

## Automated Cloud-Based Mobile Payment Solution for Vehicle Parking Areas

A.I. Tambuwal<sup>#1</sup>, B.A. Buhari<sup>#2</sup>

#1 UsmanuDanfodiyo University, Department of Mathematics, PMB 2346 Sokoto Nigeria,  
+2348067315681.

#2 UsmanuDanfodiyo University, Department of Mathematics, PMB 2346 Sokoto Nigeria,  
+2347037602702.

### ABSTRACT

Mobile devices have become more powerful and pervasive which result in mobile application development to become more important. As a result of the popularization of mobile devices and mobile operating systems market, many applications are being developed and deployed on mobile devices. This has resulted in mobile applications becoming increasingly more present in our daily lives, allowing people to perform several tasks through the use of smart phones, tablets or equivalent devices. In particular, there are many mobile applications that help people to carry out payments. We notice, however, there is also the need for mobile applications that can be used for car parking payments within Abuja. In this paper, a cloud - based mobile application that support multiple platform is presented which enable drivers in Abuja to carry out parking payments using their mobile phones. The result of the system evaluation using some selected users demonstrated that the developed mobile application have most of the functionalities that can help drivers to easily register, manage their account details, securely pay for parking, monitor and extend parking sessions remotely. The proposed application also helps the driver to remember the exact location the vehicle was parked.

**Key words:** Mobile Cloud Computing, Mobile Devices, Payment Solution, Packing Areas, Mobile Application

**Corresponding Author:** [A.I. Tambuwal](#)

### INTRODUCTION

Vehicle owners in Abuja experience a lot of problems when making parking payment such as, searching around to locate the machine for payment and running back to the machine to make another payment for exceeded parking period if they want stay longer. Since there are small numbers of payment machines, any increase in the parking areas and vehicle owners that uses them will result to the increase in difficulties in making payment for parking. In order to address this problem, there is need to provide a system that will enable the vehicle owners to pay for parking using their mobile devices. Moreover, whenever vehicle owners need to increase his period of parking, they can do that using their mobile device regardless of their location which will reduce the difficulties of running back to locate the payment machine before any payment

can be made. This is achieved by developing a cloud - based mobile application that will enable the vehicle owners to carry out their parking payment using mobile devices. The application help drivers to register with ease, securely pay for parking, monitor and extend parking sessions remotely, manage their account details, and even help them remember where they parked their vehicle.

## **RELATED WORKS**

### **MOBILE CLOUD COMPUTING**

Mobile cloud computing simply means to store data and run an application on remote servers and the mobile device acts as a client connecting to the server. Mobile cloud computing is the use of mobile devices to obtain services remotely in a wireless environment. It integrates the functionalities of mobile computing, mobile internet and cloud computing and hence can be called cloud computing in mobile internet. The advantage of cloud computing in mobile internet is to transport applicable computing and storage from the mobile device to cloud back end server, so as to reduce the processing requirements of the mobile device [1]. Even though, there is increase in the use of mobile computing, it is often difficult to make use of its full potential as a result of problems such as resource scarcity, frequent disconnections and mobility. Mobile cloud computing provide solutions to this problems by allowing mobile applications to execute on a resource that are provided external to the mobile device [2]. “However, mobile clients could face wide variations and rapid changes in network conditions and local resource availability when accessing remote data and services. As a result, one partitioning model does not satisfy all application types and devices. In order to enable applications and systems to continue to operate in such dynamic environments, mobile cloud applications must react with dynamical adjusting of the computing functionality between the mobile device and cloud depending on circumstances” [3]. Mobile technologies have also received support from the presence of cloud providers by their high uptime and reduced latency through distributed hosting. Cloud computing environment can reduce some of the security risk associated with mobile computing by dividing the data so that only non-sensitive data is stored in the cloud and accessible to a mobile device. This also minimises the exposure of reduced end-point security with regards to malware infections because; the mobile device is only connected to a public infrastructure [4].

