

---

## **Fourth Agile Iteration Report**

### **Project: Burgerator**

Client: Ammar Shallal

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

Central Washington University - 3/14/2016

# Table Of contents

---

<b>Introduction.....</b>	<b>4</b>
<b>Website .....</b>	<b>4</b>
<b>Project Overview.....</b>	<b>4</b>
Project Summary .....	4
Project Client and Stakeholders .....	4
Project Scope .....	4
<b>Project Management Plan.....</b>	<b>5</b>
Project Organization.....	5
Risk Management .....	5
Cost Risks .....	5
Scheduling Risks .....	5
Programmatic Risks .....	5
Hazy Vision.....	5
Team Issues .....	6
Software Development Tools .....	6
<b>Requirements .....</b>	<b>7</b>
Development, Operation, and Maintenance Environments .....	7
System Model .....	7
User Interaction .....	8
Functional Requirements .....	8
Nonfunctional Requirements .....	8
Feasibility.....	9
Use Case Diagram: .....	10

<b>Use Case Scenarios .....</b>	<b>11</b>
<b>Architectural Design .....</b>	<b>19</b>
Section Overview .....	19
General Constraints .....	19
Data Design .....	19
Program Structure .....	20
Alternatives Considered .....	22
<b>Detailed Design.....</b>	<b>23</b>
Data Design .....	23
UI Package: .....	23
DB Package: .....	23
Util Package: .....	23
<b>Quality Assurance Plan.....</b>	<b>24</b>
Document Standards .....	24
Coding Standards.....	24
User Interface Guidelines.....	24
Change Control Process.....	24
Testing Process .....	24
<b>Conclusion .....</b>	<b>27</b>
<b>Appendices .....</b>	<b>29</b>
Appendix A: Navigation .....	29
Appendix B: User Manual .....	32

# INTRODUCTION

Have you ever wanted to find the best or perhaps a 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 is 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 [http://designbot9000.github.io/Burgerator\\_Android/](http://designbot9000.github.io/Burgerator_Android/)

## 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 the individual who rated the burger. The ratings will need to be entered by the user as well as the image of the burger.

# 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:

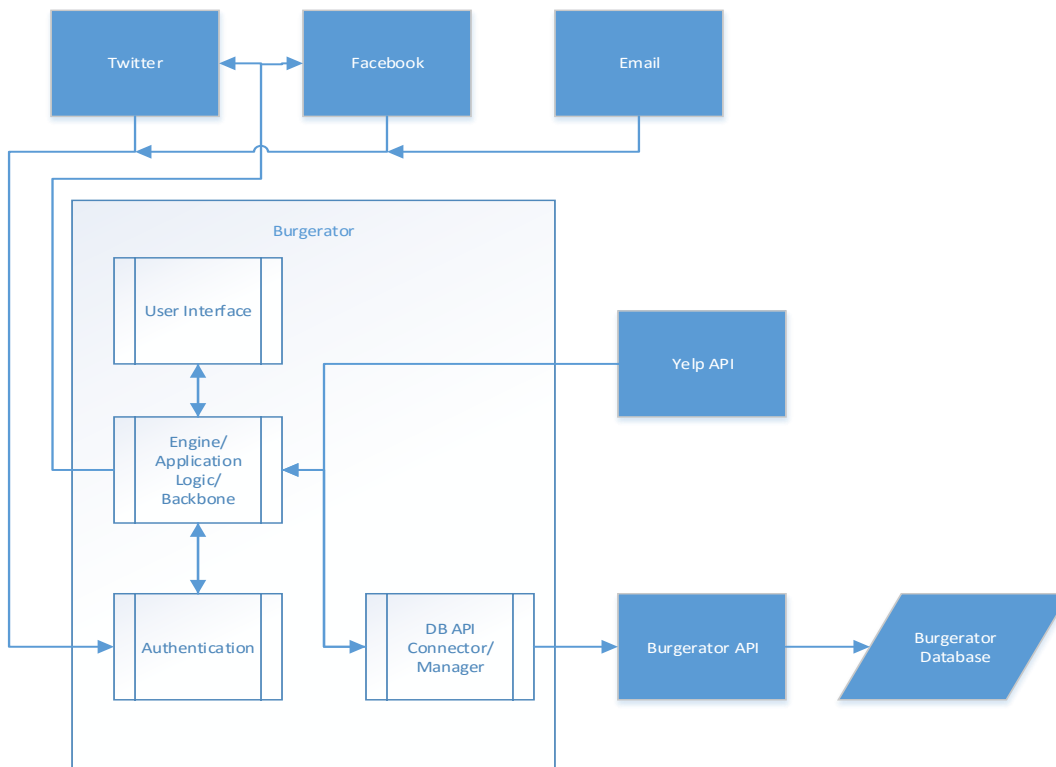


Figure 1: IOS Burgerator high level design

- 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**

- Allow users of the mobile application Android Burgerator Base 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
  - Refer to Appendix A: Navigation
- 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')
  - Logins can be completed by email, Facebook or twitter
  - Logins must be bound to each other. One user can have multiple login credentials
  - Logins 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
- 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
- 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
  - Android Burgerator Core (Figure 2)
- Android Burgerator Base Core:

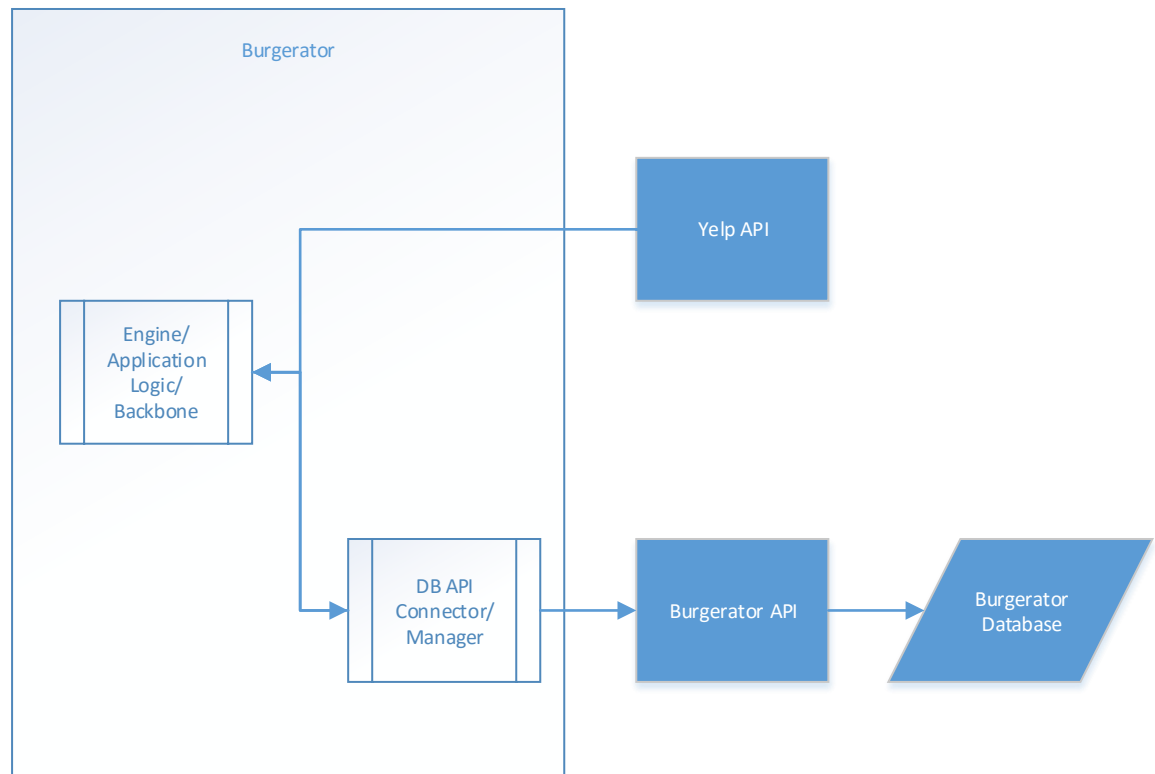


Figure 2: Base System Diagram

## USE CASE DIAGRAM:

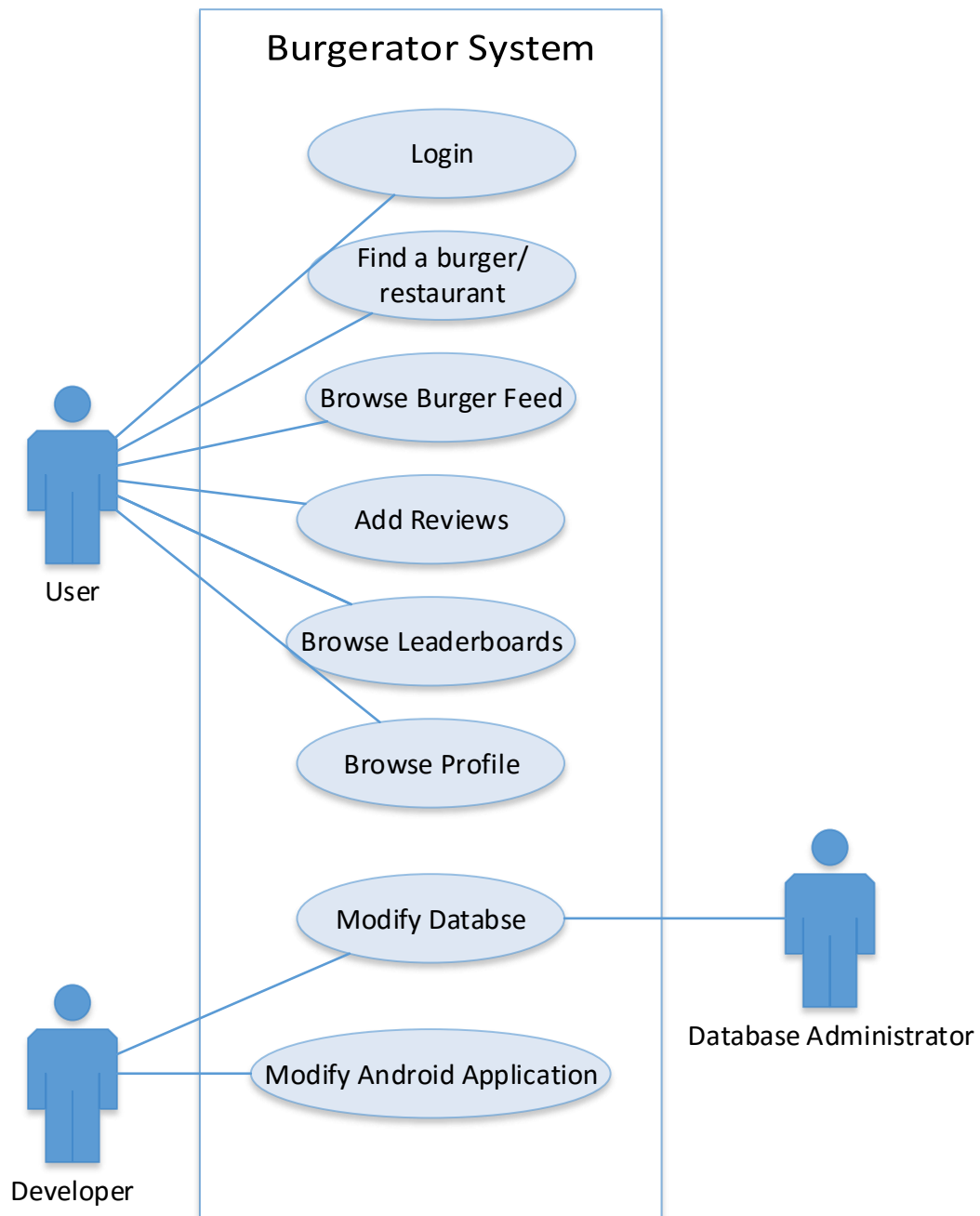


Figure 3 Use cases and actors of the Burgerator system

## USE CASE SCENARIOS

Use Case Name	Login to Burgerator
Actors	User
Summary	The user logs into the application upon first use
Pre-Conditions	<ol style="list-style-type: none"><li>1. User has the application installed</li><li>2. Internet connection is available</li><li>3. User has an Facebook account, Twitter account, or email to login with</li></ol>
Normal Flow of Elements	<ol style="list-style-type: none"><li>1. User opens the Burgerator application</li><li>2. User is taken to the Burgerator splash screen</li><li>3. User is prompted login info</li><li>4. User chooses login account</li><li>5. User is logged into Burgerator</li></ol>
Error Conditions	<ol style="list-style-type: none"><li>4a. User enters incorrect credentials</li><li>4b. User forgets account password</li></ol>
Concurrent Activities	<ol style="list-style-type: none"><li>1a. Location is ascertained</li></ol>
Post-Conditions	<ol style="list-style-type: none"><li>1. User is logged into Burgerator</li></ol>

