ASHESI UNIVERSITY COLLEGE

A WEB-BASED CUSTOMER DATABASE MANAGEMENT SYSTEM FOR CADMUS ELECTRONICS LTD. WITH BULK SMS NOTIFICATION FEATURE

UNDERGRADUATE APPLIED PROJECT

B.Sc. Computer Science

Derryck Noi Dowuona

2018

ASHESI UNIVERSITY COLLEGE

A Web-based Customer Database Management System for Cadmus Electronics Ltd. with Bulk SMS Notification Feature

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi University College, in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science.

Derryck Nii Noi Dowuona
April 2018

DECLARATION

I hereby declare that this applied project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature: ....

Candidate's Name: ....

Date: ....

I hereby declare that preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of applied project laid down by Ashesi University College.

Supervisor's Signature:

……………………………………………………………………………………………

Supervisor's Name:

……………………………………………………………………………………………

Date:

……………………………………………………………………………………………

Acknowledgements

I would like to thank God first of all, for keeping me alive. And a mighty, grateful, special thank you to my father for being able to pay my school fees for the past four years of my life in Ashesi.

Thank you to Mr. David Sampah my supervisor, for your guidance and support. You have really helped me to bring this idea into reality and I appreciate all your insights and advice.

A special thank you to Hubtel's support services for responding to my emails requesting guidance and technical support. It means a lot.

To Mr. Evans Attafuah of PsypherTxt, who helped me to learn the tools and frameworks I used in building this project, thank you very much.

My gratitude goes to Mrs. Satina Brown, the Managing Director of Cadmus Electronics for allowing me to interview and interact with you and other Cadmus employees to ensure that the proposed solution meets all requirements.

Abstract

DSTV is a satellite television service provider established by Multichoice that provides entertainment services via satellite TV to residents in African countries on a monthly basis. To continue enjoying DSTV's services, subscribers pay monthly fees via options such as Mobile Money, PayPal, and DSTV Dealers or agents. There are many DSTV Dealers in Ghana, and among the popular ones is Cadmus Electronics Ltd., which functions as both an electronics company and a Dealer for both DSTV and Canal+ (a similar satellite television service for francophone subscribers). With the existence of several DSTV Dealers in the country, Cadmus faces a tough competition of attracting customers to pay their DSTV bills through their branches in order to obtain higher revenue. Without a database of new and regular customers, Cadmus is unable to keep track of its growth rate and also promote its brand as the best DSTV dealer in the country. Cadmus desires to be the primary point of contact for all DSTV-related issues, enquiries and payments which will increase its revenue and popularity in Ghana. This project seeks to build a web-based customer database management system for Cadmus Electronics, called **OrionMonitor**, which will also have a bulk SMS feature which will notify its customers of monthly payments to be made and provide detailed options for payment (e.g. Mobile Money or Cadmus branches).

Table of Contents

Table of Figures

Chapter 1: Introduction, Background and Motivation

1.1 Introduction

Technology has become a major driving force of many business operations in the present age (Chan, 2000). The gradual acceptance of the emerging computing technologies such as advanced database management, satellite communication and data analysis has improved productivity and enhanced efficiency and rapid delivery of solutions to current and possible future problems in the world. Cloud Computing for instance, is widely used by many companies ranging small businesses to large scale enterprises. It is without a doubt that business organizations that make use of advanced technologies in their operations have greater competitive advantage than others in the corporate world (Chan, 2000).

1.2 Background

There are some companies in Ghana which without competitive advantage, find it difficult to stay ahead and earn more revenue. Cadmus Electronics, an electronics company and an accredited Dealer for Multichoice, is one such example. Multichoice is an entertainment company established in Sub Saharan Africa which mainly provides satellite television services through its products such as **DSTV** and **GOtv** (Mano, 2005). Dealers are entities that are licensed to sell products on behalf of Multichoice and other satellite television services such as **Canal+** and serve as middle men for payment of subscription fees for services provided. Other services Cadmus Electronics provides aside from selling DSTV products include sales and installation of electronic equipment, electronic repairs and service calls. Cadmus however is not the only accredited DSTV dealer in Ghana, and although it has several branches in a few major parts of the country including its capital, Accra, it faces tough competition in maintaining its numerous regular customers and attracting new customers due to competition. The challenge that comes with the presence of other dealers in the country is that customers are unable to tell the difference between the various accredited dealers; their main concern is to purchase a product or renew subscription for a service, and this poses a varying competition among the dealers.

What Cadmus Electronics Ltd desires is to have a reliable and cloud-based database of all its regular and new customers and establish a system wherein they can communicate with customers, build trust and maintain a solid relationship.

1.3 Motivation

The major problem that Cadmus faces is keeping track of all its customers without using a reliable, electronic database system. Customer data is often recorded using large notebooks which although may be suitable for small-scale businesses, cannot satisfy the large scale of Cadmus given that it has branches in many parts of Accra and beyond. It therefore requires a larger, reliable, and easily accessible database management system to keep track of its customers and maintain contact with them. With the existence of competing DSTV Dealers, losing a number of customers to them would pose a problem for the company's income revenue. Subscribers to satellite television services such as DSTV, GOTV, and Canal+ have to pay a monthly fee to continue enjoying the services, and as such, they need to be reminded to pay their subscription fees at least a few days before their subscription ends. DSTV currently has a system in place to automatically send SMS messages to customers to remind them to pay their fees via options like Mobile Money or banks, which is convenient. What Cadmus needs is to have as many customers as possible to pay their subscription fees via their branches or with Mobile Money, and market their electronic products. This will generate more revenue and propagate its popularity across the country.

1.4 Related Works

Southbank Institute of Technology, a university in Australia, conducted a study on the viability of using SMS messaging to provide services to students who use the institution's library (Herman, 2007) . In addition to their *Ask a Librarian* service, which included the use of email, chats and phone service to provide students with information, they decided to incorporate SMS service, which was found to be extremely useful in helping provide rapid response to enquiries and updates on published books. Also, due to the fact that most of the students used mobile phones on campus, preferred SMS messaging as their mode of communication which proved most convenient (Herman, 2007). This project was limited only to students of the institution, and hence was not expanded to reach out to the public.

A projected was undertaken in Bundelkhand University, India to assess the advantage of using an SMS-based alert system for its Central Library to inform students of new articles and journals by publishers (Jetty & K, 2013). The project was a follow-up on pilot project initiated by University of Swaziland and Emerald Group Publishing. The project involved gathering target recipients, creating profiles for each recipient with their preferences captured, and an attempt to automate the sending of alerts from publishers to interested recipients. The main objective of the project was to repackage email alerts from publishers into SMS messages and forward them to persons whose preferences matched that of the publishers. The problem encountered during the project's execution was that the process of converting email messages to SMS messages and sending them to target recipients could not be automated because the bulk SMS server provided no option for email gateway processing (Jetty & K, 2013).

In order to identify the main drivers of acceptance of SMS advertising and positive responses to them, a study was conducted in various locations in Seoul, South Korea, which were mostly universities (Dix, Jamieson, & Shimul, 2016). A total of two hundred and six responses were obtained and it was observed that the context of SMS advertising (in relation to the time and location) as well as trust toward the advertising companies were the main drivers of acceptance of SMS-based advertising. Advertising companies who abuse their customers' personal information and incessantly send them messages cause them to despise SMS messages, which is a relevant point to consider for customers of Cadmus Electronics Ltd.

Cadmus Electronics also requires an expedient way of making and recording sales. Sales in Cadmus branches are recorded in the traditional and outdated notebook form. These records would later have to be compiled into comprehensive reports to be sent to the Managing Director to make decisions weekly, which place a heavy burden on employees within the company.

Database systems, are one of the most sought-out technologies by companies and organizations throughout the world due to the enhanced ability to provide quick and efficient access, manipulation and distribution of information for organizations and companies (Li, 2004). The purpose of such systems is to have easy and convenient access to data files that have been organized in a comprehensive way and allow for update, deletion and insertion of new data (Yusuf & Nnadi, 1988). Many small-scale enterprises rely heavily on advanced database management systems to keep accurate records of transactions and inventory which expedites processes, enhances productivity and enables businesses to operate at a faster pace compared to traditional records in notebooks.

1.5 Project Scope

This project is aimed at creating a web-based customer database system for Cadmus Electronics to keep track of all customers to maintain a consistent relationship, and also building an inventory management system to better handle record keeping and report compilation.

1.6 Project Objectives

This project aims to primarily develop a web-based customer database system for Cadmus Electronics, named **OrionMonitor**, to keep track of all its customers and to communicate deadline payments for subscriptions and advertisement. The aim of the inventory management system is to monitor and manage records of stocks and sales to provide comprehensive reports to management for effective decision-making.

1.7 Project Significance

This applied project will be of great benefit to Cadmus Electronics Ltd. because it will give them a higher competitive advantage over other DSTV Dealers in Ghana by maintaining consistent, non-intrusive communication with customers which builds a form of trust and propagates their brand. By having more customers pay at Cadmus branches, employees receive more commission on subscription fees paid and this also increases the revenue of the company.

Chapter 2: Requirement Analysis

This chapter discusses the processes involved in gathering requirements for OrionMonitor, design decisions, detailed functional and non-functional requirements, and how users will interact with the system.

2.1 Project Scope

This project is designed mainly for use in the head office of Cadmus Electronics Ltd. in Osu, by a designated administrator and the managing director, and also in its various branches across the country. The system will be hosted on Heroku, a cloud platform service for web and mobile applications. The server-side programming of the system will be implemented using Parse; an open-source Backend-as-a-Service platform.

2.2 Feasibility

Most of the required resources are free and open source, and hence require no cost. The only aspect of the system which will incur cost will be the SMS functionality. Sending SMS messages to many users of various mobile networks incurs some charges, which need to be addressed in the development of the application. The application will be developed in such a way that SMS message configuration and notification will be done automatically with minimal input from the administrator. The cloud platform Heroku, provides a remote server with a free database package which provides a space of 0.5 Gigabytes (GB). In the event of expansion requirements of the application, upgrade packages will incur monthly fee (e.g. $15 per 1GB monthly) (“Pricing | mLab Cloud MongoDB Hosting,” n.d.).

2.3 Requirements Gathering Technique

The technique used in requirements gathering was interviews (both structured and unstructured). Unstructured interview was used initially in order to understand the issues and concerns that the company is facing, and structured was then used to get a better understanding of the requirements of the desired system.

2.4 Requirements Gathering Procedures

These were the processes involved in gathering requirements for the intended system:

a. Interview with Managing Director: unstructured and structured in order to obtain fundamental insight into the problems being faced in the company and what is expected of the proposed system.
b. Interview with Inventory Manager: To understand how inventory records were being kept and identify flaws that could be addressed by the proposed system.
c. Observing Multichoice database system: For more insight on what kind of customer data would be relevant for the proposed application.

2.5 System Requirements

These are the features and functionalities that the system is expected to implement. Because the system is database-centered, the basic requirements involve the core database operations i.e. **C**reate, **R**ead, **U**pdate and **D**elete (**CRUD**).

Customer Database System: this application is purposed to manage records of all customers of Cadmus Electronics to inform them of subscription payments and conduct analysis for management of the company.

1. The system should be able to display all customer records in a tabular form.
2. The system should be able to authorize users before modifying the database.
3. The system should be able to automatically send SMS messages to customers whose subscriptions are due at three days before termination.

2.6. Functional Requirements

Customer Database System:

1. User should be able to login with authorized credentials before using the system.
2. User should be able to view all required details of a customer.
3. User should be able to add, update and delete customer records.
4. User should be able to create SMS messages to be sent to selected customers automatically.
5. User should be able to view a comprehensive report from system for review.

2.7 Users

The main users of this database system will be the IT administrators within the company and the employees in Cadmus' branches. Both classes of users will have access to the database system and major features, with the exception of the SMS configuration and manipulating the database (adding, deleting, editing) which will be handled mainly by the IT administrators and only authorized employees.

2.8 Use Case Diagram

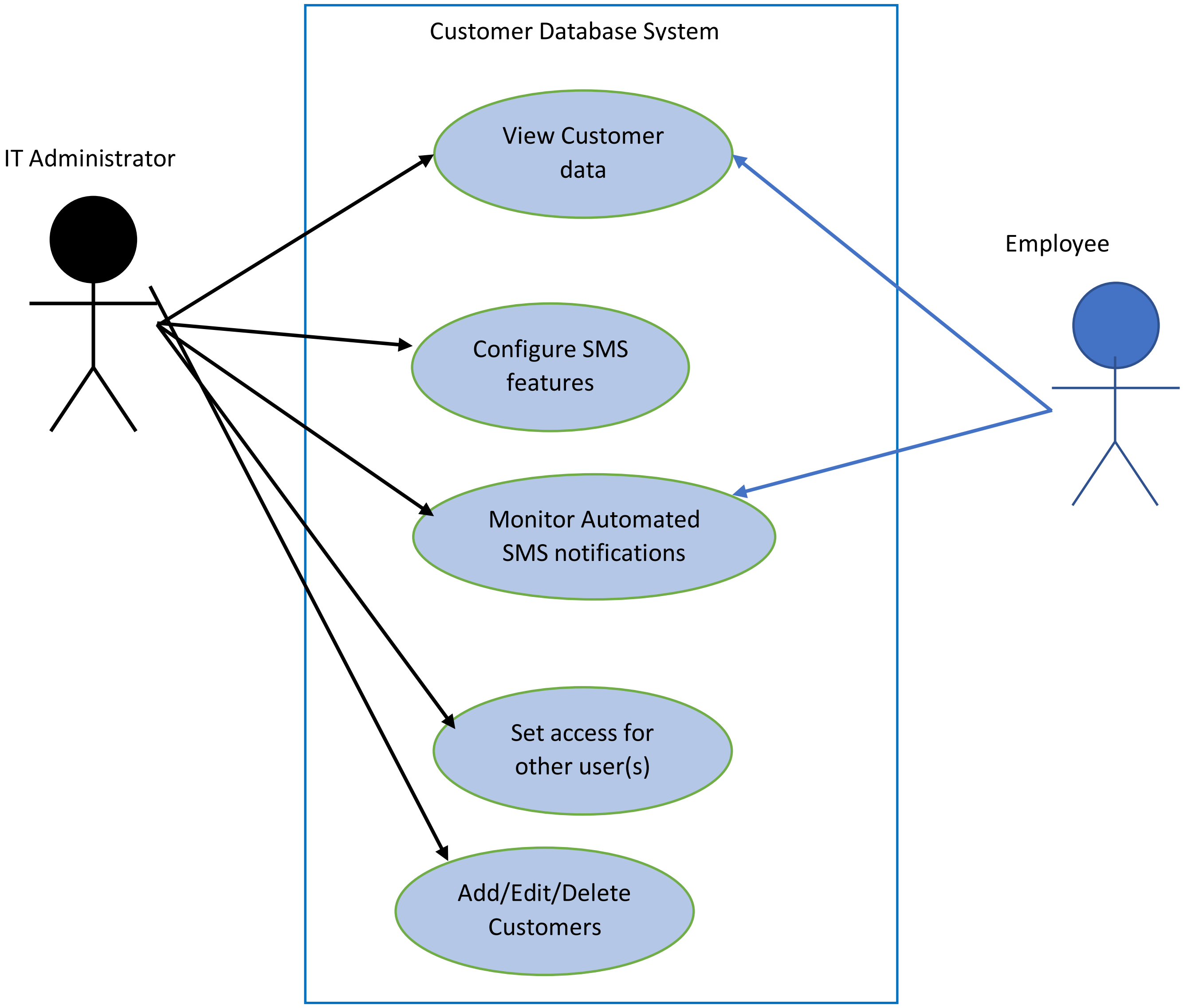

Customer Database Svstem
IT Administrator
View Customer data
Configure SMS features
Monitor Automated SMS notifications
Set access for other user(s)
Add/Edit/Delete Customers
Employee

Figure 2.1: Use Case Diagram for IT administrator and employee.

