Software Requirements Specification

UniHangouts

Date	Changes	Version
2/4/2018	Initial Document	1.0
2/7/2018	System Requirements Inspection Changes	1.1
4/18/2018	Subsequent Release Update	2.0

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to explain the requirements and features of the UniHangouts application.

1.2. Scope

This project, called UniHangouts, will be an application that allows students at UNL to discover and keep track of events that are happening on campus. Our application will give users the ability to create events, view a live feed of nearby events, and filter events using tags. The application will be accessible as a web application with the possibility to be ported to Android and/or iOS.

1.3. Definitions, Acronyms, and Abbreviations

Android: Mobile operating system developed by Google
iOS: Mobile operating system used by Apple iPhones
Xamarin: Framework for building cross-platform mobile applications through Microsoft Visual Studio
Microsoft Visual Studio: Development environment designed for the creation of applications for Windows and smartphones
Microsoft SQL Enterprise: Development environment designed for creation and management of Database and Stored Procedures
Stored Procedures: Are pre-compiled database subroutines that act like a function to get data from a database. They also provide data-validation, sanitation, and access-control to the database.
REST Service: (Representational state transfer) is a web service to provide communication over the internet. This service will manipulate the database and transfer information back to clients.
ASP.NET Web Application: Server side web application framework that provides the website.

1.4. Overview

The rest of this document will cover the overall description, the specific requirements, and an appendix for the application. The overall description will describe the overall use and purpose of the application. The specific requirements will discuss the details and design structure of the application and its features. The appendix includes screenshots of what the system will look like.

2. Overall Description

2.1. Product Perspective

2.1.1. System Interfaces

2.1.1.1. The application will interface with a Microsoft SQL database for information storage.

2.1.2. User Interfaces

2.1.2.1. The user interface will allow users to use the screen and a standard mouse and keyboard to enter input information and navigate through the application. Touchscreen support will be added if the application is eventually ported to Android and/or iOS.

2.1.3. Hardware interfaces

2.1.3.1. This application will have the capability to run on any device with an up-to-date browser.

2.1.4. Software interfaces

2.1.4.1. A REST Service will allow clients to interface with the server which provides data from the Microsoft SQL database.

2.1.4.2. An ASP.NET Web application will provide a user accessible website.

2.1.5. Communication interfaces

2.1.5.1. All communication will take place over the internet via the REST service, which will require an internet connection.

2.1.6. Memory

2.1.6.1. The application will rely on web communication and will store minimal information locally. However, the results of web communication may be stored in small local caches.

2.1.7. Operations

2.1.7.1. The main operation of the application will be displaying the feed of events to the user.

2.1.8. Site adaptation requirements

2.1.8.1. We will have to adapt the user interface when making the transition from a web application to a mobile application to accommodate for use of the touchscreen.

2.2. Product Functions

2.2.1. The function of this application is to provide a forum for students at UNL to discover events that are taking place on campus.

2.3. User Characteristics

2.3.1. The users of the application will be college students at UNL. As such, they will for the most part be young adults with a widely varying degree of experience using technology.

2.4. Constraints

2.4.1. The application will require connection to the internet to function properly, as the data for events and user data will be stored in a database.

2.4.2. Processing power will not be an issue because client apps will simply be rendering a user interface and calling web methods.

2.5. Assumptions and Dependencies

2.5.1. The application will depend on a Microsoft SQL server for maintaining the database. On the client side, it is assumed that the user is using up-to-date browser software.

3. Specific Requirements

3.1. External Interface Requirements

3.1.1. REST Service. Client apps will communicate with a server side service that will provide web methods such as the following:

3.1.1.1. Web methods for Accounts:

3.1.1.1.1. createuser

3.1.1.1.2. getuserinfo

3.1.1.1.3. login

3.1.1.1.4. logout

3.1.1.1.5. verifyapikey

3.1.1.2. Web methods for Events:

3.1.1.2.1. create

3.1.1.2.2. getdescription

3.1.1.3. Web methods for Locations:

3.1.1.3.1. create

3.2. Functions

3.2.1. First Release

3.2.1.1. Navigation

3.2.1.1.1. A UniEvents label will link to the Event Feed page.

3.2.1.1.2. An ApiTest tab will link to the ApiTest page.

3.2.1.1.3. A Login tab will link to the Login page.

3.2.1.1.3.1. The Login tab will be visible only to users who are not logged in.

3.2.1.1.4. A SignUp tab will link to the SignUp page.

3.2.1.1.4.1. The SignUp tab will be visible only to users who are not logged in.

3.2.1.1.5. An Account tab will link to the Account page.

3.2.1.1.5.1. The Account tab will be visible only to users who are logged in.

3.2.1.1.5.2. The Account tab will not display "Account," but rather the username of the current user.

3.2.1.1.6. All navigation labels and tabs will be present on all pages unless otherwise specified.

3.2.1.2. Event Feed Page

3.2.1.2.1. Upon starting the application/visiting the site, the user will be brought to the Event Feed page.

3.2.1.2.2. The Event Feed page will display events in list form, sorted chronologically, with the newest events at the top.

3.2.1.2.3. Each event in the list will display an appropriately formatted title and caption, as well as information on the event host, a brief location description, address, event type, and start and end times.

3.2.1.2.4. Clicking on an event will open an "Event Information" window that will display all of the above information as well as a description.

3.2.1.2.4.1. The window will have a functional "Close" button on the top right.

3.2.1.2.5. There will be an "add event" button on the Event Feed page that will open the Event Creation Page.

3.2.1.2.5.1. This button will not be accessible to users who are not logged in.

3.2.1.3. Event Creation Page

3.2.1.3.1. The page will contain an appropriately formatted event creation form with input fields and a submit button.

3.2.1.3.2. The submit button will not function if the user does not provide the following information: title, event type, start time, end time, city, state, country, and postal code.

3.2.1.3.2.1. The following information will be optional: caption, description, name, and address.

3.2.1.3.2.2. If submission fails, a message will warn the user, and required fields will be highlighted.

3.2.1.3.3. The start and end times will be specified using an intuitive calendar-like interface element.

3.2.1.3.4. Images may be embedded in event descriptions using HTML; they will then be displayed with the description on the Event Feed page.

3.2.1.4. API Test Page

3.2.1.4.1. A drop-down menu will exist to permit selection of an API method.

3.2.1.4.1.1. Selecting a method will populate “Route” and “Http Type” text boxes below the drop-down, insert relevant database information in a right-aligned column, and insert appropriate input fields in a left-aligned column.

3.2.1.4.2. When an input field is updated, the visual representation of the post body in the ‘Post Body’ box will be updated.

3.2.1.4.3. An execute button will exist to send the post and update the ‘Resulting JSON’ box with the response from the database.

3.2.1.4.4. A ‘Clear Inputs’ button will exist to empty every input box.

3.2.1.4.5. A timer exists to measure the response speed of the database.

3.2.1.5. Login Page

3.2.1.5.1. The page will contain a username text field and a password text field.

3.2.1.5.2. The page will contain a submit button.

3.2.1.5.2.1. If login fails, a message will inform the user of the error and.

3.2.1.5.3. The page will contain a “Create Account” button that links the user to the Account Creation page.

3.2.1.5.4. A successful login will display a message greeting the user by username and direct the user to the Event Feed page.

3.2.1.5.5. Exiting the browser will not log the user out.

3.2.1.6. Account Creation Page

3.2.1.6.1. The page will contain the following required input fields (marked with a red asterisk): username, password, city, state, country, and postal code.

3.2.1.6.1.1. The following fields will be optional: display name, first name, last name, school email, contact email, phone number, location name, and address.

3.2.1.6.2. A “Create Account” button exists to submit the input information and create the corresponding account.

3.2.1.6.2.1. When the button is clicked, invalid or missing input will prevent the account from being created, display an error message, and highlight the input boxes.

3.2.1.6.2.1.1. A username that is already taken constitutes invalid input.

3.2.1.6.3. Upon successfully creating an account, the system will display a success message and direct the user to the Login page.

3.2.1.7. Account Page

3.2.1.7.1. The page will display a list of active logins, each with corresponding date, time, and API key next to a functional logout link that can be used to end corresponding session.

