j is error term, and $\Psi_j(\cdot)$ is an activation function.

$$\Psi(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

The activation function $\Psi_j(\cdot)$ introduces a non-linear relationship between node inputs and outputs. Sigmoidal functions such as logistic and hyperbolic tangent functions are the most common choices. In this study, the sigmoid activation function (equation 8) is used. The objective is to improve weights w_{ji} to minimize Mean Absolute Error (equation 21).

3) Kalman Filter: The Kalman filter is a recursive estimator. It uses a series of measurements observed over time, incorporates statistical noise, and produces, expectedly more accurate, estimates of the target variables. The Kalman state equation is expressed in equation 9, where x_k denotes the bus travel time at current time step k that needs to be predicted, A_{k-1} represents the state transition parameter relating x_{k-1} to x_k, and w_{k-1} denotes the process noise term that has a normal distribution with zero mean and a variance of Q_{k-1}.

$$x_k = A_{k-1}x_{k-1} + w_{k-1} \tag{9}$$

The state transition parameter A_{k-1} is calculated by data in previous time step. The state x_k and measurement y_k follow equations 10 and 11 respectively. Here, v_k denotes the measurement noise term that has a normal distribution with zero mean and a variance of R_{k-1}. $w_{k-1}$and v_k are assumed to be independent.

$$x_k = x_{k-1} + w_{k-1} \tag{10}$$

$$y_k = x_k + v_k \tag{11}$$

The filtering procedure is detailed below, where $\bar{\hat{x_k}}$ denotes the prior estimate.

a) Step 1 (initialization).: Set $t = 0$. Let $E[x_0] = \hat{x_0}$, $E\{[x_0 - \hat{x_0}]^2\} = P_0$ and $E[w(i).v(j)] = 0$ for all i, j, where $\hat{x_0}$ is the predicted travel time at step 0.

b) Step 2 (extrapolation).: Extrapolate state estimate and error covariance using equations 12 and 13.

$$\bar{\hat{x_k}} = A_{k-1}\hat{x_{k-1}} \tag{12}$$

$$\bar{P_k} = A_{k-1}P_{k-1}A_{k-1}^T + Q_{k-1} \tag{13}$$

c) Step 3 (Kalman gain calculation).: The objective is to find blending factor K_k that minimizes the performance criterion. K_k is expressed in equation 14.

$$K_k = \bar{P_k}(\bar{P_k} + R_k)^{-1} \tag{14}$$

d) Step 4 (update).: Update state estimates and error covariance using equations 15 and 16.

$$\hat{x_k} = \bar{\hat{x_k}} + K_k(Y_k - \bar{\hat{x_k}}) \tag{15}$$

$$P_k = (I - K_k)\bar{P_k} \tag{16}$$

e) Step 5 (next iteration).: Let $k = k + 1$, and repeat from *Step 2* until the circulation is complete.

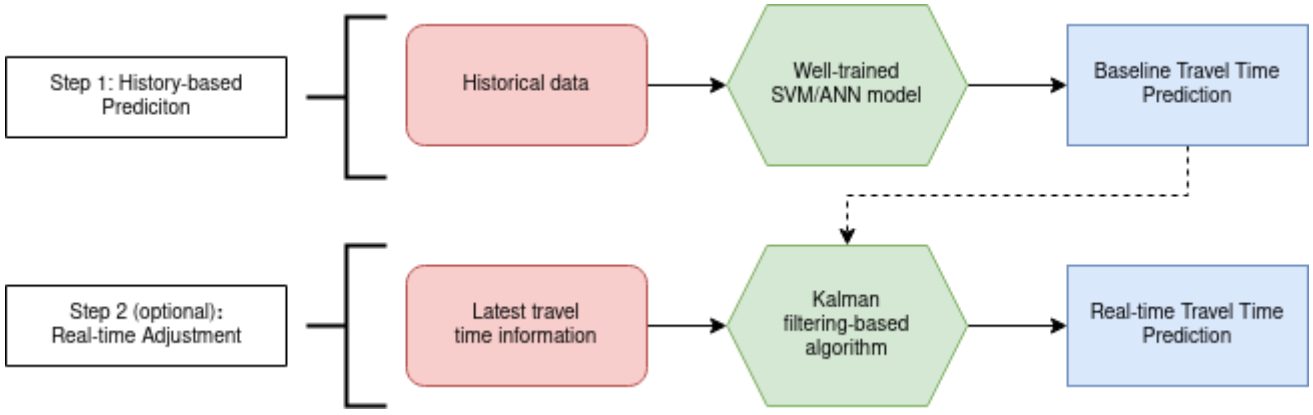

Fig. 2. Proposed schedule prediction framework.