Chapter 3: Architecture

3.1 System Architecture

The Customer Database Management System is intended to keep track of all customers of Cadmus Electronics and send SMS notifications to customers whose DSTV or Canal+ subscriptions are due (at least 3 days before) to remind them to pay at the nearest Cadmus branch. The architecture of the system consists of the database of all customers, the feature to extract and compile the phone numbers of selected customers, and the SMS API (Application Programming Interface) that will send an SMS notification to all selected customers. The system is built on a three-tier architecture client-server model, which consists of the client, the application server and the database server.

3.1.1 Client Side

The web application would be accessed using web browsers. The user would interact with the application via the web browser to communicate with the application server and database server.

3.1.2. Application Server

The application server is the software on which the web application runs. It is the part of the system that processes the application's content to be deployed to the client requesting it ("App Server," n.d.). It also serves as a connector to the database server, wherein it provides access to data stored in the database for retrieving and/or modification. For this project, the SMS feature will be implemented in the application server as well.

Extracting data from the database and automatically preparing a list of customers to send SMS notifications to will be done in this layer.

3.1.3 Database Server

The database server contains data needed by the application server to process and deliver to the client. It contains mechanisms for persisting data and ensuring that only authenticated users have root privileges to access and modify the data within it.

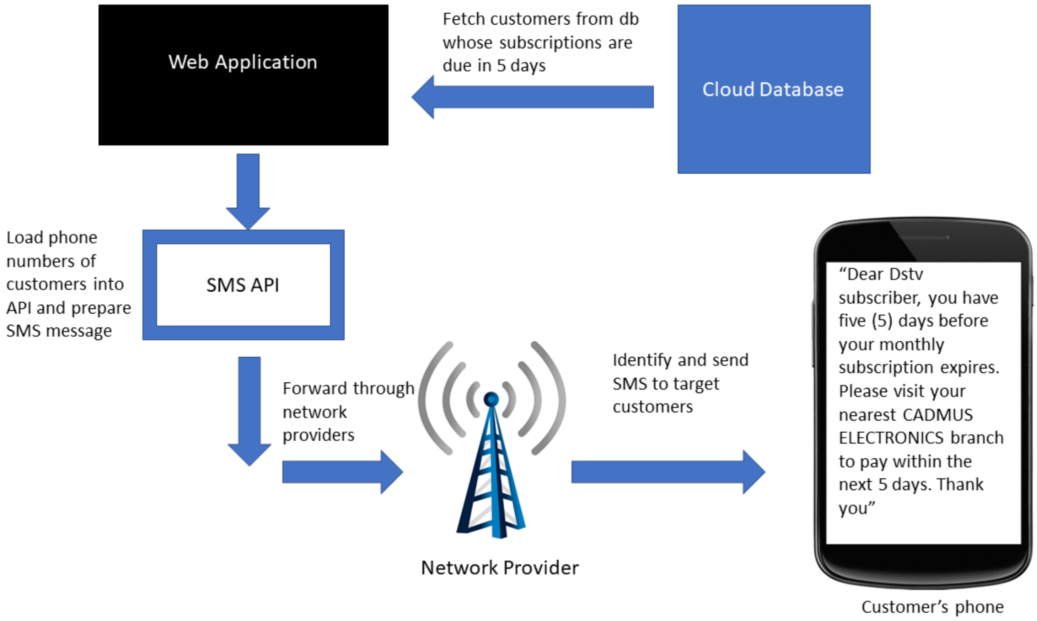

Web Application
Fetch customers from db whose subscriptions are due in 5 days
Cloud Database
Load phone numbers of customers into API and prepare SMS message
SMS API
Forward through network providers
Identify and send SMS to target customers
Network Provider
"Dear Dstv subscriber, you have five (5) days before your monthly subscription expires. Please visit your nearest CADMUS ELECTRONICS branch to pay within the next 5 days. Thank you"
Customer's phone

Figure 3.1: System architecture

The web application will be mainly used by the IT administrator regularly, to configure updates and monitor the automated SMS feature which includes checking how often SMS notifications are sent, and number of checks executed every day. It provides an interface where the administrator can easily view records of all customers from the database and configure the format of the SMS message to be sent to targeted customers. The secondary users of the application are the employees of Cadmus Electronics in various branches within the country. Employees will be given unique credentials with which they will use to be authenticated in order to use the system. The employees will use the application to add records of new and regular customers whose information are not yet captured by the company. This will expedite the process of accumulating new customer data for the application's database.

The SMS API will be used to compile selected customers' names and phone numbers from the database by the web application and send the SMS message notifying them to pay their subscription fees before their accounts are suspended. Because of monthly package subscription, the process of sending messages will be done at least few days before the end of the month, depending on the requirements of Cadmus Management team.

3.2 Tools and Frameworks

3.2.1 HTML

HTML (Hypertext Markup Language) is a computer language that allows users to create webpages in a website by manipulating tags and using hyperlinks, which can be accessed globally via the internet (Aronson, 1995, p. 3). The latest version of HTML being used currently is HTML5.

3.2.2 CSS

CSS (Cascading Style Sheets) is a styling language that is used to design the appearance of HTML elements on a webpage (Sklar, 2001). It helps to make websites look appealing to the eye. The Bootstrap template used in the implementation of this project comes with its own pre-defined CSS files.

3.2.3 JavaScript

JavaScript is a programming language with Object-Oriented functionalities that is normally used in web browsers to control interactions with users and change the appearance or behavior with HTML elements. It is normally referred to as a client-side programming language because it is normally run on client computers (computers that request services and resources from servers) (Flanagan, 2006). For this project, JavaScript was used both at the client-side and the server-side (thanks to Node.js, which will be explained later on).

3.2.4 Heroku

Heroku is a Platform-as-a-Service (PaaS) that allows developers to create and run web applications that are hosted on remote servers at a little to no cost, depending on the scale of the application (Solheim, 2015). Heroku supports multiple languages and frameworks including Node.js, a framework that enables JavaScript to be used for server-side scripting. Heroku was used to host the web application remotely, with a server URL provided for free. Heroku also provides a few resources called **addons**, which were used by the web application for specific features; Mailgun for email verification of new users created, Mongo DB’s mLab as a cloud database on which the Parse Server runs, PaperTrail for keeping logs of events and errors that occur on the server, and Scheduler for automating the sending of SMS notifications to customers according to a specified time schedule.

3.2.5 Gentellela Bootstrap Template

Gentellela Alela is a free Bootstrap template that provides an interactive and multi-layered dashboard interface integrated with many features and tools for administrative work. The template was built by Colorlib, a group of WordPress theme developers ("20 Free Bootstrap Admin Dashboard Templates 2018," 2018).

3.2.6 Parse Server

The Parse Platform is an open-source Backend-as-a-Service that provides backend resources (e.g SDKs and APIs) for developers to utilize without struggling to build a backend server from scratch ("Parse + Open Source," n.d.). Parse was once owned by Facebook but was shut down early 2017 (Lardinois & Constine, n.d.), after migrating to MongoDB, an open-source database program and now runs on mLab, its cloud database server which Heroku supports. It now runs as an open-source project. Parse also provides support for Node.js and JavaScript server-side scripting.

3.2.7 Node.js

Node.js is a JavaScript Framework that enables users to write JavaScript programs to be executed on the server. Node.js supports asynchronous programming; it can run processes without having to wait for results immediately, allowing it to run other processes and when results of a function are ready, they are accessed and displayed accordingly (Tilkov & Vinoski, 2010). Node.js enables the developer to install and integrate the Parse SDK (Software Development Kit) into this project, as well as other tools and libraries such as the Parse Dashboard, Heroku tools including mLab (MongoDB's web-based Database-as-a-Service) and Hubtel's SMS API ("parse," n.d.).

3.2.8 Hubtel

Hubtel is a web service portal that provides tools to help connect businesses with their customers effectively. Hubtel provides developers with resources such as Short Message Peer-to-Peer (SMPP), Short Message Services (SMS) and Mobile Money, which can be integrated into mobile and web applications ("Learning The Hubtel Basics," n.d.). Hubtel's SMS API was used for this project to send SMS notifications to DSTV subscribers.

3.3 Sequence Diagram and Database Schema

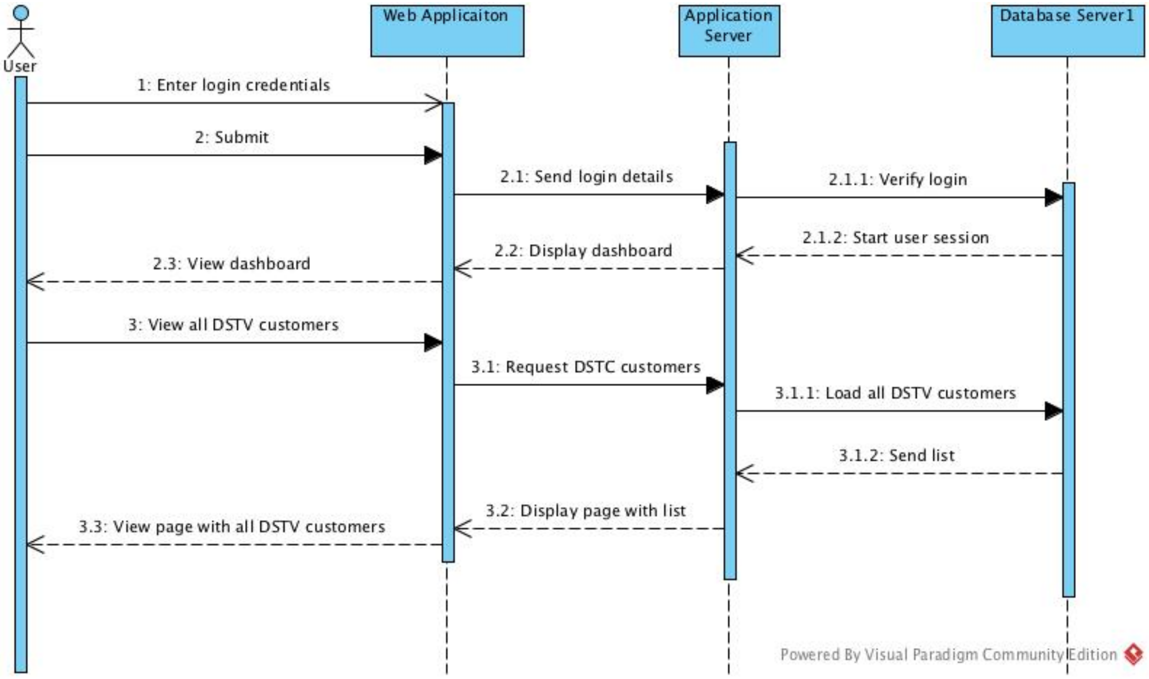
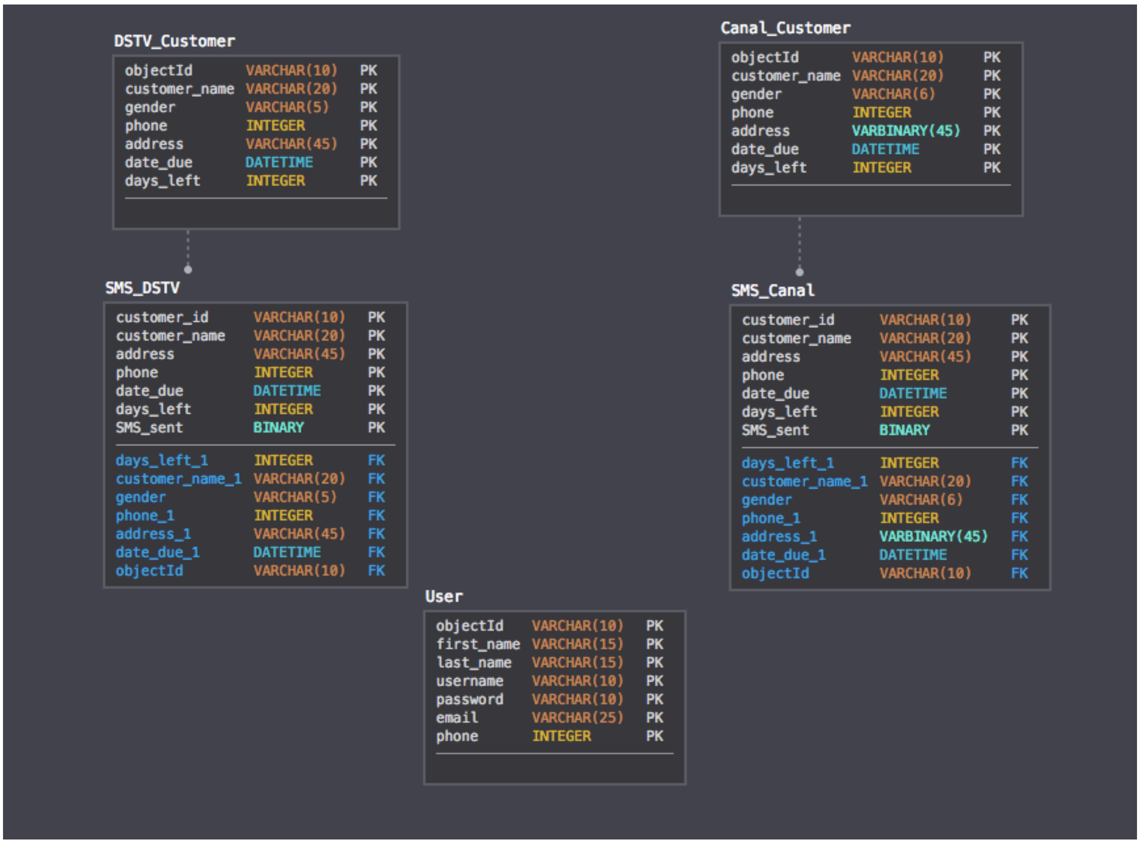

User
Web Applicaiton
Application Server
Database Server1
1: Enter login credentials
2: Submit
2.1: Send login details
2.1.1: Verify login
2.1.2: Start user session
2.2: Display dashboard
2.3: View dashboard
3: View all DSTV customers
3.1: Request DSTC customers
3.1.1: Load all DSTV customers
3.1.2: Send list
3.2: Display page with list
3.3: View page with all DSTV customers
Powered By Visual Paradigm Community Edition

Figure 3.2: Sequence diagram



DSTV_Customer

objectId	VARCHAR(10)	PK
customer_name	VARCHAR(20)	PK
gender	VARCHAR(5)	PK
phone	INTEGER	PK
address	VARCHAR(45)	PK
date_due	DATETIME	PK
days_left	INTEGER	PK

Canal_Customer

objectId	VARCHAR(10)	PK
customer_name	VARCHAR(20)	PK
gender	VARCHAR(6)	PK
phone	INTEGER	PK
address	VARBINARY(45)	PK
date_due	DATETIME	PK
days_left	INTEGER	PK

