5-2018

Investigation of Alternatives for Migrating the One-Stop-Shop (OSS) Application to a Single, Web-Based Offering that is Conducive for both Desktop and Mobile Use.

Sahiti Katragadda

Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/gradreports

Part of the Other Computer Sciences Commons

Recommended Citation

Katragadda, Sahiti, "Investigation of Alternatives for Migrating the One-Stop-Shop (OSS) Application to a Single, Web-Based Offering that is Conducive for both Desktop and Mobile Use." (2018). All Graduate Plan B and other Reports. 1198.
https://digitalcommons.usu.edu/gradreports/1198

This Report is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Plan B and other Reports by an authorized administrator of DigitalCommons@USU. For more information, please contact rebecca.nelson@usu.edu.
INVESTIGATION OF ALTERNATIVES FOR MIGRATING THE ONE-STOP-SHOP (OSS) APPLICATION TO A SINGLE, WEB-BASED OFFERING THAT IS CONducive FOR BOTH DESKTOP AND MOBILE USE

by

Sahiti Katragadda

A report submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

Approved:

Dr. Douglas Galarus, Ph.D.  Dr. Curtis Dyreson, Ph.D.
Major Professor  Committee Member

Dr. Amanda Lee Hughes, Ph.D.
Committee Member

UTAH STATE UNIVERSITY
Logan, Utah

2018
Copyright © Sahiti Katragadda 2018

All Rights Reserved
ABSTRACT

Investigation of Alternatives for Migrating the One-Stop-Shop (OSS) Application to a Single, Web-Based Offering that is Conducive for both Desktop and Mobile Use.

by

Sahiti Katragadda, Master of Computer Science
Utah State University, 2018

Major Professor: Dr. Douglas Galarus
Department: Computer Science

The One-Stop-Shop (OSS) application provides real-time data which is helpful for travelers in the Western United States in planning their travel. Included is traditional information (routing, imagery, weather), as well as points of interest and other route-specific information (elevations, rest areas, etc.). The system displays real-time data streams in a web-based application and in a separate mobile web application, which are presented to end users in a user-friendly format.

OSS web application and OSS mobile web application features have been examined and the best design features for the mobile application have been identified. Along with that, additional design features are recommended to make the mobile app user-friendly and intuitive.

The first question that needs to be addressed before starting the redesign is, “Do we need a separate mobile web application, or can we directly convert a website into a mobile application?” This is addressed in the report. Google analytics reports were also used to identify predominant features in website and mobile app that can be emphasized in the prototype. A prototype for the application has also been developed considering the essential features of the web application that can be incorporated into the mobile application.

The mobile web application cannot currently be deployed in an app store as it is not an Android or iOS app. Wrapping the mobile web application into native iOS and Android apps were examined. Different methods were evaluated to convert the application into a cross-platform, mobile application, including the usage of Apache Cordova and online app builders.

Results were documented to develop an effective, user-friendly mobile web application that can be deployed in app stores.

OSS – All Roads. All Routes. One Stop.
ACKNOWLEDGEMENTS

To my parents and my brother, Raghu Katragadda, who have been the source of great strength and support to me all through. I would like to express my sincere gratitude and deep appreciation to my esteemed Major Professor, Dr. Douglas Galarus, for providing constant support, active guidance, comments and valuable suggestions to me throughout the course of this project thus helping it to reach a successful completion.

I extend my gratitude to the Utah State University for giving me this opportunity.

My sincere thanks to Dr. Curtis Dyreson and Dr. Amanda Lee Hughes for being on my Committee.

Sahiti Katragadda
# CONTENTS

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT ........................................................................................................ iii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS .............................................................................................. iv</td>
</tr>
<tr>
<td>LIST OF TABLES .................................................................................................. viii</td>
</tr>
<tr>
<td>LIST OF FIGURES ................................................................................................ ix</td>
</tr>
<tr>
<td>CHAPTER</td>
</tr>
<tr>
<td>1 INTRODUCTION ..................................................................................................... 1</td>
</tr>
<tr>
<td>1.1 Background ....................................................................................................... 2</td>
</tr>
<tr>
<td>2 OSS WEBSITE OVERVIEW ................................................................................... 6</td>
</tr>
<tr>
<td>2.1 Screen Layout .................................................................................................. 6</td>
</tr>
<tr>
<td>2.2 Data Represented in OSS Website ................................................................. 6</td>
</tr>
<tr>
<td>2.3 Controls for OSS Website ................................................................................ 8</td>
</tr>
<tr>
<td>2.4 Sample OSS Website Feedback ...................................................................... 9</td>
</tr>
<tr>
<td>2.5 OSS Website Usage Statistics ...................................................................... 10</td>
</tr>
<tr>
<td>3 OSS MOBILE WEB APPLICATION OVERVIEW ................................................. 11</td>
</tr>
<tr>
<td>3.1 Screen Layout ................................................................................................. 11</td>
</tr>
<tr>
<td>3.2 Data Represented in OSS Mobile Web Application ........................................ 11</td>
</tr>
<tr>
<td>3.3 Best Design Features for OSS Mobile Web Application ................................ 12</td>
</tr>
<tr>
<td>3.4 Recommendations for Additional Design Features ...................................... 13</td>
</tr>
<tr>
<td>4 PROTOTYPE ......................................................................................................... 19</td>
</tr>
<tr>
<td>5 MOBILE WEBSITE VS MOBILE APP ............................................................... 22</td>
</tr>
<tr>
<td>6 WRAPPING OSS MOBILE INTO AN APP USING APACHE CORDOVA .......... 23</td>
</tr>
<tr>
<td>6.1 Apache Cordova ............................................................................................ 23</td>
</tr>
<tr>
<td>6.2 Implementing an OSS Mobile App using Apache Cordova ............................ 224</td>
</tr>
<tr>
<td>7 WRAPPING OSS MOBILE INTO AN APP USING ONLINE APP BUILDERS .... 25</td>
</tr>
<tr>
<td>7.1 GoNative ....................................................................................................... 25</td>
</tr>
<tr>
<td>7.2 Como ............................................................................................................ 26</td>
</tr>
</tbody>
</table>
7.3 IBuildApp .........................................................................................26

8 APACHE CORDOVA VS ONLINE APP CONVERTERS ..................28

9 SUMMARY ..........................................................................................29

REFERENCES .........................................................................................32

APPENDICES .........................................................................................35

APPENDIX A ..........................................................................................36

A.1 Installation and Setup of Apache Cordova on Windows ...............36
A.2 Implementation of Apache Cordova to run OSS Mobile Web App in Apache Cordova for Android.................................................................36

APPENDIX B ..........................................................................................42

B.1 Implementation of Apache Cordova to run OSS Mobile Web App in Apache Cordova for iOS .................................................................42

APPENDIX C ..........................................................................................43

C.1 Implementation of Apache Cordova to create a link in the App to launch the browser URL .........................................................................43

