

**Secondary Agile Iteration Report**

**Project: Burgerator**

Client: Ammar Shallal

Team: Kevin Haro, Luis Garcia, Alec Michael & Jonathan Hammond

Central Washington University - 12/8/2015

Contents

[Introduction 3](#_Toc435467939)

[Website 3](#_Toc435467940)

[Project Overview 3](#_Toc435467941)

[Project Summary 3](#_Toc435467942)

[Project Client and Stakeholders 3](#_Toc435467943)

[Project Scope 3](#_Toc435467944)

[Project Management Plan 3](#_Toc435467945)

[Project Organization 3](#_Toc435467946)

[Risk Management 4](#_Toc435467947)

[Cost Risks 4](#_Toc435467948)

[Scheduling Risks 4](#_Toc435467949)

[Programmatic Risks 4](#_Toc435467950)

[Hazy Vision 4](#_Toc435467951)

[Team Issues 4](#_Toc435467952)

[Software Development Tools 5](#_Toc435467953)

[Requirements 5](#_Toc435467954)

[Development, Operation, and Maintenance Environments 5](#_Toc435467955)

[System Model 5](#_Toc435467956)

[User Interaction 6](#_Toc435467957)

[Functional Requirements 6](#_Toc435467958)

[Nonfunctional Requirements 7](#_Toc435467959)

[Feasibility 7](#_Toc435467960)

[Use Case Diagram: 8](#_Toc435467961)

[Use Case Scenarios 8](#_Toc435467962)

[Conclusion 13](#_Toc435467963)

# Introduction

Have you ever thought about finding a good or new burger around town? Well, our client Ammar Shallal, thought the same thing and decided to do something about it. He and a group of friends that were on a mission to find the best burgers in New York City founded the **Burgerator** app for iOS. The Burgerator app allows users to rate, find, and keep track of the burgers they’ve had. It’s a wonderfully simple concept, and works well. However, not everyone has an iPhone, and Ammar wanted his brainchild to be accessible by a larger audience. This is where the Hamburgerler’s come in.

Our team will be developing an Android version of the app and will perform some new feature requests from the client. The new features will include more social interaction options for users, such as the ability to follow other users to see what great (or gross) burgers they’re eating. Additionally, we will be looking at performing a database restructure that moves the focus from the burger joints to the burgers themselves, and a few other requirements that will heighten the user experience of the Burgerator.

# Website

Our website is <https://trello.com/hamburgerlers>

# Project Overview

## Project Summary

Everyone has asked the question, “What is the best burger in town?” Whether it is your hometown or a town you are visiting, asking what the best burger available is a very common question. The Burgerator app gives the user the answer with one click of a button. The app will give the user a list of the best burgers in order from best to worst. Along with the name and location of the business that serves the burger. This will be developed as an Android mobile application.

## Project Client and Stakeholders

Our client is Amar Shallal. He currently has Burgerator available for IOS download. Ammar also has a database for user to input their burger ratings while the Android app is completed.

## Project Scope

Our software will make it simple for the user to find the highest rated burger in their town. The app will show the location of the restaurant where the user can purchase the burger along with an image of the burger. The app will also display the user information of whomever rated the burger. The ratings will need to be imputed by the user. As well as the image of the burger.

# Quality Assurance Plan

Document Standards

The document standards extend the styles and format currently expelified by this document. The primary font for our team is “Rockwell (Headings)”. The font size for documents is 12 point for content, 14 point for sub headings and 18 point for headings.

Presentations share the documentation font but not the font sizes. The font size for presentations is variable as to maximize the readability for the audience. In addition to readability, key principles of presentations are consistency, emphasis on illustrations, and an active narrative.

Coding Standards

Google general coding standards

Android specific coding standards

User Interface Guidelines

Replicate iOS Burgerator

Simplicity

Consistency

Change Control Process

Change is natural and encouraged in an agile methodology such as ours. Because of this, the plan to adapt to change is to talk about changes as a group and definitively step in a direction based on the change discussion. The plan of attack against requirement, or scope, creep is to properly define the boundaries of the levels of out platform specified in the requirements section.

Testing Process

The testing methodologies our group employs are ad hoc testing, system testing, and unit testing for any code developed by our team. Third party libraries will not be tested.