SMS_DSTV

customer_id	VARCHAR(10)	PK
customer_name	VARCHAR(20)	PK
address	VARCHAR(45)	PK
phone	INTEGER	PK
date_due	DATETIME	PK
days_left	INTEGER	PK
SMS_sent	BINARY	PK
days_left_1	INTEGER	FK
customer_name_1	VARCHAR(20)	FK
gender	VARCHAR(5)	FK
phone_1	INTEGER	FK
address_1	VARCHAR(45)	FK
date_due_1	DATETIME	FK
objectId	VARCHAR(10)	FK

SMS_Canal

customer_id	VARCHAR(10)	PK
customer_name	VARCHAR(20)	PK
address	VARCHAR(45)	PK
phone	INTEGER	PK
date_due	DATETIME	PK
days_left	INTEGER	PK
SMS_sent	BINARY	PK
days_left_1	INTEGER	FK
customer_name_1	VARCHAR(20)	FK
gender	VARCHAR(6)	FK
phone_1	INTEGER	FK
address_1	VARBINARY(45)	FK
date_due_1	DATETIME	FK
objectId	VARCHAR(10)	FK

User

objectId	VARCHAR(10)	PK
first_name	VARCHAR(15)	PK
last_name	VARCHAR(15)	PK
username	VARCHAR(10)	PK
password	VARCHAR(10)	PK
email	VARCHAR(25)	PK
phone	INTEGER	PK

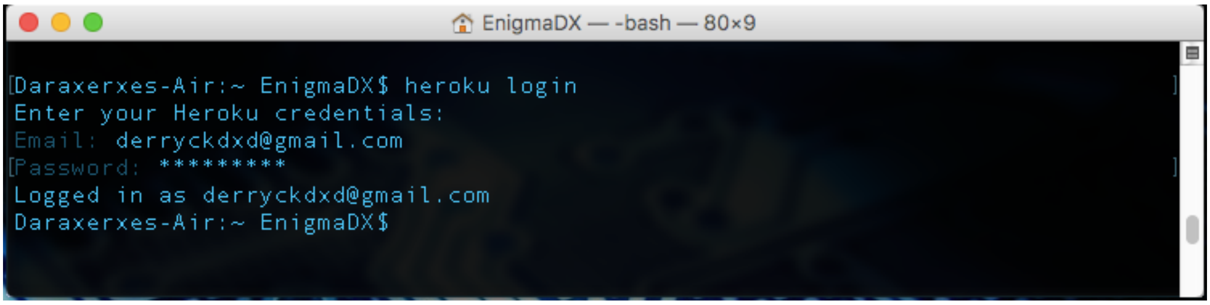
Figure 3.3: Database schema

Chapter 4: Implementation

Majority of this project's implementation was carried out with Heroku; from development and hosting on a remote server, to providing resources to support Parse Server (including the Parse Dashboard), Node.js, and MondoDB's mLab; a web-based database that Heroku supports. The SMS Messaging aspect of the project was done using Hubtel's SMS API called SMSGH. The "Gentellela" Bootstrap template provided its own HTML, CSS and JavaScript files which were edited to suit the needs of the project. This chapter provides evidence of implementation of the aforementioned tools and frameworks along with sufficient description.

4.1 Configuring Heroku and Node.js on local machine

After installing Node.js on the developer's machine, Heroku's Command Line Interface (CLI) was installed to create and manage the web application from the computer's terminal.



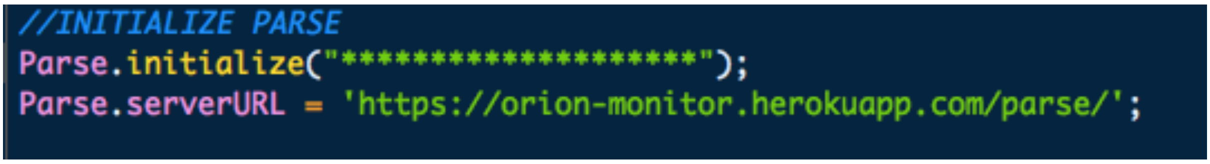
```
Daraxerxes-Air:~ EnigmaDX$ heroku login
Enter your Heroku credentials:
Email: derryckdxd@gmail.com
Password: *********
Logged in as derryckdxd@gmail.com
Daraxerxes-Air:~ EnigmaDX$
```

Figure 4.1: Heroku CLI login via terminal.

On Heroku's website, the application is created and given a name. then using the Git feature which Heroku's CLI supports, the application is cloned to the developer's computer, which contains several files and folders provided by Node.js for web server configuration.

The main configuration file in the project is the index.js file, which contains JavaScript code used to configure the application to run with Node.js and Parse. A sample Parse Serve code is cloned from GitHub and merged with the cloned application.

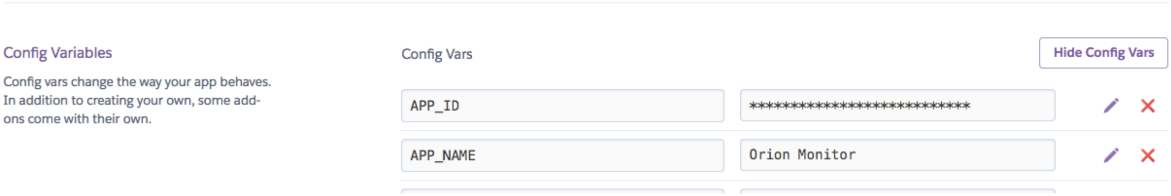
4.2: Activating Web Application with Parse



```
//INITIALIZE PARSE
Parse.initialize("********************");
Parse.serverURL = 'https://orion-monitor.herokuapp.com/parse/';
```

Figure 4.2: Code to initialize Heroku application running on Parse Server

The image above indicates the key code in the index.js file required to activate and run the web application running on the Parse server. The asterix (*) represents an alphanumeric string which is a randomly generated key that is assigned to the web application. The 'serverURL' is the URL of the web application that can be accessed in a web browser. These configuration settings must correspond with the settings in the application on Heroku's website.



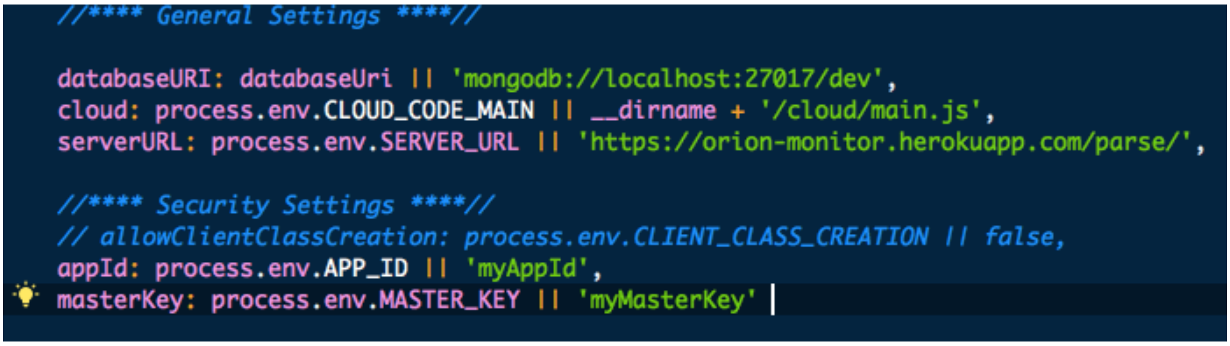
Config Variables

Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.

Config Vars

APP_ID	***************************
APP_NAME	Orion Monitor

Figure 4.3: Configuration on Heroku's website

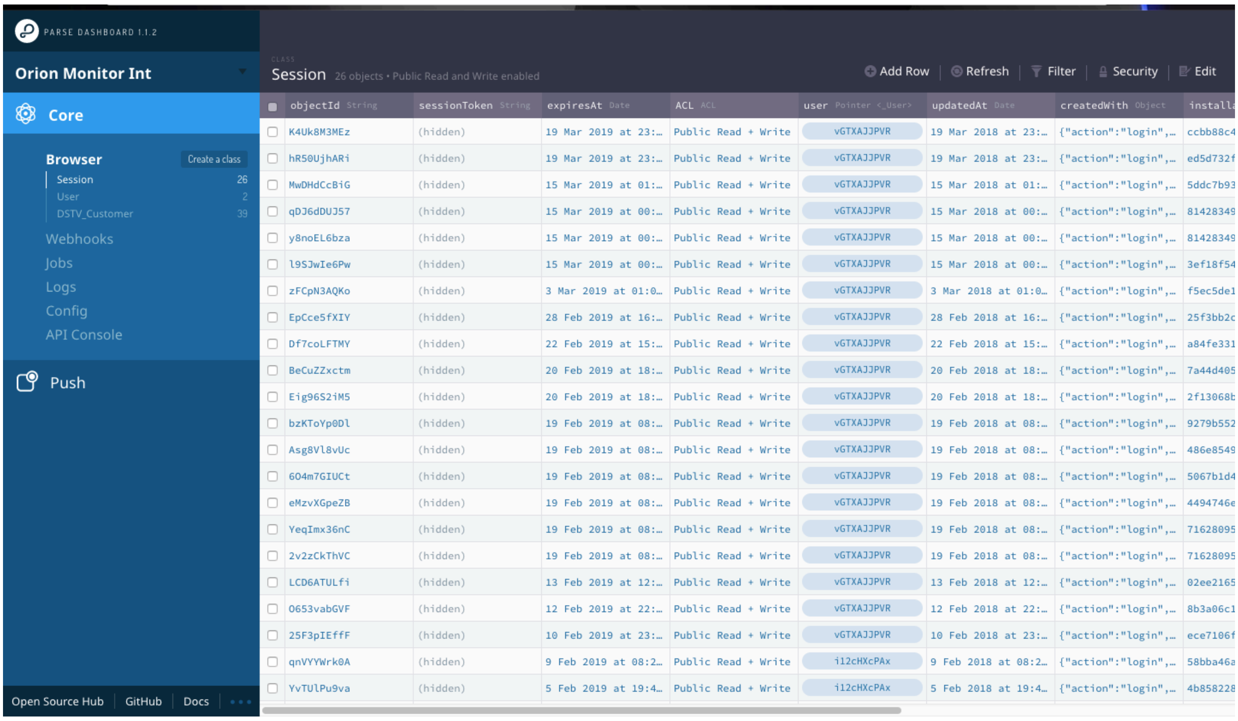


```
//**** General Settings ****//

databaseURI: databaseUri || 'mongodb://localhost:27017/dev',
cloud: process.env.CLOUD_CODE_MAIN || __dirname + '/cloud/main.js',
serverURL: process.env.SERVER_URL || 'https://orion-monitor.herokuapp.com/parse/',

//**** Security Settings ****//
// allowClientClassCreation: process.env.CLIENT_CLASS_CREATION || false,
appId: process.env.APP_ID || 'myAppId',
masterKey: process.env.MASTER_KEY || 'myMasterKey'
```

Figure 4.4: Configuration of application from Heroku

4.3 Installing Parse Dashboard

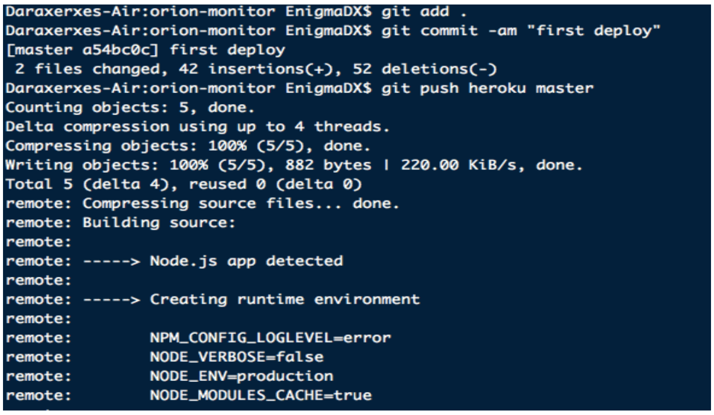
After configuring the application to correspond with Heroku, the Node Package Manager (npm) provided by Node.js, is used to install other libraries and tools relevant for the application. The first tool to download is the Parse Server which runs on MongoDB's mLab. The next tool to download is the Parse Dashboard which can be installed and run locally on a computer (*parse-dashboard* (GitHub)). It provides a holistic view of applications that are running on the Parse Server and gives details of such apps including Sessions (number of Sign ups and Logins), Parse Users, Classes created, and also gives the administrator full access to create, rename, edit and delete classes and users. The Parse Dashboard can be accessed once installed on a computer via the link *http://localhost:4040*. In this project, there is one class created initially; *DSTV_Customer*, which has the properties of a customer.



□	objectId String	sessionToken String	expiresAt Date	ACL ACL	user Pointer <_User>	updatedAt Date	createdWith Object	installa
□	K4Uk8M3MEz	(hidden)	19 Mar 2019 at 23:…	Public Read + Write	vGTXAJJPVR	19 Mar 2018 at 23:…	{"action":"login",…	ccbb88c4
□	hR50UjhARi	(hidden)	19 Mar 2019 at 23:…	Public Read + Write	vGTXAJJPVR	19 Mar 2018 at 23:…	{"action":"login",…	ed5d732f
□	MwDHdCcBiG	(hidden)	15 Mar 2019 at 01:…	Public Read + Write	vGTXAJJPVR	15 Mar 2018 at 01:…	{"action":"login",…	5ddc7b93
□	qDJ6dDUJ57	(hidden)	15 Mar 2019 at 00:…	Public Read + Write	vGTXAJJPVR	15 Mar 2018 at 00:…	{"action":"login",…	81428349
□	y8noEL6bza	(hidden)	15 Mar 2019 at 00:…	Public Read + Write	vGTXAJJPVR	15 Mar 2018 at 00:…	{"action":"login",…	81428349
□	l9SJwIe6Pw	(hidden)	15 Mar 2019 at 00:…	Public Read + Write	vGTXAJJPVR	15 Mar 2018 at 00:…	{"action":"login",…	3ef18f54
□	zFCpN3AQKo	(hidden)	3 Mar 2019 at 01:0…	Public Read + Write	vGTXAJJPVR	3 Mar 2018 at 01:0…	{"action":"login",…	f5ec5de1
□	EpCce5fXIY	(hidden)	28 Feb 2019 at 16:…	Public Read + Write	vGTXAJJPVR	28 Feb 2018 at 16:…	{"action":"login",…	25f3bb2c
□	Df7coLFTMY	(hidden)	22 Feb 2019 at 15:…	Public Read + Write	vGTXAJJPVR	22 Feb 2018 at 15:…	{"action":"login",…	a84fe331
□	BeCuZZxctm	(hidden)	20 Feb 2019 at 18:…	Public Read + Write	vGTXAJJPVR	20 Feb 2018 at 18:…	{"action":"login",…	7a44d405
□	Eig96S2iM5	(hidden)	20 Feb 2019 at 18:…	Public Read + Write	vGTXAJJPVR	20 Feb 2018 at 18:…	{"action":"login",…	2f13068b
□	bzKToYp0Dl	(hidden)	19 Feb 2019 at 08:…	Public Read + Write	vGTXAJJPVR	19 Feb 2018 at 08:…	{"action":"login",…	9279b552
□	Asg8Vl8vUc	(hidden)	19 Feb 2019 at 08:…	Public Read + Write	vGTXAJJPVR	19 Feb 2018 at 08:…	{"action":"login",…	486e8549
□	6O4m7GIUCt	(hidden)	19 Feb 2019 at 08:…	Public Read + Write	vGTXAJJPVR	19 Feb 2018 at 08:…	{"action":"login",…	5067b1d4
□	eMzvXGpeZB	(hidden)	19 Feb 2019 at 08:…	Public Read + Write	vGTXAJJPVR	19 Feb 2018 at 08:…	{"action":"login",…	4494746e
□	YeqImx36nC	(hidden)	19 Feb 2019 at 08:…	Public Read + Write	vGTXAJJPVR	19 Feb 2018 at 08:…	{"action":"login",…	71628095
□	2v2zCkThVC	(hidden)	19 Feb 2019 at 08:…	Public Read + Write	vGTXAJJPVR	19 Feb 2018 at 08:…	{"action":"login",…	71628095
□	LCD6ATULfi	(hidden)	13 Feb 2019 at 12:…	Public Read + Write	vGTXAJJPVR	13 Feb 2018 at 12:…	{"action":"login",…	02ee2165
□	O653vabGVF	(hidden)	12 Feb 2019 at 22:…	Public Read + Write	vGTXAJJPVR	12 Feb 2018 at 22:…	{"action":"login",…	8b3a06c1
□	25F3pIEffF	(hidden)	10 Feb 2019 at 23:…	Public Read + Write	vGTXAJJPVR	10 Feb 2018 at 23:…	{"action":"login",…	ece7106f
□	qnVYYWrk0A	(hidden)	9 Feb 2019 at 08:2…	Public Read + Write	i12cHXcPAx	9 Feb 2018 at 08:2…	{"action":"login",…	58bba46a
□	YvTUlPu9va	(hidden)	5 Feb 2019 at 19:4…	Public Read + Write	i12cHXcPAx	5 Feb 2018 at 19:4…	{"action":"login",…	4b858228