4) Proposed Real-Time Prediction Framework: The proposed real-time schedule prediction framework is depicted in Figure 2. As a *first step*, the baseline predictor $f\phi$, a regression model, that estimates a baseline segment travel time. This is based on recurring patterns inferred by the regression model based on historical data. To incorporate the effects of dynamic factors into the prediction, an optional *second step* applies a Kalman filter f_k that follows the update sequence in section III-C3 to adapt the baseline prediction to real-time, based on the statistics of most recent travel time observations.

Two different regression techniques are experimented with and compared, namely SVR and NN. The input features to the regression models are as follows:

a) Speed of current bus in the previous segment (v_{bus}).: The speed of the bus in the previous segment is computed as the ratio of previous segment distance to the corresponding travel time (equation 17). It is indicative of the bus's condition during this trip, such as malfunctions, crowding, or other trip-specific factors.

$$v_{bus} = \frac{l_{prevseg}}{t_{prevseg}} \tag{17}$$

where $t_{prevseg}$ represents the actual travel time of the bus in the previous segment, and $l_{prevseg}$ represents the length of the previous segment.

b) Time of day (t_d).: The time of day can influence the traffic and hence, the travel time on the road segment. This is expressed as a categorical feature. Time of day is binned into 1-hour intervals by treating the hour value in 24-hr format as its numeric class encoding. For instance, *9.28 PM* is encoded as class *21*.

c) Holiday or Working day. (h): Road traffic patterns are different on business days and holidays. Furthermore, they also correlate with t_d. h is represented as a binary categorical variable, denoted as 0 for holidays, and 1 for working days.

d) Weighted moving average of travel times of previous buses on the same segment. (t_p): The previous buses that traveled along the same road segment are good indicators of the current state of the road condition. The travel time of a preceding bus has greater influence on the current prediction when its time headway is lesser. Hence, a simple weighted method (equation 18) is used to compute t_p.

$$t_p = \sum_{i=1}^{k} \frac{\frac{1}{T_i}}{\sum_{j=1}^{k} \frac{1}{T_j}} t_{seg,i} \tag{18}$$

where $t_{seg,i}$ denotes the travel time of the ith preceding bus on the current segment, T_i represents the time headway between the current bus and the ith preceding bus, and k denotes the number of preceding buses considered to compute t_p. Following successful applications in [6] [22], this work adopts $k = 3$ i.e. three preceding buses are considered.

The regression model is a baseline predictor f_ϕ with parameter ϕ trained using historic data through supervised learning. The aforementioned features serve as inputs, and the segment travel time forms the prediction target, to produce a trained baseline predictor $f\phi$, where ϕ is the learned parameter setting. A baseline estimate of travel time can be made using equation 19. The estimate can be adjusted to real-time by using the Kalman filter as depicted in equation 20.

$$t_{seg,baseline} = f_\phi(v_{bus}, t_d, h, t_p) \tag{19}$$

$$t_{seg,realtime} = f_k(t_{seg,baseline}) \tag{20}$$

5) Deploying the Prediction Module: The schedule prediction module finds its application in two distinct use-cases: (1) *real-time schedule prediction* for buses in transit, and (2) *tentative schedule prediction* for buses that will transit in future. The former applies both steps of the proposed real-time prediction framework to predict segment travel times adjusted to real-time ($t_{seg,realtime}$), while the latter predicts $t_{seg,baseline}$ by using only the baseline predictor since real-time data is not available for buses that will commute in future.

The baseline predictor f_ϕ trained on historic data is deployed as a model specification file, along with its weight parameters on the web server. The Kalman filter f_k is stored on the server with its most recent state for each segment. Each time a prediction use-case is queried, the associated route and segment information is drawn from the database, relevant features are computed and fed to the model, and the predicted segment travel time is returned as response.

D. Central Web Server

A central web server is deployed to interconnect the deployed IoT modules. The communication happens over the internet using RESTful API. There are three key API endpoints that help maintain the state of the database in real-time:

a) Register or de-register trip code for a bus.: Before starting a trip, the bus's IoT module is used to enter the trip code that the bus will service. When the module pings this endpoint with the trip code, the web server associates it with the bus.

b) Arrival time logging.: When a bus arrives at a stop, the receiver module reads the bus's RF tag, and pings this API endpoint. In response, the web server records the arrival time of the bus at this stop for the current trip.

c) Avail imminent arrival times.: The advance intimation modules at the bus stops ping this endpoint at regular intervals to manage their PASs. The web server responds with a list of all buses that will next arrival at this stop.