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: <div>View that review</div> <div>View the review's respective restaurants</div> <div>View the review's picture</div> <div>'Pound' the review</div> 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	<ol style="list-style-type: none"> <li>1. User has the application installed</li> <li>2. Internet connection is available</li> <li>3. User location is enabled</li> <li>4. User camera is functional</li> <li>4. User is logged in</li> </ol>
Normal Flow of Elements	<ol style="list-style-type: none"> <li>1. User opens the Burgerator application</li> <li>2. User is taken to the Burgerator home screen</li> <li>3. User navigates to the 'review' tab</li> <li>4. User chooses restaurant</li> <li>5. User takes a picture of the burger</li> <li>6. User rates the burger</li> <li>7. User adds comments</li> <li>8. User can share on Facebook and twitter</li> <li>9. User submits rating</li> </ol>
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: <div>View the top burger reviews</div> <div>View the review's respective restaurants</div> <div>View the review's picture</div> 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	<ol style="list-style-type: none"> <li>1. User has the application installed</li> <li>2. Internet connection is available</li> <li>4. User is logged in</li> </ol>
Normal Flow of Elements	<ol style="list-style-type: none"> <li>1. User opens the Burgerator application</li> <li>2. User is taken to the Burgerator home screen</li> <li>3. User navigates to the 'profile' tab</li> <li>4. User views their Burgerator rank (Squire etc...)</li> <li>5. User browses previously rated burgers</li> <li>6. For every top burger, the user can: <ul style="list-style-type: none"> <li>View the top burger reviews</li> <li>View the review's respective restaurants</li> <li>View the review's picture</li> </ul> </li> </ol>
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: <div> <div>Modify user interface</div> <div>Modify database connection</div> <div>Modify yelp api connection</div> </div>
Error Conditions	None
Concurrent Activities	None
Post-Conditions	1. The application has been maintained

# ARCHITECTURAL DESIGN

## SECTION OVERVIEW

This section includes the constraints and limitations that the team has bumped into as the project comes along. This includes, but is not limited to, hardware and software constraints. As well as the data design, program structure and the alternatives we considered as a team.

## GENERAL CONSTRAINTS

The global limitations we have encountered include hardware and software constraints, Wi-Fi and cellular bandwidth, meeting performance requirements, and server constraints.

The team has encountered hardware and software constraints when it comes to testing the application. With only two of the team members having android devices, our platform options to run the app on are limited. In order to assure maximum quality, we would need to test the app on all android platforms. However, with the sheer number of Android devices and OS version, this is not feasible for such a small team. The team turned to virtual emulators to test the app on a larger scale. But the software also had its limitations by not including all the available platforms. Also, the UI looks different in the emulators compared to the hardware. This makes it difficult to get an accurate representation of how the UI will look exactly. The emulators also have the tendency to be extremely slow and therefore make it difficult for us to realize how fast the app actually runs.

The team has also encountered Wi-Fi and bandwidth constraints. The app needs Wi-Fi in order to run properly and needs a large amount of bandwidth for the images of the burgers being rated. The team has had think about how we want to deal with the scenario of a user having no internet connection. As well as finding different ways to deal with the extensive amount of bandwidth needed for the images to be stored and loaded.

## DATA DESIGN

The data design for Burgerator is represented by a few different modules. These modules consist of data storage for the database, S3 scalable bucket hosting, Elastic Beanstalk web hosting, and local persistent data on the android device.

The database runs on an amazon web services server that contains an instance of a MySQL database. This server acts as a scalable system to accommodate growing data storage and also throughput querying the database. These are the current specifications of the server instance:

Engine: MySQL 5.6.23

Avalability Zone: us-east-1e

Endpoint: mysqltest.chvc7fpwstop.us-east-1.rds.amazonaws.com

Storage Type: General Purpose

Storage: 5GB

The S3 scalable bucket hosting is a server used for housing images of burgers and restaurants that have been rated. There are two main folders on this server that are meant to hold photos that are used by the application. These folders are “uploads” which contain images of burgers and “restaurants” which contain images of restaurants that users rate burgers at.

Next, our Elastic Beanstalk web application server is used to host the database API that Burgerator connects through to perform CRUD operations on the database. This web application is predominately PHP that facilitates access through a series of endpoint PHP files that act as methods to the API. Our Elastic Beanstalk application also contains an admin panel to maintain the database.

Finally, our Android persistent data is managed by a class that implements the singleton and adapter design patterns to allow data to be quickly accessed, appropriately, and from a single location. Although not currently implemented, our persistent data will store data onto the user’s device and perform CRUD operations on it when necessary.

## **PROGRAM STRUCTURE**

The architectural model is moving from a system where the views and controller is one object towards an architectural model where the views and controller are two distinct objects. At the beginning, we decided to include the controller with the views. As the project is moving along we have come to realize that separating the views from the controller makes the architectural design more extensible.