APPENDIX D ..........................................................................................46

D.1 Demo of GoNative Online App Builder ............................................46
<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1 Apache Cordova Vs Online App Converters</td>
<td>28</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 OSS Home Page</td>
<td>6</td>
</tr>
<tr>
<td>3.1 OSS Mobile Web App Home Page</td>
<td>12</td>
</tr>
<tr>
<td>3.2 Initial Loading Details</td>
<td>13</td>
</tr>
<tr>
<td>3.3 Weather Status Shown in Color</td>
<td>14</td>
</tr>
<tr>
<td>3.4 Example Vertical Legend Feature for Weather</td>
<td>15</td>
</tr>
<tr>
<td>3.5 Example Hamburger Menu</td>
<td>16</td>
</tr>
<tr>
<td>3.5.1 Example Hamburger Menu</td>
<td>17</td>
</tr>
<tr>
<td>3.6 Tab Bar Menu</td>
<td>17</td>
</tr>
<tr>
<td>3.7 Recommended Features Overview with Hamburger Menu</td>
<td>18</td>
</tr>
<tr>
<td>4.1 Prototype: Home screen for OSS Mobile App</td>
<td>20</td>
</tr>
<tr>
<td>4.2 Prototype: Hamburger Menu on OSS Mobile App</td>
<td>21</td>
</tr>
<tr>
<td>A.1 Apache Cordova Installation</td>
<td>37</td>
</tr>
<tr>
<td>A.2 Cordova Project Creation, Adding Android Platform and Build</td>
<td>37</td>
</tr>
<tr>
<td>A.3 Successful Build Creates a Debug APK file</td>
<td>38</td>
</tr>
<tr>
<td>A.4 Installing Debug APK file on a Connected Device</td>
<td>38</td>
</tr>
<tr>
<td>A.5 Keystore File Creation</td>
<td>39</td>
</tr>
<tr>
<td>A.6 Keystore File Created in the Root Directory</td>
<td>40</td>
</tr>
<tr>
<td>A.7 Releasing the Signed APK File for Production</td>
<td>40</td>
</tr>
<tr>
<td>C.1 OSS Web Link App</td>
<td>45</td>
</tr>
<tr>
<td>C.2 Launch will Open OSS Mobile Web Application</td>
<td>45</td>
</tr>
<tr>
<td>D.1 Input the URL and Build the App</td>
<td>46</td>
</tr>
<tr>
<td>D.2 Enter App Details</td>
<td>46</td>
</tr>
</tbody>
</table>
D.3 Add an Application Icon .................................................................47
D.4 Email Containing the Results of the Build.................................47
D.5 iOS Source Code and Android APK is Available for Use...............48
D.6 OSS Mobile App is Installed and is Ready for Use .......................49
CHAPTER 1

INTRODUCTION

The One-Stop-Shop (OSS) for Rural Traveler Information project developed a website that serves many users daily during bad weather conditions, providing them with real-time road weather information to ensure safe road travel [1]. The system is a web-based application and it includes a separate mobile web application that was developed later. Both are presented to end users in a comprehensible way with a great user-experience. The original offering contains a large amount of data including current weather conditions, road/travel conditions, forecast weather, fire incidents and detections, and other travel-related information like rest areas, features of interest, truck scales, and summit locations. The mobile web application contains a subset of these features.

The features of the OSS web application and the OSS mobile web application were examined carefully to determine the best features that can be incorporated on the mobile web application. Chapter 2 discusses the details of the OSS web application. It describes the data represented in the OSS website, controls and usage statistics, which are helpful for predicting the features from the original offering that should be incorporated into the mobile web application.

Chapter 3 includes a brief introduction to the OSS mobile web application and its best design features. There are features on the OSS mobile web application that are difficult to identify without a legend like the weather details. A menu is essentially useful in the OSS mobile web application to accommodate more features from the original website. Recommendations are presented in this chapter to enhance the legibility of features and accommodate more features from the original website to make the mobile web application more intuitive and user-friendly. With these recommendations, a prototype was developed with the features suggested from Chapter 3. This prototype is discussed in Chapter 4. Chapter 5 presents the differences between a website, a mobile website and a mobile app. This chapter emphasizes the advantages and disadvantages of having a mobile app. The mobile web application cannot be deployed in an app store as it is not an Android or iOS app.

There are different approaches to deploy a web application on a mobile device. One approach includes building native apps with platform specific skills. Java and Kotlin programming languages are used to build Android apps. Objective C and Swift are used to build iOS apps. These apps have the highest performance with full access to mobile device capabilities. Another approach is to use a mobile web application that can be fully-hosted in the mobile browser. Also, the mobile web application cannot be deployed in an app store as it is not an Android or iOS app.

A third approach is a hybrid mobile application development which uses a single codebase to target multiple platforms like Android and iOS. The hybrid approach builds
the user interface with standard web development technologies like HTML, CSS, JavaScript access to data from the server. This is wrapped into a native app. The OSS mobile web application, which is developed with the standard web technologies, is wrapped into a hybrid mobile application with the help of tools like Apache Cordova and online app builders. Both are effective tools to wrap the mobile web application into native apps with one codebase that can target Android and iOS platforms. Apache Cordova uses WebViews, where, the HTML, CSS and JavaScript codebase runs in an internal browser called WebView that is wrapped into a native app. This hybrid approach reduces the time and cost to build mobile apps from a mobile web application or a web application.

1.1 Background:

The OSS application provides general information about driving conditions, road and weather information, traveler information, traffic congestion/delay information, trip planning or route planning data, trucking information, winter road and weather conditions [1]. The OSS application promotes traveler safety and mobility. This system is widely used in the Western United States. The Western States Rural Transportation Consortium (WSRTC) grouped together to promote technologies and educational opportunities to facilitate and enhance safe, seamless rural travel throughout the Western United States. Western States included in the OSS applications are Arizona, California, Colorado, Idaho, Montana, Nevada, New Mexico, Oregon, Utah, Washington, and Wyoming. This project is a continuation and broadening of the original California Oregon Advanced Transportation System (COATS) project. COATS began in 1998 as a collaborative effort among the California Department of Transportation (Caltrans), the Oregon Department of Transportation (ODOT) and the Montana State University, Bozeman’s Western Transportation Institute (WTI) to investigate the use of Intelligent Transportation Systems (ITS) in rural areas. The intent of the COATS project was to facilitate the use of ITS to enhance safety, improve the movement of people, goods, and services, and subsequently promote the economic development of the bi-state region and to begin deployment of those solutions [5].

“The long distance traveler doesn't care that he has crossed a state line or district boundary. What he wants is accurate, timely and reliable road condition and weather information all along a route - from beginning to end. OSS finally gives us the ability to get route oriented real-time Traveler Information to the public in an effective manner.”

- Ian Turnbull, Chief (retired), and original OSS project Champion Caltrans Office of ITS Engineering and Support [5]

OSS serves a large population of users that depend on real-time traveler information. These users include:

- Local, regional, and long-distance travelers,
• Local and state transportation agency personnel,
• Emergency responders,
• Commercial vehicle operators, and
• The traveling public in general.

The original OSS website link is: http://oss.weathershare.org/. The mobile OSS web application link is: http://oss.weathershare.org/m/. The mobile web application cannot be deployed in an app store because it is not an Android or iOS app.

The OSS application is very useful for transportation agencies and traveling public in general. Here are some benefits of OSS [5]:

• OSS provides easy access to all-in-one place for real-time traveler information which enhances traveler safety and mobility in rural areas.

• Travelers can be aware of the road and weather conditions which allows them to make informed decisions about their travel.

