Paper ID #20975

Capstone Project: A Cloud-based Backend Server for an Automated Bicycle Rental System (6 page paper)

Ms. Thuong N Nguyen, York College of Pennsylvania

Thuong grew up in Vietnam, came to the USA as an international student. She received a B.S. in Computer Engineering from York College of Pennsylvania in 2016. She received an A.S. in Engineering and an AAS in Computer Aided Draft and Design from Harford Community College in 2012. Her interests include mobile applications, web development, internet of thing (IOT) and mobile wearable technologies.

Mr. Justin Reichner, York College of Pennsylvania

Grew up in the small town of Shamokin, Pennsylvania. Studied engineering at York College of Pennsylvania and graduated in December 2016 with a Bachelor's of Science degree in computer engineering. Member of Alpha Chi National Honor society. Enjoys working with web based applications, and just coding in general.

Dr. James Moscola, York College of Pennsylvania

James Moscola is an Associate Professor of Computer Science and Computer Engineering at York College of Pennsylvania. He received a B.S. in Physical Science from Muhlenberg College in 2000, a B.S. in Computer Engineering, a M.S. in Computer Science, and

Dr. Kala Meah, York College of Pennsylvania

Kala Meah received the B.Sc. degree from Bangladesh University of Engineering and Technology in 1998, the M.Sc. degree from South Dakota State University in 2003, and the Ph.D. degree from the University of Wyoming in 2007, all in Electrical Engineering.

Capstone Project: A Cloud-based Backend Server for an Automated Bicycle Rental System

Thuong Nguyen, Justin Reichner, James Moscola, and Kala Meah

Department of Engineering and Computer Science,
York College of Pennsylvania, York, PA

Abstract

Automated bicycle rental systems have become an increasingly popular form of public transportation in cities and on campuses. As a senior capstone project at York College of Pennsylvania, 19 engineering students across three disciplines; computer engineering, electrical engineering, and mechanical engineering; designed and built a working Automated Bicycle Rental System (ABRS) for use by the college community. The ABRS project is designed to support multiple rental kiosks, and a multitude of bicycles, each with an embedded GPS tracking device. Each rental kiosks and GPS tracking device communicates with a common backend server to provide status updates on system health, bicycle availability, as well as rental transaction information. The backend server maintains this information and provides both an end user and administrative user interface for interacting with the system. This paper presents the design and implementation of the cloud-based backend server for the ABRS project.

Keywords

automated bicycle rental, backend server, GPS tracking, web server

Introduction

Bicycle rental systems provide an environmentally friendly method of transportation for modern society. Bicycle rental/sharing has grown significantly in Europe, North America, South America, Asia, and Australia since the 1960s. Bicycle rental systems provide a viable solution to ease congestion and transportation problems in highly populated areas. Riders can rent bicycles from an electronically controlled rental system using smart electronic devices, such as mobile phones, or magstripe cards. A rider can ride the bicycle to their desired destination and return the bicycle to a different rental station or return the bicycle to the original rental station. Bicycle rental systems can be located at mass transit hubs such as bus and train stations, near a busy city center, near a multi-housing complex, on school/college campuses, in shopping districts, near tourist attractions, and other areas with high volume foot traffic[1].

The goal of the Automated Bicycle Rental System (ABRS) project was to design and build a fully automated bicycle rental system that the community at York College of Pennsylvania could use free of charge[10]. The community includes students, faculty, and staff with a York College email addresses. The ABRS system is fully automated and requires no attendant to service customers. Customers can rent and return bicycles completely electronically. The system is comprised of four major components; bicycles, locking positions that keep the bicycles secure when they are not in use, rental kiosks that interface with users and control the locks, and a backend server to store system information and authorize rentals. To rent a bicycle, a user needs to register on the backend server via a website, generate a transaction code, and input that code at a rental kiosk. The code is then passed from the rental kiosk to the backend server so the rental request can be validated. Once validated, a bicycle is unlocked for the user.

The system utilizes a backend server to store customer information, keep a record of all rental transactions, and track system information such as the number of bicycles that are available to rent, the power levels of the rental kiosks, and any issues or malfunctions that have occurred. This information is transmitted from each rental kiosk to the backend server via a cellular network or WiFi connection. The backend server also tracks the locations of all ABRS bicycles by receiving and storing GPS coordinates transmitted by GPS tracking devices located on each bicycle[2]. Figure 1 shows the main page of the ABRS website. From this page, a user can view kiosk locations, the number of bicycles available at each of those kiosk locations, and the number of open locks available for bicycle returns.

