ECE 651 - Final Project Documentation

v1.3

genie

Jonathan Buie
Young-hoon Kim
Peter Murphy

TABLE OF CONTENTS

REVISION HISTORY

Revision 1.3

DOCUMENTATION OVERVIEW

Genie is an app that provides a safe, crowd-sourced platform that finds people in the same location based on common interest and mutual benefit. Have a homework set due in an hour and stuck on a line of code? Holler for help on Genie. Stuck in a parking lot and need to jumpstart your car? Holler for help on Genie. Feeling blue and general want to chat or grab a study break with someone? Holler on Genie. People in close proximity will see your request and come to your aid. Everyone's a Genie, they just don't know it yet.

PROJECT SCOPE

This project covered all elements of the design cycle for a mobile application. More specifically, it included the design and implementation of a crowd-sourced mobile application for Android devices. This requires a user friendly software framework in the front end, a developer-friendly backend, and thorough documentation for developers. It will extend through multiple development phases with an agile development method.

PROJECT OBJECTIVES

Genie allows users connect based on matching help requests. Ultimately, the goal for Genie is to attract as many users as possible and to do this, Genie must provide an exceptional user-experience. This requires a safe, reliable, convenient, and secure application.

The corresponding documentation intends to communicate all aspects of the design cycle.

DOCUMENTATION SCOPE AND OBJECTIVES

This documentation intends to be a technical user guide for other developers of the Genie platform. The information presenter will allow the reader to understand the requirements, design, and architecture of the application known as Genie. The scope includes all activities that promote Genie's Value Proposition of delivering fast, reliable, safe, and effective means for users to connect. It is also aimed at showing a maintainability framework for future developers, and allow the management team to ensure a compliance with usability, brand, and intellectual property.

TECHNICAL SCOPE AND OBJECTIVES

From a technical standpoint, this document will include all necessary information for determining a reasonable development and deployment plan. It is aimed at providing the management team with a list of pre-approved requirements that demonstrate a functional understanding of the application so that one can execute a business plan using only the information presented in subsequent sections. This documentation is also targeted at any current or future development teams to promote code usability.

REQUIREMENTS

FUNCTIONAL REQUIREMENTS

1) Login and Logout
    a) User shall be able to use his/her email to register an account

   b) User shall get a validation email to confirm account
   c) In the settings page the user can successfully log out of the application
2) Post Requests
   a) User shall have the option to post requests to a shared database
   b) Requests shall propagate to all other app users
   c) In the user profile page user is able to see all posts that he/she created
3) Respond to Requests
   a) User shall be able to view requests as they come up in real time
   b) User shall be able to choose specific requests and respond
   c) A private message conversation shall initiate between responder and requester
   d) Conversations shall be able to take place in real time
4) View Conversations
   a) User shall be able to view all conversations and responses for requests he/she posted
   b) User shall be able to view all conversations which he/she has responded to
   c) User shall be able to select and continue conversations from this view
5) GPS Location
   a) User shall be able to see their current location visually on a map
   b) User shall be able to see request locations in relation to their current location on map
   c) GPS shall be able to update in real time

NON-FUNCTIONAL REQUIREMENTS

1) Performance
   a) Size
      i) Less than 10MB
   b) Channel Capacity for 10,000+ users
   c) Available only Online
   d) Scalability for expanding number of users
   e) Service time less than 1 min on Duke Campus
   f) Transmission time (less than 2 minutes with full internet connection) on startup to update live feedback and update changes
2) Security
   a) Maintain Duke University Application Standards
      i) Multi-factor Authentication
      ii) Maintain FERPA Standards since containing student information
      iii) Data encryption during transit
3) Accessibility
   a) Available for Online Use
      i) Online: All features available
   b) Languages
      i) English
4) Permission
   a) Identity
      i) Email address
   b) Location
      i) Precise location (GPS and Network based)
      ii) Approximate location (Network)
   c) Device ID & Status

i) Read phone status and identity
d) Other
i) Run at startup
ii) Create accounts and set passwords
iii) View network connect
iv) Full network access
5) Platform Compatibility
a) Android
i) Requires Android 4.03 and up
b) Identity
i) Email address
c) Location
i) Precise location (GPS and Network based)
ii) Approximate location (Network)
d) Device ID & Status
i) Read phone status and identity
e) Other
i) Run at startup
ii) Create accounts and set passwords
iii) View network connect
iv) Full network access
6) Platform Compatibility
a) Android
i) Requires Android 4.03 and up