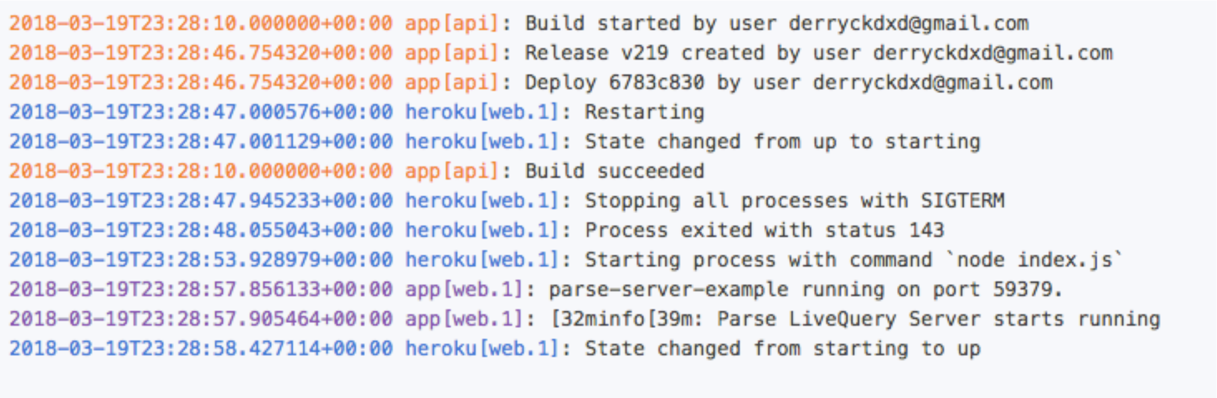
Figure 4.5: Parse Dashboard running locally.

To deploy the application to run on Heroku's server, the Git functionality is used with the CLI to push the applications folders and files to the server. Using Heroku's PaperTrail addon, the administrator can view the logs in Heroku to read the background processes running in the web application and be informed when the application has been deployed or not.



```
Daraxerxes-Air:orion-monitor EnigmaDX$ git add .
Daraxerxes-Air:orion-monitor EnigmaDX$ git commit -am "first deploy"
[master a54bc0c] first deploy
 2 files changed, 42 insertions(+), 52 deletions(-)
Daraxerxes-Air:orion-monitor EnigmaDX$ git push heroku master
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 882 bytes | 220.00 KiB/s, done.
Total 5 (delta 4), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Node.js app detected
remote:
remote: -----> Creating runtime environment
remote:
remote:        NPM_CONFIG_LOGLEVEL=error
remote:        NODE_VERBOSE=false
remote:        NODE_ENV=production
remote:        NODE_MODULES_CACHE=true
```

Figure 4.6: Snippet logs showing application being pushed to server for deployment.



```
2018-03-19T23:28:10.000000+00:00 app[api]: Build started by user derryckdxd@gmail.com
2018-03-19T23:28:46.754320+00:00 app[api]: Release v219 created by user derryckdxd@gmail.com
2018-03-19T23:28:46.754320+00:00 app[api]: Deploy 6783c830 by user derryckdxd@gmail.com
2018-03-19T23:28:47.000576+00:00 heroku[web.1]: Restarting
2018-03-19T23:28:47.001129+00:00 heroku[web.1]: State changed from up to starting
2018-03-19T23:28:10.000000+00:00 app[api]: Build succeeded
2018-03-19T23:28:47.945233+00:00 heroku[web.1]: Stopping all processes with SIGTERM
2018-03-19T23:28:48.055043+00:00 heroku[web.1]: Process exited with status 143
2018-03-19T23:28:53.928979+00:00 heroku[web.1]: Starting process with command `node index.js`
2018-03-19T23:28:57.856133+00:00 app[web.1]: parse-server-example running on port 59379.
2018-03-19T23:28:57.905464+00:00 app[web.1]: [32minfo[39m: Parse LiveQuery Server starts running
2018-03-19T23:28:58.427114+00:00 heroku[web.1]: State changed from starting to up
```

Figure 4.7: View from Heroku logs confirming that application is running.

4.4 Installing Heroku addons

For the OrionMonitor web application, four (4) resources, called addons provided by Heroku were installed. Each resource has an initial free package, which can be upgraded for a monthly cost depending on the scale and needs of the application.

4.4.1 mLab

The first addon to be installed is mLab, which is a cloud database service provider that hosts databases provided by MongoDB. This serves as the cloud database of the web application.

4.4.2 PaperTrail

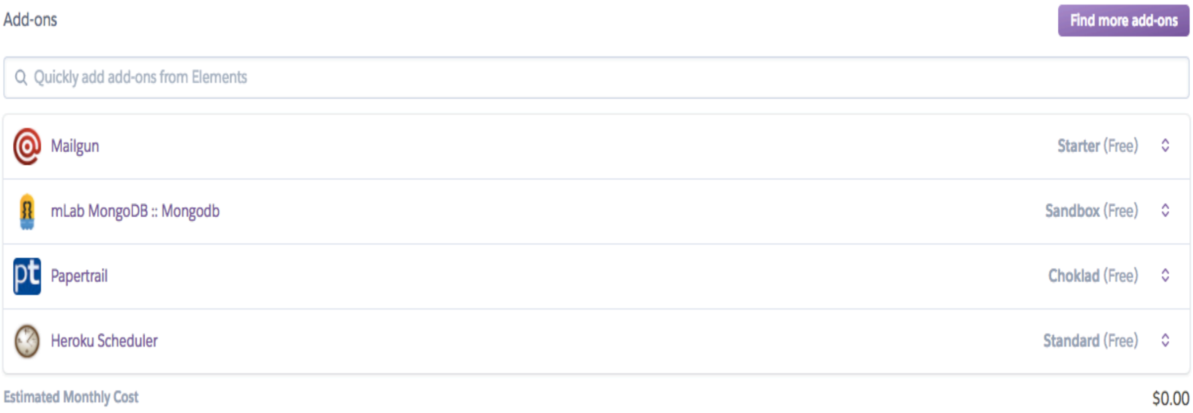
PaperTrail is a tool used to track the background processes and statuses of the application and makes it easier to locate problems and fix errors. It was used in the debugging process as OrionMonitor was being built.

4.4.3 Mailgun

Mailgun is an addon that is used by the web application to send emails to clients or customers, per the developer's choice. For an employee to sign up as a user for the application, an email would have to be sent to the head administrator for confirmation before granting the employee a user ID which will be used to login and add new customers to the database.

4.4.4 Scheduler

The Heroku Scheduler addon is used by OrionMonitor to run the SMS notification process on a daily basis. Each day, the application will check its database for a list of customers whose monthly subscriptions are due in less than a specific number of days (3 in this case), and then send them an SMS notification reminding them to pay their bills at the nearest Cadmus Electronics branch before the due date.

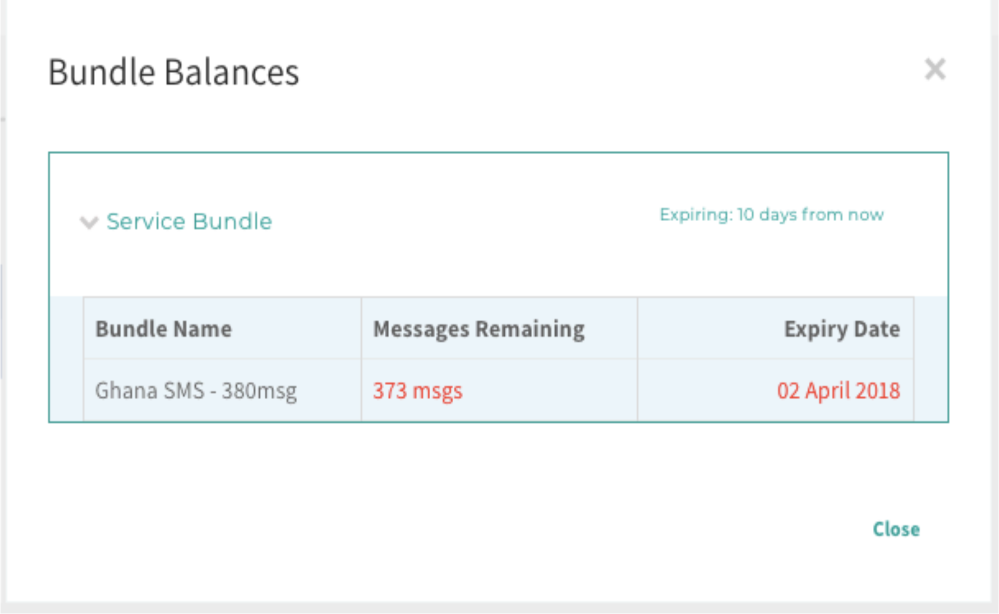


Mailgun	Starter (Free)
mLab MongoDB :: Mongodb	Sandbox (Free)
Papertrail	Choklad (Free)
Heroku Scheduler	Standard (Free)
Estimated Monthly Cost	$0.00

Figure 4.8: List of Heroku addons used by OrionMonitor.

4.4 Integrating Hubtel's SMS API

Hubtel's SMS API is integrated into OrionMonitor by first installing the SMSGH package using npm. Hubtel provides a variety of bundles to purchase monthly in order to send SMS messages. For this project, a monthly fee of GHC10 was paid for 380 SMS messages to be sent using OrionMonitor.



Bundle Balances

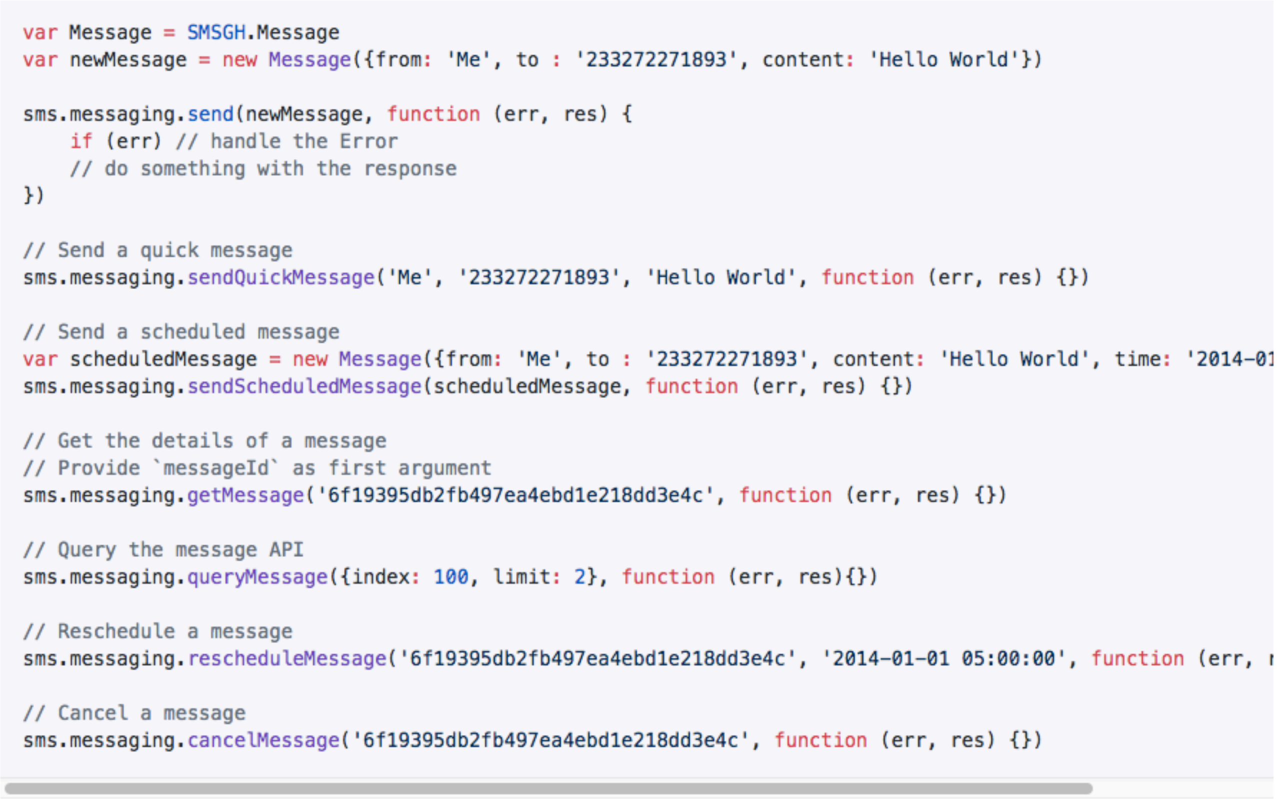
Service Bundle

Expiring: 10 days from now

Bundle Name	Messages Remaining	Expiry Date
Ghana SMS - 380msg	373 msgs	02 April 2018

Figure 4.9: Details of Hubtel's SMS messaging bundle for the month of March, 2018.

To implement the SMS API, a section of code was copied from SMSGH's source code on GitHub, which contains a variety of methods for sending SMS messages.