### **Sencha Touch 2**

The development of the frontend of the application is through using Sencha Touch 2 technology. Sencha Touch is a high-performance HTML5 JavaScript mobile application framework that enables the development of cross-platform user interface with HTML5. It solves cross-platform mobile application development by providing the developers with the tools needed to build cross-platform applications that mimic natively compiled applications, while making full use of HTML5 and CSS3. The main benefits of using Sencha touch 2 are: enable the working with variety of devices such as iOS, Android and tablets, provide uniform look and feel, beautiful user interface and rich data management and convenient ease of use comparing to other technologies such as Object C or Java.

### **PhoneGap**

One of the technologies that are used in developing the application is PhoneGap. PhoneGap is one of the most widely used cross-platform technology for creating apps that operate for multiple devices, such as the iPhone, Android, Blackberry, Windows, WebOS and Symbian. It is the only open source mobile framework that supports all the major seven platforms. PhoneGap enable mobile apps to be easily created using popular web technologies such as HTML5, CSS, and JavaScript [5]. This technology provides a rich collection of client-side JavaScript APIs with a method for hosting the web application within a native mobile application. It also provides access to portions of the native device capabilities such as Camera Support, Touch Screen, Geolocation, Hardware Sensors, Local Database and Storage, Process Management, etc. through a wrapper (usually JavaScript) to the native API [6]. Some of the benefits of using PhoneGap in the development of mobile application includes: reduce development time and save a long-time maintenance cost, fast development, access native features, deploying the applications to multiple platform and then use PhoneGap to package your work for specific mobile operating systems.

### **Heroku - Platform as a Service**

Heroku is one of the most popular platform supporting deployment and management of application in the cloud. It supports several programming languages including Java, Ruby, Node.js and Scala. In comparison with different platforms Heroku ideal for prototyping as there are no upfront costs required to deploy application in the cloud together with a database (Postgres). Once the application reach production stage, very attractive pricing plans can support backups, traffic and maintenance of the application. It also support continuous deployment by easily creating testing, staging, and production versions of an app and deploy to and between them instantly [7]. Some top benefits of heroku in cloud computing applications includes: easy deployment of the application, ideal for prototyping, no need for OS knowledge, easy interface and big list of add-ons.

### **PostgreSQL**

PostgreSQL is a database on the cloud provided by Heroku cloud platform as a service (PaaS) which is owned by Salesforce. PostgreSQL is the data store of choice for reliable web-applications which provide a powerful, reliable and durable open-source SQL-compliant database. The databases can be used from any cloud, PaaS, or your local computer and it is easy to connect from common languages such as Ruby on Rails, PHP and Java where configuration settings are generated for them automatically. With PostgreSQL the developer doesn't have to worry about daily backups, because there is continuous protection which keeps data safe. Every change made to the data is written to write-ahead logs, which are shipped to multi data centre, high-durability storage. In the event of unrecoverable hardware failure, these logs can be automatically replayed to recover the database to its last known state [8]. For the purpose of storage, we design our own database using PostgreSQL, which is one of Heroku add-ons.

## METHODOLOGY

The existence of different mobile operating systems or platforms with different programming languages and development tools can be a problem when the developer wants to develop and release an application that can be used in many platforms. Due to the specificity of each platform, a typical approach for developing a native app for a given platform requires rewriting the application to support different platforms mobile devices. In this paper, this will lead to managing the ongoing development of this mobile app for each of the platforms that we wish to support. This will be difficult, impracticable either in terms of money or development time and will result to slow in start-up for this project by requiring a lot of efforts to be made. This is as a result of different tools, APIs, devices with different capabilities and also differences among the actual platform SDKs (Software development kits) [9][10]. As it was stated, our application is develop to support multiple platforms; therefore we have decided to design hybrid application using the combination of two frameworks: Sencha Touch2 and Phone Gap. Sencha Touch2 supports implementation of native looking interface whereas Phone Gap enables the integration of this into the native application. In this paper we uses the development method that started by developing web application which is later integrated and converted using PhoneGap bridging framework to native mobile application for a particular mobile platform. This development method provides solution that can easily be converted to support multiple mobile platforms.

