Michigan State University

Team Michigan State University

Spartan Experience App

Project Plan

Fall 2017

Michigan State University Staff:

E.J. Dyksen

Tyler Olsen

Spencer Ottarson

Team Members:

Ryan Johnson

Nayana Kodur

Patrick Pale

Roy Perryman

Scott Swarthout

Table of Contents

1. Executive Summary

Michigan State University (MSU) is a public research university located in East Lansing, Michigan. With a total enrollment of 50,000+ students, MSU is amongst the ten largest public universities in the United States. The university strives to provide outstanding education at the undergraduate, graduate, and professional level and aspires to advance outreach and engagement through activities that lead to a better quality of life for individuals and communities. To enhance students' experiences, Michigan State University focuses on providing them with the technological resources needed to support their education and campus engagement.

As technology continues to enhance the daily lives of individuals, an idea was recently proposed by Michigan State University's IT Services department to develop a user-friendly mobile application that enhances the student, employee, faculty, and visitor experience by providing the information they would most like to see. Imagine you were an incoming student. Would it not be reassuring to have directions from Wilson Hall to Berkey Hall for your 8:00 AM IAH course at the tip of your thumb? Would it not be nice to know what engaging events MSU has planned for students next Sunday? What about knowing what is for dinner at The Vista at Shaw later today? These are some of the key features that the Spartan Experience Application contains.

The Spartan Experience Application is a mobile application that provides the user with information regarding building and parking lot locations and hours, dining hall menus and hours, campus events and news, as well as context-sensitive information that is personalized based on the user's location and time of the day. Users can also use their current location to obtain directions to locations on campus. The application has the concept of a role for each user. If the user is a visitor, student, employee, or faculty, the app provides additional functionality that is specific to their needs. Students can view grades and class schedules, and employees and faculty can view the university org chart. Logged in users can also store their preferences in the app, and the experience is then catered to their interests and dietary restrictions. The Spartan Experience Application is available for both iOS and Android Platforms.

2. Functional Specifications

The primary goal of the project is to develop a mobile application that enhances the student, employee, faculty, and visitor on-campus experience at Michigan State University. Many large universities have a university app that enables easy navigation of their campuses, connects users to ongoing and upcoming events, and provides up-to-date information about the university. Michigan State University has decided to develop their own application that provides these services, as well as solves problems unique to users on its campus. This application assists new students, existing students, faculty, employees, and visitors by providing relevant MSU campus information.

Building and parking lot locations and hours are provided in a user-friendly format. Navigation services are also be offered to the user, so that they may easily navigate between locations on campus. For example, a student may want to utilize the application to obtain walking or biking directions and an estimated time of arrival from their dorm building to the location of their class. Or perhaps a visitor may want driving directions to the nearest guest parking lot. The Spartan Experience Application mitigates these requests efficiently.

Additionally, all dining hall menus and their hours can be viewed in one consolidated location. Users do not have to track down the cafeteria menus of each cafeteria individually and are able to see menus for future days as well. Those who are vegetarian, and vegan can filter the menus to show only vegetarian or vegan options. This feature allows users to efficiently determine where and when they want to eat each meal.

The application advances student engagement by providing the user with information regarding campus events and a Twitter feed containing campus news. Many students are unaware of the large amount of campus events, many of which they might be interested in. Integrating an events calendar with the application provides users with information about campus events they may not have known about otherwise.

Another aspect of the application is to provide the user with context-sensitive information based on their location and the time of the day. For instance, if it is approaching dinner time and the user is close to Shaw Hall, the application's Home Screen displays an option to view the menu for The Vista at Shaw. If the user happens to be walking by the Eli and Edythe Broad Art Museum, the application's home screen may display information about an event happening there later in the day.