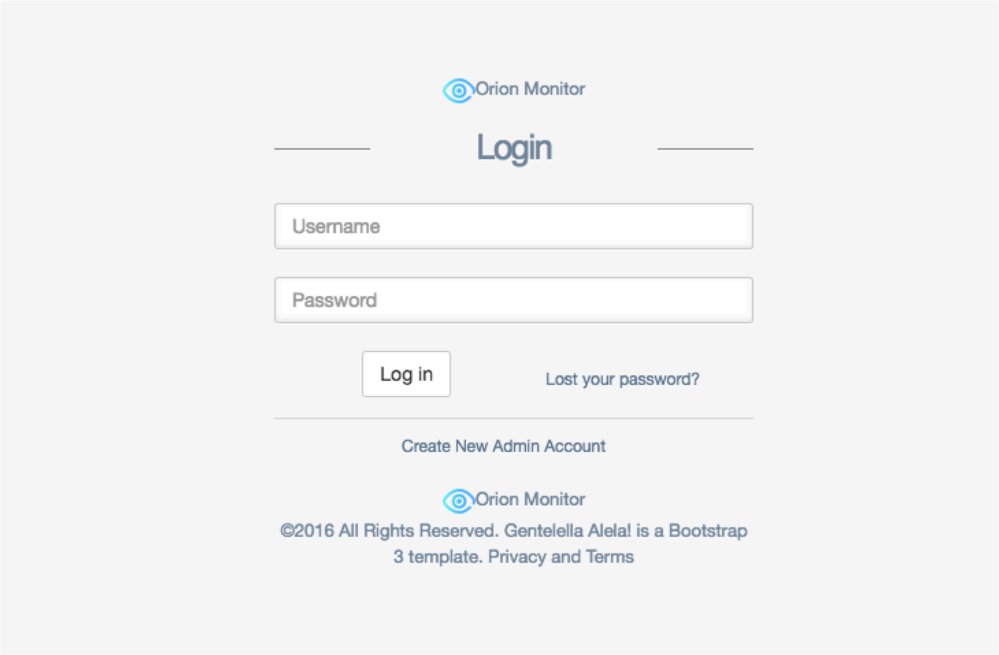


```
var Message = SMSGH.Message
var newMessage = new Message({from: 'Me', to : '233272271893', content: 'Hello World'})

sms.messaging.send(newMessage, function (err, res) {
    if (err) // handle the Error
    // do something with the response
})

// Send a quick message
sms.messaging.sendQuickMessage('Me', '233272271893', 'Hello World', function (err, res) {})

// Send a scheduled message
var scheduledMessage = new Message({from: 'Me', to : '233272271893', content: 'Hello World', time: '2014-01
sms.messaging.sendScheduledMessage(scheduledMessage, function (err, res) {})

// Get the details of a message
// Provide `messageId` as first argument
sms.messaging.getMessage('6f19395db2fb497ea4ebd1e218dd3e4c', function (err, res) {})

// Query the message API
sms.messaging.queryMessage({index: 100, limit: 2}, function (err, res){})

// Reschedule a message
sms.messaging.rescheduleMessage('6f19395db2fb497ea4ebd1e218dd3e4c', '2014-01-01 05:00:00', function (err, r

// Cancel a message
sms.messaging.cancelMessage('6f19395db2fb497ea4ebd1e218dd3e4c', function (err, res) {})
```

Figure 4.10: SMSGH source code from GitHub.

4.5 Web Interface

Once OrionMonitor is opened on the web, the initial page that loads by default is the Login page which prompts the user to enter his/her credentials to login and view the dashboard. There is also an option to create a new admin user for the application, which can only be approved of by sending a request to an existing administrator's email address.



Orion Monitor

Login

Username

Password

Log in

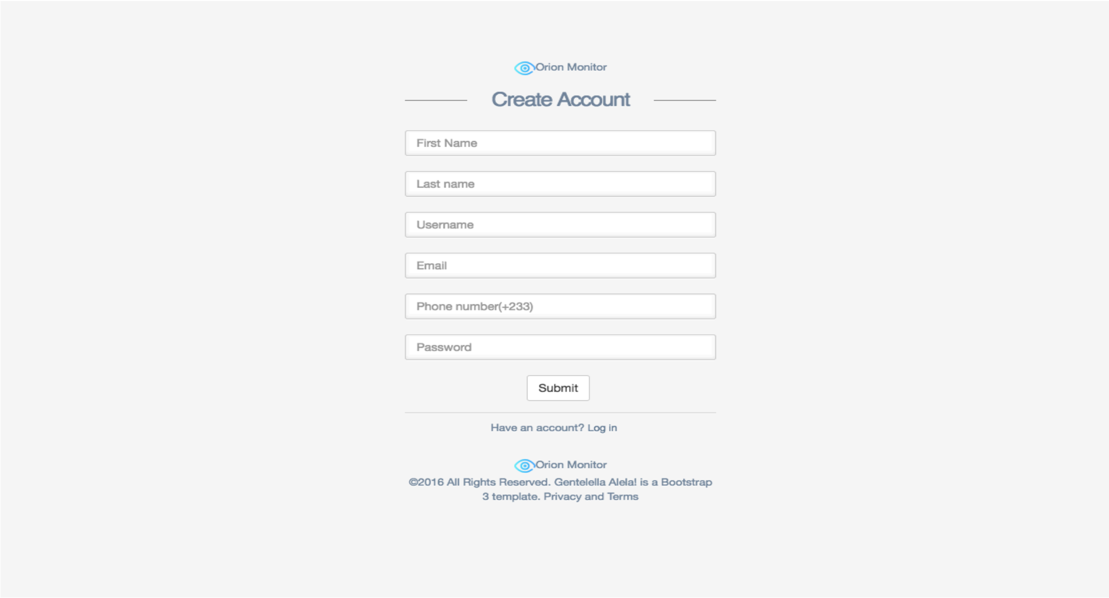
Lost your password?

Create New Admin Account

Orion Monitor

©2016 All Rights Reserved. Gentelella Alela! is a Bootstrap 3 template. Privacy and Terms

Figure 4.11: Login page



Orion Monitor

Create Account

First Name

Last name

Username

Email

Phone number(+233)

Password

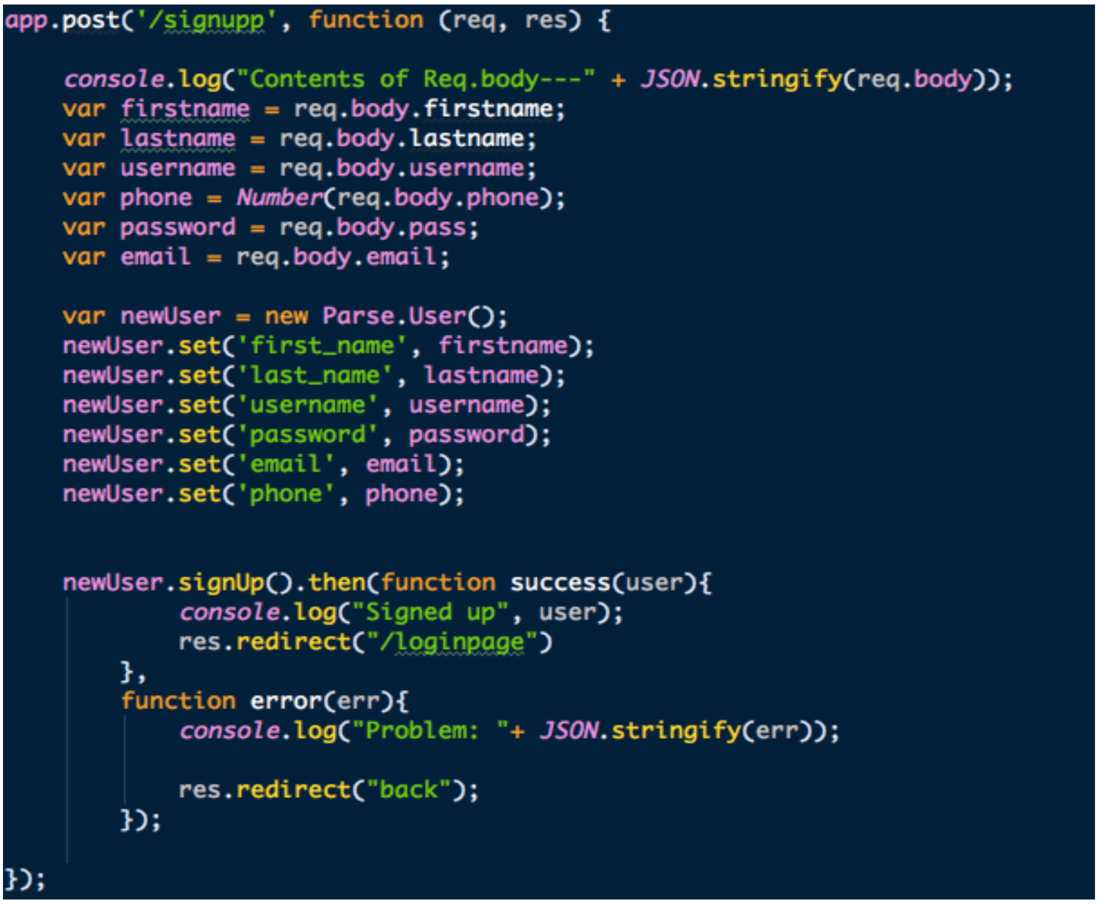
Submit

Have an account? Log in

Orion Monitor

©2016 All Rights Reserved. Gentelella Alela! is a Bootstrap 3 template. Privacy and Terms

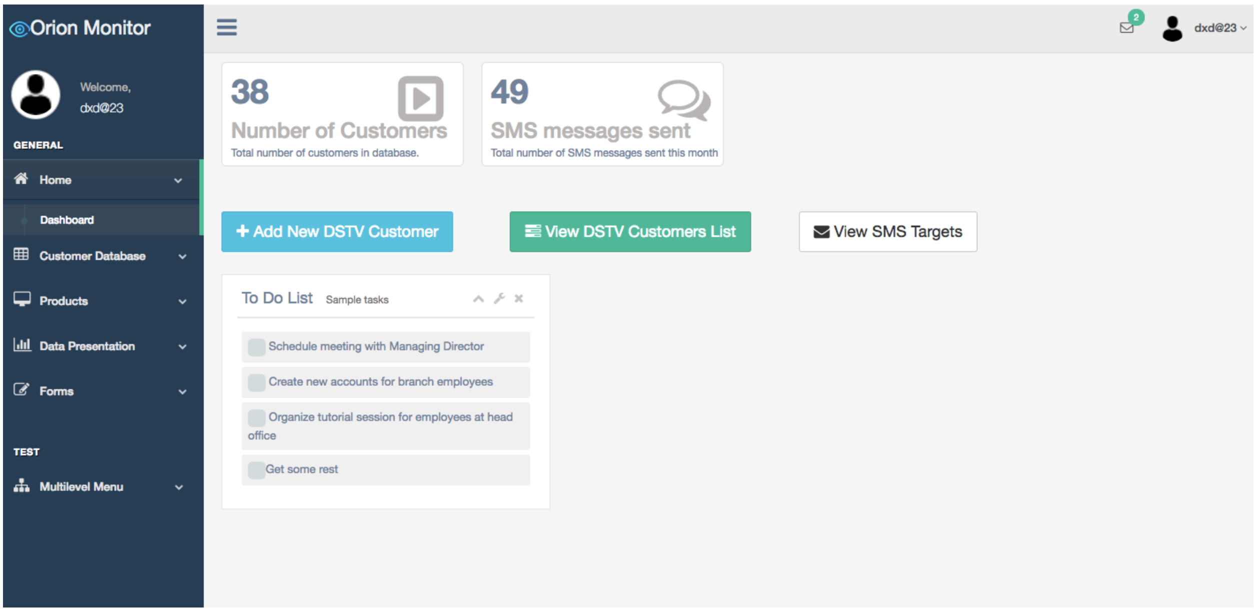
Figure 4.12: Sign up page



```
app.post('/signupp', function (req, res) {

    console.log("Contents of Req.body---" + JSON.stringify(req.body));
    var firstname = req.body.firstname;
    var lastname = req.body.lastname;
    var username = req.body.username;
    var phone = Number(req.body.phone);
    var password = req.body.pass;
    var email = req.body.email;

    var newUser = new Parse.User();
    newUser.set('first_name', firstname);
    newUser.set('last_name', lastname);
    newUser.set('username', username);
    newUser.set('password', password);
    newUser.set('email', email);
    newUser.set('phone', phone);


    newUser.signUp().then(function success(user){
            console.log("Signed up", user);
            res.redirect("/loginpage")
        },
        function error(err){
            console.log("Problem: "+ JSON.stringify(err));

            res.redirect("back");
        });

});
```

Figure 4.13: Code demonstrating creation of a Parse User from sign up form

After a new user completes the form, the fields in the form are parsed to the application server and are used to create a new Parse User, as shown in the code above. After the user is created, he/she is redirected to the login page to be authenticated before proceeding as a new user. After the new user is created, the administrator will assign him/her with an access level that will determine the level of privileges the user will have access to. Once login is completed, the user is assigned a session which grants him/her access to other pages without having to login again. The user is then redirected to the main dashboard where he/she can perform necessary administrative duties.



38
Number of Customers
Total number of customers in database.

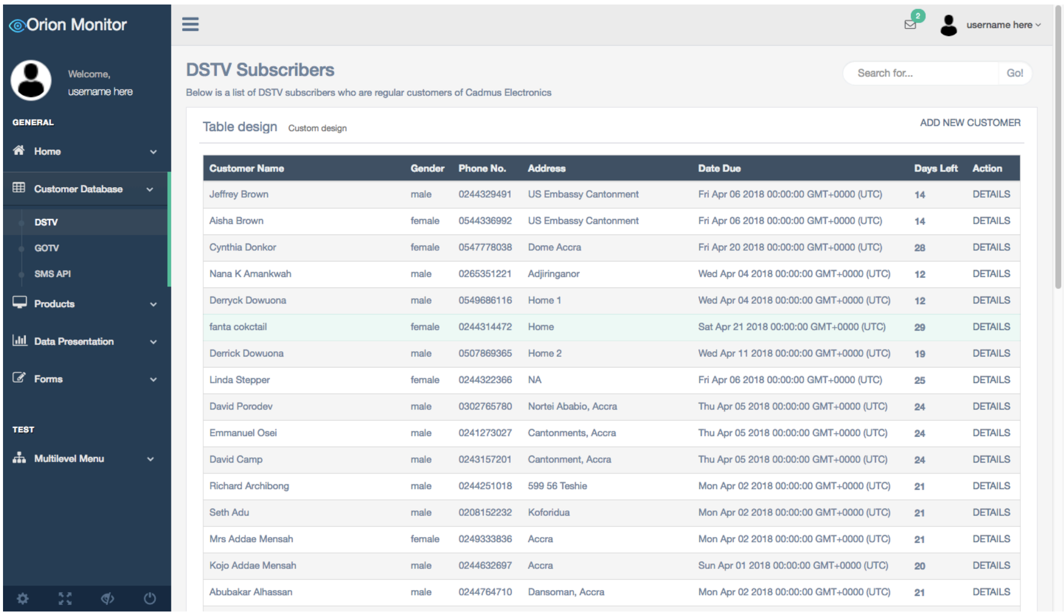
49
SMS messages sent
Total number of SMS messages sent this month

To Do List Sample tasks

- Schedule meeting with Managing Director
- Create new accounts for branch employees
- Organize tutorial session for employees at head office
- Get some rest

Figure 4.14: Dashboard view after successful login

An overview of the dashboard displays the total number of DSTV customers, the total amount of Hubtel's SMS bundle, and the total number of SMS messages sent within the week so far. The page that shows the list of DSTV customers with their details: full name, address, phone number, gender, date monthly subscription is due, and number of days left.