## SYSTEM DESIGN

The application architecture resembles client server architecture that consists of mainly three (3) parts which includes Client side, Server side and Database. All the data regarding drivers profile records, login details and parking payment transaction are pushed or requested and then retrieved from the server which makes queries to the database. The server and database resides on the cloud. The overall architecture of the application is shown in Figure 1.

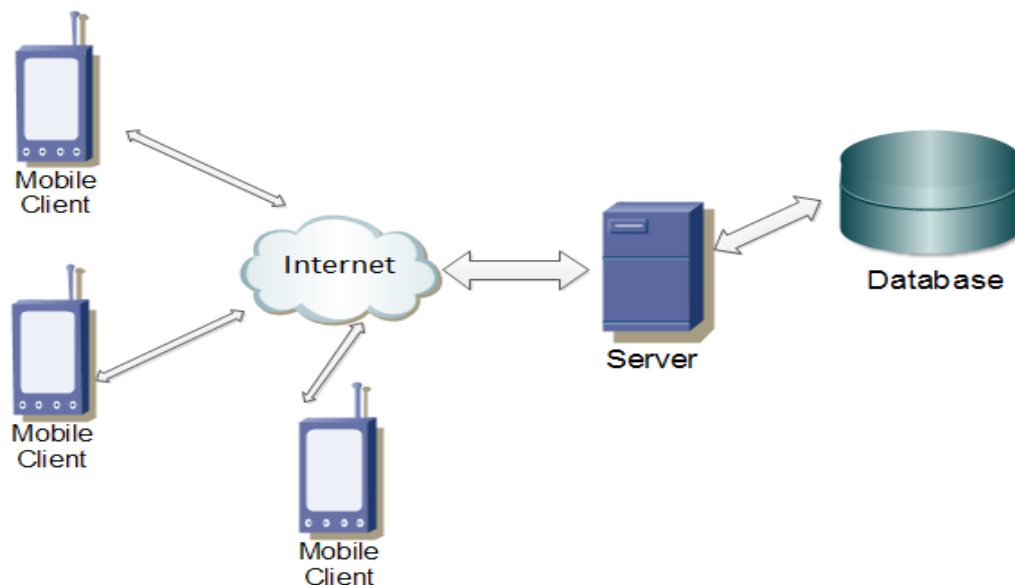


Figure 1: System Architecture

As showed in the architecture above, the user (driver) needs a mobile phone to access the client side mobile application. The web server serves as the point of entry to the website and any

backend system that it might interface with (e.g. database). There is a need of a web server in this system to provide respond to the request for web pages. So that whenever, the user (driver) made a request (e.g. login into their accounts), it is send to the web server which listens and responds to the request by first retrieving the page information from the database and display it. A web application is provided (RESTful web services) which produce a dynamic response according to the type of inputs users provided with their request.

The system requires the database for persistent data storage. The account details of drivers are sent to the database after registration which is required to verify the driver when he/she logged into his/her account. The payments records are also sent to the database to be access by the parking warden while monitoring the parking areas.

### **CLIENT SIDE**

The User Interface is designed specifically to ease the means of interaction between the user and the mobile application. A user often interacts with interfaces when using web or any software applications. This will result in interface changing over a period of time or application having many interfaces, while the underlying procedures remain constant. The developments pattern that create interface together with application data models will make it harder to migrates and reuse both the interface and procedures in other applications. Therefore, a good development pattern is to separates application data models from its interfaces [11]. MVC is a software architecture pattern which separates the representation of information from the user's interaction with it. The model consists of application data, business rules, logic, and functions. A view is any output representation of data where multiple views can be created with the same data. The Controller allows the communications between the Model and View by mediating input and converts it to commands that can be used by model or view. Although MVC was originally developed for personal computing, it has been widely used as a very common design pattern in user interface development of web applications. Some of the advantages of using MVC architecture pattern are: code maintainability, readability, reusability, decoupling and reduces complexity of Data binding.