Since different users may have different types of needs, the application considers the user’s role within the university: visitor, student, employee, faculty. For example, a student can view their class schedule and grades. They also see their next class for the day on the home screen. Employees do not see a class schedule, but can view the university organization chart. Faculty members can also view their class schedule. If a user belongs to multiple roles, they see the information relevant to all those roles. The application detects the appropriate role for the user when they login with their MSU net ID.

The first time a user logs in, they are presented with a series of prompts for their preferences on which events and twitter feeds they find interesting, as well as their dietary preferences. The application stores

this information and uses it to personalize the app to them. This data is also stored in the app's back-end, so if the user switches devices, the app preserves their preferences. If a user would like to change their preferences, they can do so at any time.

The mobile application is available on both iOS and Android platforms. Michigan State University's IT Services Mobile Application team hopes to expand this project for the future. The team is planning for this application to be a platform for other MSU services to build on. In the future, other departments and services can add functionality as plug-ins to the current architecture.

3. Design Specifications

3.1 Overview

The Michigan State University mobile application is designed for both iOS and Android platforms. A user-oriented application that is simple, visually appealing, consistent across all platforms, and useful makes this application successful. No matter the type of user, curated information is shown to them on the home screen, so that upon opening the application, the user is presented with what is relevant to them. Nearby dining halls, upcoming events, and scheduled classes are among the types of information the app presents to the user.

3.2 User Interface

Both Android and iOS mobile applications follow their specific design standards. Android focuses on a Material Design which is a design language that incorporates a physical sense to the UI, includes responsive animation, depth effects with shadows and light, and smooth transitions. IOS focuses on clarity, deference, and depth. Overall an application with fluid motion, clean and crisp text and icons, and distinctive layers between objects is presented to the user.

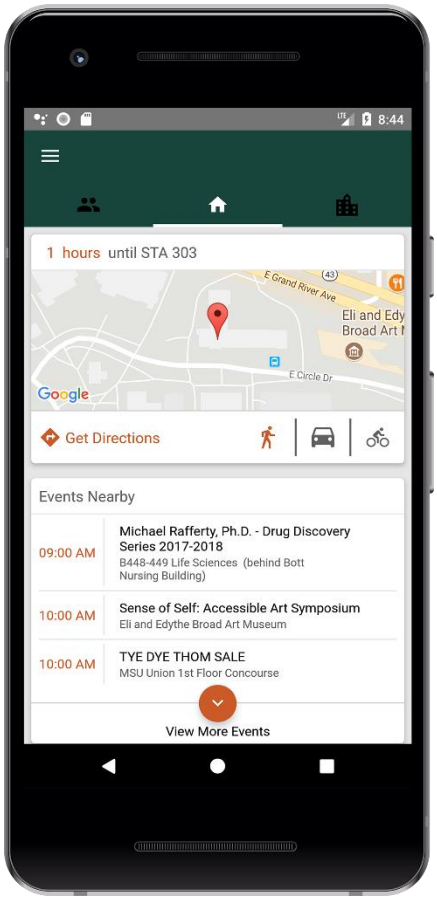


Figure 1.1

Home Screen (Android)

Figure 1.1 above displays the Android application after launching. The user is taken to the “Home” tab, indicated by the home icon. If the user of the application has logged in and is a student, the “Next Class” functionality is displayed to the them. The student is shown the time their next course begins and has easy access to start directions to their class by choosing the type of transportation and tapping the “Get Directions” text.

Below the “Next Class” section is Michigan State University related events organized by what is closest to the user. In the case there are no events for the day, events for the following day are be presented to the user. Upon scrolling the tabs are hidden to allow for better use of screen resources. The next section available to the user is be “Dining Halls Near You”. The three nearest dining halls appear along with the option of viewing their menus. Lastly, the home screen contains a section that displays three menu options from the user’s nearest dining hall.

