Wits Software Development Group 2013-2014

WITSMAPAPP SPECIFICATION

VERSION 2.0

MARCH 16, 2014

PRESENTED BY:

JASON CHALOM

Version History

VERSION	DATE	AUTHOR	DESCRIPTION
1.0	December 2013	JMC	Initial version, including custom maps API.
2.0	March 2014	JMC	Revised version including the use of Google Maps API

CONTENTS

Program Specification

INTRODUCTION

PROJECT OVERVIEW

This is the University of the Witwatersrand's Software Development Student group's 2013 (and perhaps going into 2014) project to build a mobile application for Wits Marketing.

This app will allow the user to select on a map where they would like to go and from their current position to their destination it will calculate and display the shortest path using the Google API.

It will also display information about the campus they are on and the building they are going to. Both a description and an image will be displayed respectively. The ability to display more images may be added later.

PROJECT OBJECTIVES

- Create server to support the input of custom data such as campus locations, descriptions and images.
- Have a database to store information
- Have the input system be user password protected from unauthorized access
- Have a functional and well-built mobile app based on the Phonegap framework and JavaScript libraries

SUCCESS CRITERIA

- A functional system which can tell where the user is from their mobile device
- Have a store of data about locations on campuses
- Ability to draw paths to locations, perhaps even have dynamically updateable paths.
- Have a server side administration web-app which is functional.
- All user interfaces have Wits colours and logo.

PROJECT TEAM

- Jason Chalom (@TRex22 on GitHub)
- Isaac Seshoka (@ick-seshoka on GitHub)

Program Specification

STANDARDS

DESIGN

The project web pages, which include the admin pages and the mobile app must have the Wits colours, and Logo. Ie Blue-Gold and the Logo.

API

The API should follow restful practices. (See design specification docs for more information).

The RESTful API standard used follows apigee's ebook: "*Web API Design: Crafting Interfaces that Developers Love*".

See below in deliverables for the API design.

SECURITY ISSUES

HTML

Cross site scripting is an issue where instead of a user inputting plaintext into a String they instead insert HTML markup to allow the use of scripts such as JavaScript. This security flaw can be overcome by checking every string and using HTML & clause to handle all input as strings. This flaw would only be system local but should still be considered.

MYSQL AND PHP

The main issue here is SQL injection. A user instead of entering plaintext may enter SQL statements to cause damage throughout the entire system. There are a few ways to negate this security flaw. One method is to add "" to the end of every string as it goes into the database. A better way would be to scrub the String so that no infected text goes into the database at all.

PHONEGAP UPDATES

There are a few ways to handle updates to a Phonegap app. It has a built in technology which can be added as an extra, but I don't like it since it greatly decreases performance.

DELIVERABLES

The SpecificationDesigns.pdf document has wireframe and colour layouts as well as a database UML design which are there for reference.

DATABASE

USERS TBL

- UserId (Primary Key) Autonumber
- Username (String)
- PasswordHash (String)
- Email (String)

CAMPUS TBL

- CampusID (Primary Key) Autonumber
- GPSLat1 (String)
- GPSLong1 (String)
- GPSLat2 (String)
- GPSLong2 (String)
- GPSLat3 (String)
- GPSLong3 (String)
- GPSLat4 (String)
- GPSLong4 (String)
- Description (String)
- ImageLocation (String)

BUILDINGS TBL

- BuildingID (String)
- CampusID (Foreign Key from Campus table)
- GPSLat (String)
- GPSLong (String)
- Description (String)
- ImageLocation (String)
- BuildingAbbreviations (String) There can be a few of them which will then be delimited.

Program Specification

WEB PAGES

All web pages will have a unified theme. (See the SpecificationDesigns.pdf document for all proposed wireframe designs)

There has to be a navigation bar to navigate to all pages

Any delete options will display a confirmation dialogue.

Any input pages which depend on other information in the system will display an error dialogue if that information has not yet been inputted.

LOGIN PAGE DESIGN

The login page will be basic in design. (See the SpecificationDesigns.pdf document)

FIRST TIME LOGIN PAGE DESIGN

There needs to be a page to initially create an administrator account. (See the SpecificationDesigns.pdf document)

USER ADMINISTRATION

This page is used to display all users in the system, edit user information and also delete users.

CAMPUS INPUT PAGE DESIGN

This page is where Campus information is inputted, it will use a Google Maps object to help locate a square which demarcates the campus selected. The user will 'draw' a rectangle around the area they wish to demarcate.

It will upload an image (Maybe multiple) of the campus and also a description of the campus.

CAMPUS ADMINISTRATION

This page will display a list of campuses in the system and will be able to delete a campus or edit its information.

BUILDINGS INPUT PAGE DESIGN

This is the page buildings will be inputted onto. There will be a dropdown list of campuses currently in the system to select from. If there is no campus then the page should redirect to the Campus Input page. It will also use a Google Maps object to find the GPS coordinates of the building on the specific campus. By selecting a campus from the dropdown the map will be populated at that campus. The user will then create a point on top of a building to select it. The page will then upload the building information to the database.

BUILDINGS ADMINISTRATION

This page will display a list of buildings on a specific campus in the system and will be able to delete a campus or edit its information.

MOBILE APP

The mobile app will have three swipe-able pages.

WELCOME PAGE

This page will have a campus dropdown menu which when selected will show a building dropdown menu so that the user can select a building which they wish to go to.

There will also be a search bar so that the user can search using a room number. The API will handle calculating the building from its abbreviation and the floor from the first number. (This will have to be checked in the database design and also the building input page.)

By clicking a next arrow, swiping left or clicking a button the program will bring up the second page.

SECOND PAGE

The second page will bring up a map, with the current location of the user and their destination drawn for them. It will also plot a course for them to take to get there. (There may need to be a recalculate button to recalculate the path or to have an auto-refresh ability). The Google Directions API could be used. https://developers.google.com/maps/documentation/directions/

THIRD PAGE

This page will display the information on the current campus and also the selected building. It will have images of the two and the abbreviations of the building as well. (Some information may already exist in the Google Map).

API DESIGN

The API design will be quite simple since it is a small API.

~ represents the root url of the API, in this design the API url will be assumed to be localhost at port 80. Ie http://localhost/api/ is the api link, and ~ represents http://localhost . {} represents a variable or a autonumber.

GET CAMPUS

~/api/{v1}/campuses/{CampusID}

This will return a JSON object of the specific campus its information and the http header.

GET CAMPUS LIST

~/api/{v1}/campuses/

This will return a JSON list (IEnumerable object) of campuses in the system including the http response header.

GET BUILDING

~/api/{v1}/buildings/{BuildingID}

This will return a JSON object of the specific building its information and the http header.

GET BUILDINGS LIST

~/api/{v1}/buildings/

This will return a JSON list (IEnumerable object) of buildings in the system including the http response header.

POST SEARCH CRITERIA

~/api/{v1}/search/{query}

This will return a JSON object of the campus and building or an error not found (Error 404 Bad Request in the http response header)

ERROR RESPONSES

Only a few specific errors are required:

HTTP 404 Not Found

HTTP 400 Bad Request

HTTP 500 Internal Server Error

HTTP 500 Internal Server Error (With custom message), Google API malfunction

HTTP 200 OK