In this paper, we used Model View Controller (MVC) pattern in our development where the Model component contains our database classes and methods, Controller component holds our RESTful web service methods that communicate with data model classes and the View component contains HTML pages with Java Script methods that allow them to communicate with the web service methods.

The client side user interface consists of four (4) major components which include:

- (i) Registration Page that allow the user (driver) to register by entering the values require for the registration which includes username, password, mobile number, plate number, card number, expiry date. It also contains checkboxes that allow user to select whether to receive text reminder, receipt, accept term and conditions and register button as shown in figure 2 below.

**Parking Payment**

### Please Register

**Enter User Name**  
Username

**Enter Password**  
Password

**Enter Mobile Number**  
Mobile Number

**Enter Vehicle Plate Number**  
Plate Number

**Enter Credit/Debit Card Number**  
Card Number

**Enter Card Expiry Date**  
Expiry Date

Send Text Reminders  
 Send Text Receipt  
 Accept Terms and Conditions

**Register**

Figure 2: Registration Page

- (ii) Login Page that allow user (driver) to login into his account. It contains two text boxes, username and password where the driver enters his login details and submits by clicking login button as shown in figure 3 below.

**Parking Payment**

### Please sign in Here

**Enter User Name**  
Username

**Enter Password**  
Password

Stay signed in

**Sign in**

Figure 3: Login Page

- (iii) Payment Page that allow the user (driver) to make his parking payment. The page contains textboxes that display the driver mobile number, car plate number, cost to be paid, parking expiry time and another one that allow him to enter his card CVV code. It also contains two option boxes that allow user to select how long he wishes to park and region where he is parking. Figure 4 shows a registered user making his parking payment.

The screenshot shows a mobile application interface for making a parking payment. The title bar at the top is dark and contains the text 'Parking Payment' and a hamburger menu icon. The main content area is white and has a large heading 'Make Payment'. Below the heading are several input fields, each with a blue label above it: 'Mobile Number' (08067315681), 'Vehicle Plate Number' (AA 23 ARG), 'Select Region you want to Park' (Region A), 'Select How Long You Want to Park' (One Hour), 'Cost to be charge' (£1), and 'Enter the CVV 2 - 3 code for card expiring on 2014'. At the bottom of the form is a large blue button labeled 'Pay'.

Figure 4: Payment Page

- (iv) Timer Page which contains two textboxes that display the expiry time and clock time that count down the time base on the duration driver selected. It also contains extend button that is showed after the driver is alerted when one minute remain in his/her parking period which reloads the payment page to allow driver extend his parking period. Figure 5 below shows a timer clock monitoring a parking period of one hour.