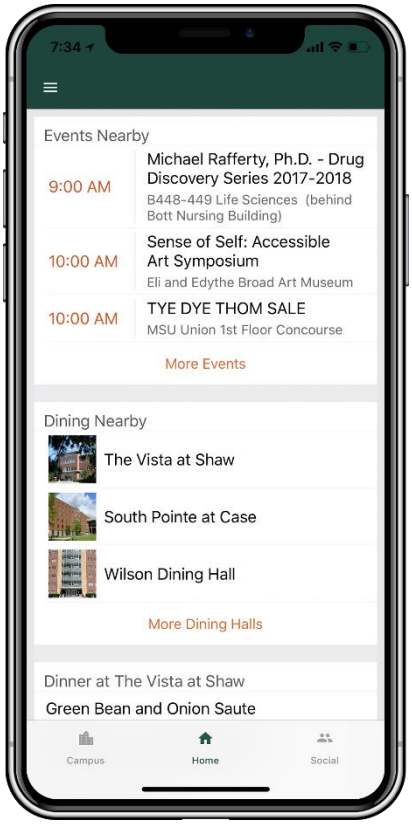


Figure 1.2
Home screen (iOS)

Figure 1.2 shows the user interface of the home screen for iOS. Design standards are followed to suit iOS human interface guidelines. As can be seen in the differences between Figures 1.1 and 1.2, the tab

layout used in Android is switched to a bottom toolbar in iOS. Additionally, the material design cards used in Android are turned into simple views stretched entirely across the screen for iOS.

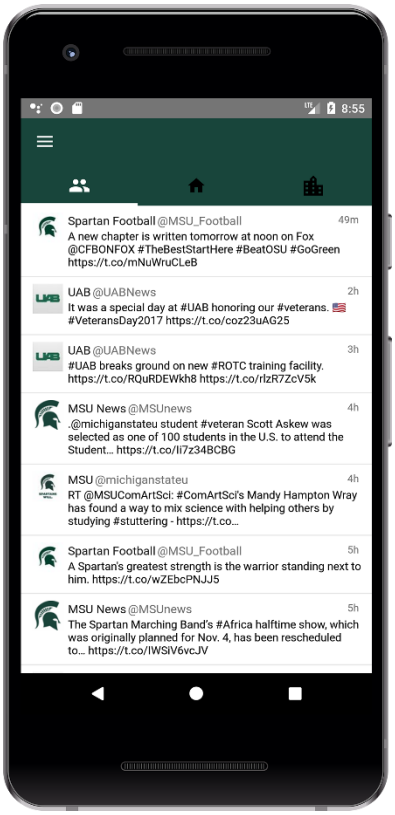


Figure 1.3
Social tab displaying tweets from various MSU twitter accounts

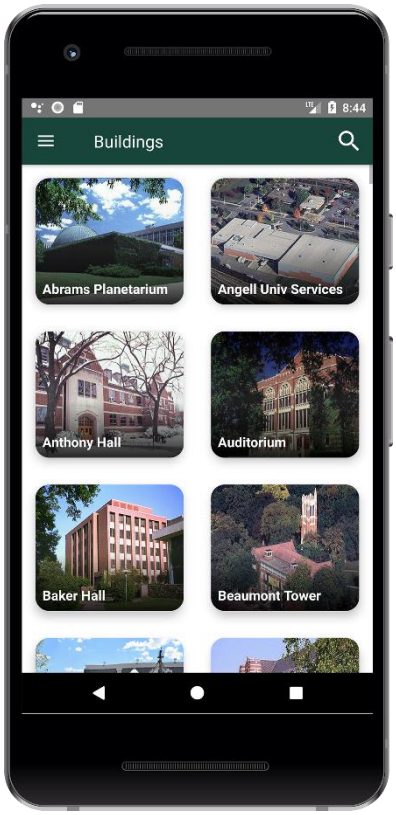


Figure 1.4
Buildings View displaying a scrollable list of MSU buildings

The social tab shown in Figure 1.3 is meant to bring a variety of news to the user from different Michigan State verified twitter profiles. The feed is included to improve the spread of news to students, faculty, employee, and general visitors to Michigan State University. Users are able to personalize the feed by selecting which Twitter accounts they would like to see by setting their user preferences.