E. Web Interface for Commuters

The web interface is essentially a user presentation of the web server. The central web server already stores real-time updates of bus arrivals based on update pings from the deployed IoT modules. The web interface, simply accesses the database to fetch this data and other route information, and computes and responds to user queries. The web interface facilitates two broad user queries: (1) *track running status*, and (2) *avail expected schedule* of buses. The running status can be accessed for buses in transit. The actual arrival time of the bus is displayed for stops that the bus has already crossed. For stops that it is yet to reach, the Estimated Time of Arrival (ETA) is predicted using the real-time schedule prediction module (both prediction steps, baseline f_ϕ and Kalman adjustment f_k, are applied). Likewise, the expected schedule of a bus trip can be availed for buses that have not departed from their source stop yet. The expected arrival times of the bus at all stops along its route is estimated using the schedule prediction module, applying only the baseline predictor f_ϕ, since real-time arrival data in not available for future transits.

IV. Results and Discussion

A. Schedule Prediction

The schedule prediction model estimates segment-wise travel times. It was trained and tested on bus arrival times in the Massachusetts Bay Area. Specific road segments were chosen to analyze the models' performance.

1) Data Collection and Cleaning: The Massachusetts Bay Transportation Authority (MBTA) provides a GTFS-realtime feed. The GTFS-realtime feed features an API with three kinds of updates, namely *trip updates*, *service alerts* and *vehicle positions*. For data collection, the MBTA Silver Line bus routes *SL4* and *SL5*, both inbound and outbound, were chosen. The routes have a good mix of common and independent road segments (refer to Figure 3). The *trip updates* API was queried regularly for four months (October 2020 to January 2021) to record bus arrival times for all trips on these routes. After cursory data cleaning, dropping corrupt and invalid values, about 550,000 arrival times remained, resulting from about 46,000 trips. The arrival times data is stored as CSV files. Of the wide-range of attributes available in the logged data, only attributes relevant for model training are retained. A snapshot of these attributes is presented in Figure 7 for a small portion of the collected data. The collected data is split into training and testing sets, where data from the first three months constitute the training set, and that from the fourth month constitutes the test set. Adopting a chronological split is representative of the real-world scenario for travel time prediction, where historic data is used to make predictions about future trends.

2) Model Training and Performance: The baseline estimator f_ϕ was trained using the training set and validated on the testing set. To ensure that features have similar ranges and allow faster convergence, all numerical features were normalized between 0 and 1 using min-max normalization. Two regression learning techniques were experimented with for the baseline predictor, SVR and NN.

a) Support Vector Regression.: Two SVR regression models were trained on historic data, one using the Gaussian kernel (SVR-Gauss), and the other using a polynomial kernel of degree two (SVR-Poly). The trained parameters were obtained, following the SVR training methods detailed in section III-C1. The model was trained using stratified 4-fold cross validation, and the performance measures were ascertained as an average over the four runs.

b) Neural Network: A single hidden-layer neural network architecture was used for the baseline predictor. It comprised of 4 neurons in the input layer corresponding to the 4 features considered, 8 neurons in the hidden layer, and 1 neuron in the output layer. The number of neurons in the hidden layer was determined after experimenting with a short range of values between 4 and 10. The sigmoid activation function was used between the input and hidden layers. The network was trained using the backpropagation algorithm at a learning rate of $3e-03$.

The error in predicted segment travel times was estimated in terms of (1) Mean Absolute Error (MAE), and (2) Mean Absolute Percentage Error (MAPE) in the forms presented in equations 21 and 22.

$$MAE = \frac{\sum |t_{observed} - t_{predicted}|}{N} \tag{21}$$

$$MAPE = \frac{1}{N} \sum \frac{|t_{observed} - t_{predicted}|}{t_{observed}} \tag{22}$$

To ascertain the real-time schedule prediction performance, three road segments were considered: Segment 1 between *Tufts Med Center* and *Herald St.*, Segment 2 between *South Station* and *Chinatown*, and Segment 3 between *Boylston* and *Tufts Med Center*. These road segmented are summarised in Table I. The segments were chosen for their significant differences in road types. For instance, segment 1 crosses an interstate, while segment 3 has two turns. Such features can significantly impact the bus travel time in the segment.