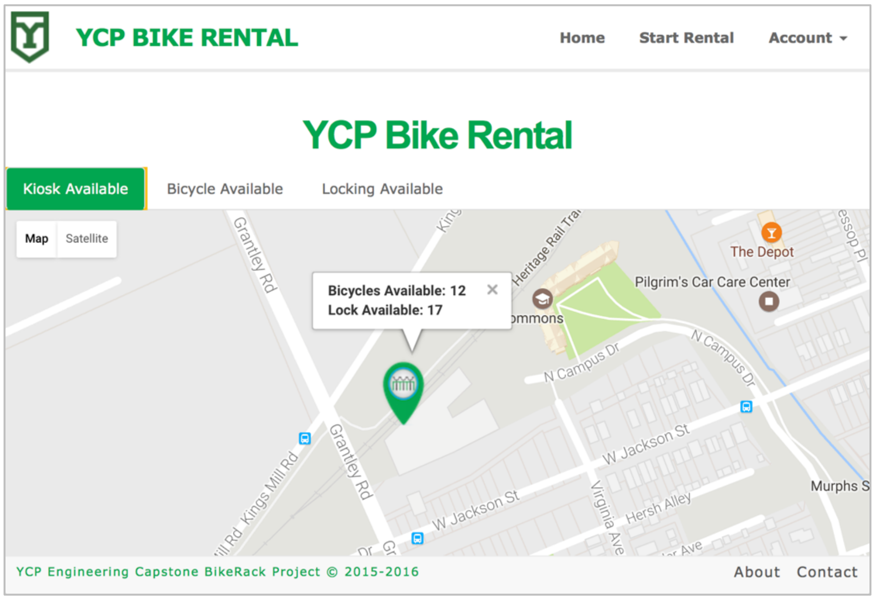


Figure 1: Main page of the ABRS website

Design Summary

The backend server for the ABRS project uses a cloud-based solution. The benefits of this approach include no initial hardware setup, high reliability, and reduced cost. Heroku[3], a cloud-based Platform-as-a-Service (PaaS) provider, was selected for the ABRS project. Heroku provides several advantages over other providers such as Amazon Web Server (AWS). Heroku is easy to use and provides a streamlined mechanism for deployment. Heroku also offers a free tier that is well-suited for this project. One limitation of Heroku's free tier is that the ABRS web application can only run 18 out of every 24 hours. However, this does not impact the ABRS system since rentals are only permitted during daylight hours. The backend server was built with Ruby on Rails and PostgreSQL[4,5]. Additionally, the ABRS web application utilizes Bootstrap[6] to ensure the website user interface scales for various screen sizes from desktop computers to mobile phones.

Implementation of the Backend

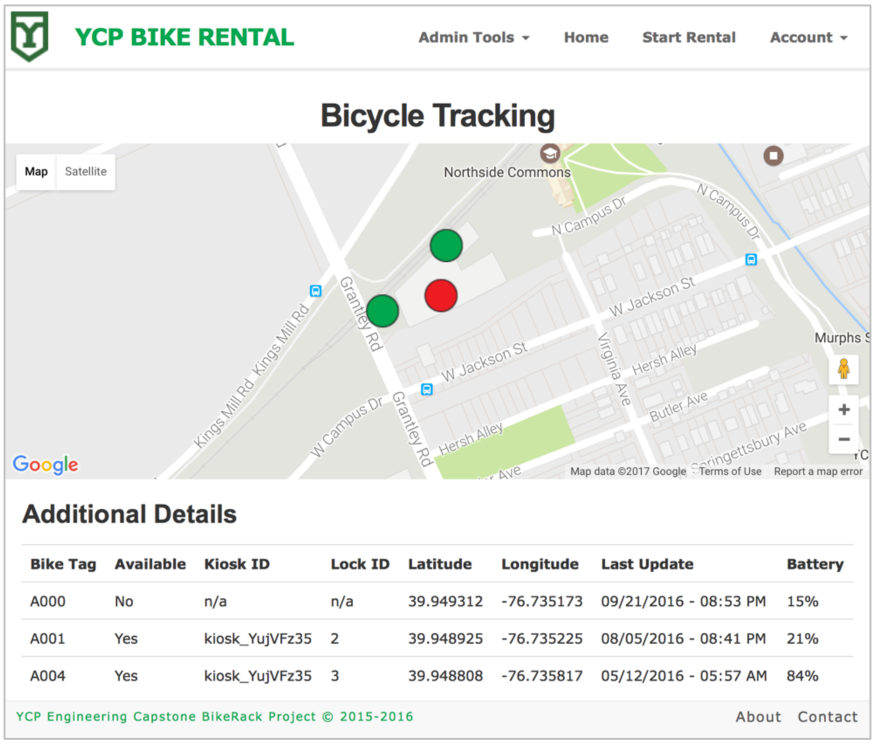
The Rails framework utilizes a Model, View, Controller (MVC) architecture for organizing web applications application. For the ABRS project, the models represent database objects such as users, kiosks, and bicycles. Controllers handle computations and database accesses, and views are used to render user interfaces on the website. Rails routes incoming HTTP requests to the appropriate controller methods using paths specified in the request. Typically, HTTP requests are generated by web browsers. However, the ABRS kiosks and bicycle GPS trackers also generate HTTP request messages to communicate with the backend server.