3.2.1.7.1.1. The current session will be clearly labeled and highlighted.

3.2.1.7.2. A “Logout Everywhere” button will exist to log the user out of all active sessions.

3.2.1.7.3. Upon ending the current session using one of the above methods, the system will display a success message and direct the user to the Event Feed page.

3.2.1.7.3.1. If logout fails for some reason, an error message will be displayed.

3.2.1.7.4. The page will display all information that the current user entered on the Account Creation page, leaving fields blank if that information was not provided.

3.2.1.8. Security

3.2.1.8.1. User passwords will be encrypted with SHA-256 before they are stored into the database.

3.2.2. Second Release

3.2.2.1. Event Tags

3.2.2.1.1. A required event tag input field will exist on the Event Creation page.

3.2.2.1.2. A button will exist on the Event Creation page to allow users to create a new tag.

3.2.2.1.2.1. The button will open a window with a close button, submit button, and the following required fields: tag and description.

3.2.2.1.3. Corresponding tags will be displayed for each event on the Event Feed page.

3.2.2.2. Event Type Creation

3.2.2.2.1. A button will exist on the Event Creation page to allow users to create a new event type.

3.2.2.2.1.1. The button will open a window with a close button, submit button, and the following required fields: name and description.

3.2.2.3. Autocomplete for Locations

3.2.2.3.1. The forms on the Event Creation and Account Creation pages will both include a “Location Search” box that allows users to search known locations and automatically fill out the relevant input fields with information about a selected location.

3.2.2.4. Email Verification.

3.2.2.4.1. A “Send Verification Email” link will be displayed next to any email addresses listed on the Account page.

3.2.2.4.2. Clicking the “Send Verification Email” link will send an email containing a verification link to the corresponding account.

3.2.2.4.3. The verification link in the email, if valid, will take the user to a landing page and display a success message.

3.2.2.4.3.1. In the database, the email should be recorded as verified.

3.2.2.5. RSVP Functionality

3.2.2.5.1. Attendance information will be added to events on the Event Feed page, consisting of numbers labeled by the following categories: “Attending,” Will Stop By,” “Later,” “Maybe Attend,” and “Will Not Attend.”

3.2.2.5.2. The Event Information window will have buttons (corresponding to the above labels) on the bottom that allows users who are logged in to update RSVP information for the event.

3.3. Performance Requirements

3.3.1. The system should receive and display information from the database within 2 seconds.

3.3.2. The REST service and Web Application will be running on UNL’s servers and should minimally impact the servers.

3.4. Logical Database Requirements

3.4.1. User Information

3.4.1.1. The database will store information about user accounts including, username, a secured version of their password, an email address, and in later stages will store user devices.

3.4.2. Event Information

3.4.2.1. The database will also store information about events including date, start time, end time, location, privacy setting, events tags, proprietor, and summary.

3.5. Design Constraints

3.5.1. For the near future, one design constraint involves storage for the back end portion of the application. Storage space limits may affect the number of events and users we can host, especially if we allow users to upload images.

3.5.2. The potentially cross-platform nature of the application means that the interface will need to be kept simple and adaptable to a variety of screen sizes and aspect ratios.

3.5.3. Because the project's budget is limited, certain features such as a map view using Google's paid services are presently out of the question.

3.6. Software System Attributes

3.6.1. Reliability

3.6.1.1. The system must preserve database integrity at all costs.

3.6.1.2. The user applications should be reliable enough to provide full functionality; no functionality-breaking bugs or crashes should be present.

3.6.2. Availability

3.6.2.1. The system should have an uptime above 95% so it is not a noticeable problem for users.

3.6.3. Security

3.6.3.1. The system will implement secure account verification and password storage practices. The application will also utilize API keys so the program has the ability to block or ignore unauthorized requests to the database.

3.6.4. Maintainability

3.6.4.1. The system will be split between a backend database, and a front end web-based user interface. Updates to the front end interface will be easily done, and can be done often when a display issue is discovered, or when a large portion user feedback requests a change. Changes to the backend will be done less often, unless a performance or organization optimization can be made efficiently with minimal down time of the system.

3.6.5. Portability

3.6.5.1. The system is designed to be a web based application, so it can be run on any device with an internet connection. If eventually ported to the Android and/or iOS operating systems, it will be downloadable from the Google Play Store and/or the Apple App Store.