USER STORY

USER STORY 1

Huey is a 1st year student at Duke and arrives on campus hoping to make new friends. He hears about the new app Genie and downloads it to interact with other students in the area. Since he considers himself to pretty introverted he decides to register a new account with the email address *huey@example.com* with the password *12345678.* After registering he logs into the application and presses the "post request" button to reach out. His request reads "Looking to play Super Smash Brothers for the next couple of hours?" with the additional information that says "I have 2 controllers feel free to bring your own", and his location is "Southgate Dorm". After filling in the text fields he submits the request and waits to hear from new potential friends at Duke. Once the post is submitted it can be seen in the main list, but Huey clicks the user profile button and can see every request he's submitted.

Features Tested: Registration, Login, Post Request, User Profile

USER STORY 2

Martin is also a first year student at Duke. He heard about Genie as a high school student and already had an account once arriving to college. After settling into his own room he finds himself with free time

and opens up the Genie app to see what's going on in the area. He logs in by typing in his email *test@example.com* and password: 12345678. He successfully authenticates and sees a request that reads "Looking to play Super Smash brothers for the next couple of hours?" This instantly excites Martin and he selects the post and opens to read more information in the "Holla Back" page. The page shows that Huey is located in the Southgate dorm and displays the GPS coordinates in the Map display. Since this is nearby, he presses the "holla back button" and starts a new conversation with the user Huey. He texts a message to Huey that says "Hey Huey I'd love to play Smash! I have a controller too! When can we start playing?" He exits this screen and scrolls other posts. He's eager to see if Huey has replied so clicks the "Conversation History" button and sees his conversation with Huey listed, but still no reply.

Features Tested: Login, View Posted Requests, Private Conversation Creation, Conversation List View, GPS Location