DSTV Subscribers

Below is a list of DSTV subscribers who are regular customers of Cadmus Electronics

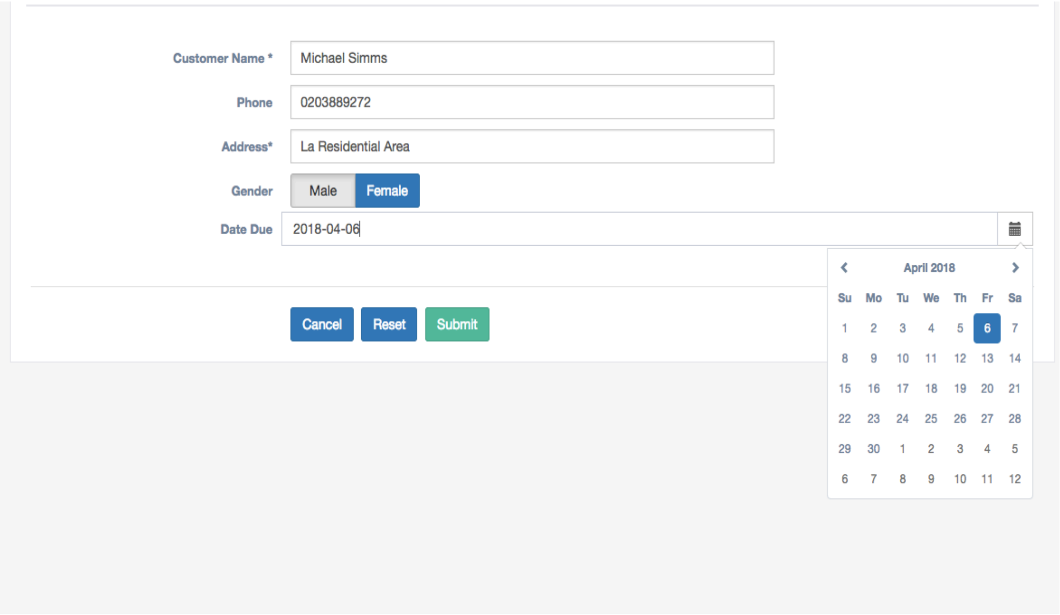
Table design Custom design

Customer Name	Gender	Phone No.	Address	Date Due	Days Left	Action
Jeffrey Brown	male	0244329491	US Embassy Cantonment	Fri Apr 06 2018 00:00:00 GMT+0000 (UTC)	14	DETAILS
Aisha Brown	female	0544336992	US Embassy Cantonment	Fri Apr 06 2018 00:00:00 GMT+0000 (UTC)	14	DETAILS
Cynthia Donkor	female	0547778038	Dome Accra	Fri Apr 20 2018 00:00:00 GMT+0000 (UTC)	28	DETAILS
Nana K Amankwah	male	0265351221	Adjiringanor	Wed Apr 04 2018 00:00:00 GMT+0000 (UTC)	12	DETAILS
Derryck Dowuona	male	0549686116	Home 1	Wed Apr 04 2018 00:00:00 GMT+0000 (UTC)	12	DETAILS
fanta cokctail	female	0244314472	Home	Sat Apr 21 2018 00:00:00 GMT+0000 (UTC)	29	DETAILS
Derrick Dowuona	male	0507869365	Home 2	Wed Apr 11 2018 00:00:00 GMT+0000 (UTC)	19	DETAILS
Linda Stepper	female	0244322366	NA	Fri Apr 06 2018 00:00:00 GMT+0000 (UTC)	25	DETAILS
David Porodev	male	0302765780	Nortei Ababio, Accra	Thu Apr 05 2018 00:00:00 GMT+0000 (UTC)	24	DETAILS
Emmanuel Osei	male	0241273027	Cantonments, Accra	Thu Apr 05 2018 00:00:00 GMT+0000 (UTC)	24	DETAILS
David Camp	male	0243157201	Cantonment, Accra	Thu Apr 05 2018 00:00:00 GMT+0000 (UTC)	24	DETAILS
Richard Archibong	male	0244251018	599 56 Teshie	Mon Apr 02 2018 00:00:00 GMT+0000 (UTC)	21	DETAILS
Seth Adu	male	0208152232	Koforidua	Mon Apr 02 2018 00:00:00 GMT+0000 (UTC)	21	DETAILS
Mrs Addae Mensah	female	0249333836	Accra	Mon Apr 02 2018 00:00:00 GMT+0000 (UTC)	21	DETAILS
Kojo Addae Mensah	male	0244632697	Accra	Sun Apr 01 2018 00:00:00 GMT+0000 (UTC)	20	DETAILS
Abubakar Alhassan	male	0244764710	Dansoman, Accra	Mon Apr 02 2018 00:00:00 GMT+0000 (UTC)	21	DETAILS

Figure 4.15: List of DSTV customers with their details

4.6 Adding New DSTV Customer

OrionMonitor allows users to add new customers to its database. It has a page that provides a form to allow the administrator/user to add a new DSTV customer to the database. Here, the employee at a particular Cadmus branch can fill in the details to add a new customer to the database. Key details include name, phone number and date on which account is due to be suspended. After filling in all fields and saving, a new user is created, and the number of days left to the customer's account suspension is calculated by taking the current date's time and subtracting it from the due date's time.

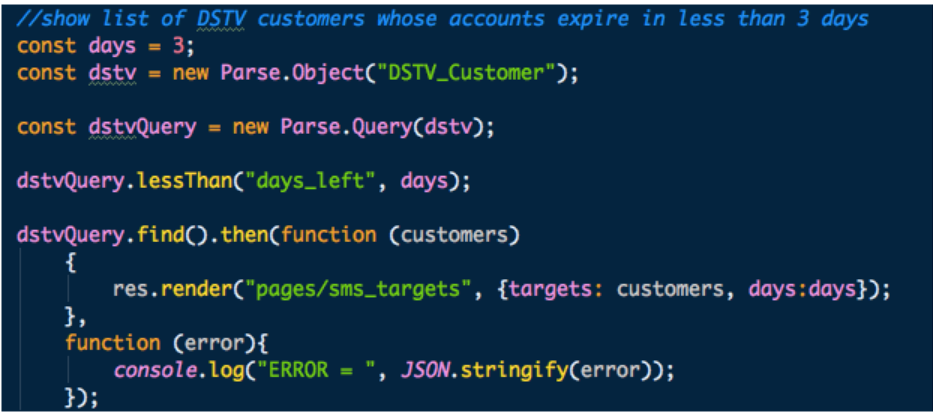


Customer Name *	Michael Simms
Phone	0203889272
Address*	La Residential Area
Gender	Male [✓ Female]
Date Due	2018-04-06

Figure 4.16: Form for adding new DSTV customer.

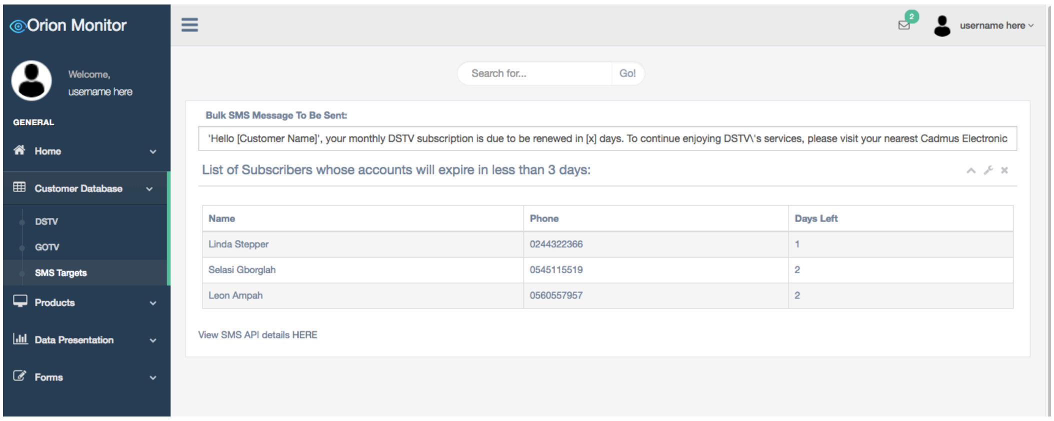
4.7 SMS Targets

The SMS Targets page shows a list of customers whose monthly subscriptions are due to expire in a specific number of days. The page also has a text field which contains the sample SMS message to be broadcasted to the list of customers, which can be edited as needed by the administrator.



```
//show list of DSTV customers whose accounts expire in less than 3 days
const days = 3;
const dstv = new Parse.Object("DSTV_Customer");

const dstvQuery = new Parse.Query(dstv);

dstvQuery.lessThan("days_left", days);

dstvQuery.find().then(function (customers)
    {
        res.render("pages/sms_targets", {targets: customers, days:days});
    },
    function (error){
        console.log("ERROR = ", JSON.stringify(error));
    });
```

Figure 4.17: Parse Query of DSTV customers whose accounts are due in less than 3 days.



Bulk SMS Message To Be Sent:

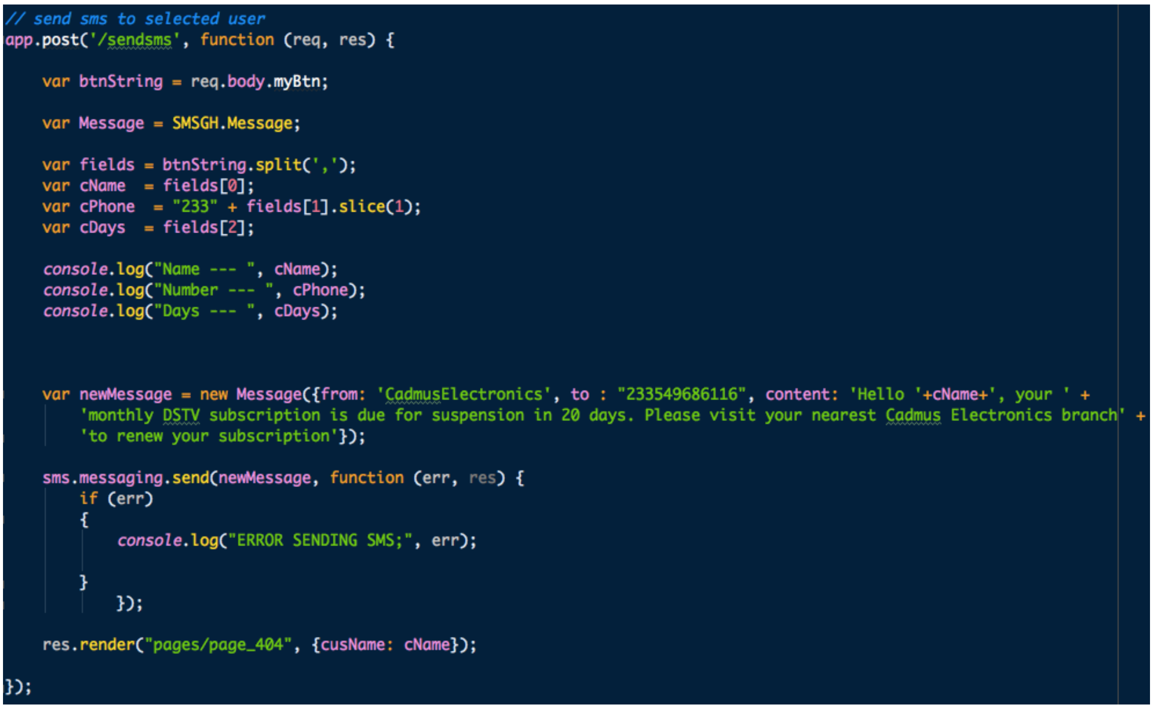
'Hello [Customer Name]', your monthly DSTV subscription is due to be renewed in [x] days. To continue enjoying DSTV\'s services, please visit your nearest Cadmus Electronic

List of Subscribers whose accounts will expire in less than 3 days:

Name	Phone	Days Left
Linda Stepper	0244322366	1
Selasi Gborglah	0545115519	2
Leon Ampah	0560557957	2

View SMS API details HERE

Figure 4.18: List of customers whose accounts expire in less than 3 days in SMS Targets page



```
// send sms to selected user
app.post('/sendsms', function (req, res) {

    var btnString = req.body.myBtn;

    var Message = SMSGH.Message;

    var fields = btnString.split(',');
    var cName  = fields[0];
    var cPhone  = "233" + fields[1].slice(1);
    var cDays  = fields[2];

    console.log("Name --- ", cName);
    console.log("Number --- ", cPhone);
    console.log("Days --- ", cDays);


    var newMessage = new Message({from: 'CadmusElectronics', to : "233549686116", content: 'Hello '+cName+', your ' +
        'monthly DSTV subscription is due for suspension in 20 days. Please visit your nearest Cadmus Electronics branch' +
        'to renew your subscription'});

    sms.messaging.send(newMessage, function (err, res) {
        if (err)
        {
            console.log("ERROR SENDING SMS;", err);

        }
            });

    res.render("pages/page_404", {cusName: cName});

});
```

Figure 4.19: Code snippet of function that sends an SMS notification to one customer

Chapter 5: Testing and Results

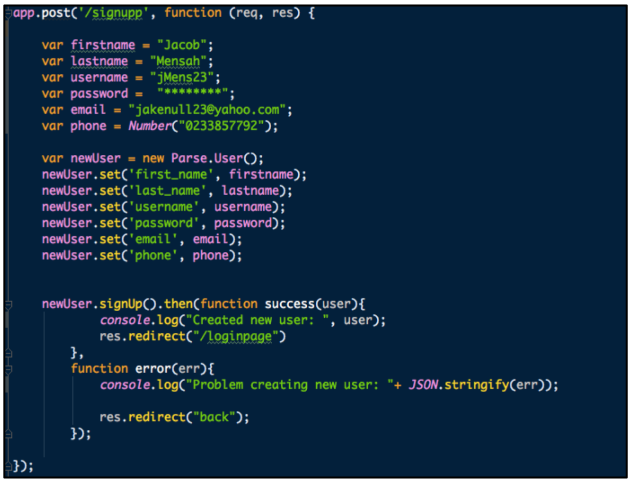
This chapter explains the results obtained from performing various tests on the application to ensure that its core functions meet the requirements highlighted in chapter 2. The tests performed were in three categories: unit tests, system-level tests, and user test.

5.1 Unit Test

The unit tests involving preforming tests on functions and features individually to ensure that desired results are obtained, and unexpected errors are corrected before integrating them into the system.

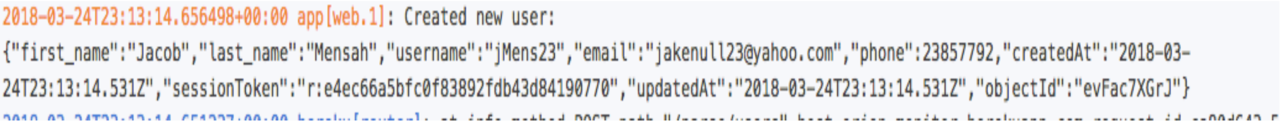
5.1.1 Creating and Logging in a Parse User

The Parse Server's API provides an option to create a Parse User class for an application running on its server. The key parameters for creating a new Parse User were username and email address. Additional attributes such as first name, last name, and phone number were included.



```
app.post('/signupp', function (req, res) {

    var firstname = "Jacob";
    var lastname = "Mensah";
    var username = "jMens23";
    var password =  "********";
    var email = "jakenull23@yahoo.com";
    var phone = Number("0233857792");

    var newUser = new Parse.User();
    newUser.set('first_name', firstname);
    newUser.set('last_name', lastname);
    newUser.set('username', username);
    newUser.set('password', password);
    newUser.set('email', email);
    newUser.set('phone', phone);


    newUser.signUp().then(function success(user){
            console.log("Created new user: ", user);
            res.redirect("/loginpage")
        },
        function error(err){
            console.log("Problem creating new user: "+ JSON.stringify(err));

            res.redirect("back");
        });

});
```

Figure 5.1: Server code to create new Parse User.