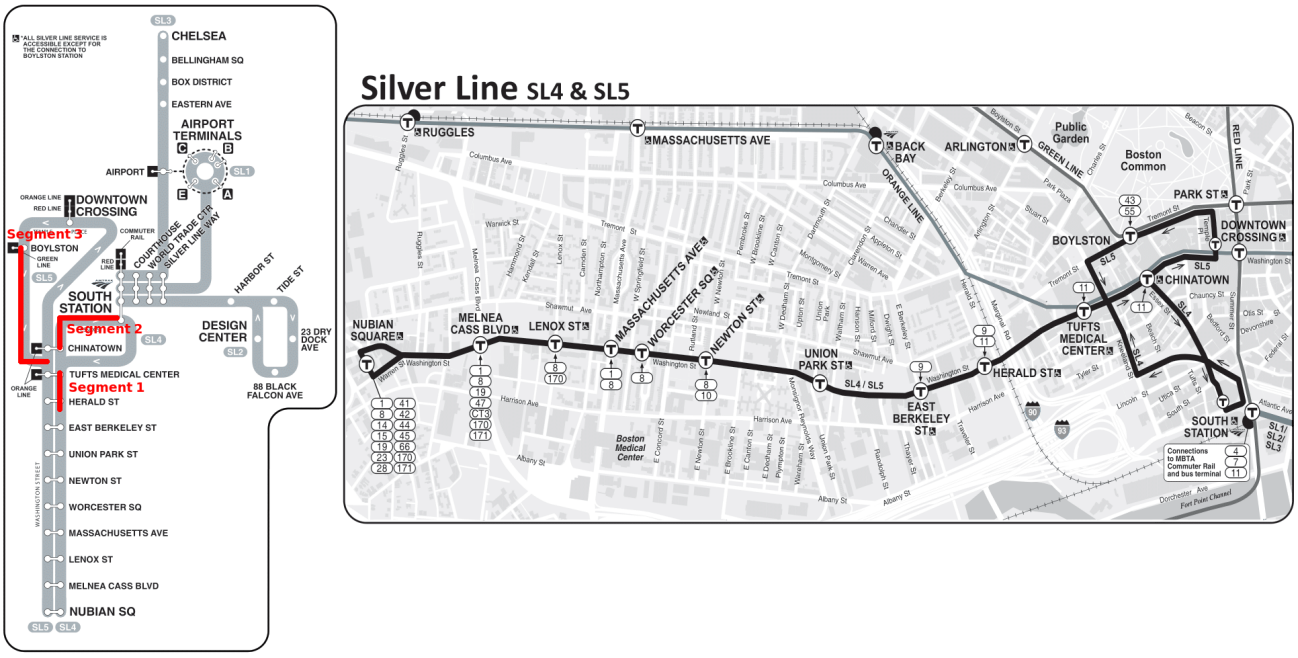

Fig. 3. Maps of MBTA's SL4 and SL5 bus routes. The highlighted portions represent the segments chosen for analyzing the prediction models.

ServiceDate	Route	HalfTripId	Stop	TimepointOrder	Scheduled	Actual
2020-10-01 0:00:00	1	48885644	110	1	1900-01-01 5:05:00	1900-01-01 5:05:24
2020-10-01 0:00:00	1	48885644	67	2	1900-01-01 5:07:00	1900-01-01 5:07:42
2020-10-01 0:00:00	1	48885644	72	3	1900-01-01 5:10:00	1900-01-01 5:10:13
2020-10-01 0:00:00	1	48885644	75	4	1900-01-01 5:13:00	1900-01-01 5:12:37
2020-12-31 0:00:00	SL5	50031615	60	10	1900-01-02 0:10:00	1900-01-02 0:12:17
2020-12-31 0:00:00	SL5	50031615	61	11	1900-01-02 0:11:00	1900-01-02 0:12:42
2020-12-31 0:00:00	SL5	50031615	64	12	1900-01-02 0:12:00	1900-01-02 0:13:15
2020-12-31 0:00:00	SL5	50031608	49001	1	1900-01-02 0:13:00	1900-01-02 0:35:01
2020-12-31 0:00:00	SL5	50031608	8279	2	1900-01-02 0:15:00	1900-01-02 0:35:48
2020-12-31 0:00:00	SL5	50031608	49002	3	1900-01-02 0:18:00	1900-01-02 0:37:43
2020-12-31 0:00:00	SL5	50031608	49003	4	1900-01-02 0:19:00	1900-01-02 0:39:48
2020-12-31 0:00:00	SL5	50031608	5098	5	1900-01-02 0:20:00	1900-01-02 0:40:32
2020-12-31 0:00:00	SL5	50031608	5100	6	1900-01-02 0:22:00	1900-01-02 0:41:59
2020-12-31 0:00:00	SL5	50031608	19402	7	1900-01-02 0:23:00	1900-01-02 0:42:41
2020-12-31 0:00:00	SL5	50031608	15176	8	1900-01-02 0:24:00	1900-01-02 0:43:00

Fig. 4. Snapshot of selected attributes for a portion of the bus arrival times data.

TABLE I
ROUTE SEGMENTS CONSIDERED FOR PREDICTION PERFORMANCE EVALUATION. DISTANCES ARE EXPRESSED IN *meters*, APPROXIMATELY.