• OSS can reduce significant delays by informing the users about road incidents like vehicle accidents, work zones, and road closures.

• OSS facilitates operations for transportation, emergency response and public safety agencies allowing them to assess conditions, coordinate resources and expedite response activities.

• OSS is a cost-effective approach as it makes critical road, weather and related information available without costly infrastructure and installations.

• For commercial drivers, OSS can serve as a valuable routing and scheduling tool to improve performance and maximize efficiency for freight movement and other services.

• OSS uses data from various sources and presents them on a single site. This allows users to depend on a single website rather than visiting various websites.

• OSS improves the delivery of traveler information throughout the region. At one web location, travelers can see what conditions exist and what conditions are expected over their entire route.

The One-Stop-Shop is affiliated with organizations like Caltrans District, Caltrans Division of Research, Innovation and System Information, Montana State University.
(MSU), Western Transportation Institute (WTI), and Western States Rural Transportation Consortium (WSRTC) [5].

The data displayed in OSS, on a single platform, is the combined data from many data sources. While many individual states operate information portals for their own highways, OSS was one of the first systems to synthesize data from multiple state information portals. The following departments of transportation provide data to OSS [6]:

- Arizona Department of Transportation (ADOT)
- California Department of Transportation (CalTrans)
- Colorado Department of Transportation (CDOT)
- Idaho Transportation Department (ITD)
- Montana Department of Transportation (MDT)
- Nevada Department of Transportation (NDOT)
- New Mexico Department of Transportation (NMDOT)
- Oregon Department of Transportation (ODOT)
- Utah Department of Transportation (UDOT)
- Washington Department of Transportation (WSDOT)
- Wyoming Department of Transportation (WYDOT)

The following entities provide weather and climatic condition data for OSS [6]:

- National Oceanic and Atmospheric Administration (NOAA)
- National Oceanic and Atmospheric Administration (NOAA) MADIS
- National Weather Service (NWS)
- National Weather Service (NWS) Advanced Hydrologic Prediction Service (AHPS)
- National Weather Service (NWS) National Digital Forecast Database (NDFD)
• University of Utah (MesoWest)

Additional data includes google maps, traffic, directions, elevation, summit locations, rest area details, fire incidents and detection, and road incidents reports. Sources for this data include [6]:

• California Highway Patrol (CHP)

• Google Maps, Traffic, Directions, Elevation

• United States Department of Agriculture (USDA) Active Fire Mapping Program

• Nevada Department of Public Safety (NVDPS)

• New Mexico Tourism Department (New Mexico True).

The following state traveler information sites provide details about traffic congestion / delays, road incidents and message signs specific to their states [6].

• Arizona: http://www.az511.com/traffic/

• California: http://quickmap.dot.ca.gov/, https://go511.com/

• Colorado: http://www.cotrip.org/home.htm

• Idaho: http://lb.511.idaho.gov/idlb/

• Montana: http://roadreport.mdt.mt.gov/travinfomobile/

• Nevada: http://www.nvroads.com/

• Oregon: https://tripcheck.com/Pages/Road-Conditions

• Utah: http://udottraffic.utah.gov/

• Wyoming: http://www.wyroad.info/

Data is obtained from various data sources for the different states and integrated into one application, OSS. It provides travelers with real-time details so that they can plan ahead of their trip and be aware of road and weather conditions in the Western United States by accessing a single application, OSS.
CHAPTER 2

OSS WEBSITE OVERVIEW

2.1 Screen Layout

The OSS homepage initially shows a map with “Help”, “Road/Travel Conditions”, “Current Weather”, “Fire”, “Forecast Weather”, and “Other Info” tabs on the toolbar towards the upper left corner of the map, allowing users to access drop-down menus to change the information layer displayed on the map. See Figure 2.1.

Figure 2.1: OSS Home Page [1]

2.2 Data Represented in the OSS Website:

OSS provides users with useful information about road and weather conditions. This information is presented in a simple and comprehensible way that reduces the effort required to get the information. The data is widely distributed [3].

- The Help tab contains a drop-down menu to view a user guide or a basic usage tutorial in the form of a video, to provide feedback via an online survey, to view
a data and links page and to view information about the system.

- The **Road/Travel Conditions** tab can be used to view chain restrictions, road information (construction), incidents, CMS (changeable message signs), CCTV (closed circuit TV images), and Google Traffic by selecting or unselecting the options with checkboxes.

- The **Current Weather** tab provides information about air temperature, relative humidity, 1-hour precipitation, 24-hour precipitation, AHPS 24-hour precipitation, and wind.

- The **Fire** tab provides access to display fire incidents and fire detections.

- Under the **Forecast Weather** tab, are links for air temperature, wind speed, wind gust speed, humidity, sky cover, 12-hour chance of precipitation, 6-hour amount of precipitation, snow, and weather forecasts.

- The **Other Traveler Info** tab allows users to view locations of rest areas, features of interest, truck scales, and summit locations.

- The **Route Planner** button helps to plan a route between starting and ending locations by showing reference points on the map.

- The **Link to Current View** button on the upper right of the toolbar creates the URL for your current information layer, map center, and zoom level so that you can set a bookmark to this page with your current settings.

- The **Google Map controls** on the left side (+, -) help to zoom in or out; pan right, left, up, or down. Zoom in/out can also be performed using the scroll wheel on your mouse or zoom in by double-clicking on the map. The Google control on the top right of the page gives an option to select from a map view, a satellite view, or a terrain view.

- The **Restore Initial View** button allows users to return to their original view.

- The **Share** option on the top right of the toolbar functionality is implemented using “AddThis.com” to share the application through Facebook, Twitter, Email,
Pinterest, Gmail, Google+, etc.

- A legend is available at the bottom left of the map for each option. The legend can be expanded or collapsed to show or hide detail.

2.3 Controls for the OSS Website:

The information from various sources is neatly depicted on one site, OSS. OSS uses controls for the website that are widely used and are obvious so that the users need not learn new things or technological controls to access the website. The website controls are built in such a way that even a novice can access the website and view the information in an easy way [3].

- Zoom in and out:

  The zoom control allows users to view detail information more clearly by clicking on “zoom for detail” or by clicking the “+” icon in the bottom right corner of the web page. A “drag zoom” option is available in the upper left corner to select a portion on the map to zoom in for more details about a region. Zooming out gives an overview of the entire region. Zooming out can be achieved by clicking the “-” icon that is available in the bottom right of the web page.

- Pan up, down, right, left:

  This control is used to move around the map in all four directions.

- Menu selections:

  These are particularly useful to select a specific option that is of interest to the user. Menu selections display all the features that are available in OSS and allow users to select data layers.

- Click on markers to display more information:

  Some of the markers on the map contain additional information, which can be displayed by selecting a marker.

- Display modes in Map view, Satellite view, and Terrain view

  These modes provide different map views.
• A Legend:

The OSS application uses different representations on the map to present data. The legend explains the data that is shown on the map. The legend is shown in a box placed in the bottom left of the web page. The information provided is helpful for understanding the map.

The details of each options can be found in the user guide [3].