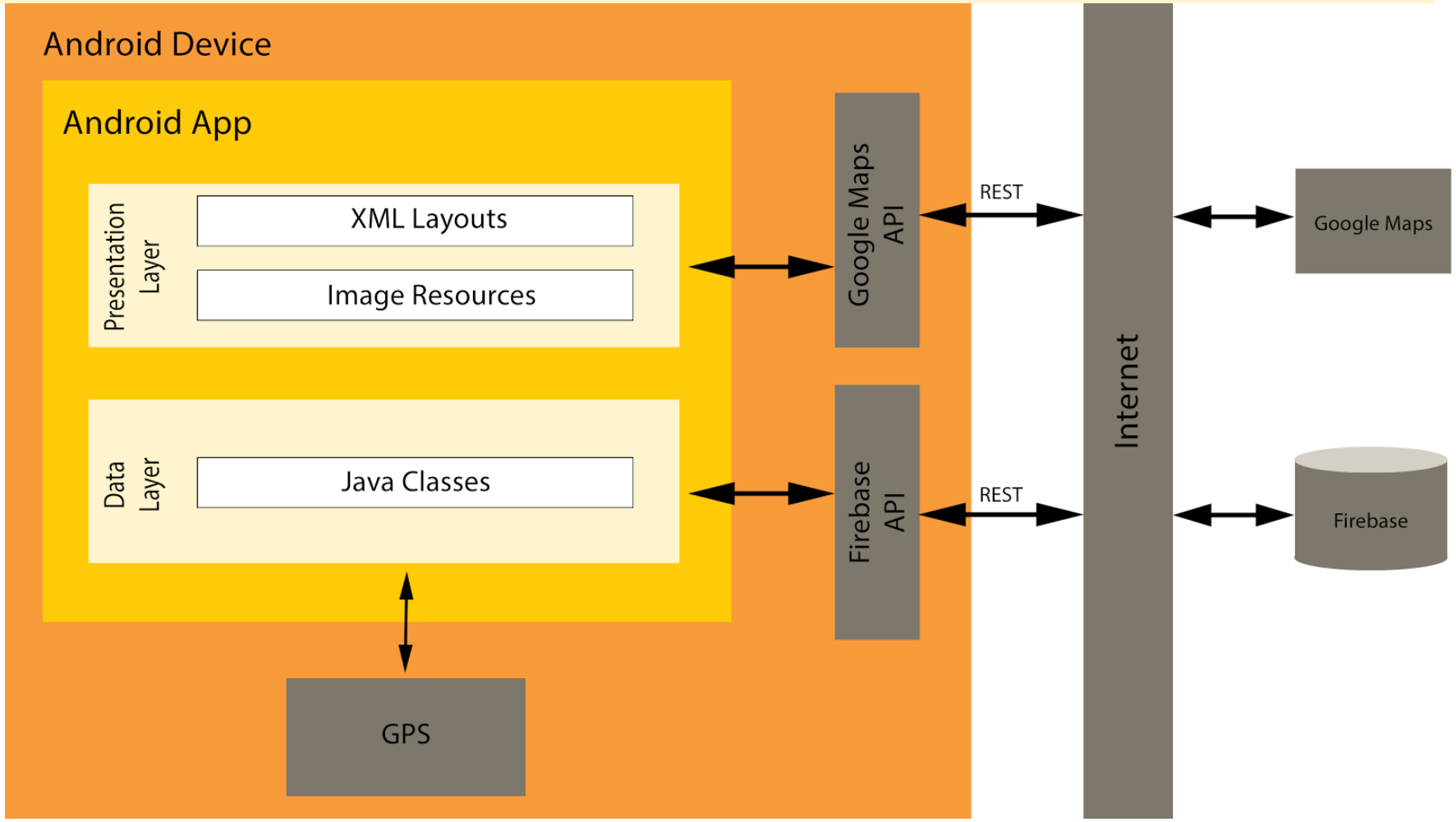
USER STORY 3

Huey had the app closed and reopens Genie. Huey notices that he has a new message in his conversation history page. Under the title conversations with me he see's a user Martin has messaged him. The message reads "Hey Huey I'd love to play Smash! I have a controller too! When can we start playing?" He replies "Hey Martin I will be playing in 20 minutes come to room 222 in Southgate, bring your controller too :) ". He is excited to finally meet someone new and while he waits he decides to log out of the app and get his room setup for a fun time playing video games. He does this by pressing the settings button in the toolbar and pressing the logout button.

Features Tested: Remember Authentication State, Conversation List View, Private Messaging, Logging out

ARCHITECTURE

SYSTEM ARCHITECTURE



PRESENTATION LAYER

This layer contains all the XML layout files and resources used to display relevant information on screen.

DATA LAYER

This layer contains all java classes for activities and java classes needed to contain and transfer information.

GPS

The physical device's GPS is used to locate the current user's location which is stored in Firebase and used in Google Maps to visually mark the user's position.

APIS

The bridge between the android application and external software. Contains the set of protocols and routines used to establish communication and ensure transfer of information.