Segment No.	Starting Stop	Ending Stop	Distance
Segment 1	Tufts Med. Center	Herald St.	565
Segment 2	South Station	Chinatown	644
Segment 3	Boylston	Tufts Med. Center	483

The performance of the three models evaluated on the three segments, with and without the Kalman filter for real-time adaptation, is presented in Table II. It is evident that the Neural Network model consistently outperforms both the SVR variants, although the SVR with a Gaussian kernel performs quite well. Furthermore, the application of the Kalman filter leads to significant performance gains, exhibiting very low errors values in comparison to their non-Kalman counterparts.

TABLE II
PERFORMANCE OF PREDICTION MODELS. KALMAN FILTER VARIANTS ARE SUFFIXED WITH *-Kmn* REPRESENT. *MAE* AND *RMSE* ARE EXPRESSED IN *seconds*. *MAPE* IS IN %.

Model	Segment 1		Segment 2		Segment 3	
	MAE	MAPE	MAE	MAPE	MAE	MAPE
SVR-Poly	29.2	14.26	31.3	15.22	32.6	16.24
SVR-Gauss	23.4	11.10	24.4	11.84	26.4	13.22
NN	22.5	9.76	24.2	9.81	25.8	10.01
SVR-Poly-Kmn	18.3	5.56	17.4	5.96	21.3	6.44
SVR-Gauss-Kmn	11.4	5.42	13.2	5.48	15.3	6.02
NN-Kmn	10.2	3.86	12.6	4.12	14.4	4.59

The MAPE values of the best prediction models namely, *NN* and *NN-Kalman*, are compared with other similar work

in literature, that attempt to predict segment-wise bus travel times [22] [23] [24]. This comparison is presented in Table III. MAPE values are evaluated as a normalization over actual travel times. Hence, although predictor performance is contingent on a range of factors pertaining to the road segments and the city or town considered: the quality of roads; length of the segments; nature of bus travel; and traffic conditions, among others; MAPE is a reasonable metric to perform a primitive, sparing and approximate comparison of the prediction models.

TABLE III
COMPARISON OF PERFORMANCE OF THE PROPOSED BEST PREDICTION MODEL, WITH OTHER WORK BASED ON SEGMENT TRAVEL TIME PREDICTION. KALMAN FILTER VARIANTS ARE SUFFIXED WITH *-Kmn* REPRESENT. THE *MAPE* METRIC IS USED FOR COMPARISON, AND IS EXPRESSED IN %.

Model	MAPE(s)	Avg. MAPE
Historical Avg. [24]	19, 17, 11	15.67
Kalman [24]	16, 8, 8	10.67
ANN [24]	12, 9, 8	9.67
SVM [23]	11.65, 9.75, 9.50, 8.87	9.94
ANN [23]	9.92, 8.55, 10.42, 8.19	9.27
SVM [22]	10.24, 16.73, 11.91	12.96
ANN [22]	10.52, 17.60, 12.70	13.61
NN Ours	9.76, 9.81, 10.01	9.86
SVM-Kalman [22]	4.33, 6.82, 4.46	5.20
ANN-Kalman [22]	4.34, 6.96, 5.10	5.47
NN-Kalman Ours	3.86, 4.12, 4.59	4.19

The authors of [22] follow a very similar approach to the proposed: uses a Kalman filter approach to improve baseline prediction, and considers three different road segments for evaluating the prediction model. The table lists the MAPE values on each of these segments, and also presents the average of these values. The proposed method outperforms both, its baseline and its Kalman variants in terms of the average MAPE values. In [23], however, only one segment is considered. The MAPE values are compared for four different configurations of times of the day and location within the city (referred to as *groups*). The table lists the four MAPE values and averages them. The proposed baseline model performs at par with theirs while, the proposed Kalman variant significantly outperforms their models. The authors of [24] evaluate their prediction models on three different segments. However, the segments here include other intermediate stops, and are not strictly between consecutive stops. Three different overlapping segments are considered, chosen such that each is longer than the previous. Since it presents the MAPE values only graphically, Table III rounds the MAPE values to the nearest integer value when listing them, and averages the performance on the three segments. Here too, the proposed baseline performs at par with their predictors, and the Kalman variant outperforms all of them.