2.4 Sample OSS Website feedback:

The following are the sample feedback responses obtained from OSS users:

“[OSS is] very helpful in helping me to decide when to leave on my run, especially up into Oregon, as that weather changes abruptly at times. The weather data provided (both current and forecast) helps me greatly when traveling on US97. I-5 gets all kinds of informational coverage but finding information for the other highways can be very, very trying. [It] also helps to know what to expect in certain trouble prone areas.”

- Surveyed OSS user [5]

“In recent lightning induced fires in Northern California, [OSS] was instrumental in gauging where the fires could be headed based on wind speed... This allowed our center to be better prepared for all of the many ‘what if’ situations we were faced with.”

- Dispatcher, Caltrans Transportation Management Center [5]

“It will be really useful for planning our trips when we are doing field work or visiting our agencies.”

- Surveyed OSS user [5]

“It can’t be beat for travel planning and checking on road conditions, accidents, looking at web cam, etc. Beats calling in for road conditions.”

- Surveyed OSS user [5]

“OSS was very helpful when driving from Montana to Oregon and back in the middle of
winter. It aided in planning my trip by allowing me to see the upcoming road conditions, knowing when the best times of the day would be to go over the many mountain passes to minimize travel on snow and ice, avoid traffic congestion, view highway cameras, and see upcoming travel alerts and weather forecasts. OSS is a great tool that is easy to use and removes the need to visit multiple websites to retrieve weather and road conditions in each state. I now primarily use OSS when traveling through the states covered by the web application.”

- Student, Montana State University [5]

“The OSS team has produced a world-class system that has a unique approach to solving multiple complex problems in the aggregation and delivery of traveler information. Through a strong technical team, partnerships and a steady vision, the OSS platform is the definitive answer to rural traveler information in the western United States.”

- Sean Campbell, Chief and OSS Project Manager, ITS Special Projects Branch, Caltrans Division of Research, Innovation, and System Information [5]

2.5 OSS Website Usage Statistics:

The following reference provides information about the usage of OSS website during a period. This information is helpful to identify the predominant features that users use on the OSS.

“We started collecting statistics on OSS usage in November 2011. Since then, the number of users and the amount of information they are viewing has increased dramatically. To date (as of September 16, 2016), we have had over 400,000 user sessions and over 16,000,000 CCTV camera images have been viewed. OSS has been accessed from over 120 countries, from all 50 U.S. States, over 4200 cities and towns in the US, and over 700 cities and towns in California. On December 24th, 2015, there were 6190 user sessions and over 320,000 CCTV images served. We expect the upcoming winter to be another record year for the One-Stop-Shop, particularly if we have a rough winter.”

- Doug Galarus, Program Manager and Senior Research Scientist, WTI/MSU (and Principal Investigator on OSS) [5]
CHAPTER 3

OSS MOBILE WEB APPLICATION OVERVIEW

3.1 Screen Layout

The OSS Mobile home page initially displays the entire region in a high-level view. As we zoom in for details, the data becomes clear. The mobile view (shown in Figure 3.1) implements a subset of the functionality and features, unlike the website view which has more room to accommodate all the features [2].

3.2 Data represented in the OSS Mobile Web Application:

The OSS mobile web application contains essential features that are widely used in the website. The OSS mobile web application is a subset of the original web application presenting a minimum set of features that addresses typical user interests.

- **CCTV:**
  
  CCTV provides images taken from CCTV cameras which are opened in a small window when the CCTV marker is selected.

- **CMS (Changeable Message Signs):**
  
  CMS convey real-time information to drivers and call for additional attention to hazards identified by warning messages. These are opened in small windows when the CMS marker is hovered.

- **Incidents:**
  
  Incidents provide information about real-time road traffic collisions and other events. These reports the area affected, type of incident, and responding officer status in a small window when the incidents icon is selected.

- **Chain restrictions:**
  
  Chain restrictions for specific vehicle types provide details including elevation, route, status and restriction information that is depicted on a small window which is opened when the chain restriction icon is selected.

- **Current weather:**
  
  Current weather conditions are depicted on the map as a raster depicted with colors corresponding to the weather type.

- **Weather forecast information:**
  
  Weather forecast information is available when each marker is selected.
There is a link on the bottom of each marker’s information bubble that will open a window to the National Weather Service’s mobile site, providing forecast information for that location.

![Figure 3.1: OSS Mobile Web App Home Page](image)

3.3 Best Design Features for the OSS Mobile Web Application:

Designing a mobile site is often a challenge. The design features are essential because they make the site legible. There are many design features to consider in developing a mobile web application. OSS uses the following features to make it stand out and be more presentable to the user.

- The content is kept to minimum i.e. the user is presented with only what they most-likely want to know. The features that users select most on the original website are shown in the OSS mobile web application.
- It uses a simple design with less interface elements to keep the user at ease with the application.
- The OSS mobile web application is consistent with the original OSS website. This means the web service and the mobile application share similar characteristics. This allows users to make frictionless transitions between the mobile and web
application.

- When the mobile web application is opened, the real-time data is loaded in the application. The user is notified about this with the help of several messages on the screen as shown in Figure 3.2. This keeps the user engaged in the application and makes it clear when loading is taking place.

  Welcome to OSS Mobile.
  OSS Mobile works best with an LTE or high-speed Wi-Fi connection.
  Please wait while we download up-to-date traveler information to your device...

  Initializing Map...
  Retrieving Data...
  Retrieving CCTV Data...
  Retrieving Weather Forecast Data...
  Retrieving CMS Data...
  Retrieving Incident Data...
  Retrieving Chain Control Data...
  Done Downloading Incident Data...
  Done Downloading Chain Control Data...
  Done Downloading Weather Forecast Data...
  Done Downloading CMS Data...
  Done Downloading CCTV Data...

  Figure 3.2: Initial Loading Details [2]

- Parameters are set in the URL corresponding to the map view. This gives an overview of the data on the map at a glance and provides quick access to the view if bookmarked or the site is reloaded.
  http://oss.weathershare.org/m/?clat=40.5&clng=-114&zoom=5

  The URL indicates that latitude = 40.5, longitude = -114.0 and zoom Level = 5.

- Icons are shown for incidents, CCTV, CMS can chain restriction locations. With a mouse-over or click event, the marker is zoomed in and the sticky window opens. As the mouse-out or second click event occurs, the sticky window closes.

3.4 Recommendations for Additional Design Features:

The most important thing to keep in mind when designing a mobile app is to make sure that the app is intuitive. If the app is not intuitive, it cannot be useful or have real value to the user. The OSS mobile web application has a weather feature which contains weather-related information that is depicted in colors without a legend. This
kind of representation makes it difficult for users to understand the significance of the color which relates to the status of current weather conditions.

As seen from Figure 3.3, it may be difficult to identify weather features on OSS Mobile. It is not clear as to which color signifies what kind of weather condition (snow, rain, etc.) due to the lack of a legend. Note that the colors correspond to National Weather Service standards. Providing a legend for weather display could make things clearer to the user.

Figure 3.4 shows an example legend from an existing, unrelated, third-party application called Radar App. A vertical legend as shown in this figure can not only make information clear but also will occupy relatively little screen space.

Radar App contains an expandable feature at the top which signifies crucial details for current weather status. Having this kind of feature on the OSS Mobile App will require an extra expandable display as shown in the Figure 3.4 that is specific to the weather conditions.
Another recommendation for the OSS mobile web application is to include a menu feature. This is useful to accommodate more features from the original website.