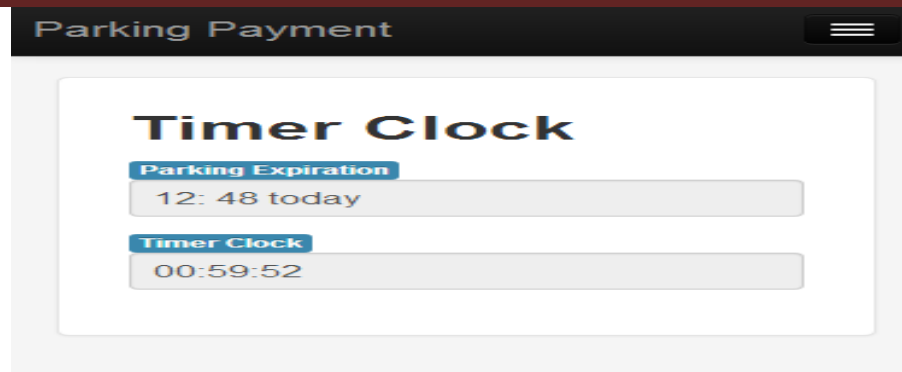


Figure 5: Timer Page

### **SERVER SIDE**

For backend of the applications, cloud server solution was provided. All data together with its operational logic resides in cloud. This decision was made based on benefits that it brings in comparison with hosted solutions. Cloud computing supports start-up companies as it promotes paying (monthly) for used infrastructure or services rather spending vast upfront costs. Some of the advantages of this choice of technology are: pay per usage (hardware on demand), no system administrators needed, includes server monitoring and easy scaling up/down of instances (auto balancing).

### **DATABASE**

For the database design, we used Data Access Object pattern which is one of the core J2EE pattern commonly used in many enterprise application where data storage is required. This pattern enables separation of the data access logic from business logic [12]. Some of the advantages of using data access object pattern are: it separates business logic from data access logic, it provides easy way to retrieve objects, it supports scaling up database in organised way and it allows easy integration of multiple/different storage types.

### **SYSTEM IMPLEMENTATION**

This section describes the actual implementation of the cloud – based mobile application based on the system architecture specified in Figure 1 above. The implementation started by developing web application which is later integrated and converted using PhoneGap bridging framework to native mobile application for a particular mobile platform. The system components and how they are implemented to serve the needed functionalities of the developed system are specified.

### **CLIENT SIDE INTERFACE**

As mention earlier the user interface pages were implemented using Sencha Touch 2 framework which is a high-performance HTML5 JavaScript mobile application framework that enables the development of cross-platform user interface with HTML5. Also, in order to retrieve data from the database that resides in the cloud, the project uses REST architecture to communicate with the data from the database, allowing adding and filtering of relevant records from the database. REST (Representational State Transfer) is the major model used in implementing web services. The key sources of information are resources. Every resource can be addressed and accessed with a unique URI. The RESTful service deployed was a Jersey service and developed several

service methods. Table 1 summarises the web service methods developed, their path URL's and their functions.

Table 1: RESTful Service methods

Method	Method Name	Path URL	Functions
GET	<i>findAllRecords()</i>	rest/users	Return all registered users (drivers).
	<i>findAllPaymentRecords()</i>	rest/users	Return all payments drivers made.
POST	<i>addUser()</i>	rest/users	Adds a new driver after registration.
	<i>addUserPayment()</i>	rest/users/payments	Adds payment made by the user (driver).
	<i>login()</i>	rest/Webservice/login?username, password=param	Search and login driver using a given username and password
	<i>search()</i>	Rest/users/search?region=param	Search all payments made in a given region
PUT	<i>updateUser()</i>	rest/users/username	Update user (driver) records
DELETE	<i>deleteUser()</i>	Rest/users/username?username=param	Deletes a user (driver) records with a given username.

The implementation provides only a simple prototype to demonstrate payment system, by making assumption that payment is made and payment records uploaded to the database. In real system, the application will use online payment methods which allow drivers to carry out their parking payments using their Credit/Debit cards. To provide the payment service, this research work proposes the use of PayPal which is a global e-commerce business allowing payments and money transfers to be made through Internet. PayPal provides APIs that allow developers to set up instant payment in their applications. PayPal provides three (3) set of APIs for developers as follows: adaptive Payments API allow developers to set up automated payments by creating applications that manage payments, payment pre approvals, and refunds; express Checkout allows developers to minimize the number of steps customers must complete when they checkout and PayPal Payments Advanced which accept credit cards, PayPal, and Bill Me Later online with PayPal Payments Advanced.