It can be observed that errors in travel time predictions of the proposed system are the highest for *segment 3*, followed by *segments 2 and 1*. This relative ordering is consistent across all models. The two traffic junctions in *segment 3* likely explain the high rate of errors, since patterns of traffic at junctions are intuitively more complex than at plain roads; and therefore, harder for an ML model to capture. On the contrary, although *segment 1* is characterized by an interstate crossing, it engenders lesser prediction error than *segment 2*. This can be explained by the significantly higher distance covered by buses in *segment 2*, possibly overshadowing the effect of busy and unpredictable traffic at the intersection in *segment 1*. These observations suggest that road segment length is an importance factor. Concomitantly, however, it is not the only factor that induces complexity into the problem of travel time prediction. Rather, other aspects of the road segment also factor in, and considering features based on speed and activity of previous buses is vital; supporting the choice of features chosen in this study. Furthermore, this work restricts segments to only sections of roads between consecutive stops, in contrast to studies such as [24] [23], where segments comprising intermediate stops are allowed. In such models, since not all of the aforementioned segment-specific factors can be concretely laid down, and presented as input to the model like segment length, over-generalization of traffic patterns across road segments is likely; consequently worsening the prediction error.

B. Web Server, Interface and IoT Prototype

A miniature prototype of the LocateMyBus system was built and simulated using IoT modules for two buses, and four bus stops namely, *Majestic*, *Halasuru*, *Indiranagar* and *CV Raman Nagar*, a simplified version of a bus route in Bangalore, KA, India. Each bus is equipped with a transmitter module comprising an RF tag, and a trip code entry interface through a laptop. The bus stops have the receiver module with an RFID reader, and an advance intimation module that displays bus arrival information. All IoT devices are networked using LAN to communicate with a central server, that houses the the prediction module. The IoT systems were simulated on the Proteus software, and prototyped with Arduino and Raspberry Pi boards.

Before the bus departs from source, the trip code is entered, and the server is pinged to associate the bus with the trip. At regular intervals, the bus stops' intimation modules query the server to get expected arrivals, and display them on their PAS. When a bus arrives at a stop, the receiver module uses the RF reader to detect the bus, and pings the server to log the arrival event. Throughout, the web interface displays the running status of the bus by directly querying the server.

V. CONCLUSION AND FUTURE WORK

This paper proposes the LocateMyBus system that leverages Internet of Things (IoT) and Machine Learning (ML) to ease the uncertainty associated with bus commute in cities. It enables commuters to track live running status, and avail estimated schedule of buses through a web interface, as well as through PA systems at bus stops. The system employs a Neural Network and Kalman filter based prediction model to estimate segment-wise travel times of buses based on historic and real-time data of bus arrivals and route information. Further, the system proposes a set of IoT modules comprising low compute power edge devices to log bus arrival times, paving the way to building real-time General Transit Feed Systems (GTFSs).

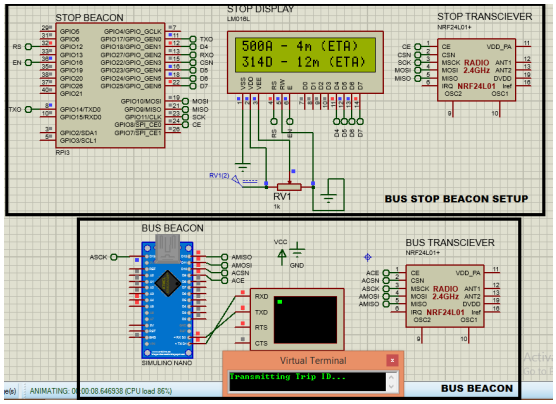

Fig. 5. IoT system simulation on the Proteus software

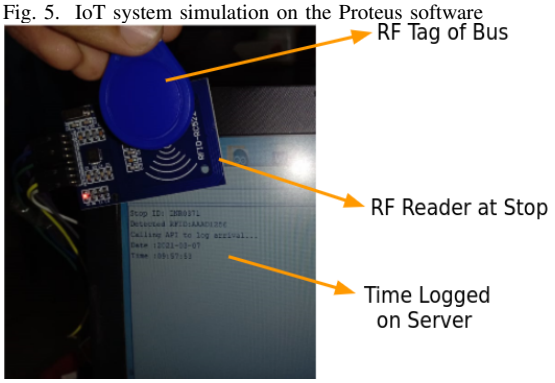

Fig. 6. IoT system prototype built using Arduino and Raspberry Pi boards

The proposed system can be extended as a distributed network in future to handle large volume of requests from buses, stops and commuters across a city. Other future directions can aim to improve the schedule prediction model by incorporating logged arrival data, to update the learned by the model, through incremental learning.

ACKNOWLEDGMENTS

The authors acknowledge and thank the support of the Research Center at Sri Sivasubramaniya Nadar College of Engineering, Chennai, India in funding this research work.

REFERENCES

[1] J. Houghton, J. Reiners, and C. Lim, "Intelligent transport: How cities can improve mobility," *IBM Institute for Business Value*, pp. 1–6, 2009.