A menu is a set of options presented for selection to the user. Different menu presentation mechanisms are available for mobile apps [16].

A hamburger menu [16] is a classic example of a hidden menu as shown in the Figure 3.5. It does not occupy screen space until the user calls it with a tap or swipe.

The advantages of using a hamburger menu are [16]:

- Saves space on the screen.
- Accommodates a large number of features.
- It is well-known.

The disadvantages of using a hamburger menu are [16]:

- Hides context, so users don’t know which navigation options are available until they open this menu.
Users must do one or more actions before reaching another screen.

A tab bar [16] is usually located at the top or bottom of the screen as shown in the Figure 3.6. Users can reach it from anywhere in the app. It is always located in the thumb-friendly zone. A tab bar can be a good solution if the navigation system is not too branched out.

The advantages of using a tab bar menu are:

- Easy to reach, which makes the user interface more smooth.
- It is always in sight, so people will use this more often than hidden menus.
- Navigation to screens occurs with a single tap.

The disadvantages of using a tab bar menu are:

- A tab bar can hold a limited number of options (5 or less).
- If icons on the tab bar don’t have labels, it may be hard to guess what they mean.

Figure 3.5: Example Hamburger Menu [17]
On the OSS application, using a hamburger menu option on the left corner could be used to accommodate more features when compared to tab bar menu. Figure 3.7 shows an example from the Missouri Department of Transportation’s Traveler Information app.
There are features in OSS website that can be implemented in OSS mobile app to provide similar functionality. The Route Planner is not present in OSS mobile but is present in the original web app and could be implemented in OSS mobile app. A mechanism for sharing the application via various social networking sites is available in the website. A share option in OSS mobile could be helpful. Additionally, mobile phones support push notifications feature. This can be used in OSS mobile app to give timely updates i.e. to report status about the current location might also be considered.

Figure 3.7: Recommended Features Overview with Hamburger Menu
Source: MoDOT Traveler Information Map [22]
CHAPTER 4

PROTOTYPE

To develop a prototype OSS Mobile App, the essential features from the desktop versions should be incorporated into the mobile application. The first question that needs to be addressed before starting the design is “Do we need a separate mobile web application, or can we directly convert the website into a mobile application?” The (original) OSS website includes all the core features and some additional features. It generally requires a wide screen to provide room for display of all information.

For a mobile application, users do not necessarily need all the functionality of a web application. When all the features from the website are added to the app, the app could become too complicated with many screens, and users would find it difficult to use. Building an app with the core features first and then adding complimentary features may be a viable approach.

For the prototype, the goal is to build an effective user interface for OSS mobile to accommodate more features from the original website. In order to implement more features from the original website, we can use a hamburger menu as shown in Figure 4.1 on the OSS mobile app. In this way, more features can be accommodated in OSS mobile app.

Google Analytics reports are used to track usage of the original and mobile versions of the One-Stop-Shop. The predominant event action on the original and mobile versions is CCTV (camera selection). The event actions that are widely used in the original version are CCTV, CMS, incident, chain requirements, weather-related information (air temperature, wind), and fire-related information. Fire incident events occurred more frequently than fire detection according to the statistics in the reports. This may be a by-product of defaulting to display fire incidents when fire information is selected. Other information includes summits, rest areas, and features of interest. Mobile event actions that are widely used by users are CCTV, CMS, incident, and chain requirements.

The OSS mobile app can accommodate more features from the OSS website with the help of hamburger menu that is scrollable. This is depicted in the Figure 4.1 and Figure 4.2.
Figure 4.1: Prototype Home Screen for OSS Mobile App
Figure 4.2: Prototype: Hamburger Menu on OSS Mobile App
CHAPTER 5

MOBILE WEBSITE VS MOBILE APP

Apps and mobile websites are typically accessed on handheld devices like smartphones. A mobile website is like any other website. It consists of browser-based HTML pages that are linked together and accessed via the Internet. The distinguishing feature between a website and a mobile website is the fact that mobile websites are designed to be accessed using smaller displays. Apps are applications that are downloaded and installed on mobile devices rather than being rendered in a browser.

Mobile websites can be inexpensive. They have a price similar to that of designing and maintaining a regular website. A mobile website is instantly accessible to users via a browser across a range of devices like iPhone, Android, BlackBerry, Windows Phone, etc. They don’t need a separate site for iOS/Android. Also, mobile websites are easy to set up and easy to maintain and update. They need not be submitted on app stores [21]. Mobile websites simply require a domain and hosting. Apps require the users to download and install the app on their devices.

There are some critical disadvantages for mobile websites. They do not have offline access, and app store presence. If a mobile website is designed poorly, it greatly affects its performance on mobile devices with a cluttered design which results in poor usability.

Mobile apps can do things which websites cannot, like push notifications and offline access to offloaded data. The design for mobile apps are typically less cluttered and are more specific to the mobile device platform. Mobile apps are available on the app stores which are always placed on the home screen of the mobile devices when they are downloaded. This kind of ease of access enhances user access by keeping them engaged better in the app.

There are some disadvantages too with mobile apps. They incur extra expense to deploy an app on app stores. An app needs to be set up and submitted to app stores. A mobile website is much more dynamic than an app in terms of pure flexibility to update the content. To change the design or content of the mobile website, simply edit the site and publish it to make the changes immediately visible [21].

Unlike a mobile website, mobile app updates should be pushed to an app store which in turns requires installing the update on the devices.
CHAPTER 6

WRAPPING OSS MOBILE INTO AN APP USING APACHE CORDOVA

There are various approaches to convert a mobile web application to mobile apps. The first approach deals with building native apps with platform-specific skills like Java and Kotlin for Android apps, Objective C and Swift for iOS apps [13].

A second approach is to use a mobile web application. The OSS mobile web application can be fully hosted in a mobile browser. The major drawback with this approach is that the mobile web application cannot be deployed in an app store as it is not an Android or iOS app.

A third approach is hybrid mobile application development, which uses a single codebase to target multiple platforms like Android and iOS. This approach builds the user interface with standard web development technologies like HTML, CSS, JavaScript with server access to data. This is wrapped into a native app. The OSS mobile web application, which is developed with the standard web technologies and JSON server data, is wrapped into a hybrid mobile application with the help of tools like Apache Cordova and online app builders. Ideally, we can use standard web development tools to create and deploy mobile applications from a single codebase across different platforms. Technologies available to achieve this task include Apache Cordova and online app builders [13].

6.1 Apache Cordova

Apache Cordova is a free, open-source platform for building native mobile applications using HTML, CSS, and JavaScript. Apache Cordova enables developers to build applications for mobile devices using HTML, CSS, and JavaScript instead of relying on platform-specific API’s like those in Android, iOS, or Windows Phone [4].

It enables wrapping HTML, CSS and JavaScript code depending on the platform of the device. This allows developers to target platforms with just one code base. The resulting applications are hybrid, meaning that they are not truly native mobile applications because all the layout rendering is done via WebViews instead of the platform’s native UI framework, nor purely web-based because they are not just Web apps, but are packaged as apps for distribution and have access to native device APIs.