Figure 1.4 displays the Buildings View of the application. It is composed of a scrollable list of MSU buildings listed in alphabetical order. Tapping on a building takes the user to a page with additional information about the specific building.

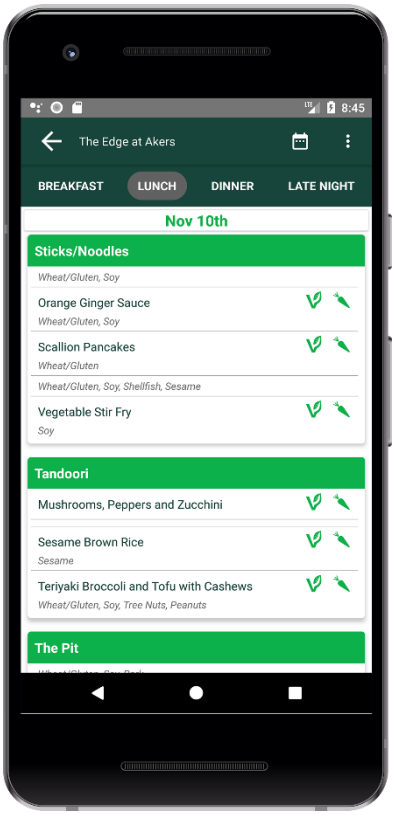


Figure 1.5
Dining Hall Menu View

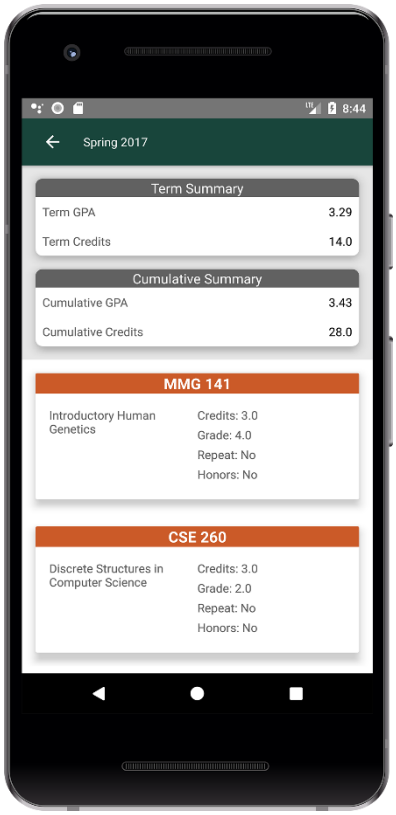


Figure 1.6
Grades View

Figure 1.5 shows the dining hall menus view for the application. Users can select different dining halls to see what the menu for a specific meal is. Menu items are sorted by the cafeteria station they belong to and include allergy/dietary information.

If the user is a student, they have access to the grades view. The grades view contains grade information for a student's previous semesters and is organized by semester.

