**Pointer Toolkit**

**Fall 2016 Team:** Nicholas Kubley, Kierstan Leaf, Dakota Oakley

**Fall 2015 Team:** Will Robinson, Dan Sowka, Emily Wiedoff

**Table of Contents**

**Overview**

**Development**

**Design**

**Android Development Progress**

**iOS Development Progress**

**Campus Maps**

**Bus Routes**

**My Notes**

**Quick Links**

**Athletic Events**

**Dining Hours**

**Useful Tools**

**Offline Storage**

**GitHub**

**Overview**

Pointer Toolkit was originally created in the Fall of 2015 as a CNMT 480 project. During this semester Campus Maps and Bus Routes were created and functional. In the Fall of 2016, again as a CNMT 480 project, four new modules were added: My Notes, Quick Links, Athletic Events, and Dining Hours. Campus Maps was also updated to include more functionality such as 'Take Me There'.

Pointer Toolkit was published to both Android and iOS at the end of the Fall 2016. The Android version is currently live at <https://play.google.com/store/apps/details?id=edu.uwsp.pointerToolkit> while the iOS version is waiting for approval.

Below you will find basic documentation on each module, GitHub, design, some special JavaScript files, and development processes.

**Development**

Continual development of it requires all tools needed to run Ionic apps including, but not limited to Node.js, npm, Leaflet, D3, cordova, ionic, bower, and gulp. OpenStreetView is also used on the Campus Maps module.

Resources, such as splash screens and icons, are built across two folders to accommodate different operating system build processes as well as Cordova’s updates.

**Design**

Pointer Toolkit follows a mix of Material Design and UWSP's color standards.

https://material.io/guidelines/

https://www.uwsp.edu/infotech/web/Pages/standards.aspx

All of the basic icons were taken from one of the following Material Design Libraries:

http://ionicons.com/

https://materialdesignicons.com/

We stuck with a few basic colors throughout the app, they are as follows:

#51328e //Dark Purple

#664a9c //Mid Purple

#a075b2 //Light Purple

#ffffff //White

#f2af2b //Yellow

**Android Development Progress**

As of December 20, 2016, version 1.0 of the app is published to the Google Play Store here: <https://play.google.com/store/apps/details?id=edu.uwsp.pointerToolkit> . Downloading and running through here is as simple as downloading and installing any other app from Google Play.

**iOS Development Progress**

As of December 21, 2016, the app is in the process of being submitted to the Apple Store.

**Campus Maps**

The campus maps module shows a map of campus along with most buildings located on campus. Building information is stored in the campusmap.geojson file. Information stored includes the building’s boundaries, name, ID, location, center, image thumbnail, and the file names for the floor plans. The Chemistry building is not included, as construction was not finished at the time of writing.

The map is built using the Leaflet plugin. Map data is pulled from OpenStreetMaps. Map data is not stored locally and is pulled live from the internet using the Leaflet API. The plugin EdgeBuffer forces the map to load a few tiles that are off the screen to help minimize loading. Routing is done through Mapzen. There is a limit of 50000 routes per day and 2 routes per second. The key is included in the code. An internet connection is required for routing to function.

This module contains 3 modals. The BuildingInfo modal is the popup when a building is clicked. This contains the building’s address and thumbnail. The “Interior Maps” button opens the BuildingInterior modal, which is a list containing links to all of the available floor plans. The links take you to the school’s website, which requires a log-in. The “Take Me There” button initiates routing. If you are outside, the app takes you to the closest entrance of the chosen building. If you are inside, it shows you the closest entrance of the building you’re in as the starting point. The final modal, BuildingDropdown, is accessed by clicking the floating action button. It contains a list of buildings on campus. Clicking on one will open the BuildingInfo modal.

*Known Issues*

-[iOS] The building polygons do not render.

-[Android] Some users may need to active the locate control manually before routing will function.

**Bus Routes**

This module has a similar set-up to the Campus Map module. It uses Leaflet and OpenStreetMaps to draw the map. The bus routes are stored in json and geojson files under www/assets/json. For functionality, the Bus Routes Module shows all bus stops for the selected route, how long until the next bus arrives, and an estimation for the bus’s current location.

*Known Issues*

-[iOS] Some of the elements for the bus routes, including the bus location, do not render.

**My Notes**

The purpose of the My Notes module is to allow users to write and read their own notes. These notes may store any content they so choose for as long as they need. Examples could be reminders, lists, or schedules. After starting the module, all notes presently made show in a scrollable list. Tapping of a note shows its details in the node editor. A new note can also be started by tapping the pencil in the bottom-right corner.

From the note editor, a user may type a note title, and their notes. The checkmark indicates to save it. The trash can (if available) will delete the note after confirming with the user. The X indicates to close the note editor without saving changes. Note titles cannot be duplicated. If an error occurs, the user will be notified and be given a chance to fix it.

More specific function documentation is in the source code.

**Quick Links**

The purpose of the Quick Links module is to provide quick access to common UWSP pages as well as any custom links. When the module is loaded, all UWSP links and custom links are shown in a scrollable list. Stars near the custom links indicate they are custom links. A gold star indicates a link is favorited. The links are sorted by favorite status, and then by time of last update.

To visit a link, simply click on it. The device’s default tool for loading a website, often a browser, will open with it.