Cordova applications are ordinarily implemented as a browser-based WebView within the native mobile platform. Apache Cordova is a good option for those who already have a web application and want to port it to different mobile platforms. Cordova development uses existing HTML + JavaScript + CSS applications to build a cross-platform mobile application for iOS, Android, Windows Phone, and other platforms [4].

The OSS Mobile website is well-suited for conversion to a Cordova app. The front-
end uses HTML, CSS, JavaScript. The back-end server provides JSON data files and the target platform is Android, iOS, Windows Phone. Cordova is a wrapper, an application that has an embedded web browser when the web app is loaded.

6.2 Implementing an OSS Mobile App using Apache Cordova

Apache Cordova can be used to convert the OSS Mobile Web Application into a native app or cross-platform app with just one codebase [4]. The web application or mobile web application developed in HTML, CSS and JavaScript with JSON data in the back-end is packaged into Cordova application using a Cordova container which acts as a wrapper and generates native APK file for Android and IPA file for iOS that can be deployed onto Google’s Play Store and Apple’s App Store. To achieve this, we must move front-end code from the web server to a Cordova app. Front-end code must be repackaged in a Cordova-friendly fashion so that it can run in the Cordova client. For details on Installation and Setup of Apache Cordova on Windows, refer Appendix A. Also, refer to Appendix A for a detailed implementation of Apache Cordova to run the hosted OSS Mobile Web App in Apache Cordova for Android. For details about Implementation of Apache Cordova to run the hosted OSS Mobile Web App in Apache Cordova for iOS, refer Appendix B.

In this way, a cross-platform app is generated that can be deployed on app stores for Android and iOS. As Apache Cordova is an open source project, it is reliable and dependable to wrap the OSS application into a Cordova container to generate platform-specific applications under one codebase. However, there is a major disadvantage with this approach. Live updates cannot be reflected in the app. Each time a new update is added to the application, the above procedure of building and deploying the application needs to take place which can be tedious. Another approach to overcome this problem requires the creation of a Cordova project with a static mobile web application which consists of a “Launch” option to open the browser-based mobile web application. In this way, the app can be deployed onto the app stores without having to rebuild each time an update is made to the application. All live updates to the mobile web application will be reflected in the app that is deployed in to the app stores. Now, there won’t be any issues with live updates as the launch option opens the browser URL of the mobile web application. Updates to the app in the app store are not required if the mobile web application is updated in the browser[4].

An advantage with this approach is the user interface of the app in the app store can be updated easily and less frequently. It is less expensive to maintain the application which launches the web URL directly. However, there are some major disadvantages with this approach. This application is a static application which simply launches the OSS mobile web URL link. Users could by-pass the app and go directly to the website. This might not be desirable. Also, there is a risk of the app being rejected by Google or Apple to deploy the app on their app stores as the apps simply directs to the website with a launch option.

To see the process of Apache Cordova creating a link in the app to launch the browser URL, go to Appendix C.
CHAPTER 7

WRAPPING OSS MOBILE INTO AN APP USING ONLINE APP BUILDERS

Online app builders are used to convert a website into a native mobile application within minutes. This does not require any additional coding efforts. Once the interactive website which is usable on small devices is built, the URL of website is used to convert it into a mobile application.

The technology behind online app builders is built to take existing web applications and seamlessly convert them into native apps for Android and iOS. They have custom-built Android and iOS native containers to convert an existing web application to a hybrid mobile app. Online app builders support publishing and distributing a web app as a native app through Apple iTunes and Google Play. All updates to the website automatically cascade to the mobile app, and no app store updates are required. [10]

7.1 GoNative

GoNative (https://gonative.io) was founded in early 2014 to “create smart solutions for challenging problems.” It all starts with hosting the existing web application in the custom-built Android and iOS native containers. This helps to publish and distribute the existing “web app” as a "native app" through Apple iTunes and Google Play, and all updates to the site automatically cascade to those native apps, without requiring any app-store updates.

GoNative acts as better wrapper as other wrappers leverage PhoneGap or Cordova which requires static template files and AJAX for all network requests. GoNative has been designed from scratch to work with existing live web applications.

There are 4 steps:

1. Enter the URL of the website to wrap.
2. Receive the source via email.
3. Install it on the mobile device.
4. Tap to launch.

A demo of GoNative Online App Builder is provided in Appendix D.

GoNative requires a single application license for both Android and iOS priced at $790. It can publish or distribute apps including versions for both Android and iOS. There is full
access to source code with the ability to modify. A one-time license fee is incurred to build the cross-platform apps. The app can be updated and rebuilt an unlimited number of times. More details on pricing can be found at https://gonative.io/pricing.

7.2 Como [10]

Como app builder (also known as Como DIY) was recently rebranded to Swiftic. Como App maker has been in the app creation business since 2014 and has helped small businesses all over the world build over a million apps.

Como requires a fee of $33 per month for building cross-platform apps directly. It supports app submission to Google’s Play Store, Apple’s app store, Amazon’s app store with unlimited downloads. More details on pricing can be found at https://sites.google.com/site/comopricing/

7.3 IBuildApp [10]

IBuildApp is a very easy to use mobile app creation, hosting and management platform, established in 2011. The platform allows businesses to create and publish iPhone and Android apps in a matter of minutes.

This online platform builds an app in 3 simple steps:

1. Choose a ready-made template or use your own website application
2. Edit an app according to your preferences
3. Request publishing on Google’s play store and Apple’s app store.

IBuildApp is rated as one of the best app makers of 2017 is termed as a fine-for-your-business solution with a wide variety of features to choose from. Pricing options include $59 per month. More details can be found at https://ibuildapp.com/pricing.php

Other app builders like Mind Studios and Zeroqode are also available, and they work in the same way but with different pricing and support.

Apache Cordova or online app builders may be used for building hybrid mobile apps. A hybrid app is a combination of a native app and a web app. Users can install it on their device like a native app, but it is a web app. These types of apps are built with JavaScript, HTML, CSS and run in WebView. It is easier and faster to build because there is a single codebase.
Hybrid apps, unlike native apps, are built using HTML, CSS and JavaScript and are subsequently placed in a wrapper which can generate an APK file for Android and an IPA file for iOS. Native apps are platform specific. Android apps are developed in Java or Kotlin, iOS apps are developed in Objective C or Swift, and C# for Windows. Native apps outperform other apps because they are developed against both technical and user experience guidelines.

There is a major disadvantage of developing hybrid apps. The user experience is often sacrificed as we cannot customize the app based on the respective platform.

The decision to build either a native, web, or hybrid mobile app should be based on business objectives:

- Timeline for having the app,
- Quality of user experience,
- Complexity of features, and
- Budget

Online app builders are easy to use, quick to implement, and reasonably priced. They are a quick solution if we want to convert a website into an app. We can instantly get the full source code for Objective C and Java apps. The only technical requirement for online app builders is that the website should work well on smaller displays too.
CHAPTER 8

APACHE CORDOVA VS ONLINE APP CONVERTERS

Apache Cordova and online app converters are powerful wrappers that package HTML, CSS, and JavaScript applications into Cordova containers which can produce hybrid applications that can be deployed on Android and iOS devices. However, they have some advantages and disadvantages, and these are compared in Table 8.1.