### SERVER SIDE

As mention earlier, at the back end we provide a cloud server solution. In this subsection we discussed how the client side web application was deployed on the cloud as follows:

The implementation involves installing HerokuGit in eclipse plug-ins which allows developing a Heroku application. A Heroku Jersey template application was developed to host the parking

payment web application. The Heroku Jersey template application has similar structures with the web application developed above which uses Jersey web service. After the application was developed, it was imported to eclipse through importing an existing Heroku application. The Model, Controller and View files including libraries files were copied one after another from the web application to Heroku application, committing changes and pushing them to the remote application at each stage. The URLs in Java Script methods that make called to the web service methods using local server were changed with the Heroku app URLs. Lastly, dependencies codes that link the application to the PostgreSQL database already on the cloud were also added.

## **DATABASE**

The Database was setup in the HerokuPostgreSQL. A Java class has been implemented with a method to establish connection to the database and return a connection string that enables remote communication with the database. Java methods are also implemented to connect and create the Tables in the database. Two (2) Tables were created which includes Registration Table, Payment Table and Signing Table. The Table structures and schema are as follows:

- The registration table stores the registration details for a new user (driver) that signs up for a new account. The table consists of the following columns:
  - Username: A text field to hold the username provided by the user (driver) which will be used for login verification.
  - Password: A text field required is required to store the user (driver) password for login verification
  - Mobile number: A field required to hold the mobile number of the user (driver) which will be used to send text messages for parking charges.
  - Number plate: This field stores the car licence plate number which can be used to identify the car parked.
  - Card number & Expiry date: These fields store credit card details for payment. The credit card details are stored in the database to save the user from having to provide his/her card details whenever he/she is paying.
- Payment table stores the records of users currently logged in and payments made by the users. The Table has the following columns/fields:
  - Region: This field stores the region where the user wants to park.
  - Number plate: This field stores the car licence plate number which can be used to identify the car parked.
  - Period: This field is used to store how long the driver would like to park his/her car for and will be used to calculate the amount to be paid.
  - Amount: This field is used to store the amount paid by the user.
  - Expiry: This field is used to store the expiry time that parking is due to expire.

## CONVERSION OF WEB APPLICATION TO NATIVE MOBILE APPLICATION

As stated earlier, the implementation started by developing web application which is later converted to mobile application. This section described how the web application was converted to mobile application. For the implementation of the native mobile applications, Android SDK is installed in the eclipse plug-in that allow Android Application project development. Two Android applications for driver and parking warden were developed respectively using PhoneGap.

### SYSTEM EVALUATIONS

After implementing the developed mobile parking payment application, the performance of the system has been evaluated by testing the functionalities provided by the system which includes driver registering, login in, making payments, alerting him when the time is about to expired and extending his parking period. In order to evaluate the application functionalities, ten (10) drivers were registered. Each driver has his account with login details that he can log in, update his details if he wish and makes payment. Figure six (6) below shows a driver login in his account and changing his records.