[2] K. K. Dewan and I. Ahmad, "Carpooling: a step to reduce congestion," *Engineering Letters*, vol. 14, no. 1, pp. 61–66, 2007.

[3] V. Kumar, B. A. Kumar, L. D. Vanajakshi, and S. C. Subramanian, "Comparison of model based and machine learning approaches for bus arrival time prediction," Tech. Rep., 2014.

[4] R. Jeong and R. Rilett, "Bus arrival time prediction using artificial neural network model," in *Proceedings. The 7th international IEEE conference on intelligent transportation systems (IEEE Cat. No. 04TH8749)*. IEEE, 2004, pp. 988–993.

[5] L. Vanajakshi, S. C. Subramanian, and R. Sivanandan, "Travel time prediction under heterogeneous traffic conditions using global positioning system data from buses," *IET intelligent transport systems*, vol. 3, no. 1, pp. 1–9, 2009.

[6] B. Yu, W. H. Lam, and M. L. Tam, "Bus arrival time prediction at bus stop with multiple routes," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 6, pp. 1157–1170, 2011.

[7] C. Ma, Y. Wang, and K. Chen, "Competition model between urban rail and bus transit," *Journal of Transportation Systems Engineering and Information Technology*, vol. 7, no. 3, pp. 140–143, 2007.

[8] T. Szincsák and A. Vágner, "Public transit schedule and route planner application for mobile devices," in *Proceedings of the 9th International Conference on Applied Informatics*, vol. 2, 2014, pp. 153–161.

[9] W. Roush, "Welcome to google transit: How (and why) the search giant is remapping public transportation," *Community Transportation*, 2012.

[10] B. Rashid and M. H. Rehmani, "Applications of wireless sensor networks for urban areas: A survey," *Journal of network and computer applications*, vol. 60, pp. 192–219, 2016.

[11] J. Engelbrecht, M. J. Booysen, G.-J. van Rooyen, and F. J. Bruwer, "Survey of smartphone-based sensing in vehicles for intelligent transportation system applications," *IET Intelligent Transport Systems*, vol. 9, no. 10, pp. 924–935, 2015.

[12] J. Lohokare, R. Dani, S. Sontakke, and R. Adhao, "Scalable tracking system for public buses using iot technologies," in *2017 International Conference on Emerging Trends & Innovation in ICT (ICEI)*. IEEE, 2017, pp. 104–109.

[13] J. T. Raj and J. Sankar, "Iot based smart school bus monitoring and notification system," in *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*. IEEE, 2017, pp. 89–92.

[14] K. G. Subadra, J. M. Begum, and H. Dhivya, "Analysis of an automated bus tracking system for metropolitan using iot," in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. IEEE, 2017, pp. 1–5.

[15] V. M. Anu, D. Sarikha, G. S. Keerthy, and J. Jabez, "An rfid based system for bus location tracking and display," in *International Confernce on Innovation Information in Computing Technologies*. IEEE, 2015, pp. 1–6.

[16] S. I.-J. Chien, Y. Ding, and C. Wei, "Dynamic bus arrival time prediction with artificial neural networks," *Journal of transportation engineering*, vol. 128, no. 5, pp. 429–438, 2002.

[17] J. Patnaik, S. Chien, and A. Bladikas, "Estimation of bus arrival times using apc data," *Journal of public transportation*, vol. 7, no. 1, p. 1, 2004.

[18] Y. Ramakrishna, P. Ramakrishna, V. Lakshmanan, and R. Sivanandan, "Bus travel time prediction using gps data," *Proceedings Map India*, 2006.

[19] Z. Obermeyer and E. J. Emanuel, "Predicting the future—big data, machine learning, and clinical medicine," *The New England journal of medicine*, vol. 375, no. 13, p. 1216, 2016.

[20] Y. Bin, Y. Zhongzhen, and Y. Baozhen, "Bus arrival time prediction using support vector machines," *Journal of Intelligent Transportation Systems*, vol. 10, no. 4, pp. 151–158, 2006.

[21] R. Padmanaban, K. Divakar, L. Vanajakshi, and S. C. Subramanian, "Development of a real-time bus arrival prediction system for indian traffic conditions," *IET Intelligent Transport Systems*, vol. 4, no. 3, pp. 189–200, 2010.

[22] C. Bai, Z.-R. Peng, Q.-C. Lu, and J. Sun, "Dynamic bus travel time prediction models on road with multiple bus routes," *Computational intelligence and neuroscience*, vol. 2015, 2015.

[23] T. Yin, G. Zhong, J. Zhang, S. He, and B. Ran, "A prediction model of bus arrival time at stops with multi-routes," *Transportation research procedia*, vol. 25, pp. 4623–4636, 2017.