Methods for ad hoc testing will be used to identify obvious bugs. Our ad hoc testing procedures will consist of distributing the current application to the development team and attempting to identify bugs with the application.

Methods for system testing will include testing the different modules in the Burgerator system. Specifically, this is ensuring that the Burgerator engine can handle user authorization, database connectivity, and yelp API connectivity.

Methods for unit testing will include testing all source code written by our development team at a unit level. This will be facilitated by the use of java testing suites such as jUnit testing.

Client acceptance testing will be incrementally evaluated via content demonstrations. The demonstrations will occur as frequently as our client would like, or tri weekly.

# Project Management Plan

## Project Organization

We have chosen to use the agile software development process. Consisting of a series of short iterations, each ending with an update of some form delivered to the client. The agile process will allow flexibility and easy change while all team members are on the same page, equally informed and applicable for effective risk management.

The Hamburgerler’s consists of Team Lead Kevin Haro, Quality Assurance Lead Jonathan Hammond, Documentation Lead Alec Michel, and Design Lead Luis Garcia. We will meet as a team for twenty minutes three times a week, and two scrum meetings for an hour.

Our longer meetings will consist of working individually, planning, troubleshooting, and announcing other issues that are needed.

## Risk Management

We have chosen to use the tips for managing risks as outlined in Practical Tips for Software-Intensive Student Projects. The risks presented are clear, comprehensive, and have good pointers for mitigation

## Cost Risks

The likely hood of risk around cost will be understood by exploring the places our project incurs a cost, and determine our situation. We will clarify with the client, if a budget for web hosting and server space will be planned accordingly. The damage of these risks is very low, since we know what to expect.

## Scheduling Risks

For scheduling, the team has come up with meeting on a daily to weekly basis. Thus communication can be easily accessed verbally. Questions and concerns are appropriately addressed throughout each meeting. However, the only current plan for dealing with the complete absence of a team member is to split the members work among the others. This would be a challenging situation, which is why all team members are strongly informed of each other’s progress. The likelihood of a complete absence risk is low, since work can be completed separately from the group. But the impact is high since there are only four members on the team.

## Programmatic Risks

Programmatic risks are relatively low, since it is a small group project. Group expectations, faculty advising and clients requirements are strongly and clearly stated.

## Hazy Vision

The risk of building the wrong project is medium. Although the requirements are well-established, the risk lies within the complexity of our project. Since the team is working off of a previous projects work, the results of the previous project are different from this current project. That is why contact with the client is critical for a clear understanding.

## Team Issues

The risk of problematic team members is high. The team has done very well to reduce the risk. Our team is well open to communication, and has agreed to be open to change. But the main problem is the size of the team. A mere four-person group is very small for a project of this unknown proportion. This means that team members will need to pull their own weight. This is how controversy arises when a member feels like they are doing more work than the others, etc. The plan is to pair program as needed, so that each team member can work to the best that they can.

These are the main risks addressed. Overall, the team is open to confronting new risks that develop.

## Software Development Tools

The Team is using Trello for issue tracking with a Kanban method of focus, GitHub for version control, Trello and DropBox for document sharing, and Facebook for project scheduling. Current knowledge, simplicity and easy access are reasons why the team chose these development tools. Trello and Facebook are particularly easy to access, since they have mobile apps that come with the tool.

# Requirements

## Development, Operation, and Maintenance Environments

Android Burgerator will run as a native Android application on Android devices. Development, operation, and maintenance will utilize physical Android phones as well as virtual Android emulators. Minimum and target Android API’s will change based on the technological needs of the application combined with maintaining the maximum possible user base.

## System Model

* + Existing high-level system view:



Figure : IOS Burgerator

* IOS Burgerator is what currently exists. It is a mobile application that allows users to rate burgers within a geographic location (predominantly used in New York). Android Burgerator Base is intended to be an exact replication of IOS Burgerator(Figure 1).
  + Proposed Systems:

Android Burgerator Base

Android Burgerator Goal

* The proposed system Android Burgerator Base is ideally identical to IOS Burgerator. However, the Android Burgerator Goal is intended to be a more flexible extension of Burgerator that incorporates more social media aspects into Burgerator. For example, allowing you to share burgers with friends.