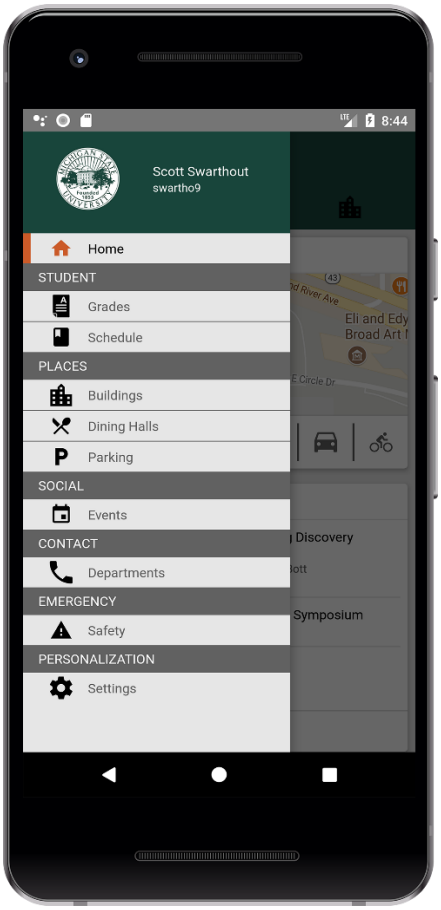
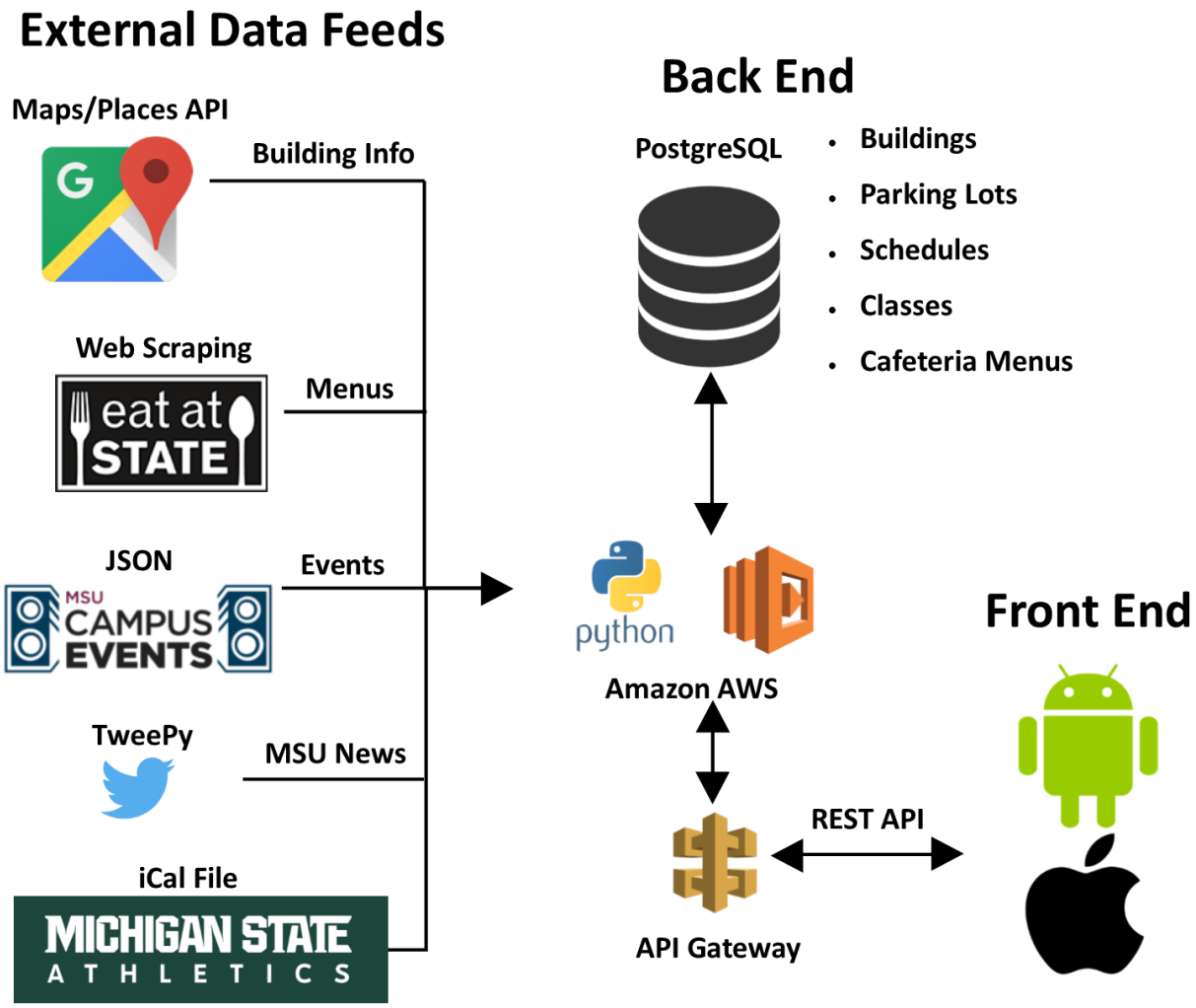