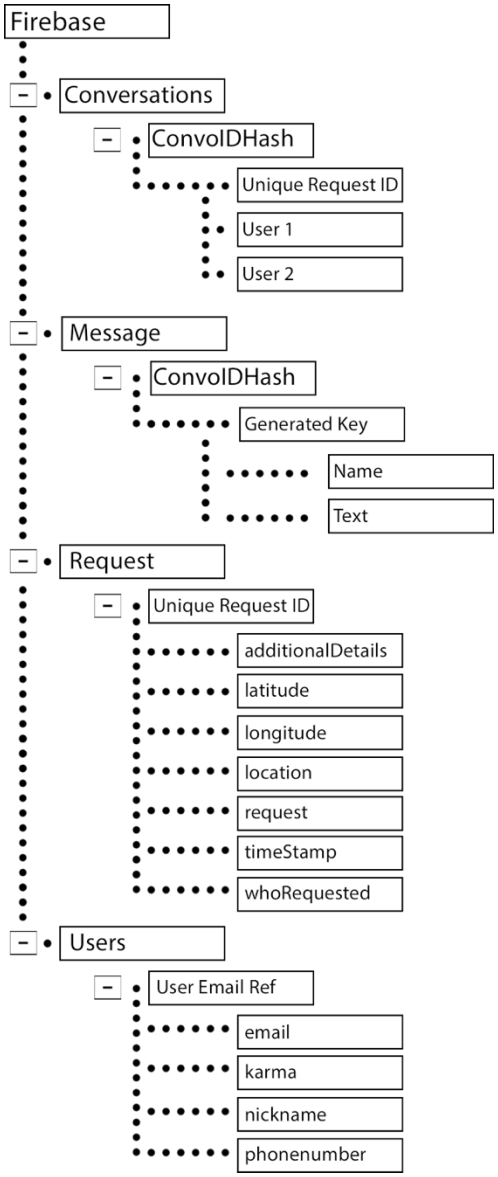
FIREBASE

Online database used to store and transfer all of the app's data. This includes authentication, user information, requests, and messages.

GOOGLE MAPS

Mapping software provided by Google to visually represent user's location and nearby areas of interest. Latitude and longitude information is collected by the android device and used with Google Maps to show the user's current location. Google Maps also allows for real time updates and tracking.

DATABASE ARCHITECTURE



FIREBASE

Online database used to store and transfer all of the app's data. The design of the NoSQL data structures requires planning and favors a flattened structure. This is a consideration for the reading of data at a later time and a nested structure is not preferred. The 4 main children in our flattened structure include Conversations, Requests, Messages, and Users.

REQUESTS

When we create new request we create "unique Request ID" which is a hash of the request, the user who requested, and the timestamp. The user inputs the data through the UI for the request, additional

details, and location. The whoRequested, latitude, longitude, and timestamp are automatically generated once the post is created. Once we have all the necessary data PostRequest Class is created and pushed to the database.

CONVERSTATIONS

A conversation is created with relation to a request. User1 field is filled taking the current user's name, and the user2 field is the name of the user who requested the post. Finally the last field is filled with the unique ID of post itself. The name of the child is referred to as the conversation ID and is generated as a hash created by the user1, user2, request and timestamp fields from the post (convoIDHash). With the necessary data a Conversation class is created and pushed into the database.

MESSAGES

The messages have the same key name which is generated when a conversation is created (convoIDHash). Each message that is sent has the generated key created through Firebase and each individual message contains the name of who sent the message and the text of the message itself. In the private message activity each time a user presses the "send" button a new ChatMessage Class is created.

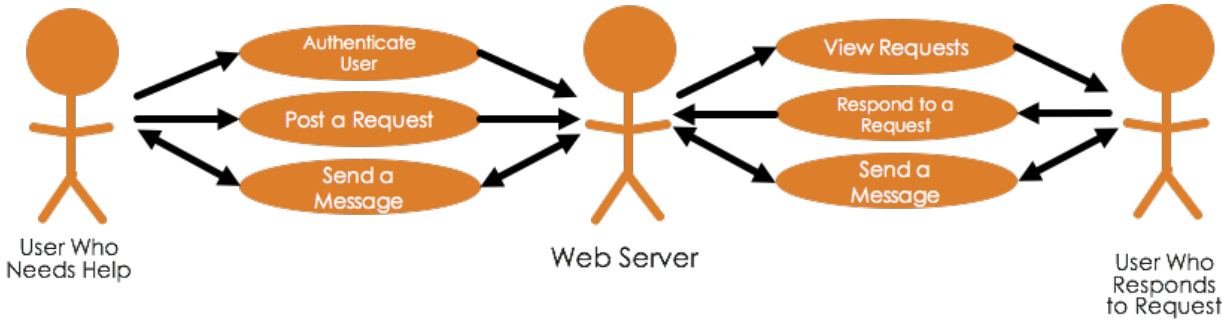
USERS

The user class is an extension of the authentication information. For security the password and authentication information is not available, however the "user email ref" is where the additional information is located. This information is optional and obtained through the UI when the user registers. Upon registration a User Class is pushed to the database and the data can be viewed and edited through the User Profile page.