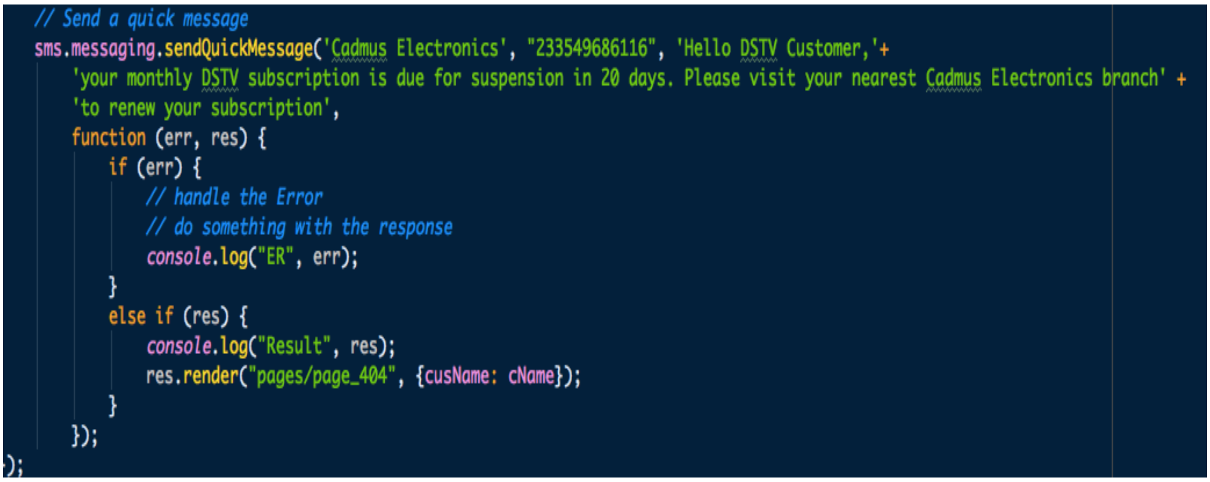
```
2018-03-24T23:13:14.656498+00:00 app[web.1]: Created new user:
{"first_name":"Jacob","last_name":"Mensah","username":"jMens23","email":"jakenull23@yahoo.com","phone":23857792,"createdAt":"2018-03-
24T23:13:14.531Z","sessionToken":"r:e4ec66a5bfc0f83892fdb43d84190770","updatedAt":"2018-03-24T23:13:14.531Z","objectId":"evFac7XGrJ"}
```

Figure 5.2: Logs from Heroku's server confirming new Parse User created.

According to the logs provided by Heroku's PaperTrail addon, the new Parse User was successfully created with all of its associated attributes. On the Parse Dashboard, the User class has its first instance which is the new user created.

5.1.2 Send SMS Notification to Customer

Hubtel's SMS API contains a variety of functions to use when composing and sending SMS message. The main method used is sending an SMS message was "**QuickMessage()**". This function takes in three parameters; the sender's name or phone number, the recipient's phone number (in the format "233*********"), and the message content, all of which are in the form of strings (within double or single quotes).



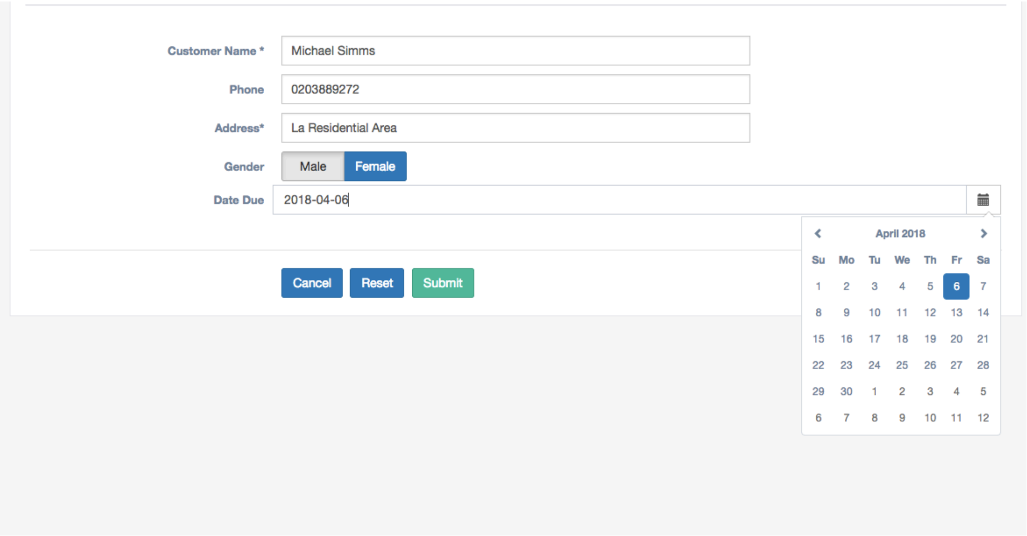
```
    // Send a quick message
    sms.messaging.sendQuickMessage('Cadmus Electronics', "233549686116", 'Hello DSTV Customer,'+
        'your monthly DSTV subscription is due for suspension in 20 days. Please visit your nearest Cadmus Electronics branch' +
        'to renew your subscription',
        function (err, res) {
            if (err) {
                // handle the Error
                // do something with the response
                console.log("ER", err);
            }
            else if (res) {
                console.log("Result", res);
                res.render("pages/page_404", {cusName: cName});
            }
        });
);
```

5.3: Function to send quick SMS message to selected user.

5.2 Integrated Testing

5.2.1 Creating Parse Object (DSTV Customer)

Orion Monitor has a webpage which allows the user to add new DSTV subscribers visiting a Cadmus branch to the database. In this page the user fills all the details in the form and submits the form. Using JavaScript, the application ensures that all fields in the form must be filled before submitting the form to the server. On the application server, all the details are taken from the form and are used to create a new instance of DSTV_Customer class in the database, which can be viewed on the Parse Dashboard.



Customer Name * Michael Simms

Phone 0203889272

Address* La Residential Area

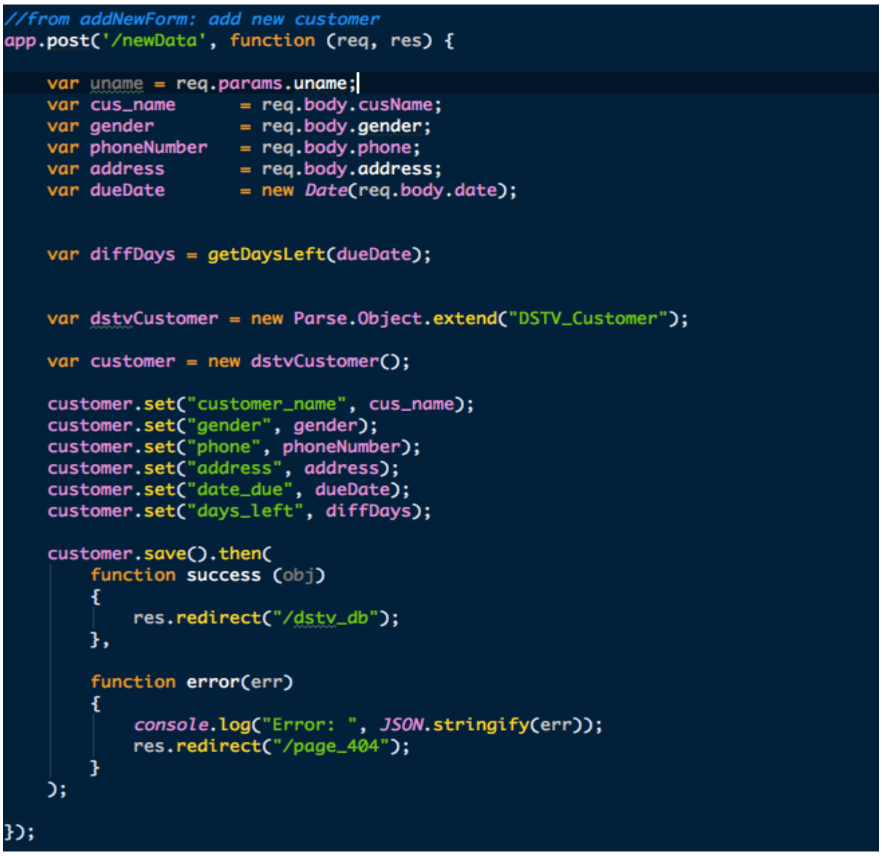
Gender Male [✓ Female]

Date Due 2018-04-06

April 2018

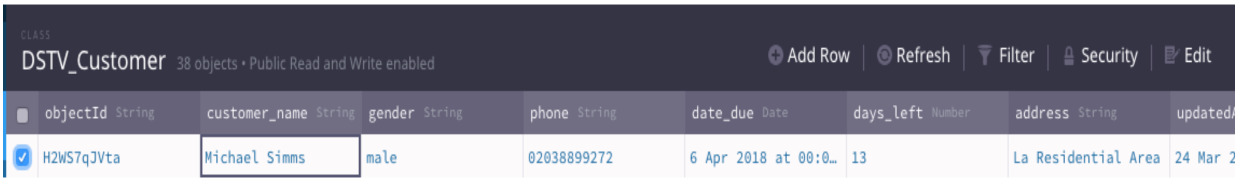
Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5
6	7	8	9	10	11	12

Figure 5.4: Form to add new DSTV_Customer



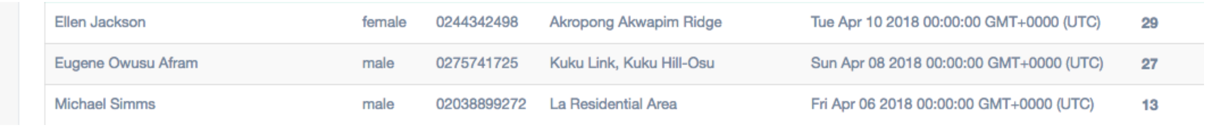
```
//from addNewForm: add new customer
app.post('/newData', function (req, res) {

    var uname = req.params.uname;
    var cus_name        = req.body.cusName;
    var gender          = req.body.gender;
    var phoneNumber     = req.body.phone;
    var address         = req.body.address;
    var dueDate         = new Date(req.body.date);


    var diffDays = getDaysLeft(dueDate);


    var dstvCustomer = new Parse.Object.extend("DSTV_Customer");

    var customer = new dstvCustomer();

    customer.set("customer_name", cus_name);
    customer.set("gender", gender);
    customer.set("phone", phoneNumber);
    customer.set("address", address);
    customer.set("date_due", dueDate);
    customer.set("days_left", diffDays);

    customer.save().then(
        function success (obj)
        {
            res.redirect("/dstv_db");
        },

        function error(err)
        {
            console.log("Error: ", JSON.stringify(err));
            res.redirect("/page_404");
        }
    );

});
```

Figure 5.5: Parse code to create new DSTV_Customer instance



	objectId String	customer_name String	gender String	phone String	date_due Date	days_left Number	address String	updatedA
☑	H2WS7qJVta	Michael Simms	male	02038899272	6 Apr 2018 at 00:0…	13	La Residential Area	24 Mar 2

Figure 5.6: Parse Dashboard showing new DSTV_Customer created



Ellen Jackson	female	0244342498	Akropong Akwapim Ridge	Tue Apr 10 2018 00:00:00 GMT+0000 (UTC)	29
Eugene Owusu Afram	male	0275741725	Kuku Link, Kuku Hill-Osu	Sun Apr 08 2018 00:00:00 GMT+0000 (UTC)	27
Michael Simms	male	02038899272	La Residential Area	Fri Apr 06 2018 00:00:00 GMT+0000 (UTC)	13

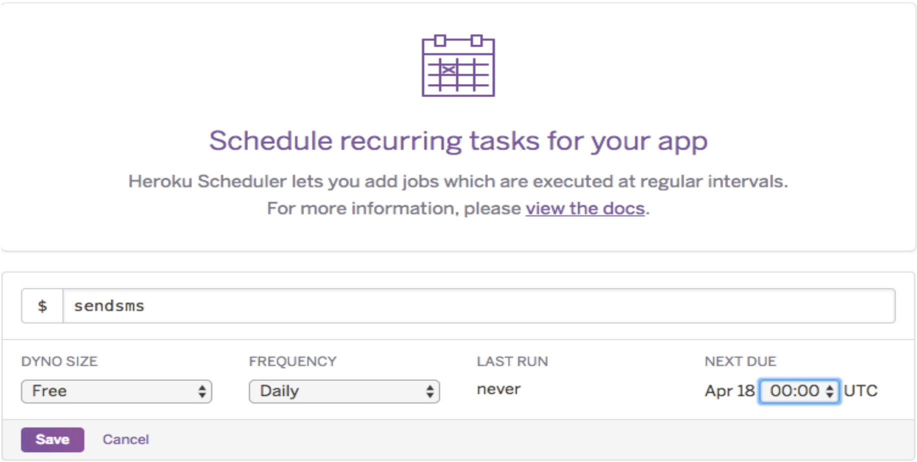
Figure 5.7: New DSTV customer added to list on webpage.

5.3 Automating SMS notifications

To use Heroku’s Scheduler, a file is created and named *sendsms* with no extensions. This file is then stored in the root directory of the project. The code in the *sendsms* file does the following:

1. Query the Parse database to find a list of customers whose monthly subscriptions are due to be suspended in less than 3 days.
2. Each customer’s name, phone number and number of days left are retrieved and stored as parameters.
3. The function that sends the SMS notification is called and the parameters for each customer are passed to it.

On Heroku, the Scheduler addon is opened and a new task is created. The name of the task is the name of the file that contains the query and SMS codes (*sendsms*). A schedule is set for the task (in this case, every day at 12am).