The controllers handle requests for different data formats such as HTML or JavaScript Object Notation (JSON)[7]. The data format used for communication depends on the source of the request; web browsers interact with the backend via HTML; kiosks and GPS trackers interact with the backend via JSON. The Rails controllers create new model instances, or update attributes of existing instances using Rails' Active Record. In Rails, each Active Record object represent an instance of a model object.

Administrator Role – Administrators of the bicycle rental system can monitor and control the system through a secure administrative web interface. An administrator can view which bicycles are rented, access the GPS coordinates of rented bicycles, and see the list of users who are currently renting bicycles. The administrator can also add or remove kiosks, bicycles, and locks from the backend database using the web interface.

An admin can track the location of bicycles using the web interface as shown in Figure 2. The bicycle tracking page includes a Google map with markers indicating the geographic locations of each bicycle. The map was generated using the 'gmaps4rails' Ruby gem[8]. This gem includes many JavaScript functions for interacting with the Google map that can be integrated directly into the Ruby code. Only bicycles that are currently rented are displayed on the map. Bicycles that are locked at a rental station have known locations and are not displayed. Clicking on a bicycle marker reveals additional information about the bicycle such as the bicycle's identifier and the battery level of the GPS tracker. The markers on the map update dynamically so that a full page refresh is not necessary to update the markers. This dynamic updating was achieved through the use of Asynchronous JavaScript and XML (AJAX) requests. With AJAX the view sends an HTTP request to the server and a controller returns the latest marker data. The view then uses this new marker data to dynamically update the page.

Additional web pages were created to give an administrator the ability to monitor several other aspects of the system. One such page provides a list of all completed or ongoing rental transactions. Another page displays a paginated list of registered users. Yet another page displays the health of the kiosks, bicycles, and locks in the system. The health page displays mechanical issues that require maintenance such as a flat tire on a bicycle or a broken keypad on a kiosk.



Bicycle Tracking

Additional Details

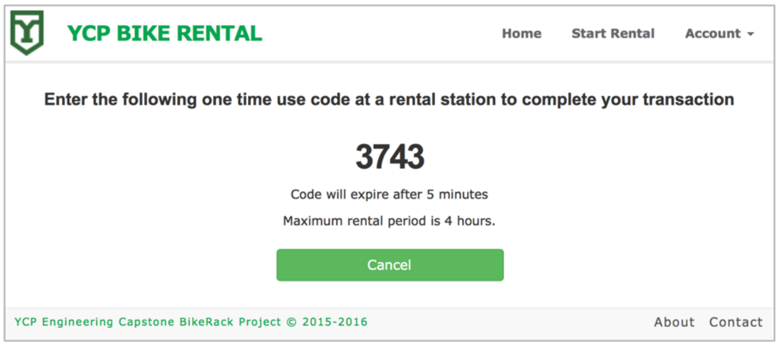
Bike Tag	Available	Kiosk ID	Lock ID	Latitude	Longitude	Last Update	Battery
A000	No	n/a	n/a	39.949312	-76.735173	09/21/2016 - 08:53 PM	15%
A001	Yes	kiosk_YujVFz35	2	39.948925	-76.735225	08/05/2016 - 08:41 PM	21%
A004	Yes	kiosk_YujVFz35	3	39.948808	-76.735817	05/12/2016 - 05:57 AM	84%

YCP Engineering Capstone BikeRack Project © 2015-2016 About Contact

Figure 2: Administrator interface, bicycle tracking

User Role – Before a user can rent a bicycle, they must register via a signup form on the ABRS web site. The signup form contains fields for the user's name, email, and password. There are some client-side validations on the form inputs that are performed with JavaScript. The email must have a 'ycp.edu' domain, and the password must be at least 8 characters long. If these validation checks fail, the form is not submitted and the user is notified of any issues. Upon a valid sign-up, a new User object is created in the database, however the account is initially set to an unactivated status. An email containing an activation link is generated by the backend server and emailed to the email address supplied by the user. The user activates the account by clicking the link in the email. Prior to activating their account a user cannot access the account and cannot rent any bicycles. This ensures that only users with a valid York College of Pennsylvania email address can use the bicycle rental service. Activation emails are sent using the third-party service SendGrid[9].