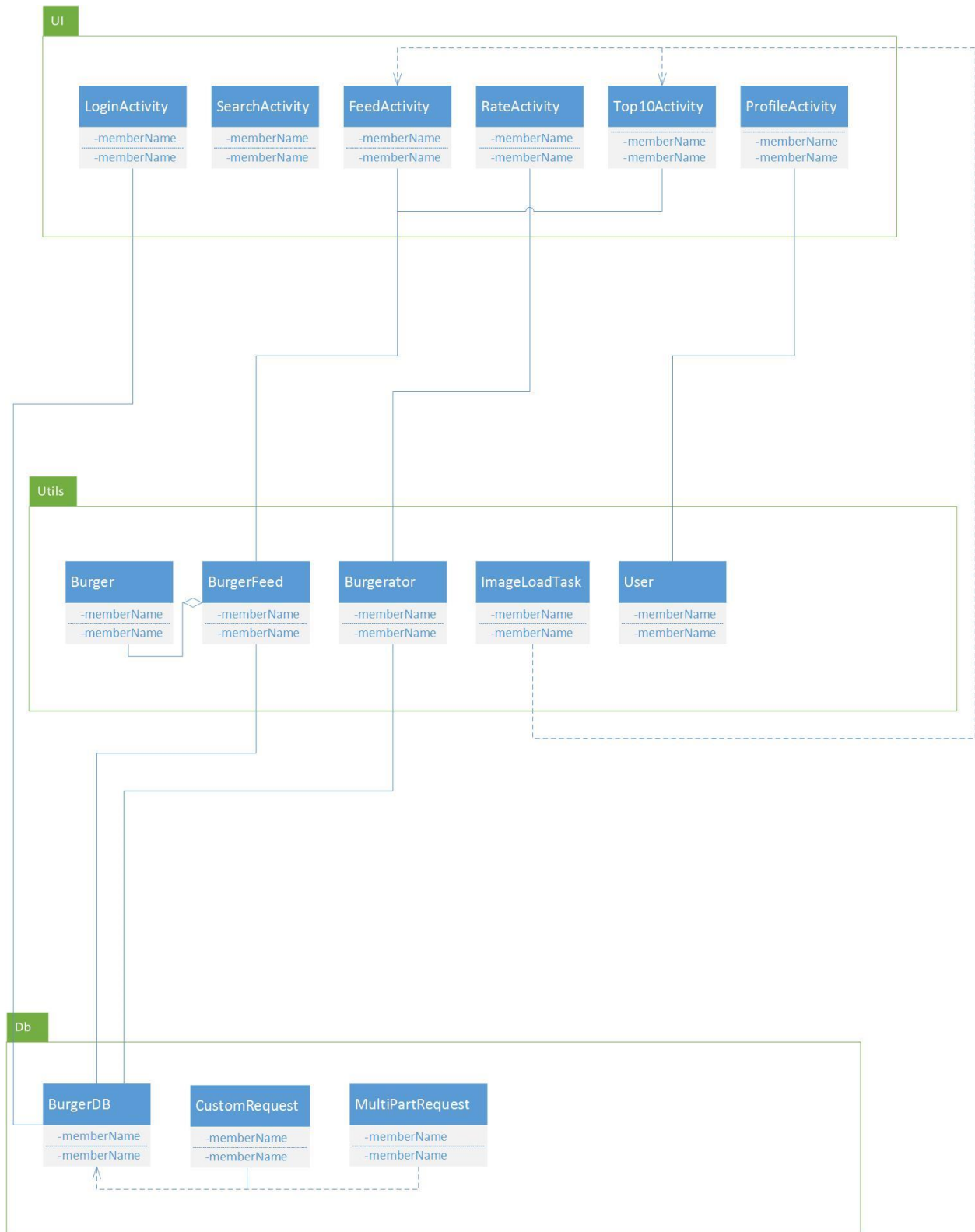


Figure 4: Burgerator for Android - current architecture model

## **ALTERNATIVES CONSIDERED**

At the beginning we considered to not have any architectural model at all and just begin programming. We came to the conclusion that this would create a lot of problems for the team in the future than solutions. It seemed like a good idea at first because we would waste no time getting started on the project, but it would create a variety of problems as the project came closer to conclusion. We decided that setting time aside to develop our model view controller would be more beneficent to the team and thus, will spend a small amount of time refactoring our code.

# DETAILED DESIGN

## DATA DESIGN

This section includes our classes along with a description of the class and description of the included methods.

### UI PACKAGE:

**FeedActivity:** The FeedActivity class loads the burger feed window. The class contains the banner settings as well as the tab buttons settings. The class also holds the specifications for loading and displaying each burger in the feed.

**LoginActivity:** The LoginActivity class loads the login window. The class handles the user's choice for login authentication (Facebook, Twitter, email). The class then takes the username and password and checks it with the server for authentication.

**ProfileActivity:** The ProfileActivity class loads the profile window. The class handles the way the users information will be displayed and retrieve the users rated burgers.

**RateActivity:** The RateActivity class loads the rate window. This class accesses the users camera to take a picture of the burger they are going to rate. It also handles the users rating input and sends the information to the server.

**SearchActivity:** The SearchActivity class loads the search window. It handles the location services and loads the appropriate information for that location.

**Top10Activity:** The Top10Activity class loads the top 10 burgers window. The class loads the information, including the image and burger info, of the top 10 burgers from the database.

### DB PACKAGE:

**BurgerDB:** The BurgerBD class interacts with the database API. It's a utility class that interacts by http request. This class takes the user email and password and verifies the user. This class also handles when the user forgets his/her password. It also retrieves the top 10 burgers content and the burger feed content.

### UTIL PACKAGE:

**Burger:** The Burger class is a container for the burger information that will be displayed. This class holds that information so the app does not need to make http request every time it needs the information.

**Burgerator:** The Burgerator class is a container for the input of the user. This class holds all the data and packages it to be sent to the server.

**BurgerFeed:** The BurgerFeed class is an array that holds the burgers that have been rated. It holds the information for both the burger feed window and the top 10 window.

**ImageLoadTask:** The ImageLoadTask class is used to handle the way a picture is taken and loaded into the burger rate window.

**User:** The User class is a container to hold the users information. This way the app does not need to make http request every time it needs that information.

## **QUALITY ASSURANCE PLAN**

### **DOCUMENT STANDARDS**

The document standards extend the styles and format currently demonstrated by this document. The heading/display font for our team is “Rockwell.” Headings have 18 point font and subheadings have 14 point font. The font size for documents content is 11 point with a font of “Calibri (Body)”.

Presentations share the documentation font styles 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**

The following hyperlinks embody the practices that this development team strives for. Because the Android platform is written in the programming language Java, we must have excellent Java proficiency. Furthermore, the Android platform has its own coding standards, project guidelines, and development design patterns that must be considered when developing a mobile application in Android.

[Google Java general coding standards](#)

[Android coding standards](#)

[Android project guidelines](#)

[Android development guidelines](#)

### **USER INTERFACE GUIDELINES**

In addition to the reference links below, some user interface principles to abide by are simplicity, consistency, and to maintain the custom user interface aesthetic presented in iOS Burgerator.

[Android user interface best practices](#)

[Android user interface design](#)

### **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 our platform specified in the requirements section. These guidelines will be presented to the development group until a unanimous agreement on concrete requirements has been achieved.

### **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. The bugs will be found by testing all existing use cases and use case scenarios.

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. This system level testing will take place on physical and nonphysical devices.

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.