Figure 1.7
Opened navigation drawer in the application

The above figure (Figure 1.7) shows the navigation drawer of the application. The navigation drawer provides access to other features not directly available from the main dashboard's three tab sections. Features available in the navigation drawer are settings, building locations and hours, dining hall menus and hours, campus events, parking lots, department contact information, and emergency phone numbers. The user in the figure above is a student so two additional features (Grades and Schedules) are added to the navigation drawer.

4. Technical Specifications

4.1 System Architecture



4.2 System Components

4.2.1 External Data Feeds

The app ingests data from many sources, both from MSU and from third party services. Each data feed provides some functionality to the app, and our architecture allows for additional data feeds to be added in the future.

4.2.1.1 Google Maps/Places API

The Google Maps and Google Places API is used for building information, maps, and directions. For each building, the information provided by MSU such as the building name and daily hours is joined with Google maps information such as the GPS coordinates, the street address, and any photo URLs Google Maps has for the location.

4.2.1.2 Eat at State

MSU Residential and Hospitality Services (RHS) provides a website where all the menus for the cafeterias are posted. A web scraping bot written in python visits every cafeteria's menu once a day and scrapes it for all the menu information. This scraper uses the Beautiful Soup python library for making requests and parsing the html pages. This data is stored in the database for quick access from the mobile app. The web scraper runs in an AWS Lambda function and is scheduled to run once per day.

4.2.1.3 Campus Events

MSU publishes an events feed of all events on campus. This is provided as an RSS feed. Each time the mobile app requests events from the back-end, a lambda function is executed which fetches the XML from the RSS feed, converts it into JSON, and returns it to the client. This API allows the mobile clients to get events for a specific day, or for a specific category of events. The events feed is segmented into several categories, including academic calendar, performing arts, and museum exhibits.

4.2.1.4 Twitter

The python backend uses the tweepy library for connecting to the twitter API. The app aggregates the twitter profiles of notable MSU accounts, including UAB, Eat at State, and The State News. The application provides an authentication token to the twitter API, and queries for recent tweets from each MSU profile.

4.2.1.5 Michigan State Athletics

The MSU campus events RSS feed does not provide information for sporting events. This is provided by Michigan State Athletics. The calendar is available as an iCalendar file. A python script in a Lambda function fetches the calendar, parses it, and stores it in the database in the same table as the campus events. This allows the app to easily present athletic events along with all other events.

4.2.1.6 Employee Organizational Chart

When an employee logs in to the app, they can view the MSU org chart. To power this feature, MSU IT services traversed Active Directory to recursively fetch all employees and their direct manager. This data is processed and stored in the python backend as a python pickle object, and employee details can easily be fetched on each request.

4.2.2 Back End

4.2.2.1 Python API layer

A Python web server interfaces the external data sources and the mobile applications with a convenient REST API. The project is deployed with AWS Lambda, a product which allows for code to be executed whenever a new request arrives from the mobile apps, but without a traditional virtual machine running all the time. This simplifies the system administration that needs to be done to maintain and scale the application, since AWS Lambda automatically scales the application by running code in response to each trigger. The code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload.

4.2.2.2 PostgreSQL

This application uses PostgreSQL as the database for several reasons. First, it is well supported in the AWS ecosystem, and AWS RDS (Relational Database Service) allows applications to run Postgres databases in their managed environment. Also, Postgres is useful because the data model for the product fits well into a relational model, so SQL provides an easy way to query for data. Finally, there are robust, well documented, and well supported libraries and tooling for using Postgres with Python.