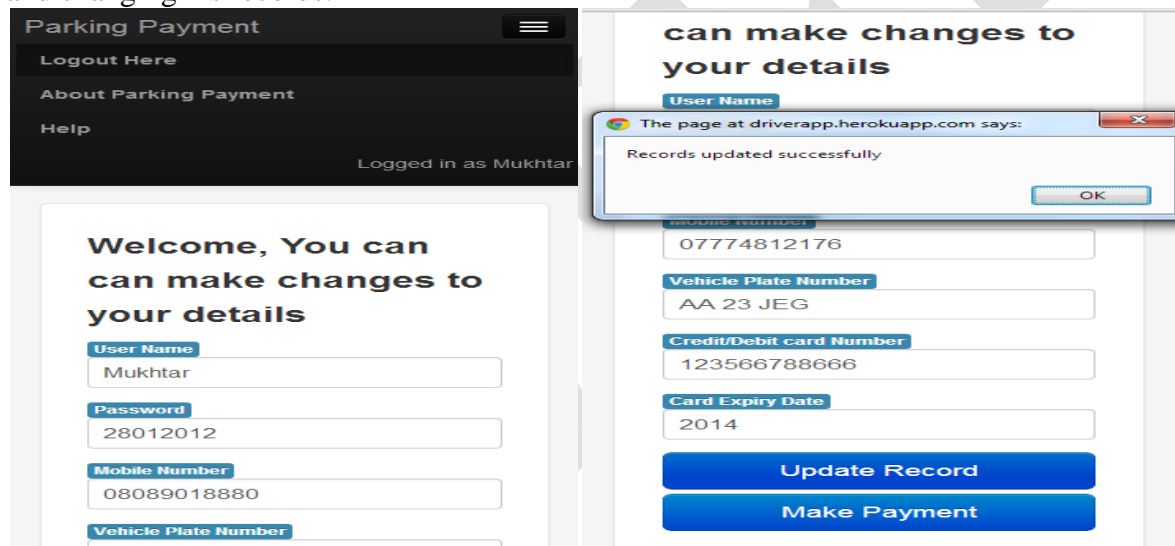


Figure 6: Driver updating his account details

In figure seven (7) below, a driver successfully makes his payment and an alert is shown to him when one minute is remaining in his parking period.

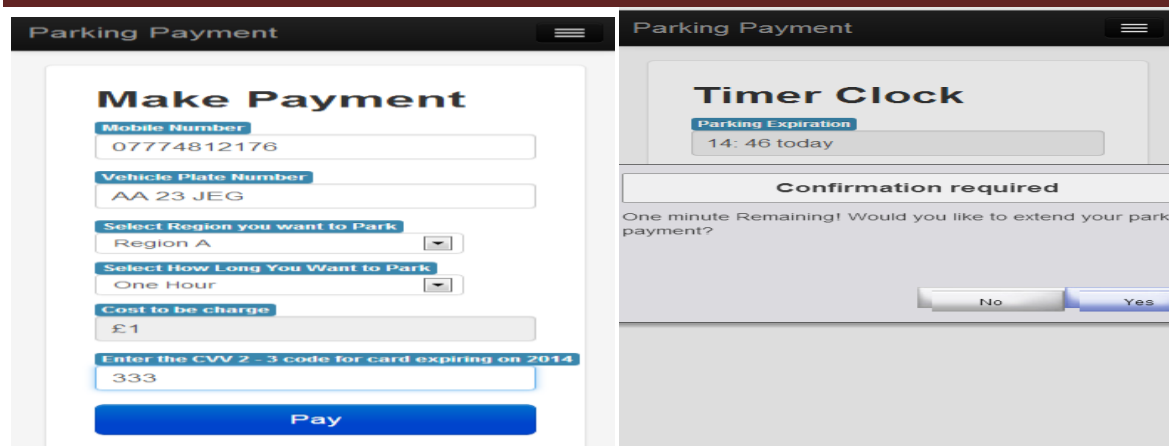


Figure 7: Driver making payment

Table 2 below summarises the functionalities that are achieved, those that are partially achieved or not achieved with explanation of why they are not achieved. In the Table, the fulfilment column contains numerical values 1 and 2 which shows the level at which the functionality rich completion. One (1) partially achieved and two (2) fully achieved.

Table 2: Summaries of the functionalities that are achieved

Requirements	Fulfilment	Comments
Provision of persistent data storage.	2	Database was successfully set and created to store the data.
Registration	2	Driver can successfully register to create an account and his details are added to the database.
Login	2	Registered drivers successfully logins to their account
Updating account details	2	After login, drivers can successfully make changes and update their account details.
Payments	1	In order to implement full payment functionality, a payment system such as PayPal shall be used. This is not fully implemented because it requires budget. A prototype of payment system is provided to demonstrate drivers making payments when logged in.
Reminder	2	A Timer is used which monitors the time and alert the driver when it remains one minutes before the period expired and when it expired.
Parking extension	2	Drivers can successfully extend their parking period after receiving an alert.
Provision of Cloud solution	2	The application was successfully deploy on the cloud using Heroku cloud platform as a service.