DESIGN

USE CASE DIAGRAMS

The following figure is a use case model that shows the actors in all user stories. The user on the left represents a user requesting help on Genie, the web server represents the firebase database, and the user on the right represents a user who may choose to respond to a help request. The five most important tasks are shown in the subsequent figures as task cards.



AUTHENTICATION	
Actors	User, Database
Description	User posts email and secure password a database
Data	Email, Password, Display Name
Stimulus	User command issued by software application
Response	System confirms authentication and valid ID
Comments	Identifier must be unique.

POST A REQUEST	
Actors	User, Database
Description	Adds an item to a database that lists the help wanted by the poster, along with his or her coordinates and timestamp
Data	Latitude, longitude, location, request, timespan, requesting user email, additional information.
Stimulus	User command issued by software application
Response	Pop up prompt confirming that the help request has been successfully posted.
Comments	The post can be seen on the users homepage

SEND A MESSAGE	
Actors	Post Request User, Database, Accepting-Request User
Description	This is a typical chat message system. It is initiated upon acceptance of help requests.
Data	Simple strings represent the body of each message. Each string includes the user that sent it along with a hash key.
Stimulus	User command issued by device prompts message push. Server update prompts message pull.
Response	Message added to conversation list.
Comments	This message system is private. It can only be seen by the two conversing users.

RESPOND TO A REQUEST	
Actors	Responding-User, Database
Description	User can view available help requests in database and can click on one to accept, if desired.

Data	Accepting-user's unique identifier
Stimulus	User command issued by device
Response	Initiate chat conversation with help-requesting user
Comments	Once a request has been accepted, it cannot be closed

VIEW REQUESTS	
Actors	Database, user
Description	All help requests are shown on a single page. A user becomes an "Accepting-User" when he or she accepts a help request
Data	All available help-requests posted by users
Stimulus	Any update from the database by a user pushing a new help-request. The list will be populated upon launch of the home screen
Response	Visual changes to home page display
Comments	Requests will never disappear. This will be fixed in a future release.

CLASS DIAGRAMS

IMPLEMENTATION

README

HOW TO RUN THE APP FROM A PHYSICAL ANDROID DEVICE

1) Clone a copy of Master branch onto local drive.
2) Ensure device meets minimum requirements. App minimum SDK version 16, targetSDKVersion 23
3) Load up android studio, load project, and plug in android device.
4) Run -> Run "app" -> select physical device
5) App should be up and running.

HOW TO RUN THE APP FROM A PHYSICAL ANDROID DEVICE

1) Clone a copy of Master branch onto local drive.
2) Load up android studio and load project.
3) Tools -> Android -> AVD Manager
4) Select Create Virtual Device
5) Select Nexus 5, then Next
6) Select Marshmallow API 23, Android 6.0 (with Google API's) as your system image, then Next
7) Feel free to rename device and select finish.
8) To run the app on the emulator, select Run -> Run "app" -> select the virutual device you created above -> select ok
9) App should run as intended

HOW TO RUN THE APP FROM A PHYSICAL ANDROID DEVICE

The google maps API will not be able to obtain coordinates for the user's current location from the device unless the GPS coordinates are also emulated using the android device monitor, instead the coordinates will be set to lat:0, long: 0

FUTURE DEVELOPMENT

- Restrict search range depending on user's specific preferences
    - User should be able to specify a distance range from their current position and filter requests that are made by that distance range.
- Text fields with predetermined tags for users interests and skills
    - Requests should be able to be filtered and searchable by tags.
    - Priority of requests should be made by tags/skills/interests that the user has predetermined.
- Implement heuristic to determine a user's karma score
    - As a way to promote good deeds. Each user can be given a karma score that increases for assisting others requests, and decreased if they fail to assist another user.
- Allow user to request, accept or deny ability to communicate with another user
    - When a user responds to a post request if they are not currently friends the user should have the option to accept or deny the chance to communicate with this unknown user.

