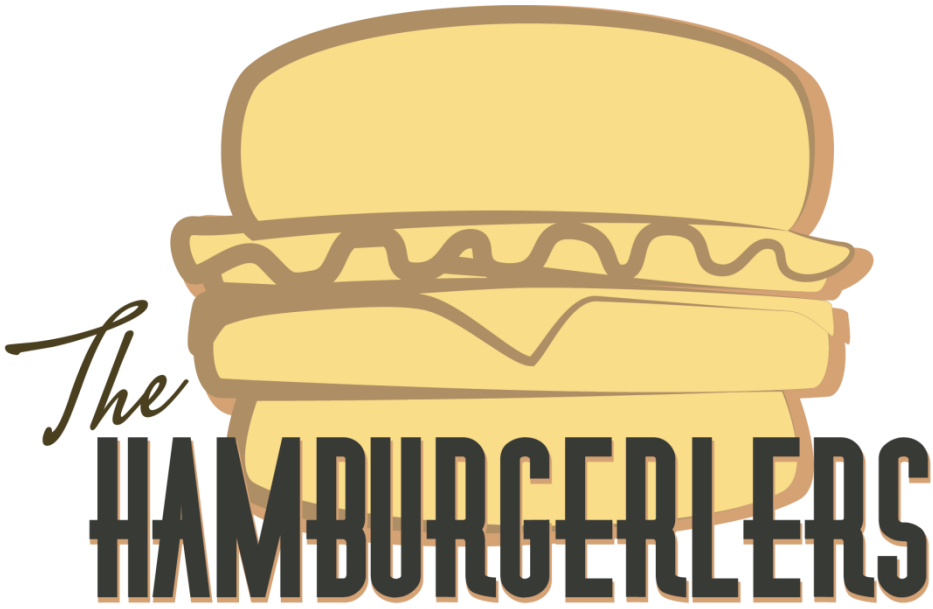

Third Agile Iteration Report

Project: Burgerator

Client: Ammar Shallal

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

Central Washington University - 2/6/2016

Contents

Introduction 4

Website 4

Project Overview 4

- Project Summary 4
- Project Client and Stakeholders 4
- Project Scope 4

Project Management Plan 5

- 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

Requirements 7

- Development, Operation, and Maintenance Environments 7
- System Model 7
- User Interaction 8
- Functional Requirements 8
- Nonfunctional Requirements 8
- Feasibility 9
- Use Case Diagram: 10

Use Case Scenarios 11

Architectural Design 19

- Section Overview 19
- General Constraints 19
- Data Design 19
- Program Structure 20
- Alternatives Considered 22

Detailed Design 23
Data Design 23
UI Package: 23
DB Package: 23
Util Package: 23
Quality Assurance Plan 24
Document Standards 24
Coding Standards 24
User Interface Guidelines 24
Change Control Process 24
Testing Process 24
Conclusion 25
Appendices 26
Appendix A: Navigation 26

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/

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:

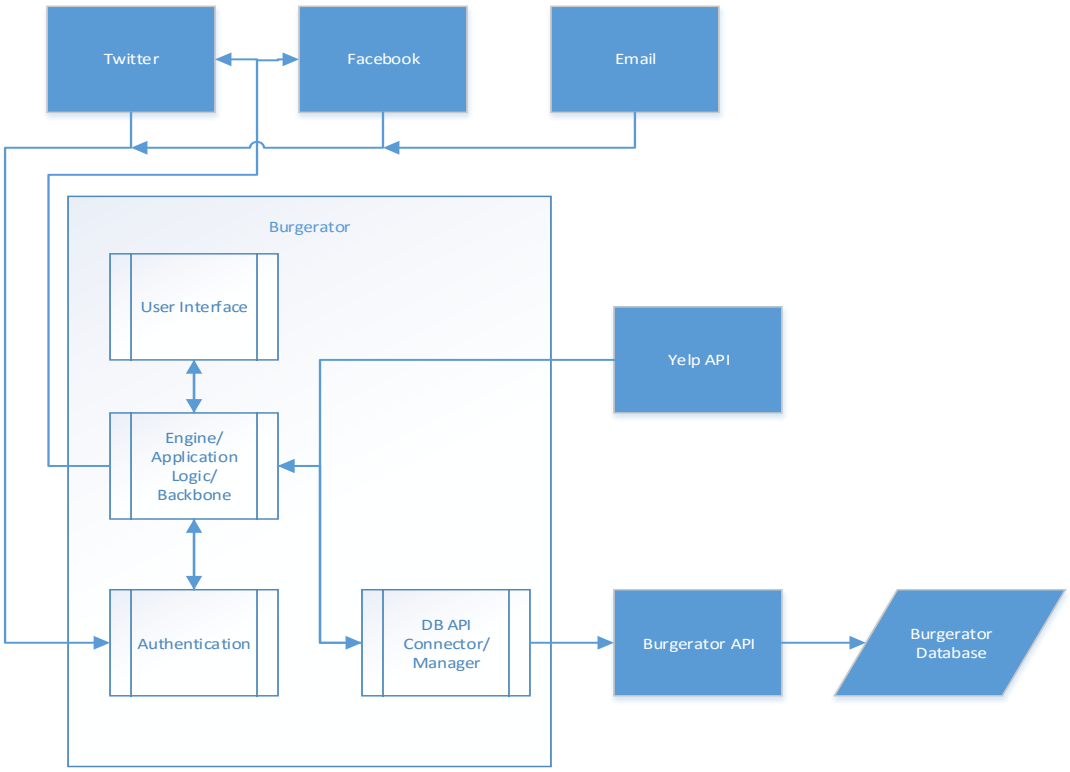

Figure 1: IOS Burgerator high level design

- 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:

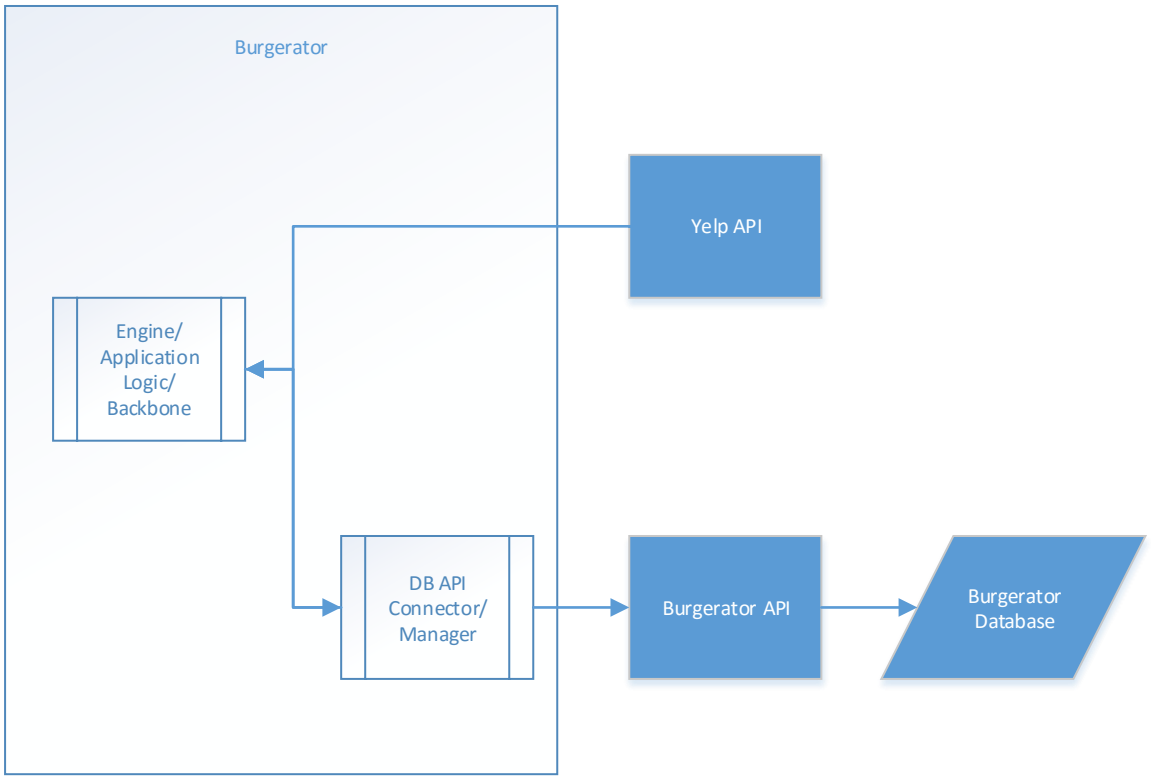

Figure 2: Base System Diagram

USE CASE DIAGRAM:

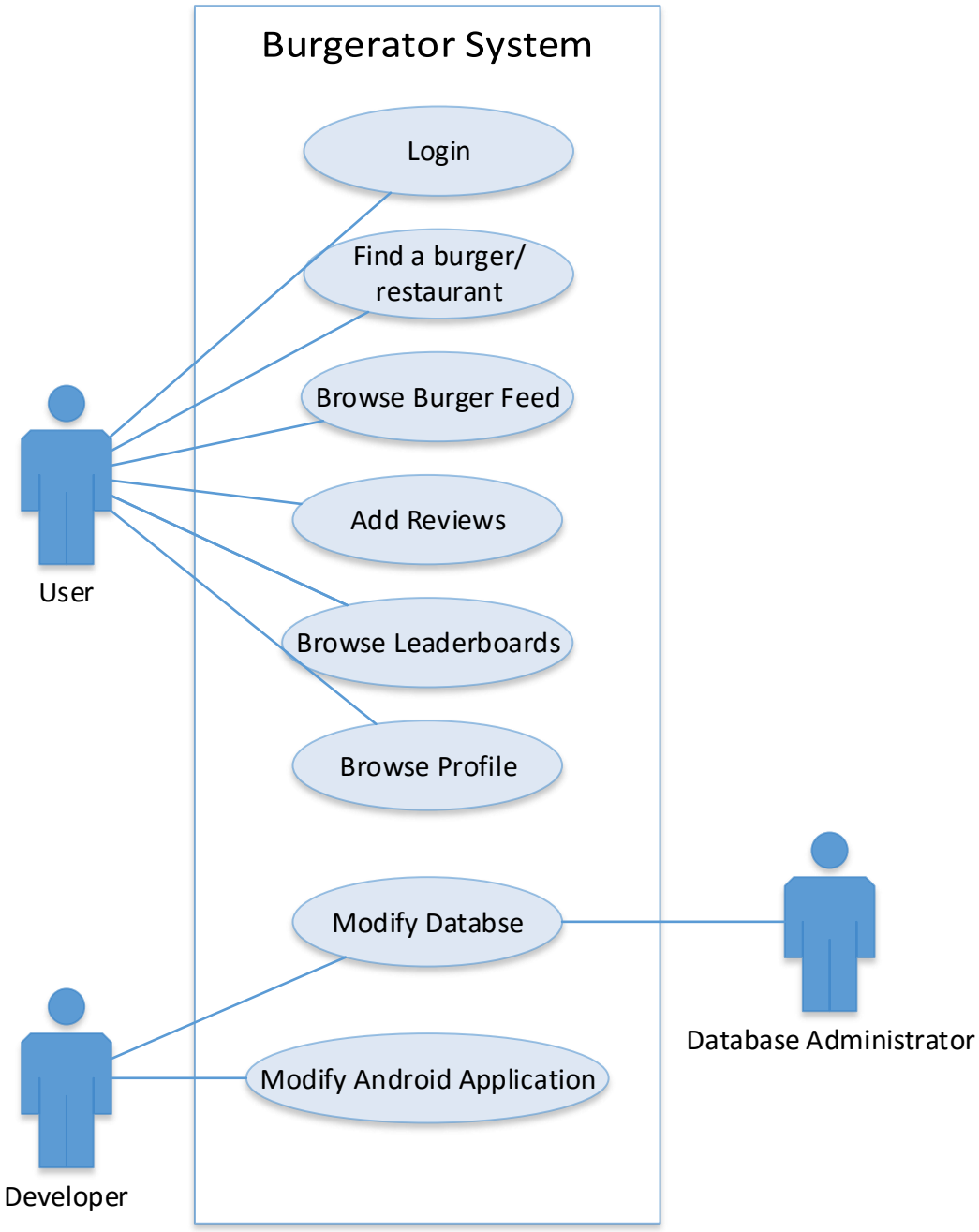

Figure 3 Use cases and actors of the Burgerator system

USE CASE SCENARIOS

Use Case Name	Login to Burgerator
Actors	User
Summary	The user logs into the application upon first use
Pre-Conditions	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

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 or 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 preform 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. Out 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 preform 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.