Renting and Returning Bicycles – Once a user has signed up and activated their account, they may request a single-use code that can be entered at a rental kiosk to rent one or more bicycles. Each generated code belongs to a specific user. A link displayed as 'Start Rental' on the web interface calls a controller action to generate a new unique code and the code is displayed to the user as shown in Figure 3. The Rails application generates the four-digit one-time use codes using a variant of the Rails 'SecureRandom' function. An integer in the range 0-9999 is converted to a string and padded to support leading zeros. The generated code is checked against the backend database to ensure uniqueness. If the code already exists in the database a new random code is generated to ensure each user receives a unique code. Generated codes expire after 5 minutes.



Enter the following one time use code at a rental station to complete your transaction

3743

Code will expire after 5 minutes

Maximum rental period is 4 hours.

YCP Engineering Capstone BikeRack Project © 2015-2016

Figure 3: Generation of one-time use code

When a user inputs a rental code into a kiosk, it is transmitted to the backend server where the server verifies the validity of the code and ensure that it has not yet expired. The server also validates that the number of requested bicycles is available and that each of the requested bicycles has a GPS tracker with battery level of at least 70%. The GPS tracking devices located on each bicycle begin to lose accuracy once the battery drops below 50%. Ensuring that the battery level of at least 70% ensures the user has at least 4 hours of rental time before the battery reaches 50%. This also provides enough time for missing bicycles to be tracked and located before the battery in the GPS tracker is completely depleted. If a GPS tracker battery drops below 50% the user who rented the bicycle and the administrator of the system are notified via email. These emails are generated by the backend server and sent using SendGrid[8].

When a rental is validated the backend server creates a new Transaction object for each bicycle being rented. Transaction objects for ongoing rentals include the following fields: the bicycle identifier, the email of the user renting the bicycle, and the start time of the rental. Transaction objects can be viewed by administrators via the web interface. When a bicycle is returned to a kiosk the server verifies that the return is valid and ensures that the bicycle being returned is not already parked at a rental station. If a return is valid, the database is updated to indicate the returned bicycle's new location and the bicycle is made available to rent once again. Once a bicycle is returned the Transaction object associated with its rental is changed to a Transaction History object. The Transaction History objects include the end time of the rental and can be viewed by administrators via the web interface to provide administrators with a complete history of rentals.

Integration with Kiosks and GPS Tracking Devices

Each rental kiosk communicates securely with the backend server using HTTP over SSL (HTTPS). The Rails application running on the backend server routes incoming requests to controller actions that interact with the backend database and respond to the request as necessary. Both the backend server and the kiosks send/receive parameters as JSON formatted data over the HTTPS connection.

Kiosk Communication for Rentals and Returns – When a one-time use code is entered at a rental kiosk, that kiosk generates an HTTP POST request for the rental to the backend server. The HTTP POST request contains the one-time use code and identifiers for the requested bicycles as JSON parameters. If the code and all bicycle identifiers are valid the server returns a successful status to the kiosk. Otherwise, an error code is returned. Possible error codes for a rental request include:

"*expired code*", "*invalid bicycle identifier*", and "*GPS tracker has insufficient battery*". Renters see a message on the kiosk screen describing the returned error

Bicycle returns are also handled via an HTTP POST. The request contains the identifier of the returned bicycle and is transmitted as JSON formatted data. The response includes a status code of 200 for successful return, or a code of 100 for an invalid return.

Kiosk Communication for System Health Status – The health of the kiosks, bicycles, and locks can be updated through HTTP POST request. Each object has its own POST URL for updating its health condition. The backend expects a condition code detailing the issue being reported. Each condition codes identifies a specific problem with the system such as a flat tire on a bicycle or a broken locking mechanism. For bicycles and kiosks the power status is periodically sent to the backend server to keep this information up to date. Responses from the backend server include a status code of 100 for unsuccessful updates and 200 for successful updates.

Synchronizing the Kiosk Status with the Backend – Periodically, each kiosk sends an HTTP request detailing which bicycles are currently locked at the kiosk, and in which locks those bicycles are located. This method is useful to update the backend server if bicycles have been added or removed outside of the normal rent/return process. This may occur if a bicycle is stolen or removed manually for maintenance.