## User Interaction

* + The user of the mobile application Android Burgerator Base is to allow individuals to visit a restaurant, take photos of a hamburger they have ordered, leave a rating for said burger, and other features related to rating burgers.
  + The use-case diagram and scenarios describe the interaction between the user and the mobile application.
  + Refer to the use case diagram and corresponding scenarios(Figure 3)

## Functional Requirements

* + Allow the user to login (Use case ‘Login to Burgerator’)
  + Login’s can be completed by email, Facebook or twitter
  + Login’s must be bound to each other. One user, can have multiple login credentials.
  + Login’s must be secure and use proper authentication practices.
  + Allow the user to explore the five main windows of the application(Find A Burger, Burger Feed, Burger Rating, Top Burgers, User Profile)
  + Find A Burger: Search for burgers based on GPS location, zip code, or keyword. (Use case ‘Find a burger/restaurant’)
  + Burger Feed: A list of burger reviews that are somehow (location, friends, pervious views, interests) relevant to you. (Use case ‘Browse burger feed’)
  + Burger Rating: A virtual form to complete a review of a burger. (Use case ‘Rate a burger/ Add review’)
  + Top Burgers: A list of the top 10 rated burgers in the world. (Use case ‘Browse burger leaderboards’)
  + Allow the user to control setting, such as location, Facebook posts, linked accounts, and logout, from within the application

## Nonfunctional Requirements

* + Given that Burgerator is location based, there must be access to location or a manual way to enter the location.
  + Constraints that the hardware imposes on the application are the same that other applications have. Memory, data, and battery constraints should be minimal.
  + The portability of the project is apparent given the underlying Android platform. This advantage opens up to application to the majority of the mobile market share.
  + The reliability of the application will rely on the servers that support it.

## Feasibility

* + Although ‘Android Burgerator Goal’ is the vision for Burgerator there exist two significant benchmarks for the development team, Android Burgerator Base and Android Burgerator Base Core (Figure 2).
  + Android Burgerator Base Core:



Figure : Base System Diagram

## Use Case Diagram:



Figure

# Use Case Scenarios

|  |  |
| --- | --- |
| Use Case Name | Login to Burgerator |
| Actors | User |
| Summary | The user logs into the application upon first use |
| Pre-Conditions | 1. User has the application installed  2. Internet connection is available  3. User has an Facebook account, Twitter account, or email to login with |
| Normal Flow of Elements | 1. User opens the Burgerator application  2. User is taken to the Burgerator splash screen  3. User is prompted login info  4. User chooses login account  5. User is logged into Burgerator |
| Error Conditions | 4a. User enters incorrect credentials  4b. User forgets account password |
| Concurrent Activities | 1a. Location is ascertained |
| Post-Conditions | 1. User is logged into Burgerator |

|  |  |
| --- | --- |
| Use Case Name | Find a burger/restaurant |
| Actors | User |
| Summary | Once logged into Burgerator, the user is in search of a burger/restaurant |
| Pre-Conditions | 1. User has the application installed  2. Internet connection is available  3. User location is enabled  4. User is logged in |
| Normal Flow of Elements | 1. User opens the Burgerator application  2. User is taken to the Burgerator home screen  3. User navigates to the ‘find a burger’ tab  4. User sorts by keyword, distance, or rating  5. User browses restaurants  6. User chooses restaurant  7. User chooses burger  8. User goes to restaurant |
| Error Conditions | 4a. No results returned  8a. Restaurant closed or burger no longer served |
| Concurrent Activities | None |
| Post-Conditions | 1. User has found a desired burger |

|  |  |
| --- | --- |
| Use Case Name | Browse burger feed |
| Actors | User |
| Summary | Once logged into Burgerator, the user browses the burger feed |
| Pre-Conditions | 1. User has the application installed  2. Internet connection is available  4. User is logged in |
| Normal Flow of Elements | 1. User opens the Burgerator application  2. User is taken to the Burgerator home screen  3. User navigates to the ‘burger feed’ tab  4. User browses other reviews  5. For every review, the user can:  View that review  View the review’s respective restaurants  View the review’s picture  ‘Pound’ the review  6. User continues to browse the burger feed |
| Error Conditions | 3a. Burger feed does not load |
| Concurrent Activities | None |
| Post-Conditions | 1. User has viewed rated burgers |