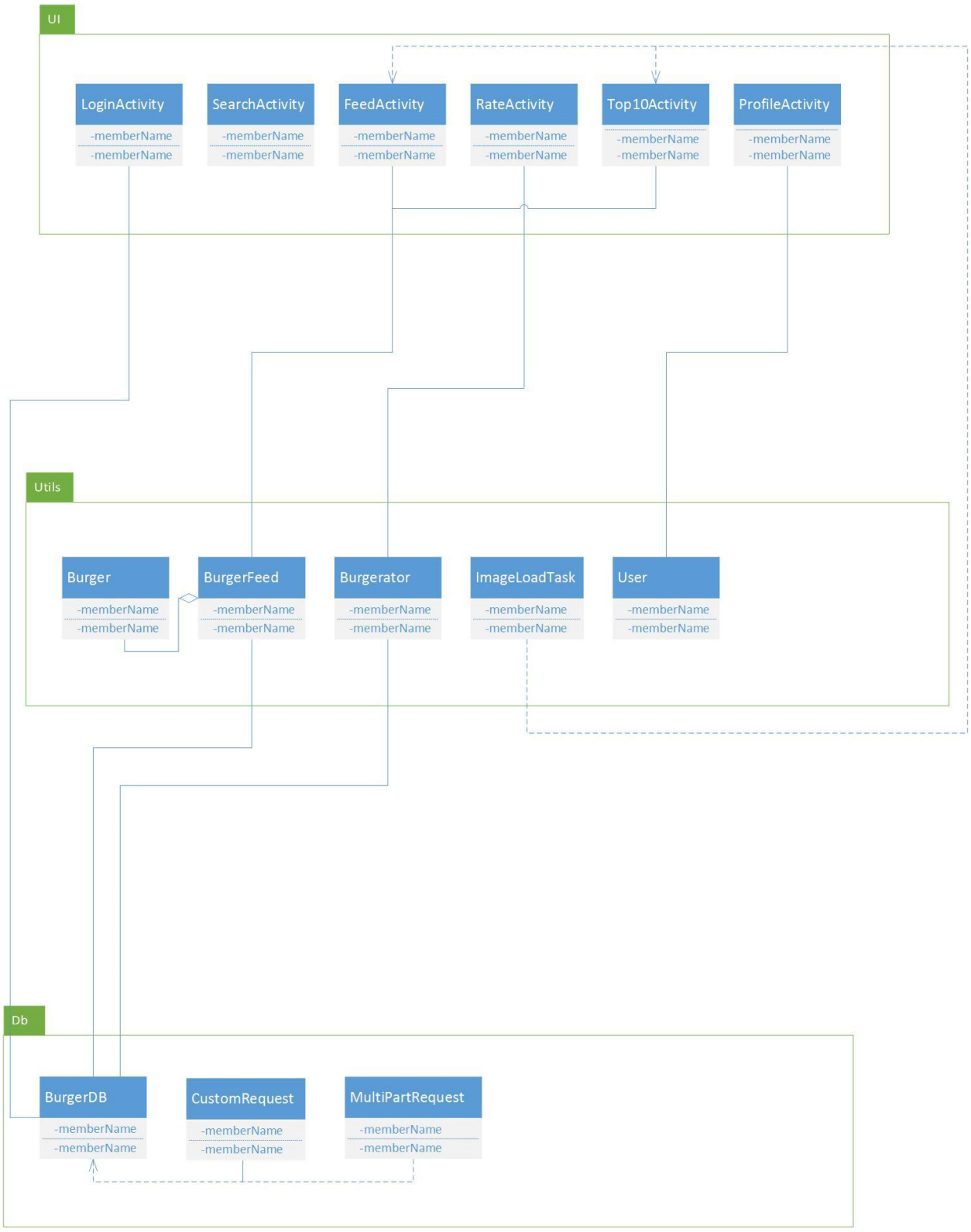

Figure 4: Burgerator for Android - current architecture model

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 out 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.