## **SYSTEM TESTING**

System testing for Burgerator has been implemented through use of functional acceptance testing, ad hoc testing, alpha testing, and beta testing. Functional acceptance testing is applied at the functional, or feature, level and is used to test the stability of a function that is being integrated into Burgerator. Ad hoc testing implores the use of heuristic troubleshooting techniques to test the stability of the system at different points. Alpha and beta testing expose Burgerator to the intended demographic and work to collect user experience feedback.

In addition to testing methodologies nested within system testing, the following will also be covered in the system testing section: the testing process, requirements tractability, testing items, testing schedule, test recording procedures, hardware and software requirements, constraints, and test cases.

### **TESTING PROCESS**

The testing process and the time of testing for Burgerator depends on which of the testing methodologies is being used. If functional acceptance testing is being used then then it is time for a new feature to be added. The process for functional acceptance testing is to: ensure the feature satisfies the requirement, ensure edge case testing has been completed, and test the features stability on multiple physical devices and emulators.

Furthermore, if Burgerator is about to deploy a stable release then is it time for ad hoc testing and a new release (alpha, beta, launch, or update). The process for these types of release testing is similar in that they both find bugs and yield insight about the user experience. The process for ad hoc testing is to heuristically find bugs in the system, document the bugs or issues on version control or through external documentation, and notify the developers of these problems or enhancements.

Similarly, release testing is defined by presenting a stable version of Burgerator to the intended user base and listening for bug reports, word of mouth, and genuine user feedback.

## REQUIREMENTS TRACEABILITY

The requirements tractability for Burgerator is represented by the following nested list. The least indented lines represent higher level requirements while deeply nested elements represent low level requirements. The three highest level requirements in the traceability graph are: replace existing iOS user interface, query APIs for data, and incorporate GPS.

Replace existing iOS user interface

- Engineer user interface in Android

Query APIs for data

- Engineer ability to make HTTP requests (Volley)

  - Google maps

  - Yelp

  - Burgerator API

    - Ability to log in users

  - Facebook

    - Ability to log in users

  - Twitter

    - Ability to log in users

Incorporate GPS

## TESTING ITEMS

Burgerator has a plethora of testing items to be investigated and conditions to be made for each. In general, each activity, each activity transition, and sub systems reused throughout Burgerator (like requests and asynchronous threads) each have their own testing items. Because of the rapid pace of our development team, testing items are generated and completed within the lifecycle of functional acceptance testing and ad hoc testing. If the testing items are of particular interest at the current stage in the application they will be listed in [Burgerator's repository issues](#).

## TESTING SCHEDULE

The testing schedule for Burgerator has been primarily limited to merging projects at the end of a functional acceptance test and also when a new release of Burgerator is coming out. These are the only time we test in Burgerator. No formal testing schedule is defined because the philosophy of the team is to develop the application to its fullest and test once we need a stable version. This development philosophy can also be thought about as a functionality first approach.

## TEST RECORDING PROCEDURES

The test recording procedures implemented by the Burgerator development team is to track all issues, bugs, or enhancements through use of the repositories version control process. When a user of the application would like to submit feedback, they can report a bug within the application or contribute directly to the repository by opening a new issue. Developers follow the same methodology by reporting issues to the repositories issue tracker during development.

## HARDWARE AND SOFTWARE REQUIREMENTS

Hardware requirements for Burgerator include a camera, a GPS unit, a wireless unit, and a device with an android API above 16. The Android API is also discussed in the software requirements but API level is an adequate reflection of minimum processor and memory specifications.

Software requirements for Burgerator include an Android API above 16 and also the ability to allow extra heap allocation. Other software requirements for android are to allow installation from unknown sources while the application is not on the google play store.

## CONSTRAINTS

Due to the nature of this project, some implementation and testing constraints exist within Android Burgerator. One constraint is not running Burgerator directly off of Burgerator's actual live server. This has forced the development team to spin up a new development server that is an exact copy of the iOS server. Additional testing and implementation constraints have been not being able to modify the backend code to increase robustness and functionality of Burgerator. Other constraints have been having a limited number of testing devices, time, and money.

## TEST CASES

Similar to The Hamburgerlers' approach to testing items, because of the rapid pace of development, test cases are generated and completed within the lifecycle of functional acceptance testing and ad hoc testing. If the test cases are of particular interest at the current stage in the application they will be listed in [Burgerator's repository issues](#).

## CONCLUSION

In conclusion, The Hamburgerlers have a nearly completed product. The app successfully leverages Yelp, Burgerator, Google Maps, Glide, Volley, and GSON APIs at the time of this writing. However, the few features that have yet to be fully implements and/or bug tested are the burger selection for the rate activity, the "pound it" button & counter, updating user data after the account is created, and logging in with Twitter.

The team has pushed through all of the big hurdles to create the core of the app, and have a nearly complete user interface. We believe our client is pleased with our work, and will be referring up to a profession app development studio for mentorship to get the app past the final stages and published on the Google Play Store.

All in all, we've learned much regarding Android app development, working with 3<sup>rd</sup> party APIs, asynchronous tasks, multi-threaded programming model, and event-based programming. We all look forward to new and exciting projects in mobile development.