GPS Tracker Communication – Each GPS tracking device communicates securely with the backend server using HTTPS. These HTTP requests include the type of message and the bicycle identifier as request header fields. There are three types of messages: a startup message, a location update message that updates the GPS coordinates of the bicycle, and a message to signal the docking or undocking of the bicycle. The HTTP POST parameters vary by message type. This GPS location update message includes the latitude, longitude, the battery level of the GPS tracker, and additional information about the accuracy of the coordinates. The latitude/longitude coordinates are sent in degrees-minutes format. The backend converts these values to decimal format before saving the updated values to the database.

Conclusions

A backend server is a key part to running and managing an automated bicycle rental system. In this paper, a cloud-based backend server solution was presented for an automated bicycle rental system. Using a cloud-based server provider allowed for faster and smoother setup and development. The server was developed using Ruby on Rails and is hosted on Heroku. The rental kiosks communicate with the backend server over HTTP to authorize bicycle rentals, and report unexpected status or health changes. Bicycles that are equipped with GPS tracking devices can also send GPS location data over HTTP to the backend server. This gives administrators of the system the ability to track rented bicycles and locate stolen bicycles. Users can easily register and begin renting bicycles from any web browser. Additionally, a full administrative website provides an administrator a convenient interface for monitoring and managing the system. Unique features of the York College of Pennsylvania ABRS system include GPS tracking of bicycles and the administrative control interface.

References

1 Paul J. DeMaio, "Smart Bikes: Public Transportation for the 21st Century" Transportation Quarterly, Vol. 57, No. 1, Winter 2003.

2 Kala Meah, James Moscola, and Andrew Warner, "Design and Implementation of a Wireless GPS-Based Bicycle-Tracking Device for Capstone Design," ASEE 124th Annual Conference & Exposition, Columbus, OH, June 25-28, 2017.

3 James Lindenbaum and Adam Wiggins, "Cloud Application Platform | Heroku." *Cloud Application Platform | Heroku*. Salesforce App Cloud, 2007.

4 Michael Hart, "Ruby on Rails Tutorial (3rd Ed.)," *Ruby On Rails Tutorial*. Addison-Wesley Professional Ruby, ISBN: 978-0134077703.

5 "Ruby on Rails IDE :: JetBrains RubyMine," *JetBrains*. Ed. IntelliJ IDEA, 2000. [Online]. Available: https://www.jetbrains.com/ruby/

6 Mark Otto, "Bootstrap, The World's Most Popular Mobile-first and Responsive Front-end Framework," [Online]. Available: http://getbootstrap.com/

7 JavaScript Object Notation (JSON). [Online]. Available: http://www.json.org/

8 Benjamin Roth, "Gmaps4Rails," *Gmaps4Rails*. Apneadiving, 14 Feb. 2011.

9 "Marketing & Transactional Email Service | SendGrid." [Online]. Available: https://sendgrid.com/

10 Scott Kiefer, Tristan Ericson, Kala Meah, and James Moscola, "Design, Build, and Installation of an Automated Bike Rental System as a Part of Capstone Design," ASEE 123rd Annual Conference & Exposition, New Orleans, LA, June 26-29, 2016.

Thuong Nguyen

Thuong received a B.S. in Computer Engineering from York College of Pennsylvania in 2016. She received an A.S. in Engineering and an AAS in Computer Aided Draft and Design from Harford Community College in 2012. Her interests include mobile applications, web development, and mobile wearable technologies.

Justin Reichner

Justin received a B.S. in Computer Engineering from York College of Pennsylvania in 2016. His interests include embedded systems and web development.

James Moscola

Dr. Moscola is an Assistant Professor of Computer Science and Computer Engineering at York College of Pennsylvania. He received a B.S. in Physical Science from Muhlenberg College in 2000, a B.S. in Computer Engineering, a M.S. in Computer Science, and a Ph.D. in Computer Engineering from Washington University in St. Louis in 2001, 2003, and 2008 respectively. His interests include embedded systems and hardware acceleration.

Kala Meah

Dr. Meah received the B.Sc. degree from Bangladesh University of Engineering and Technology in 1998, the M.Sc. degree from South Dakota State University in 2003, and the Ph.D. degree from the University of Wyoming in 2007, all in Electrical Engineering. Currently, Dr. Meah is an Associate Professor in the Department of Engineering and Computer Science at York College of Pennsylvania. His research interest includes electrical power, HVDC transmission, renewable energy, energy conversion, and engineering education.