Conversion of web application to a native mobile app	2	The web application is successfully converted to a native Android mobile application
--	---	--

## CONCLUSION AND FUTURE WORKS

In this paper, a cloud - based mobile application that support multiple platform is presented which enable drivers in Abuja to carry out parking payments using their mobile phones. System evaluation with some selected users show that the application allow drivers to easily register, manage their account details, securely pay for parking, monitor and extend parking sessions remotely. The work started by developing web application which is later integrated and converted using PhoneGap bridging framework to native mobile application for Android mobile platform. Similarly, the application can be easily modify to support any mobile platform of choice by simply importing PhoneGap libraries that makes it possible to access all native functionalities supported by PhoneGap and adding codes that define support for screens and permission to use those native features.

However, as a future work, the research should focus on integration and use of mobile device GPS to get current location, Google Map to set up current location and converting the web application to native mobile application using different mobile platforms. The future research will also focus development of another mobile application that will help parking wardens to manage and control parking payments for parking areas within Abuja metropolis.

## REFERENCES

- [1] Song, W., & Su, X. (2011). "Review of mobile cloud computing". In Communication Software and Networks (ICCSN), IEEE 3rd International Conference, pp. 1-4, May 2011. IEEE Xplore [Online]. Available at: <http://ieeexplore.ieee.org/> (Accessed 2<sup>nd</sup> August, 2013).
- [2] Niroshine Fernando\*, Seng W. Loke\*, and Wenny Rahayu (2012) "Mobile cloud computing: A survey." Future Generation Computer Systems, pp84-106. June 2012. SciVerse ScienceDirect [Online]. Available at: [www.elsevier.com/locate/fgcs](http://www.elsevier.com/locate/fgcs) [Accessed 20<sup>th</sup> July, 2013].
- [3] Kovachev, D., Cao, Y., & Klamma, R. (2011). "Mobile cloud computing: a comparison of application models". arXiv preprint arXiv:1107.4940.
- [4] John Rhoton (2010) "Cloud Computing Explain", 2<sup>nd</sup> edn. UK & US. Recursive press.
- [5] PhoneGap. Available at: <http://phonegap.com/> (Accessed 24 Feb., 2013)
- [6] Allen, Sarah, Vidal Graupera, and Lee Lundrigan. (2010) "PhoneGap" in Pro Smartphone Cross-Platform Development, pp. 131-152 [Online]. Available at: <http://link.springer.com/> (Accessed 23 Feb., 2013)
- [7] Heroku Cloud Application Platform. Available at: <http://www.heroku.com/> (Accessed 17 Feb., 2013).
- [8] Heroku PostgreSQL. Available at: <https://addons.heroku.com/heroku-postgresql> (Accessed 13 Feb., 2013)
- [9] Ribeiro, André, and Alberto Rodrigues da Silva. (2012) "Survey on Cross-Platforms and Languages for Mobile Apps" Eighth International Conference on the Quality of Information

- 
- and Communications Technology, [Online]. Available at: <http://inesc-id.pt.com> (Accessed 20 Feb., 2013).
- [10] Cernea, Victor, and José Calla Guzmán. (2012) "Cross-platform development of mobile applications-Evaluation of the PhoneGapframework."Bachelor Thesis. Gothenburg: Chalmers University of Technology [Online]. Available at: <http://publications.lib.chalmers.se/>
- [11] Deacon John (2009)."Model-view-controller (mvc) architecture." [Online] Available at: <https://techsimplified2.com/> (Accessed 20<sup>th</sup> June, 2013).
- [12] LONG, S. Y., LIN, H., & CHEN, X. Y. (2004). "Data Access Object Pattern".Computer and Modernization, pp5,2004 [Online]. Available at: <http://en.cnki.com.cn/> (Accessed 6<sup>th</sup> August, 2013)