4.2.2.3 API Gateway

To use Lambda functions as a REST API, an API Gateway must be set up in front of the Lambda functions. API Gateway is a product from AWS where Lambda functions are assigned to HTTP endpoints. When a request is made a one of these endpoints, the parameters of the request are forwarded to the Lambda as function arguments. API Gateway spawns a new Lambda function for every request, so if multiple requests arrive concurrently, they are all processed in parallel.

4.2.3 Front End

4.2.3.1 Android

The Michigan State University Android application is developed on Android Studio 3.0 and targets devices using SDK level 21+. It follows design standards based on Material Design founded by Google. A clean user interface gives the user a much better experience across the application. Other than design, the android application consumes multiple SDKS's to improve the quality of code produced and allows easier adaptability for teams taking over the project. Dagger2 is a dependency injection framework that allows your code to be modular. It takes away the uncertainty and stress of creating dependencies throughput the application yourself. Not only does it enhance the dependencies of objects, but it allows for more efficient and easier unit testing.

4.2.3.2 iOS

The Michigan State University iOS application follows the design requirements documented in Apple’s iOS Human Interface Guidelines. The application is developed using the most recent versions released which include iOS 11, Swift 4 and Xcode 9 beta. Like the android version, a clean and user-friendly interface is developed. Additionally, the application is be compatible with all iOS devices.

4.2.3.3 Android & iOS

Both mobile applications connect to the same services provided by Google. The two API's we are consuming from Google are Maps and Places. The Maps API provides information about a certain location on Michigan State's Campus and give us the opportunity to customize information on a map fragment. The Places API provides fixed responses in the form of JSON. The Places API provides photos of places, short summed up information, and allows querying that automatically places markers on a map fragment that decreases the amount of programming required.

4.2.4 Hardware Components

- AWS Lambda
- AWS API Gateway

4.2.5 Software Components

- iOS App Developer (Xcode 9) with Swift
- Android App Developer 3.0 with Java
- Python 3.6
- PostgreSQL Database

4.3 Testing Plan

In preparation for a public release, there are several staged test releases where feedback on usability of features, performance, and bugs are be submitted to the team for improvement. The first stage of testing is internal testing, or “dogfooding”.

Each time a new feature is merged into the master branch of the git repository, the current version of the applications are loaded on to each team member’s phone. The team has access to 2 Android phones and 4 iPhones, so the team can get a reasonable, although not comprehensive impression on the progress of the apps. The app is also shared with the project contacts at IT services for their input. The GitLab Releases feature are used to easily share new versions of the app with the team. Binaries and project files can be included in a release, so the Android APK and iOS IPA files can be included, as well as the source code required to build the app.

Once the app is feature complete, a private beta test can be created. This private beta includes all team members, client contacts, friends, family, capstone students and faculty, and others. This helps the team rapidly receive feedback on issues with the design or bugs in the app. To implement this beta test, the team uses Google Play Beta testing and Apple Testflight. The team collects email addresses of prospective beta tester who can download the app on their own devices.

5. Risk Analysis

There are multiple risks needing to be addressed, and then mitigated throughout the development of this project. These risks vary in difficulty and topic, and are listed below from what was determined to be hardest to easiest.