Schedule recurring tasks for your app

Heroku Scheduler lets you add jobs which are executed at regular intervals.
For more information, please view the docs.

```
$ sendsms
```

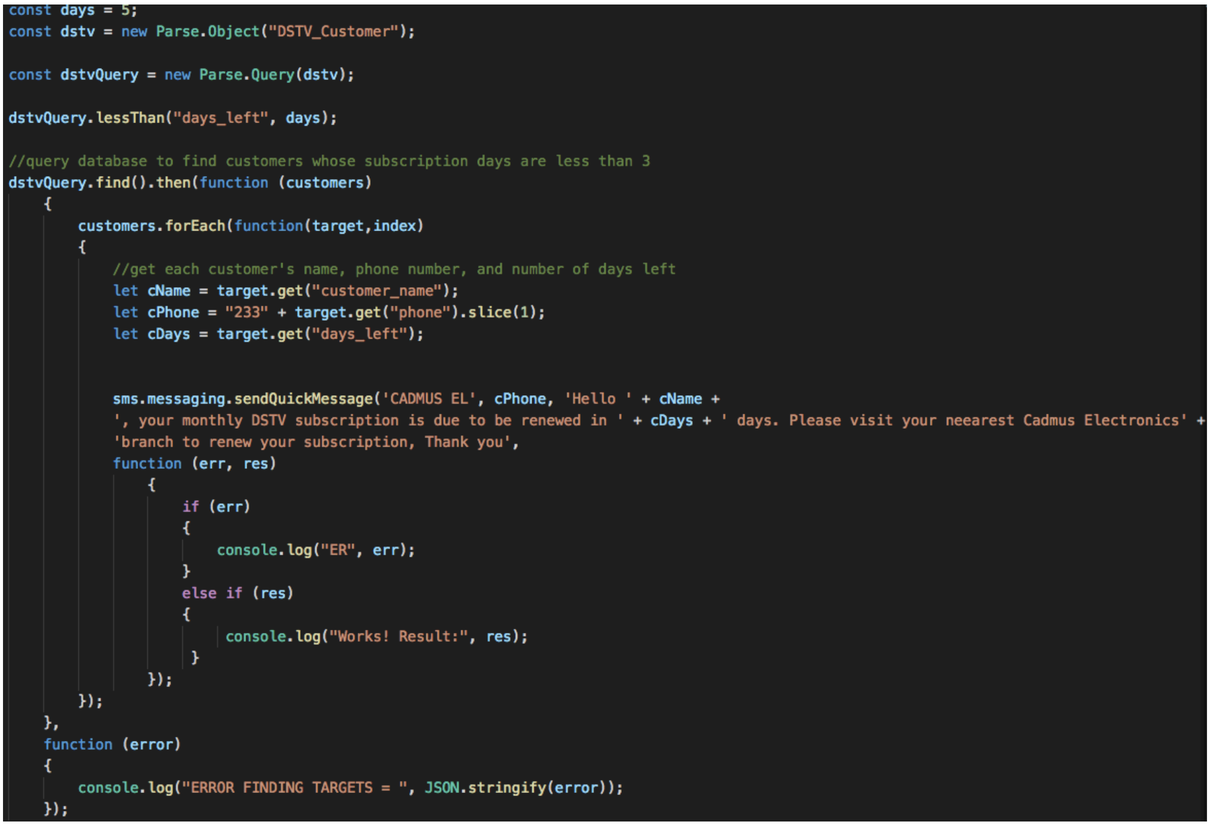
DYNO SIZE: Free

FREQUENCY: Daily

LAST RUN: never

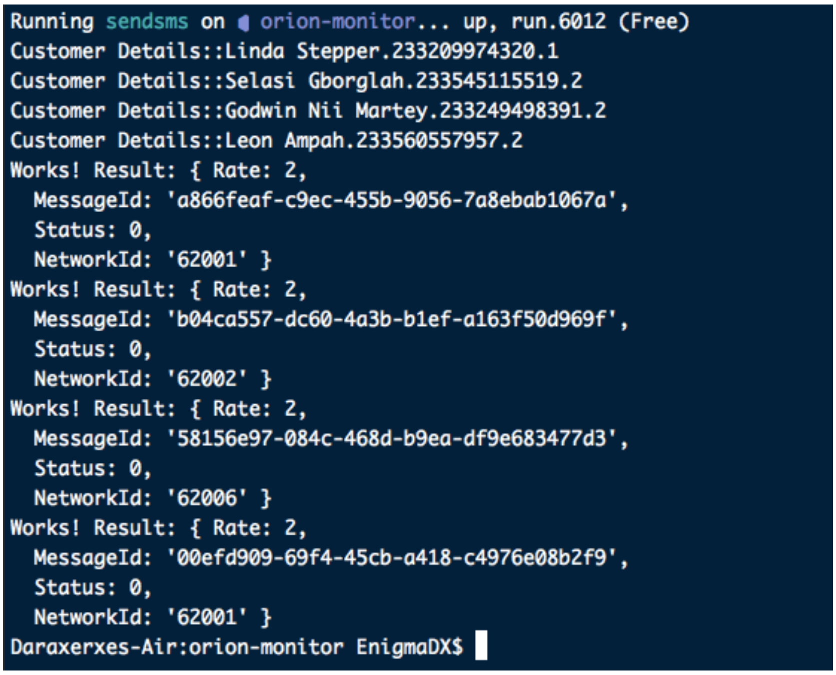
NEXT DUE: Apr 18 00:00 UTC

Figure 5.8: Creating new task with Heroku Scheduler addon.



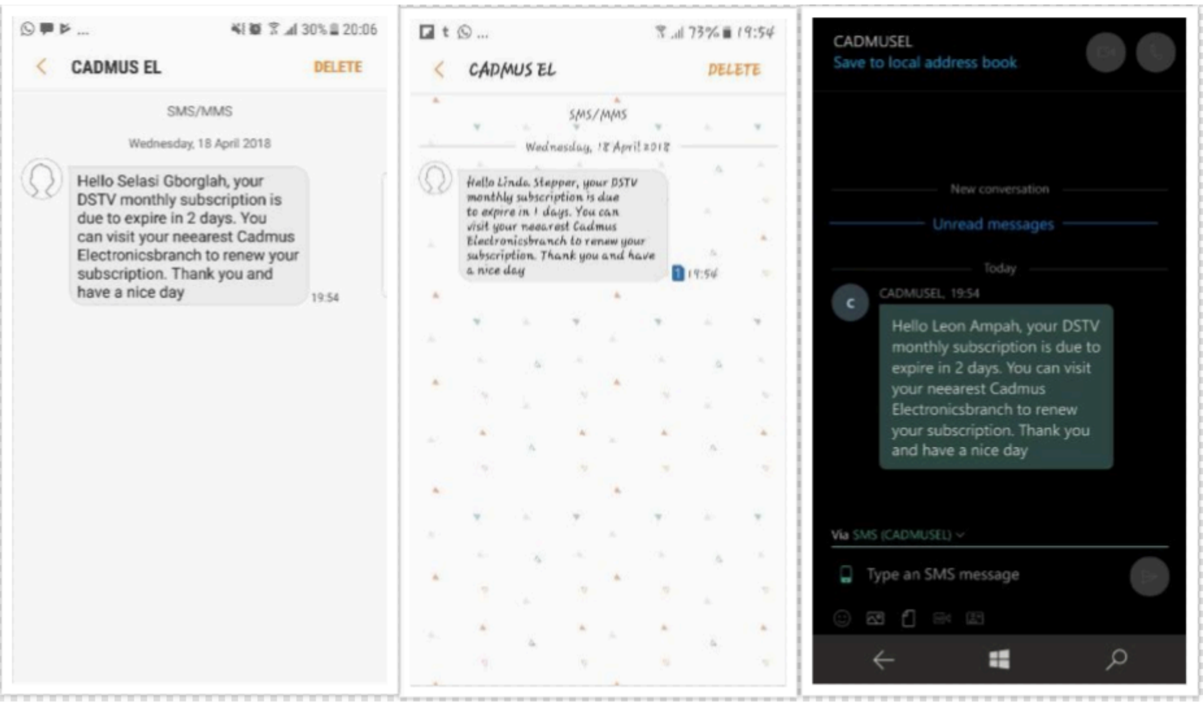
```
const days = 5;
const dstv = new Parse.Object("DSTV_Customer");

const dstvQuery = new Parse.Query(dstv);

dstvQuery.lessThan("days_left", days);

//query database to find customers whose subscription days are less than 3
dstvQuery.find().then(function (customers)
    {
        customers.forEach(function(target,index)
        {
            //get each customer's name, phone number, and number of days left
            let cName = target.get("customer_name");
            let cPhone = "233" + target.get("phone").slice(1);
            let cDays = target.get("days_left");


            sms.messaging.sendQuickMessage('CADMUS EL', cPhone, 'Hello ' + cName +
            ', your monthly DSTV subscription is due to be renewed in ' + cDays + ' days. Please visit your neearest Cadmus Electronics' +
            'branch to renew your subscription, Thank you',
            function (err, res)
                {
                    if (err)
                    {
                        console.log("ER", err);
                    }
                    else if (res)
                    {
                          console.log("Works! Result:", res);
                      }
                });
        });
    },
    function (error)
    {
        console.log("ERROR FINDING TARGETS = ", JSON.stringify(error));
    });
```

Figure 5.9: sendsms file with Parse Query and Hubtel SMS code.



```
Running sendsms on ◖ orion-monitor... up, run.6012 (Free)
Customer Details::Linda Stepper.233209974320.1
Customer Details::Selasi Gborglah.233545115519.2
Customer Details::Godwin Nii Martey.233249498391.2
Customer Details::Leon Ampah.233560557957.2
Works! Result: { Rate: 2,
  MessageId: 'a866feaf-c9ec-455b-9056-7a8ebab1067a',
  Status: 0,
  NetworkId: '62001' }
Works! Result: { Rate: 2,
  MessageId: 'b04ca557-dc60-4a3b-b1ef-a163f50d969f',
  Status: 0,
  NetworkId: '62002' }
Works! Result: { Rate: 2,
  MessageId: '58156e97-084c-468d-b9ea-df9e683477d3',
  Status: 0,
  NetworkId: '62006' }
Works! Result: { Rate: 2,
  MessageId: '00efd909-69f4-45cb-a418-c4976e08b2f9',
  Status: 0,
  NetworkId: '62001' }
Daraxerxes-Air:orion-monitor EnigmaDX$ 
```

Figure 5.10: Scheduled task sendsms running in background.



Hello Selasi Gborglah, your DSTV monthly subscription is due to expire in 2 days. You can visit your neearest Cadmus Electronicsbranch to renew your subscription. Thank you and have a nice day

Hallo Linda Stepper, your DSTV monthly subscription is due to expire in 1 days. You can visit your neearest Cadmus Electronicsbranch to renew your subscription. Thank you and have a nice day

Hello Leon Ampah, your DSTV monthly subscription is due to expire in 2 days. You can visit your neearest Cadmus Electronicsbranch to renew your subscription. Thank you and have a nice day

Figure 5.11: Screenshot of SMS notifications received on customers' phones.

5.4 Challenges

The main challenge in the application's testing was assigning specific user IDs to employees of Cadmus whose main use of OrionMonitor is inputting new customer data to be added to the database. The only way to identify a logged in user is by using a session token, which is checked before opening subsequent pages in the application. Also, Parse has a function that is used to retrieve the current logged in user (Parse.User.current()), but cannot however be used in Node.JS environment. This therefore makes it difficult to give a unique ID to a set of users to distinguish them from each other apart from the main administrator of the application.

Another challenge faced during development was using the Mailgun addon to send emails to the administrator in order to grant access to a new user.

5.5 Analysis of Results

At the end of the testing phase, most of the results prove to be satisfactory in meeting the requirements. The web application is hosted on Heroku’s server and can be accessed via web browsers. The Parse Dashboard can be accessed locally on the developer’s computer and allows for more oversight of the application’s database operations. Users (employees working at various branches of Cadmus) of OrionMonitor can add new DSTV customers with ease and can view the number of days left for a customer’s subscription to end.

Chapter 6: Conclusion and Recommendations

6.1 Current Limitations

Cadmus Electronics is yet to begin collection of details of Canal+ subscribers to be added to the web application's database. Hence only DSTV subscribers' details are available in OrionMonitor's database.

6.2 Future Works

One major issue to consider is how OrionMonitor will know when a DSTV subscriber has paid his/her monthly DSTV subscription fees. A possible solution would be to grant OrionMonitor access to DSTV's subscribers database for solely querying purposes so that customers who have already made payments in advance will be exempted from the SMS target list.

Another future addition to OrionMonitor is making use of email as an alternate means of communicating DSTV and Canal+ subscription reminders to SMS. Having a secondary means of communicating with customers would prove efficient in the event that the SMS fails in unexpected events.

6.3 Summary

The aim of this project was to provide Cadmus Electronics with an electronic database that would enable them to keep track of the regular customers they have and the new customers that will be added as time passes. The project's implementation so far fulfils the core requirement of Cadmus Electronics having a database of its regular and new DSTV customers. With this application, employees working in various branches of Cadmus will be able to add new customer records as and when they become available. Thanks to Hubtel's SMS API, Cadmus can keep in contact with its customers regularly and also build a strong relationship with new customers. This will hence promote the brand of Cadmus as the number one DSTV and Canal+ Dealer in Ghana.

References

20 Free Bootstrap Admin Dashboard Templates 2018. (2018, February 15). Retrieved February 19, 2018, from https://colorlib.com/wp/free-bootstrap-admin-dashboard-templates/

App Server. (n.d.). Retrieved March 17, 2018, from http://www.theserverside.com/news/1363671/What-is-an-App-Server

Aronson, L. (1995). *HTML 3 Manual of Style* (2nd ed.). Hightstown, NJ, USA: Ziff-Davis Publishing Co.

Chan, S. L. (2000). Information technology in business processes. *Business Process Management Journal*, *6*(3), 224–237. https://doi.org/10.1108/14637150010325444

Dix, S., Jamieson, K., & Shimul, A. S. (2016). SMS advertising the Hallyu way: drivers, acceptance and intention to receive. *Asia Pacific Journal of Marketing and Logistics*, *28*(2), 366–380. https://doi.org/10.1108/APJML-09-2015-0146

Flanagan, D. (2006). *JavaScript: The Definitive Guide*. O'Reilly Media, Inc.

Herman, S. (2007). SMS reference: keeping up with your clients. *The Electronic Library*, *25*(4), 401–408. https://doi.org/10.1108/02640470710779817

Jetty, S., & K, J. P. A. (2013). SMS-based content alert system: a case with Bundelkhand University Library, Jhansi. *New Library World*, *114*(1/2), 20–31. https://doi.org/10.1108/03074801311291938

Learning The Hubtel Basics. (n.d.). Retrieved March 19, 2018, from http://help.hubtel.com/the-hubtel-platform/introduction-to-hubtel/learning-the-hubtel-basics

Li, F. (2004). Database Technology and Database-Driven Web Applications. *Strategic Planning for Energy and the Environment*, *24*(1), 49–65. https://doi.org/10.1080/10485230409509655

Mano, W. (2005). Exploring the African view of the global. *Global Media and Communication*, *1*(1), 50–55.

parse. (n.d.). Retrieved March 20, 2018, from https://www.npmjs.com/package/parse

Parse + Open Source. (n.d.). Retrieved February 19, 2018, from http://parseplatform.org/

parse-dashboard (GitHub): (2018). JavaScript, Parse. Retrieved from https://github.com/parse-community/parse-dashboard (Original work published 2016)

Pricing | mLab Cloud MongoDB Hosting. (n.d.). Retrieved February 6, 2018, from https://mlab.com/plans/pricing/#plan-type=shared

Sklar, J. (2001). *Cascading Style Sheets* (1st ed.). Boston, MA, United States: Course Technology Press.

Solheim, J. (2015). Web Apps in the Computer Science Curriculum: A Guide Using Heroku, Java Servlets, and Postgres. *J. Comput. Sci. Coll.*, *30*(5), 126–133.

Tilkov, S., & Vinoski, S. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing*, *14*(6), 80–83. https://doi.org/10.1109/MIC.2010.145

Yusuf, A., & Nnadi, G. (1988). Structure and Functions of Computer Database Systems. *Vikalpa*, *13*(4), 37–44.