<table>
<thead>
<tr>
<th>Apache Cordova</th>
<th>Online App Converters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free and open-source platform for building mobile applications using HTML, CSS, JavaScript.</td>
<td>There are more than a dozen of programs aimed to convert a website to an app.</td>
</tr>
<tr>
<td>Can be deployed on various platforms like Android, iOS, Windows Phone, etc.</td>
<td>Most of the tools target Android platforms and work poorly on iOS.</td>
</tr>
<tr>
<td>Cordova is owned and maintained by Apache and will always be maintained as an open source project.</td>
<td>100% Automated. Works and behaves in the same way as the one in a mobile browser. Any updates to the mobile browser automatically reflects in the app.</td>
</tr>
<tr>
<td>For any modifications in the browser, the app needs to be built again and published in the app stores. Resubmission to app stores for updates.</td>
<td>Live updates to the mobile browser will automatically be reflected in the app. Updates to the site will automatically cascade to the app.</td>
</tr>
<tr>
<td>Time-consuming to perform build, create a keystore (for Android), app ID and provisioning profile (for iOS) and publish the app.</td>
<td>Can generate a publish-ready app in minutes, directly from the website. Can also download the source code of the app to extend the app in any way.</td>
</tr>
</tbody>
</table>

Table 8.1  Apache Cordova Vs Online App Converters
CHAPTER 9

SUMMARY

The One-Stop-Shop (OSS) application is widely used in the Western United States and provides real-time road and weather information to ensure safe travel. OSS serves a large population of users including local, regional, long-distance travelers, transportation agency personnel, emergency responders, commercial vehicle responders and the traveling public. The OSS project team developed a fully-functional website, which was followed by a mobile application with a subset of features from the original website. The original website-based offering contains a large amount of data includes current weather conditions, road/travel conditions, forecast weather, fire incidents and detections, and other travel-related information including rest areas, features of interest, truck scales, and summit locations.

The OSS application proves to be very useful to the users as it presents travel-related data about road and weather conditions from different data sources in one single location. Users can access comprehensive information related to the road and weather conditions in the Western United States from one application. At one web location, travelers can see what conditions exist and what conditions are expected over their entire route. This is a cost-effective approach since it does not require costly infrastructure and installations.

The OSS web application includes features like a “Help” tab which contains a drop-down menu to view a user guide or, a basic usage tutorial in the form of a video, to provide feedback via an online survey, view a data and links page and view information about the system. The road/travel conditions tab provides details about road/travel conditions which contains chain restrictions, road information (construction), incidents, CMS (changeable message signs), CCTV (closed circuit TV images), and Google Traffic. The application also provides critical details related to current weather and weather forecasts, fire incidents and fire detection. The OSS web application provides valuable information that allows users to view locations of rest areas, features of interest, truck scales, and summit locations. Also, there is a route planner in the application which is useful in planning a route between starting and ending locations by showing reference points on the map.

The controls on the OSS web application make it flexible to look at the data more clearly by zooming in and out, panning up, down, right and left, selecting different options available in the menus, clicking on markers to display more information, and viewing the application in different modes like map view, satellite view and terrain view.

The OSS mobile web application contains only the essential features that are widely used in the website. The OSS mobile web application is a subset of the original web application with a minimum set of features that are more specific to user interests. This includes CCTV information, CMS data, incidents, chain restrictions, weather-related information and weather forecast information.
The design principles of the OSS mobile web application have been examined and additional design features are recommended. With these additional features, a prototype is developed with essential features from the original website.

The OSS Mobile web application cannot be deployed on an app store as it is not an Android or iOS app. In order to deploy the app on the Google Play Store, it should have an APK file. The same is reflected in iOS apps as it needs an IPA file to deploy on Apple’s Play Store. Wrapping the OSS mobile web application into native Android or iOS apps and generating these files is necessary to deploy onto the respective app stores.

To wrap the OSS mobile web application, technologies like Apache Cordova and online app builders may be used. The implementation details for Cordova and online app builders are discussed and compared. Apache Cordova is a free and open-source platform for building mobile applications using HTML, CSS, JavaScript.

Apache Cordova is used to develop cross-platform apps with a single codebase that can be used to deploy apps onto Apple’s App Store and Google’s Play Store. In Cordova, if there are any modifications in the application, the app needs to be built again and published in the app stores. Resubmission to app stores for updates is always required. The major disadvantage of this methodology is that the live updates to the website are not reflected in the app. A separate support team must be maintained for the mobile app to add the features from the website back into the mobile app, build the app, and deploy the newly updated version onto app stores. To reduce this complication, the mobile app can just maintain a link to launch the web browser in the app. This method proves to be efficient when it comes to live updates, i.e. a browser update would immediately be reflected in the app too as the mobile app launches the browser directly. However, it proves to be inefficient because it seems to be like a static application which has access to the web browser link.

The Apache Cordova method is time-consuming to perform a build, as it involves the creation of a keystore (for Android), app id and creation of a provisioning profile (for iOS) and finally publishing the app.

Comparatively, online app builders are 100% automated. Online app builders provide an efficient way to convert the website directly into a mobile app with little efforts and pricing to deploy and maintain the app on app stores. They work and behave in the same way as an application would in a mobile browser. Any updates to the mobile browser automatically reflect in the app. Live updates to the mobile browser will automatically be reflected in the app i.e. the updates to the site will automatically cascade to the app. They can generate a publish-ready app in minutes, directly from the website. We can also download the source code of the app to extend the app in any way. However, there is a paid subscription for the service and might not very reliable to depend on such a tool. There is less flexibility as the automated tool deals with the wrapping and deployment process for
the app. Also, most of the online app builder tools target Android platforms and work poorly on iOS.

Therefore, using Apache Cordova as a wrapper to convert a web or a mobile web application into a hybrid mobile app proves to be a reliable, and an effective approach. The only disadvantage with this approach is live updates to the web or to the mobile web application cannot be reflected in the app immediately. The app needs to get updated by rebuilding, redeploying the app and the users are required to install the updates on their devices. If this can be managed, then Apache Cordova is a good approach.
REFERENCES


[17] Hamburger Menu Figure 3.5 Source, [Online]. Available:
   http://justuxdesign.com/blog/my-beef-with-the-hamburger-menu,

[18] Tab bar Menu Figure 3.6 Source, [Online]. Available:

   https://play.google.com/apps/publish/signup/


APPENDICES
APPENDIX A

A.1 Installation and Setup of Apache Cordova on Windows [4]

Step 1: Install Node JS

Step 2: Add C:\Program Files\nodejs\ path to environment variables.

Step 3: To install dependencies: npm install

Step 4: Install Cordova: npm install -g cordova

Step 5: Add %appdata%\npm\ path to environment variables to perform Cordova related operations.

A.2 Implementation of Apache Cordova to run the hosted OSS Mobile Web App in Apache Cordova for Android

Step 6: Create Cordova project using the following command.
    Command: cordova create myfirstapp com.hybrid.Android myApp
    Com.hybrid.Android is the unique domain name of the app.
    This creates a new Cordova project which contains hooks, node_modules, platforms, plugins, res, www directories along with config.xml, and package.json.

Step 7: Add Android platform
    Command: cordova platform add Android
    Go to platforms and check if there is a folder called Android.
    Go to WWW folder and delete the entire content and paste the code of the web application which need to be converted to a mobile app.