[24] W. Fan and Z. Gurmu, "Dynamic travel time prediction models for buses using only gps data," *International Journal of Transportation Science and Technology*, vol. 4, no. 4, pp. 353–366, 2015.

Karthik Desingu Karthik Desingu is an undergraduate student at the Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai. His areas of research interest include Machine Learning, Computational Biology and Computer Vision.

Daniel Mark Isaac Daniel Mark Isaac is an undergraduate student at the Department of Electronics and Communication Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai. His research interests include Machine Learning and Internet of Things.

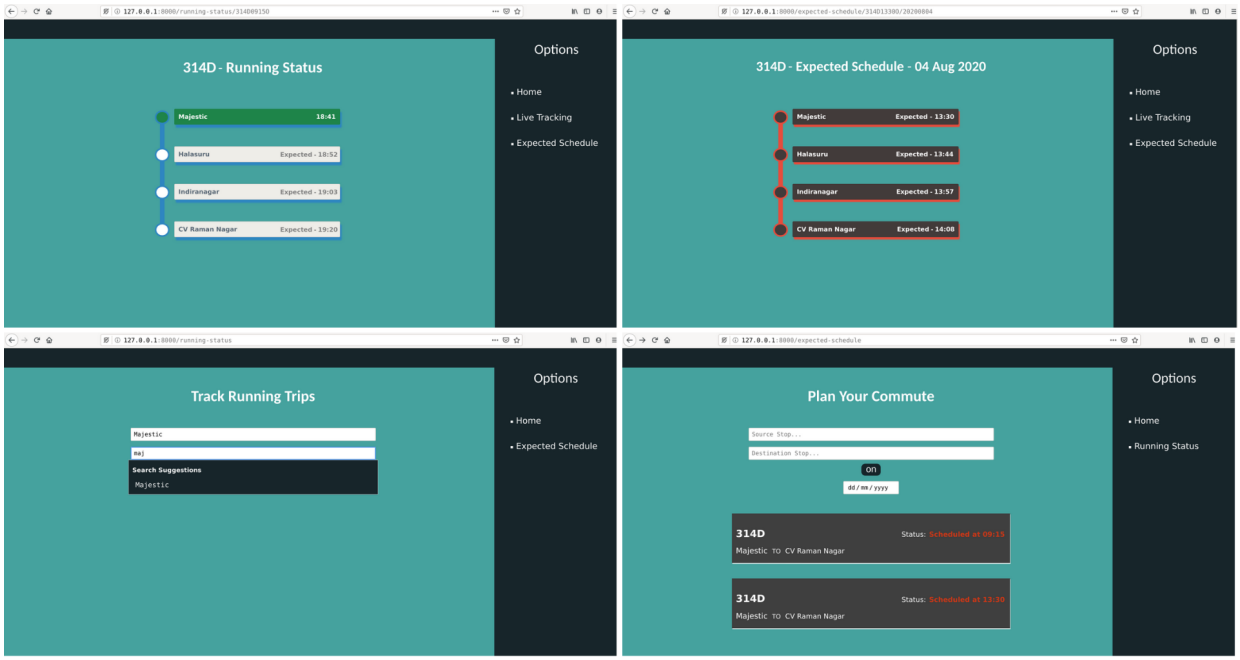

Fig. 7. Screenshots of the commuters' web interface.

Mirunalini P. P.Mirunalini working as Associate professor at department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai. Her primary research includes medical image processing and analysis, pattern recognition, Computer Vision, Machine learning and Deep learning. She has published papers in international journal and conferences.

Bharathi B. B. Bharathi, Associate Professor, Department of CSE, Sri Sivasubramaniya Nadar College of Engineering, Chennai, India. Her research interests are Speech and Natural language processing. She is working in the areas of isolated and continuous speech recognition, multilingual speech recognition, speech emotion recognition, natural language processing, speech synthesis, speaker verification / identification, machine Translation and anti-spoofing methods for automatic speaker verification. She also worked in native language identification, code mixed data, sentiment analysis, offensive language identification, and fake news detection for low resource languages, Internet of things. She has published around 90 papers in international journals and conferences.She has organized international conferences and workshops.

Cherry Mathew Philipose Cherry Mathew Philipose is an Assistant Professor of English at Shiv Nadar University, Chennai. He has his PhD in English (English Language Education – ELE/ELT) from the renowned English and Foreign Languages University (EFL-U, formerly CIEFL), Hyderabad. Cherry has four publications to his credit, and has presented research papers on English language teaching (ELT) at several international and national conferences. He has also conducted several teacher-training workshops in educational technology and in ELT for practising teachers in several places. His areas of academic interest are teacher education, continuing professional development, educational technology, information and communication technology (ICT), heutagogy, and teaching listening.