# APPENDICES

## APPENDIX A: NAVIGATION

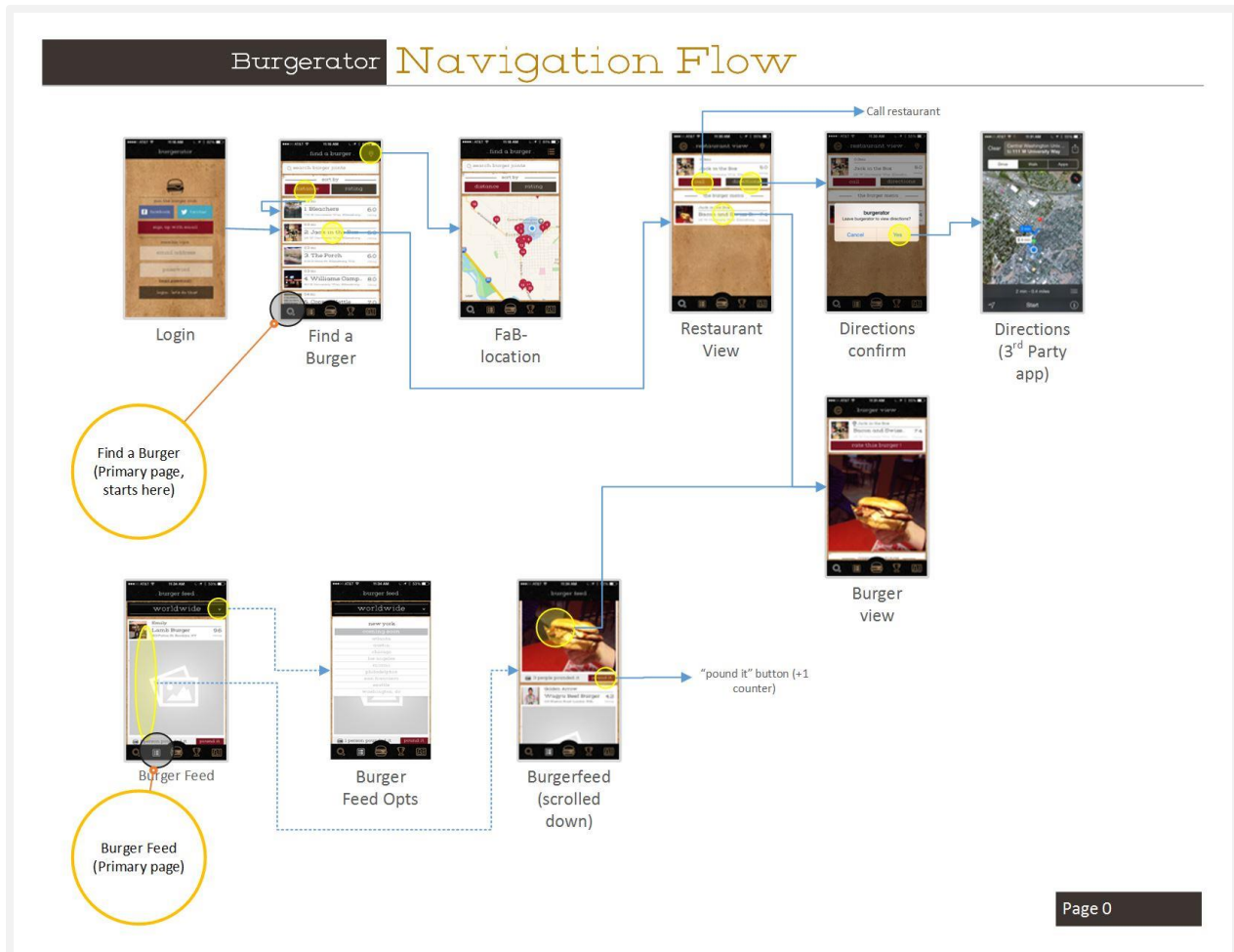


Figure 5, Burgerator navigation prototype version 1, page 1

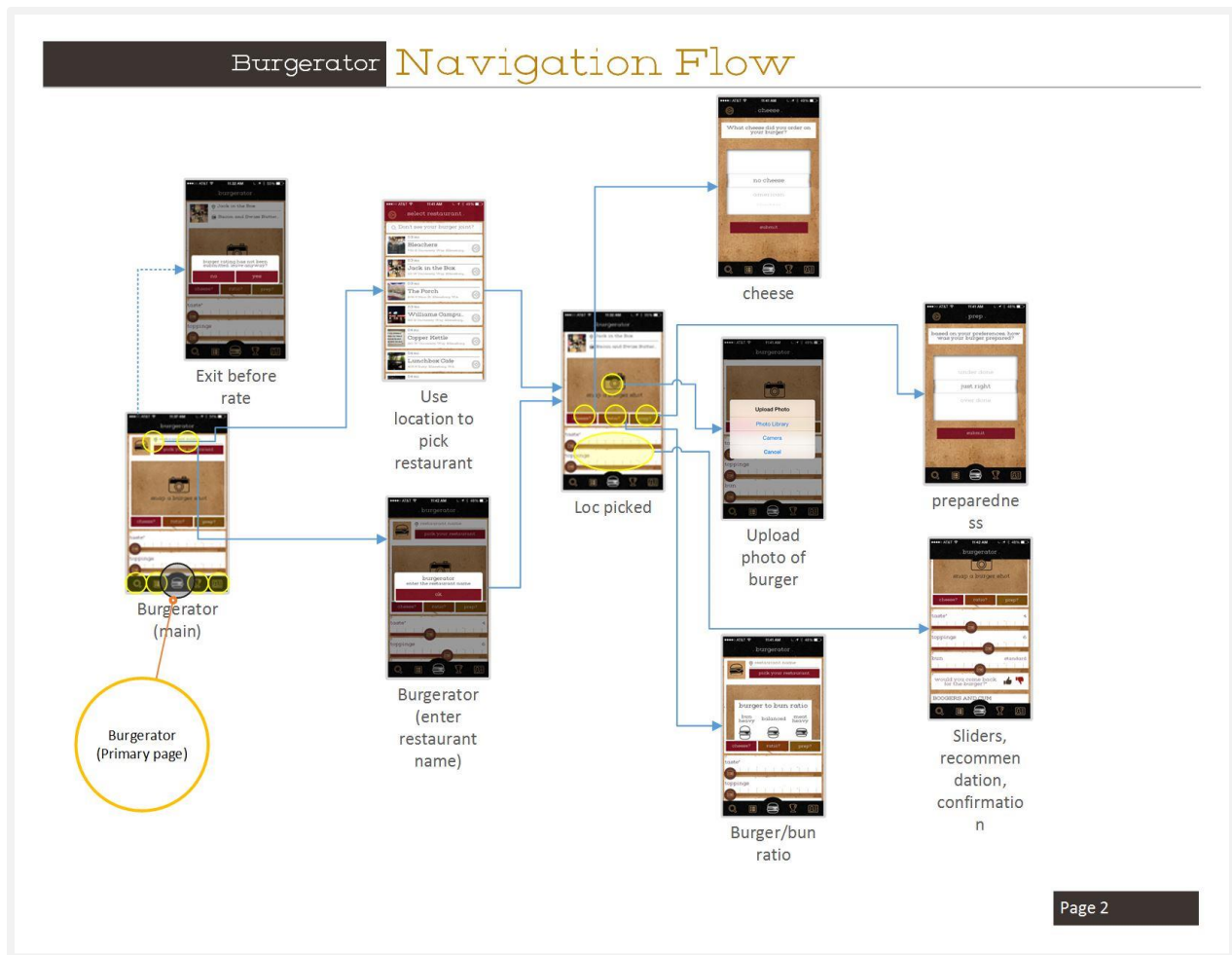


Figure 6, Burgerator navigation prototype version 1, page 2

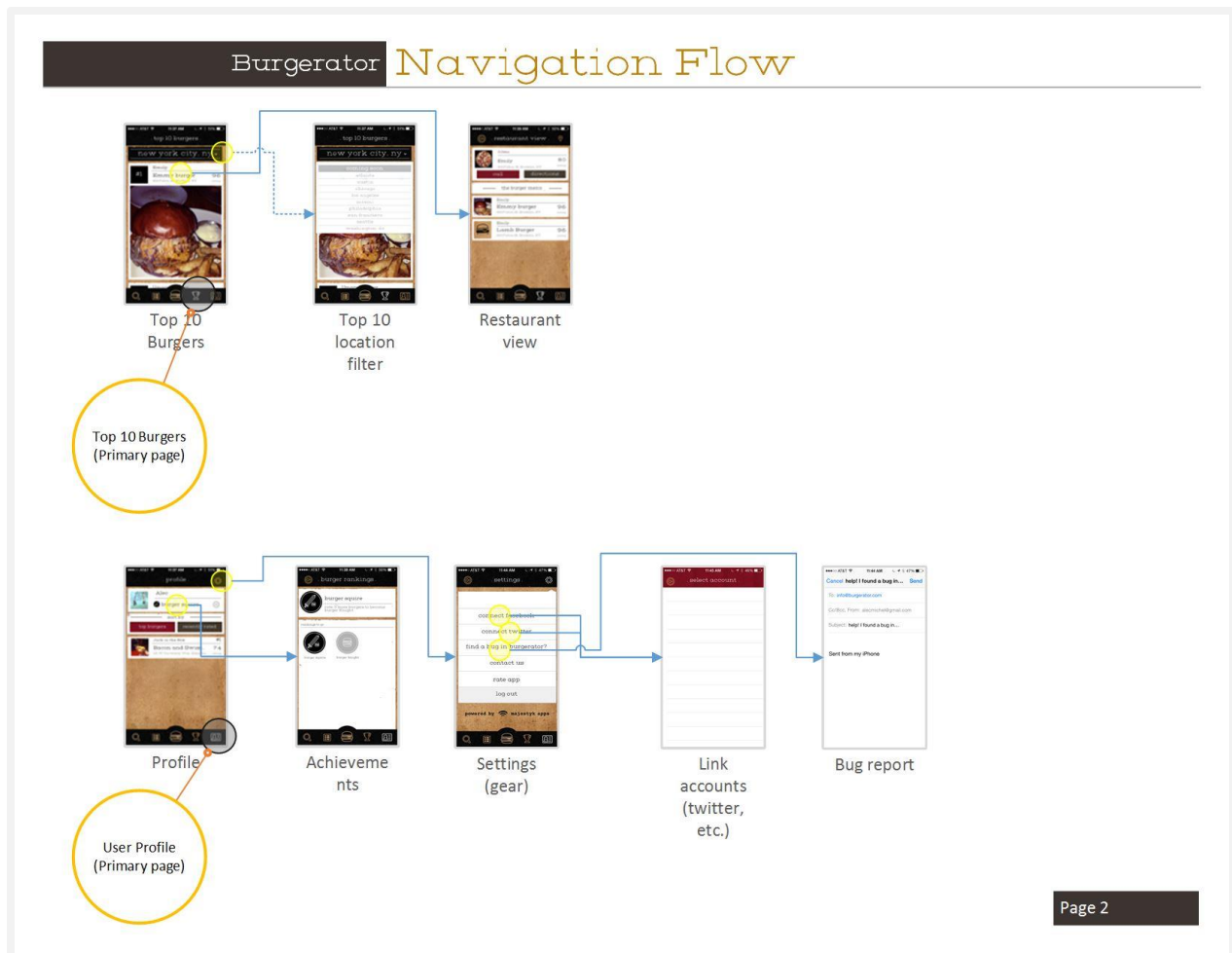


Figure 7, Burgerator navigation prototype version 1, page 3

## **APPENDIX B: USER MANUAL**

See following page for beginning of User Manual



# BURGERATOR FOR ANDROID

## User Guide



## **TABLE OF CONTENTS**

<b>Introduction.....</b>	<b>35</b>
<b>Installation.....</b>	<b>35</b>
Install Burgerator for Android (release version).....	35
Install Burgerator for Android Beta.....	35
<b>Procedures.....</b>	<b>36</b>
Logging into the app .....	36
Creating a new account: .....	37
Registering by Email.....	37
Registering by Facebook .....	37
Login with Email & Password.....	38
Login with Facebook .....	38
Navigating Burgerator for Android.....	38
Using the Search Feature.....	38
Using the Burger Feed.....	39
Using the Burgerator (a.k.a., rating a burger) .....	39
Using the Top Ten.....	40
Using the profile .....	40

# INTRODUCTION

Greetings! Thank you for your interest in Burgerator for Android. Burgerator for Android is meant for burger lovers of all ages. The interface is intended to be user friendly, and intuitive to navigate. In this manual, you will find useful information to make your Burgerator experience an optimal one.

## INSTALLATION

Burgerator for Android will come in two versions. Most users will want to install Burgerator for Android (release). However, for the users that want to get the bleeding edge features, or want to contribute to the development of Burgerator for Android, you can download Burgerator for Android Beta.

### INSTALL BURGERATOR FOR ANDROID (RELEASE VERSION)

The release version is what you will see on the Google Play Store. If you already know how to install apps from the Play Store, just search for Burgerator for Android. If you're new to Android, and you want a little more detail, please see the instructions below

**\*\*This app is not yet released. When it is, use this install method. If you're interested in contributing to the development of Burgerator, please see the Burgerator for Android Beta installation\*\***

1. Open up Google Play Store
2. Use the Play Store's search feature and enter "Burgerator for Android"
3. Select the Burgerator for Android from the list
4. Press the install button located under the title.
5. Burgerator for Android is an intuitive experience. However, if you want to be sure you're getting the fullest Burgerator experience, please see the section titled "How to use Burgerator for Android"

### INSTALL BURGERATOR FOR ANDROID BETA

Burgerator for Android Beta is only meant for experienced users! The software comes as-is, and may have serious bugs. Additionally, the Beta also requires you to side-load the app which if not handled properly, can compromise the security of your device!