Step 8: Now perform Build for Android
    Command: cordova build Android
    Check for a successful build.

Step 9: Successful build generates a debug APK file as shown in Figure A.3.
    Android debug.APK can be installed on phone. Run the file using adb install APK file name as shown in FigureA.4.
Figure A.1: Apache Cordova Installation

```
C:\WINDOWS\system32\npm install -g cordova
```

Figure A.2: Cordova Project Creation, Adding Android Platform and Build

```
C:\Users\sahilt\Desktop\myFirstApp cordova create myFirstApp com.hybrid.android myApp
Creating a new cordova project.
```

```
C:\Users\sahilt\Desktop\myFirstApp\myFirstApp\cordova platform add android
```

```
C:\Users\sahilt\Desktop\myFirstApp\myFirstApp\cordova build android
```

This plugin is only applicable for versions of cordova-android greater than 4.0. If you have a previous platform version, you do *not* need this plugin since the whitelist will be built in.

```
C:\Users\sahilt\Desktop\myFirstApp\cordova build android
```

```
BUILD SUCCESSFUL
```
This debug APK file cannot be published on Google Play Store. The Google Play Store will not approve it. To publish this application, it needs to be signed. To sign it, a key-store file should be created.

**Step 10:** Create a keystore file.

To generate the key, use a KeyGenerator:

**Command:**

```
keytool -genkey -v -keystore app-key.keystore -alias app-key -keyalg RSA -keysize 2048 -validity 10000
```
keystore password: firstkey

Figure A.5: Key store File Creation

Step 11: Add build.json file with keystore that is created to the root folder as shown in the above screenshot with the following code.

```json
{
    "Android": {
        "release": {
            "keystore": "app-key.keystore",
            "storePassword": "firstkey",
            "alias": "app-key",
            "password": "firstkey",
            "keystoreType": ""}
    }
}
```
Keystore password is firstkey. Keystore file name is app-key.

Figure A.6: KeyStore File Created in the Root Directory

**Step 12:** Release the signed APK file (using keystore) for production.
Command: cordova build Android --release

Figure A.7: Releasing the Signed APK file for Production
The APK file “app-release.APK” is used for publishing the app on Google Play Store.

**Step 13:** Publish the app on Google play store.

To upload the app to google play store [19]:

- You must have an account on google play store.  
  25$ is the one-time subscription fee  

- Then choose “Add new application”

- Add a title for the app.

- Upload signed release APK file obtained from the above steps.

- Upload icon, screenshots of the app.

- Add license agreement with terms and conditions.

- Click on “Publish app”.

  The app may take 6-7 hours to become live on Google Play Store.
APPENDIX B

B.1 Implementation of Apache Cordova to run the hosted OSS Mobile Web App in Apache Cordova for iOS [4]

Step 1: Create the Cordova application like Android and add iOS platform using the following command

Command: Cordova platform add ios

Step 2: Go to iOS platform and open the XCode project and run the build.

Step 3: To publish the app to App Store, you need a paid developer account (99$ for 1 year) [20].


Once registered, you'll need to generate your signing certificates.

- Go to account in developer.apple.com
- Create app ID, bundle identifier which matches with that in XCode.
- This app ID can be used to create the provisioning profile.
- In the build, click archive.
- Validate the app and upload the app to the app store.

Demo: https://www.youtube.com/watch?v=sCtjUCUxrhw
APPENDIX C

C.1 Implementation of Apache Cordova to create a link in the App to launch the browser URL [4]

Step 1: Create the Cordova project OSSMobileApp and add Android and iOS platforms.

Step 2: Update the Cordova project.

- Upon creating the project, config.xml file is generated. Add `<allow-navigation href="http://oss.weathershare.org/m/" />` to config.xml file

  This entry instructs the Cordova project's whitelist plugin to allow the application's WebView to navigate to the hosted site.

- Double-click on the Cordova project's www\scripts\index.js file to open it for editing, remove all the code in the file, and replace it with the following:

```javascript
var app = {
  // Application Constructor
  initialize: function() {
    this.bindEvents();
  },

  bindEvents: function() {
    document.addEventListener('deviceready', this.onDeviceReady, false);
  },

  onDeviceReady: function() {
    app.receivedEvent('deviceready');

    // Here, we redirect to the web site.
    var targetUrl = "http://oss.weathershare.org/m/";

    var bkpLink = document.getElementById("bkpLink");
    bkpLink.setAttribute("href", targetUrl);
    bkpLink.text = targetUrl;
    window.location.replace(targetUrl);
  },

  // Note: This code is taken from the Cordova CLI template.
  receivedEvent: function(id) {
```
var parentElement = document.getElementById(id);
var listeningElement = parentElement.querySelector('.listening');
var receivedElement = parentElement.querySelector('.received');

listeningElement.setAttribute('style', 'display:none;');
receivedElement.setAttribute('style', 'display:block;');

console.log('Received Event: ' + id);
}

app.initialize();

This code sets an event listener for the Cordova deviceReady event. This event fires when the Cordova container finishes initializing, and all the installed plugins are available. In the event, the code sets the URL of the native WebView window (using window.location.replace) to point to our remote web application. This causes the application to pull content from that site and render it in the Cordova application's main page (the WebView).

● Double-click on the project's www/index.html file and replace the markup in the <body> element with the following:

<a id="bkpLink" href="http://oss.weathershare.org/m/">OSS Mobile App</a>

<div class="app">
  <h1>Apache Cordova</h1>
  <div id="deviceready" class="blink">
    <p class="event listening">Connecting to Device</p>
    <p class="event received">Device is Ready</p>
  </div>
</div>

<script type="text/javascript" src="cordova.js"></script>
<script type="text/javascript" src="scripts/index.js"></script>

● In the project's www/index.html file's <head> section, replace the existing Content-Security-Policy (CSP) <meta> element with the following:

By adding the remote web app URL to the CSP specify that it is a trusted domain, and content from this site will be allowed in the hosted app.

**Step 3:** Build the updated Cordova project.

**Step 4:** Get the debug APK file and release APK file as shown in the above procedure and install the APK file on the device and test it.

![OSS Web Link App](Image)

Figure C.1: OSS Web Link App
Figure C.2: Launch Opens OSS Mobile Web Application
D.1 Demo of GoNative Online App Builder [10]

GoNative online App builder is used to wrap a web or mobile web application into Android and iOS. This application takes the URL link of the web app and converts it into the following.

`https://gonative.io/

Figure D.1: Input the URL and Build the App

Figure D.2: Enter App Details
Figure D.3: Add an Application Icon

- Click on “build”.
- The app is built, and the email as shown in Figure D.4 is received.

Figure D.4: Email Containing the Results of the Build

- Navigate to the private management URL.
- Download the APK that is generated.
- Transfer the APK to Android device, install it and test the app on the device.
Figure D.5: iOS Source Code and Android APK is Available for Use
Unlike Android, where we can directly download and install the Android APK on the device, Apple makes this process a bit more complex on iOS. In fact, there is no way to directly download the iOS app from GoNative and get it running on the iOS device.

In order to install the iOS app onto Apple’s device, the app must be built from source code in XCode which is a program provided by Apple that runs only on Mac computers. It requires a valid Apple developer account to do this process and to publish the app on app store.