To create a new link, click the plus. This will launch the quick links editor. If links are present, one can also edit already made links by clicking on the pencil on the middle right, and then the link to edit.

From the quick link editor, one can assign a title for the link, as well as its URL. For example, Google and http://www.google.com . After clicking save, the app will attempt to verify the website to make sure it exists, and if appropriate, is given the right protocol. If it cannot find it or the website doesn’t exist, the user will be warned of the issue. If the title already is in use, the user will be warned of the issue. In each case, the user will then be given the opportunity to fix it, and continue. Unverified URLs can also be forced in by the user’s request.

Warning: Websites that do not begin with the standard http:// or https:// are not reliably supported.

**Athletic Events**

The purpose of the Athletic Events module is to give a list of all upcoming athletic events for the campus. On the loading of the module, a scrollable list of all known events is shown. If the events couldn’t be loaded online, the last known copy is shown. A warning to the user is shown in this situation.

Clicking on an event shows the time it takes place. The user may also star events to move them to the top for faster reference. A toast message indicates this to the user.

The user may also add a filter to the event list by clicking on the magnifying glass in the bottom right corner. From here, the user may choose to filter by their star state, the date of the event, sport, and gender.

More specific function documentation is in the source code.

**Dining Hours**

The Dining Hours module displays each food location on campus: DUC Food Court, The Red Vest, Homegrown, Food for Thought, Upper Debot, Lower Debot. The Fork and Knife icon will be gold when the location is open and grey when it is closed. The module refreshes every 30 seconds to keep current with the time. The information on this module is not live. Rather it is hard coded in the dining.controller.js file.

The DayTimes arrays hold information [weekday, open hour, closed hour] and it goes by military time. For example if the Food Court is open from Monday 8:00AM to 10:00AM and Monday 11:00AM to 2:00PM. The array would look like the following:

var foodCourtDaysTimes = [

[1, 8, 10],

[1, 11, 14],

];

The Day Saysings is the times the location is open that day, 0 = Sunday, 6=Saterday.

var foodCourtDaySayings =

["Sorry, we are closed today",

"8:00AM to 10:00AM<br />11:00AM to 2:00PM",

"8:00AM to 10:00AM<br />11:00AM to 2:00PM",

"8:00AM to 10:00AM<br />11:00AM to 2:00PM",

"8:00AM to 10:00AM<br />11:00AM to 2:00PM",

"8:00AM to 10:00AM<br />11:00AM to 2:00PM",

"Sorry, we are closed today"];

Week Times is the weekly times a location is open.

var foodCourtWeekTimes="Monday to Friday<br />8:00AM to 10:00AM<br/>11:00AM to 2:00PM";

**Useful Tools (usefulTools.js)**

The Useful Tools is a small library of functions for general purposes that are used throughout the app. Use outside of the app for your own purposes doesn’t require any licensing or permission as far as I know. Sources are also listed, and I would recommend including that.

These tools include: escaping text for HTML use, unescaping text for HTML use, preparing the system for neat Ionic popups, making a simplified neat popup, clearing an array of all null values, testing a URL for existence, getting the best form of a URL provided, formatting a time in a neat form, formatting a date to YMD form, formatting a date to YMDHMS form, formatting a date to a neat readable form, getting the next date that is a specific day of the week, get the first day of the next month, get the first day of the next year, and navigating to a URL to an external browser.

More specific function documentation is in the source code.

**Offline Storage (offlineStorage.js)**

The Offline Storage is a small but useful library that provides functionality to read and write data for long-term storage. It utilizes the local web storage system, allowing it to work on both regular websites and hybrid apps like the Pointer Toolkit.

Three functions are offered for this. This includes one for initializing the storage, and verifying storage is functional. The next allows saving some data by its content. This includes automatically handling JSON and string processing when the key name (like file name) ends with .json. The last function is for loading an object. Like saving, function includes translating the file as a JSON. In addition, there is a parameter for a default return. This is in place for when the file retrieval fails.

More specific function documentation is in the source code.

**GitHub**

Current GitHub repository address: <https://github.com/NKubley/UWSPToolkitApp>

Git documentation - <https://git-scm.com/documentation>

**Getting code from GitHub**

The best way to do this is to clone a completely new copy of the project to your local drive, this ensures the the git files for tracking changes to the project will be up-to-date and will capture any changes you make.

**Command to clone from GitHub**

Set the working directory to the location in which you want to clone the project directory and it’s contents. Ex. Desktop. The project directory will automatically be created and the entire contents of the project will be downloaded.

*git clone https://github.com/NKubley/UWSPToolkitApp*

**Command to push code changes back to GitHub repository**

*git commit -am “example message text”*

This creates a commit file for all of the files that you have modified locally since the last pull from the repository. It also lets you put a message associated with the commit; always do this it will help you manage the repository immensely.

*git push origin master*

This pushes the commit file that you have created back to it’s origin(being tracked by the local git file) and to the master branch. If you are pushing to a separate branch, you will want to replace “master” with the name of that specific branch.

If you have edited files locally that a teammate has also edited and they have committed their changes to the repository, you will have to merge your changes with those that already exist. This topic will not be covered in this documentation as there are too many variable involved. Refer to the Git documentation at the top of this section.