1. Go to your device settings and the label "Security"
2. In the security pane, you will check the item that says allow installation from "unknown sources"
3. Open your preferred browser and type in the URL: <http://tinyurl.com/burgdroid>
4. Once it has finished downloading, you will be prompted to open the file
  - a. Select the "Package Installer"
5. Once the app has finished installing, go back to your device settings
6. Select security once again
7. Uncheck allow installation from "unknown sources"

Now you can experience the latest version of Burgerator for Android!

Please help us with identifying and issues you may encounter, feature requests, and general feedback by submitting to our issue tracker at [https://github.com/designbot9000/Burgerator\\_Android/issues](https://github.com/designbot9000/Burgerator_Android/issues)

Thank you for helping us to make Burgerator for Android an awesome experience!

## PROCEDURES

After Installing Burgerator for Android, you can now open the app. To do this, press the app icon located in your “All apps” icon on your Android device. After the app starts:

The login screen will appear, please log in with your credentials (See: *Logging into the App* for more info)

Once you are logged in, you will be taken to the search screen

The main navigation for the app is located at the bottom of the screen. From left to right the icons are:

1. Search
2. Burger Feed
3. Burgerator
4. Top 10
5. User Profile

Additional context based navigation will appear on the title bar at the top of the screen where appropriate.

## LOGGING INTO THE APP



## ***CREATING A NEW ACCOUNT:***

Creating a new account is meant to be an easy and hassle-free process. There are two processes for this: either you can register by email or use Facebook to automatically create your Burgerator account.

### **Registering by Email**

1. On the Login screen, which is displayed once you open the app, press the button with the text “Register by Email
2. You will be taken to a new screen that has text-entry boxes for:
  - a. Email
  - b. First Name
  - c. Last Name
  - d. Password
  - e. Zip code
3. Enter your credentials for each of the fields
  - a. **\*\*Note:** You do not have to enter your Last Name if you choose to do so
4. Once you’re finished entering your zip code, press the Enter key on your keyboard
  - a. For most keyboards it will display “Go” rather than the Enter symbol
5. Once your account has been successfully created you will automatically be logged in and can start using Burgerator for Android



Should an issue arise when creating a new account, a “toast” message will appear with an appropriate error message.

Such errors could be:

- A field other than Last Name was left blank
- A field contains invalid characters
- Your device could not communicate with the server

### **Registering by Facebook**

1. Press the Facebook button
2. A Facebook login window will display
3. Enter your Facebook credentials
4. Confirm the use of your Facebook credentials
  - a. If you don’t accept the App permissions, you can always create an account via “register by email”
5. If the Facebook authentication is successful, your Burgerator account will be created using your public Facebook data

### ***LOGIN WITH EMAIL & PASSWORD***

Within the Login screen, which appears when the app is opened, tap the boxes labeled Email and Password and enter your credentials into those fields. Once complete, press the Button labeled “Let’s Do This.” The interface will display an error message if authentication fails do to typos, missing data, or a bad connection.

### ***LOGIN WITH FACEBOOK***

Press the Facebook button and enter your credentials. If you’ve already authenticated with Facebook, the window will state that. In this case, merely press the button labeled “ok.”

### **NAVIGATING BURGERATOR FOR ANDROID**

The main navigation for the app is located at the bottom of the screen. From left to right the icons are:

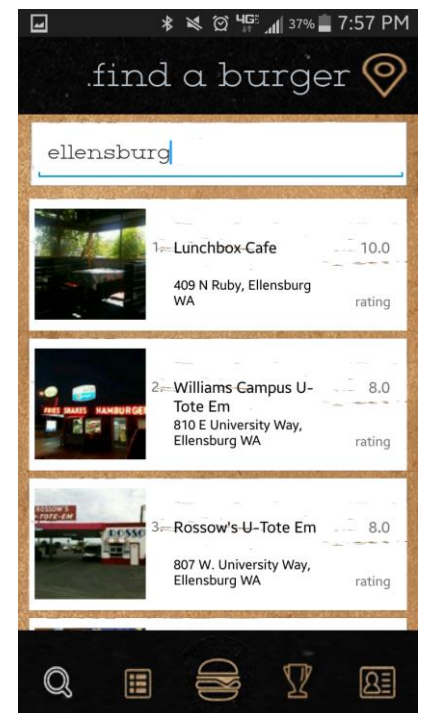
1. Search
2. Burger Feed
3. Burgerator
4. Top 10
5. User Profile

Additional context based navigation will appear on the title bar at the top of the screen where appropriate.

### **USING THE SEARCH FEATURE**

The Search screen allows the user to find nearby burger restaurants. This list provides the name, address, image, and rating of the restaurant. Additionally, the list defaults to the zip the user entered when creating an account. However, using the text field provided at the top, a user can enter any location they want to, and the list will populate with the top 25 nearby locations. This text field can accept a zip code, city, and city & state.

There is also a map feature, which the user can navigate to by pressing the map marker icon in the top right of the screen. This will display the same list as before but as location markers within google maps. The search box located at the top of the map view works the same as within the search view.



## USING THE BURGER FEED

The Burger Feed is a list of 25 of the most recently rated burgers. The most recently rated burger will appear at the top of the list. Each item displays the user who rated the burger, the restaurant name, name of the burger, address, rating the user gave it, and an image of the burger.



## USING THE BURGERATOR (A.K.A., RATING A BURGER)

The Rating view allows the user to submit a rating of a burger to the Burgerator. Required fields will be marked with an asterisk (\*).

1. First, snap an image of the Burger \*
2. Press the ratio button
3. Select the burger-to-bun ratio\*
4. Press the cheese button
5. Select the appropriate cheese\*
6. Use the three sliders to select the appropriate rating for Taste, Toppings, and Bun
7. Press the thumbs up ("yes") or thumbs down ("no") for, "Would you come back for the burger?"\*
8. Add a comment in the comments box
9. Press the button at the bottom to "Submit Rating"



After you submit your rating a success dialog will pop up.

If there was an issue with the connection or a required field was missing information, a dialog box will pop up stating what happened.

## USING THE TOP TEN



The Top 10 will look and function familiarly to the Burger Feed. The main difference is that the Top 10 displays the Top 10 rated burgers for New York, New York.

## USING THE PROFILE

The profile screen shows the user's profile data. It displays the user's profile image, name, and Burgerator ranking. The user can press the ranking text/icon to view a large detail view of all the badges that have been collected by rating burgers. As the user rates more and more burgers, the user will earn new ranks accompanied with badges.