- **Context-Sensitive Information**
  - **Difficulty:** Hard
  - **Description:** No well-defined list for context-sensitive information categories, and no team experience developing predictive algorithms.
  - **Mitigation:** Present a list of possible categories to our client, and prioritize them. Develop proof-of-concept predictive algorithm.
- **Levels of access to university services**
  - **Difficulty:** Medium
  - **Description:** There are different levels of access between the separate university services used by our application.
  - **Mitigation:** Take an inventory of which services are feasible to use and prioritize them.
- **Beacon Push Notifications**
  - **Difficulty:** Medium
  - **Description:** No team experience working with beacon push notifications.
  - **Mitigation:** Find example applications and implement a simple proof-of-concept application utilizing the technology.
- **RHS RSS Feeds**
  - **Difficulty:** Easy
  - **Description:** RHS just updated the dining hall website, so the RSS Feeds no longer exist.
  - **Mitigation:** We have to implement web-scraping techniques.

6. Schedule

- **Week 1: 8/30 – 9/3**
  - Team assignment
  - Initial team meeting
  - Workstation setup & software installation
  - Create Slack channel to communicate
- **Week 2: 9/4 – 9/10**
  - Initial conference call with Michigan State University client
  - Determine functional requirements
  - Status Report
  - Explore RSS feeds
  - Start Sketch mockups for Project Plan
  - Set up AWS Lambda environment
- **Week 3: 9/11 – 9/17**
  - 9/13 – Status Report Presentation
  - Design system architecture
  - Design iOS/Android Mockups
  - Complete Project Plan
  - Backend: Write dining hall web scraper and Twitter proxy
  - Android: Set up application and design basic layout
  - iOS: Set up application with storyboards
- **Week 4: 9/18 – 9/24**
  - 9/18 - Project Plan due
  - Backend: Add events and buildings to database
  - Android: Events, Emergency Numbers (front end)
  - iOS: Design basic layout
- **Week 5: 9/25 – 10/1**
  - Backend: Add dining halls to database
  - Android: Implement buildings, dining halls, and twitter views
  - iOS: Integrate Apple Maps, and develop buildings, departments and emergency numbers views
- **Week 6: 10/2 – 10/8**
  - Start working on Alpha presentation
  - Begin working on context-sensitive information
  - Backend: Add parking lots to database
  - Android: Add dining hall menus, events and parking and set up geofences
  - iOS: Add dining hall menus and events and set up geofence prototype
- **Week 7: 10/9 – 10/15**
  - Complete and practice for Alpha presentation
  - Backend: Add location information to dining halls and build nearby API
  - Android: Add events information view, design home view and add push notifications
  - iOS: Design home view, social view and event information view and add push notifications

- **Week 8: 10/16 – 10/22**
  - 10/16 – Alpha Presentation Due
  - Backend: Add user and roles API and add schedules to the database
  - Android: Integrate a login screen and roles into the application
  - iOS: Integrate a login screen and roles into the application and set up all geofences
- **Week 9: 10/23 – 10/29**
  - Backend: Add finals and terms to schedules API
  - Android: Add user preferences (personalization after logging in)
  - iOS: Add images into the application and build schedule view
- **Week 10: 10/30 – 11/5**
  - Start working on Beta Presentation
  - Backend: Create org chart API and transcript API
  - Android: Build settings view and schedule view
  - iOS: Add user preferences (personalization after logging in) and grades view
- **Week 11: 11/6 – 11/12**
  - Complete and practice for Beta Presentation
  - Backend: Add user preferences and parking lot information to database
  - Android: Design campus tab
  - iOS: Design campus tab
  - Clean up apps
- **Week 12: 11/13 – 11/19**
  - 11/13 - Beta presentations due
  - Clean up iOS and Android apps, resolve bugs
  - Start working on project video
  - Start Design Day presentation
  - Test: iOS - Testflight, android - through Google Play
- **Week 13: 11/20 – 11/26**
  - Polish iOS and Android apps
  - Continue working on and filming project video
- **Week 14: 11/27 – 12/3**
  - Complete project video
  - Practice for Design Day Presentation
  - Present product to client (Michigan State University)
  - Have all documentation and code ready to be turned over to client
- **Week 15: 12/4 – 12/8**
  - 12/4 – Project Videos due
  - 12/7 - Design Day set up
  - 12/8 – Design Day