BUG REPORTS

As described in the test plan bug reports will be made during the testing phase the week of April 20th - April 27th, 2016.

TEST CASES

AUTHENTICATION

TESTING REQUIREMENTS

- Registration saves user data to database
- User must input a password
- Logging out unauthorizes user on device
- There cannot be duplicate users

TESTING METHODOLOGY

- Register user Foo
  - Confirm password is required
- Log in as Foo with incorrect credentials
  - Should fail and throw “incorrect” error message
- Log in as Foo with password
- Log out
- Register user Foo
  - Should fail and throw “Duplicate User” error message

GOOGLE MAPS

TESTING REQUIREMENTS

- The Google Maps Interface must be able to locate the current user and visually show the user’s current location.
- The user’s location information must be able to update periodically and reflect user’s most recent location.
- Locations should be stored and accessible after selecting requests.

TESTING METHODOLOGY

- User Story Test (Features must satisfy this case to pass testing)
- User wants to post a request. The post request page should show the user’s current location visually.
- When user posts a request, the coordinates of the user’s location is stored.
- The user find a request that he/she wants to respond to. After selecting the request, a marker showing where the request was made should be made visible. The user’s location is also shown on the map to give an estimate of how far apart he/she is from the requester’s location.

PEER-TO-PEER MESSAGING

TESTING REQUIREMENTS

- Must verify Wifi is available & set up Wifi connection permissions on current device
  - Notify user if Wifi unavailable

- Generate a private chat window between two users
- Allow messages to be sent and received between users
    - Store Conversations in the database
- View previous conversations between users in old requests until closed

TESTING METHODOLOGY

- Send message from TestUser0 to TestUser1
    - Confirm it was sent and saved successfully
- Send message from TestUser1 to TestUser0
    - Confirm it was sent and saved successfully
- Close application and re-open the conversation between TestUser0 and TestUser1
    - Confirm that the conversation is still visible

USER PROFILES

TESTING REQUIREMENTS

- User successfully creates a profile with name and password
- Prompt user to select a variety of “tags” that indicate their interests and store in database
- Ensure Users Karma Score is displayed based on information in database

TESTING METHODOLOGY

- Register User “temp” with password “temp”
- Pick 5 “tags” and verify they are displayed in the User Profile view page
    - Sports, math, computer science, dancing, cats

POST/RESPOND REQUESTS

TESTING REQUIREMENTS

- Activity for Holler Requests posts to all users
- Click a Holler Request and open options for direct conversation
- Keep track of conversations for request and responses from users

TESTING METHODOLOGY

User Testing

- testUser0 creates a Holler request with Request “computer science”
- testUser1 should see 1 new post in the main screen
- testUser1 can select the post and have an option to start a new conversation with testUser0
- Close app and confirm state is preserved

FRONT END DESIGN

TESTING REQUIREMENTS

- A splash page must be visible before entering the home screen.

- All features of the app must have an appropriate visual screen.
- Text and images must be properly displayed without distortion or crowding.
- Navigation of the app must be both functional and intuitive.

TESTING METHODOLOGY

- Functionality of User Interface will be compared to that of the original mockup design.
- All interactive buttons will be tested to check for functionality.
- Visual components and layouts will be judged subjectively for readability.

DATABASE INTEGRATION

TESTING REQUIREMENTS

- Users and attributes must be saved to database
- Conversations must be saved on database

TESTING METHODOLOGY

- Create new user and Save settings
- Send messages between two test users
- Set user location
- Confirm that preceding information is saved on Firebase Server

TEAM

JONATHAN BUIE, DATABASE & ALGORITHMS LEAD

Jonathan is a MS student in Electrical Engineering. His work with Intel and Dow Chemical has given him the tools to research and design efficient use of the database and passing data between activities.