CONCLUSION

In conclusion, Team Hamburgerler has much work ahead. We need to identify how to connect to the Burgerator database, and how to connect to the Yelp API Database. In addition to these requirements, we need to also learn how to authorize users with their favorite social media accounts like Facebook and Twitter. The user interface in the application also needs to be recreated to replicate the iOS Burgerator application.

Scheduling conflicts have been minimal but not negligible. Looking forward, it would be ideal for development to maintain momentum and for team meeting to happen when scheduled. Also, more concrete guidelines will be established as prototype development continues.

All in all, this project is just beginning but we look forward to having a prototype in the next few weeks.

APPENDICES

APPENDIX A: NAVIGATION

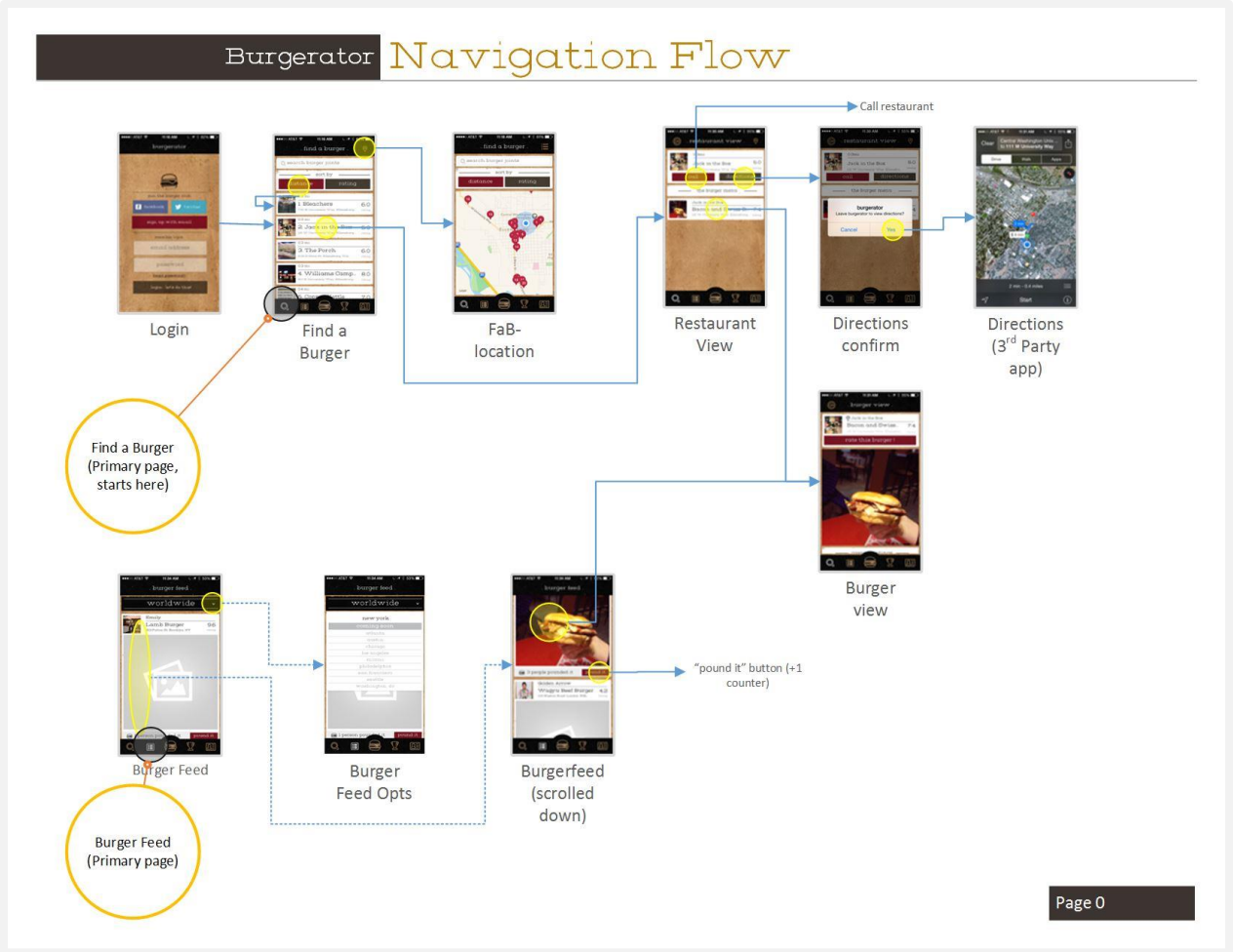

Figure 5, Burgerator navigation prototype version 1, page 1

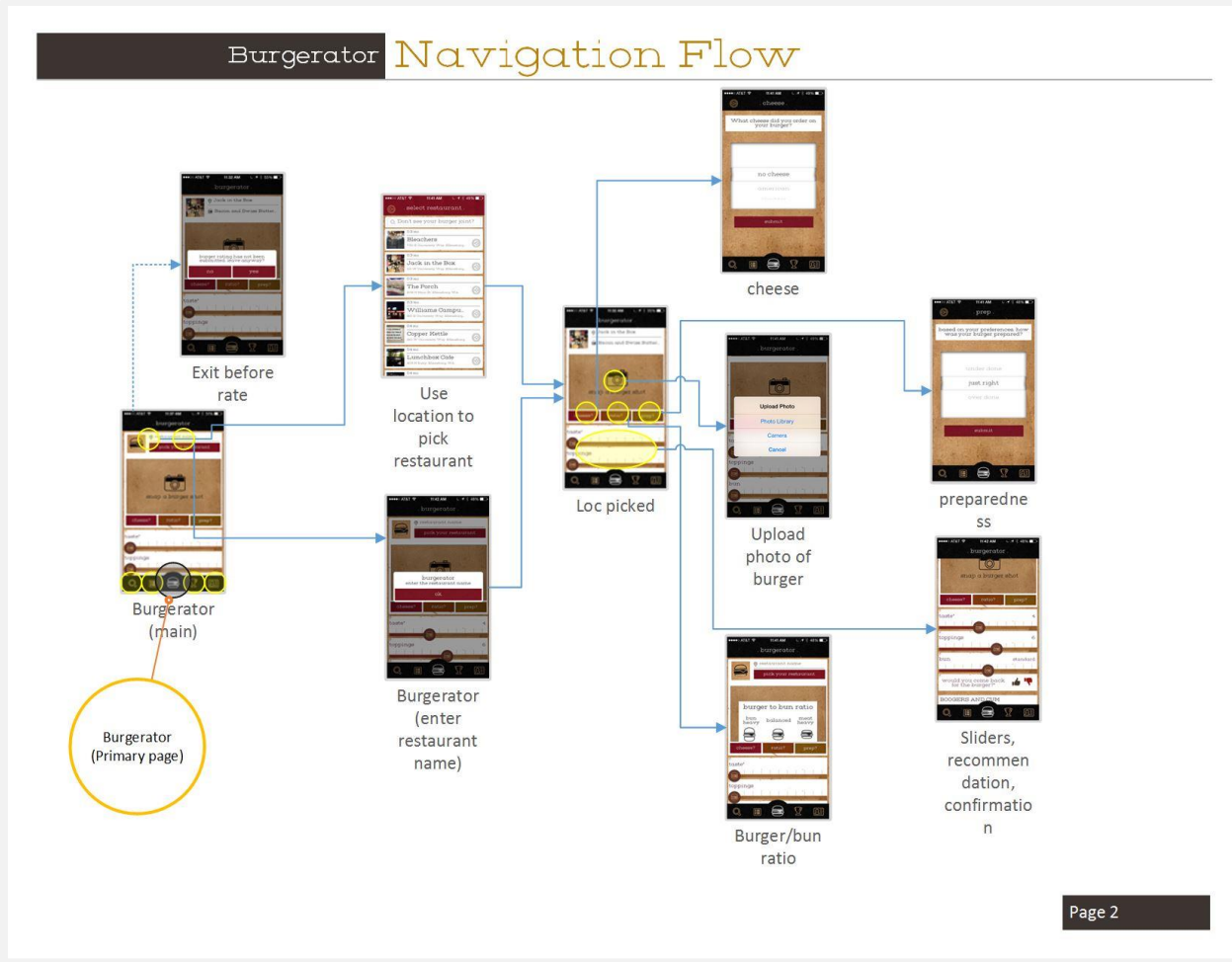

Figure 6, Burgerator navigation prototype version 1, page 2

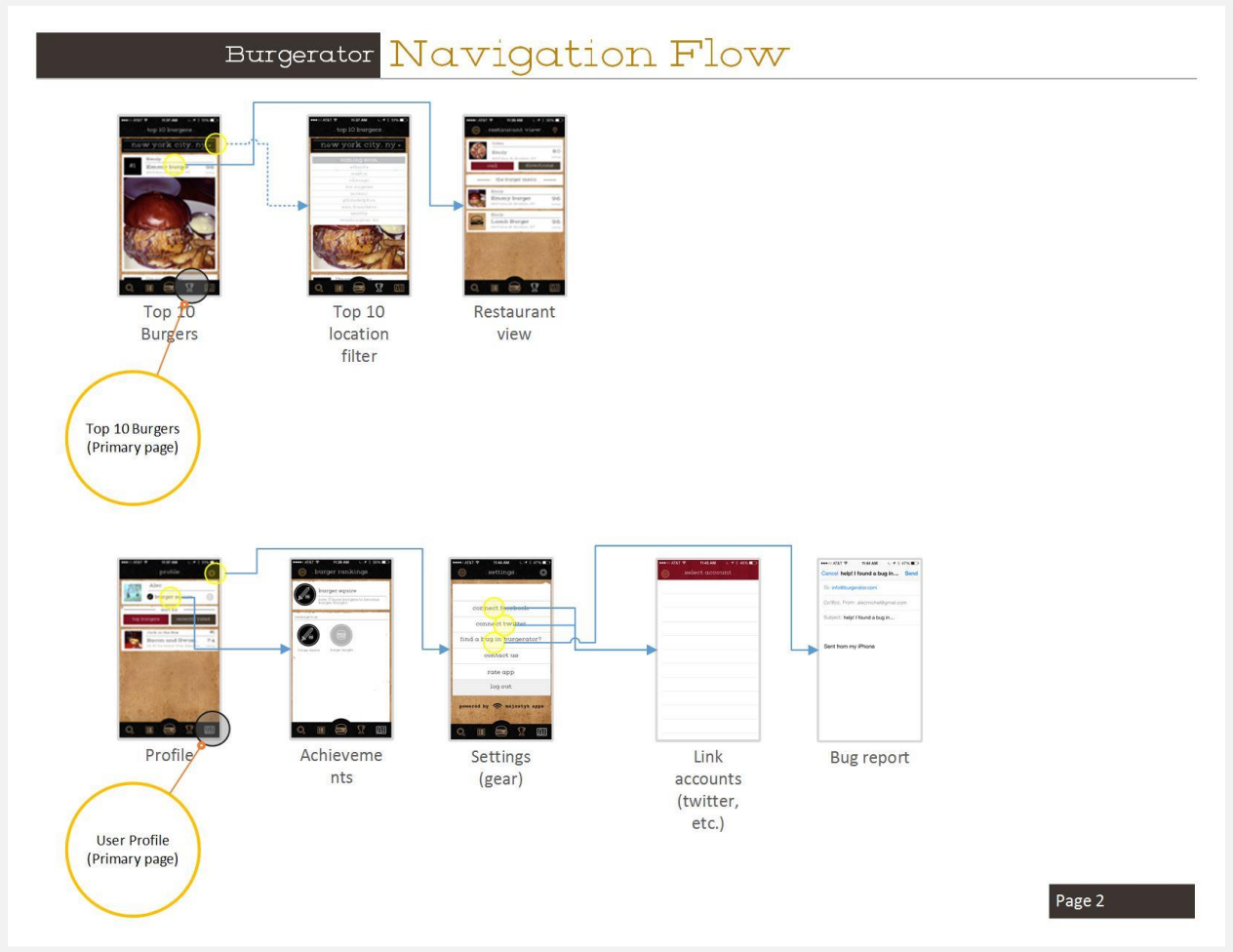

Figure 7, Burgerator navigation prototype version 1, page 3