|  |  |
| --- | --- |
| Use Case Name | Rate a burger/ Add review |
| Actors | User |
| Summary | Once logged into Burgerator, the user attempts to review a burger |
| Pre-Conditions | 1. User has the application installed  2. Internet connection is available  3. User location is enabled  4. User camera is functional  4. User is logged in |
| Normal Flow of Elements | 1. User opens the Burgerator application  2. User is taken to the Burgerator home screen  3. User navigates to the ‘review’ tab  4. User chooses restaurant  5. User takes a picture of the burger  6. User rates the burger  7. User adds comments  8. User can share on Facebook and twitter  9. User submits rating |
| Error Conditions | 4a. User cannot find restaurant |
| Concurrent Activities | Content may or may not be posted to Facebook and twitter |
| Post-Conditions | 1. User has rated a burger |

|  |  |
| --- | --- |
| Use Case Name | Browse burger leaderboards |
| Actors | User |
| Summary | Once logged into Burgerator, the user browses the burger leaderboards |
| Pre-Conditions | 1. User has the application installed  2. Internet connection is available  4. User is logged in |
| Normal Flow of Elements | 1. User opens the Burgerator application  2. User is taken to the Burgerator home screen  3. User navigates to the ‘top 10 burgers’ tab  4. User browses top burgers  5. For every top burger, the user can:  View the top burger reviews  View the review’s respective restaurants  View the review’s picture  6. User continues to browse the burger feed |
| Error Conditions | 3a. Burger feed does not load |
| Concurrent Activities | None |
| Post-Conditions | 1. User has viewed top burgers |

|  |  |
| --- | --- |
| Use Case Name | Browse personal profile |
| Actors | User |
| Summary | Once logged into Burgerator, the user browses their profile |
| Pre-Conditions | 1. User has the application installed 2. Internet connection is available 4. User is logged in |
| Normal Flow of Elements | 1. User opens the Burgerator application 2. User is taken to the Burgerator home screen 3. User navigates to the ‘profile’ tab 4. User views their Burgerator rank (Squire etc…) 5. User browses previously rated burgers 6. For every top burger, the user can:  View the top burger reviews  View the review’s respective restaurants  View the review’s picture |
| Error Conditions | None |
| Concurrent Activities | None |
| Post-Conditions | 1. User has viewed their profile |

|  |  |
| --- | --- |
| Use Case Name | Manage database |
| Actors | Database Administrator (DBA) |
| Summary | The database administrators role is to clean garbage inputs from the system, modify the relational schema, and otherwise maintain the database |
| Pre-Conditions | 1. The DBA has access to the database  2. The DBA knows how to access the database |
| Normal Flow of Elements | 1. The DBA has the ability to:  Insert Inputs  Remove inputs  Modify the schema |
| Error Conditions | 1a. Database is unavailable due to hosting problems |
| Concurrent Activities | None |
| Post-Conditions | 1. The database has been maintained |

|  |  |
| --- | --- |
| Use Case Name | Maintain/Modify Android Application |
| Actors | Developer |
| Summary | The developer’s role is to create and maintain the application. |
| Pre-Conditions | None |
| Normal Flow of Elements | 1. The developers have the ability to:  Modify user interface  Modify database connection  Modify yelp api connection |
| Error Conditions | None |
| Concurrent Activities | None |
| Post-Conditions | 1. The application has been maintained |

# Conclusion

In conclusion, Team Hamburgerler is off to a good start. This project is a big one, and started out a bit slow due to having to learn the basics of Objective C, and sorting through a codebase with little to no documentation. However, the team has done well with that and is maintaining cohesion.

There are some potential trouble areas such as maintaining accountability, and scheduling conflicts. Additionally, more time will likely need to be allotted for the project than what is currently being designated. The technology that has to be learned in order to succeed is somewhat high. There are several APIs we must familiarize ourselves with, in addition to learning the Android API. However, we will push through and make a great app for our client Ammar.

All in all, this project is starting to come together and we look forward to the result.