- Design the database and anticipate use for Android and iOS applications
- Oversee code development to ensure efficient data access from Firebase system

YOUNG-HOON KIM, FRONT END & MAPS LEAD

Young-hoon is a MEng student in Electrical and Computer Engineering. He has completed multiple hardware and software projects as well as taken multiple business courses.

- Design front end components and ensure app flows in an intuitive manner
- Implement Google Maps API and all necessary functions

PETER MURPHY, AUTHENTICATION & DOCUMENTATION LEAD

Peter has worked in the semiconductor industry for the previous 4 years before deciding on a master's degree in software engineering at Duke.

- Has an in-depth knowledge of engineering and technical solutions to oversee the Authentication for all Android users
- Leads and organizes the effective documentation of entire application including any design and test planning requirements.

APPENDIX

The following sections is a culmination of previous documentation created for this project. It is included as a reference for the reader.

PROJECT DOCUMENTATION PLAN

OUTLINE OF DELIVERABLES

Deliverable	Task	Writer/ Owner	Reviewers	Start Date	Pub Date	Notes
User Requirements	Create new user requirements. Complete with management team.	Peter	Jonathan, Judy	3/2/2016	3/8/2016	
System Requirements	Work with Tech Lead to write and finalize system requirements for Native Android Genie Application.	Jonathan	Peter	3/2/2016	3/8/2016	http://sebokwiki.org/wiki/System_Requirements
System Architecture	Update and finalize the system architecture for the app.	Young-Hoon	Genie Tech Lead (GTL)	3/2/2016	3/8/2015	
Use Case Model	Translate existing use case models into documentation.	Peter	Judy	3/8/2016	3/15/2016	Updates and additions based on Help desk reporting.
Class Diagram	Generate new class diagram for Genie and verify with technical lead.	Jonathan	Peter, Young-hoon	3/2/2016	3/8/2016	https://en.wikipedia.org/wiki/Class_diagram
App Wireframe	Take current wireframe for the iOS app and finalize for Native Android.	Young-Hoon	Jonathan, Peter	3/8/2016	3/15/2016	
Class Description	Provide information on variables, methods, and descriptions for each class	Rush Hour IV team	Rush Hour IV team and GTL	3/8/2016	4/21/2016	
Test Plan	Develop and write concrete test plan for future application development	Rush Hour IV team	Rush Hour IV team and GTL	3/2/2016	3/8/2016	Homework 6 Assignment due 3/8/2016
Supplementary Features	Auto-generate table of contents and other appendices. Also write and spell check Introduction section.	Peter	Rush Hour IV team	4/16/2016	4/19/2016	Some features can be autogenerated in text editors such a Microsoft Word. Final step in document creation.

DETAILED CONTENT PLAN

DELIVERABLE 1: USER REQUIREMENTS

PURPOSE

The user requirements are intended to give a detailed use-case on what the user expects the the document to do. It charters a general use case for the Genie user, and provides future readers with some perspective on the applications purpose.

AUDIENCE

The user requirements are intended to be used by novice Genie users and management team.

DELIVERABLE 2: SYSTEM REQUIREMENTS

PURPOSE

All System-level requirements that describe the system as a whole and should fulfill to satisfy the stakeholder needs and requirements.

AUDIENCE

The user requirements are intended to be used by novice Genie users and management team.

DELIVERABLE 3: SYSTEM ARCHITECTURE

PURPOSE

Designed to be a conceptual model that defines the structure, behavior, and necessary components for the application to function. It also depicts how all of the components interact with each other.

AUDIENCE

Management and development team.

.

DELIVERABLE 4: USER REQUIREMENTS

PURPOSE

The user requirements are intended to give a detailed use-case on what the user expects the the document to do. It charters a general use case for the Genie user, and provides future readers with some perspective on the applications purpose.

AUDIENCE

The user requirements are intended to be used by novice Genie users and management team.

DELIVERABLE 5: USE-CASE MODEL

PURPOSE

The use-case model defines the structure, behavior, and views of a system. It is used as a form of contractual agreement between the developers and management team. Anything not in the use case model will not be implemented in the Genie application.

AUDIENCE

Management and development team.

DELIVERABLE 6: APPLICATION WIREFRAME

PURPOSE

Displays the layout for all activity pages involved in the android app as well as how the user will interact with the app.

AUDIENCE

Management and development team

DELIVERABLE 7: CLASS DESCRIPTION

PURPOSE

The class description diagram will provide the developer with all necessary information regarding all classes. Such information includes a list of variables and their types, methods, and a brief description of the class.

AUDIENCE

All developers

DELIVERABLE 8: SUPPLEMENTARY MATERIALS

PURPOSE

The supplementary features include all documentation material that improve the overall quality of the documentation. Such improvements in quality directly translate to improvement in readability and organization of the documentation.

AUDIENCE

All users

CODE DEVELOPMENT PLAN

This plan is intended to be reference material for all members in the Android development team. It will be used as a starting framework for scheduling the project deliverables.

OUTLINE OF DELIVERABLES

Feature	Feature Description	Writer/ Owner	Sprint Iteration	Dependencies	Notes
Authentication	Provides a secure authentication system for the user.	Peter	1	Database Integration, User Profiles	Can be developed before User Profiles, but not complete until functionality is combined.
Google Maps Interface	Identify nearby users based on their location and prioritize their posts and responses based on proximity.	Young-Hoon	2/3	Database Integration, User's Profile Creation and Location Based services	
Peer-to-Peer (P2P) Messaging	System to allow users in Genie Communicate about post requests	Jonathan	1	User Profile Creation, Database integration	Will need to link to unique profiles before sprint 3
User Profiles	Unique profiles that user generates to	Jonathan	2	Authentication features	
Post/Respond To Requests	Each user has the ability to "Holler" a request and other users can respond and "grant a wish"	Jonathan	2/3	Database Integration, P2P messaging capabilities	
Scoring System	Implement algorithms to determine user karma score	Peter	3	Database Integration	
Front-end Design	Provide the visual components needed for the user to interact with the app in an intuitive and user-friendly manner.	Young-Hoon	1/2	App Wireframe	Work with iOS mockups and generate for Native Android
Database Integration	Integrate application data (user information, help requests, etc) with Firebase	Peter	1	n/a	

FEATURE 1: AUTHENTICATION

GOALS AND TASKS

Authenticates user through safe and secure database methods.

- Register user with database
- Confirm user credentials with database
- Unauthorize user after logging out

ACTORS

All users

FEATURE 2: GOOGLE MAPS INTERFACE

GOALS AND TASKS

Integrate and utilize Google Maps API to locate user and nearby Genie users.

- Connect with Google Maps API to send and receive data.
- Acquire geographical information of user.
- Search and find other Genie users based on your location.
- Restrict search range depending on user's specific preferences.

ACTORS

Google Maps API, all Users

FEATURE 3: PEER-TO-PEER MESSAGING

GOALS AND TASKS

Allows users to communicate from person to person to the completion of a requested action. When a user "Hollers" for help, other users in the community can communicate with them to help accomplish their goal.

- Verify Wifi is available & Set up Wifi connection permissions through application
- Allow user to request, accept or deny ability to communicate with another user
- Generate a private chat window between two users
- Send, receive and store user conversations

ACTORS

Application Users, Wifi capable device

FEATURE 4: USER PROFILES

GOALS AND TASKS

Creates a unique profile for each user. Each profile will allow a users to select their areas of interest as well as "expertise". Profiles will be used to identify a user in the Genie community.

- Create a username and password for new profiles or delete existing profiles
- Allow for user to have photo of themselves for identification
- Text fields with predetermined tags for users interests and skills

ACTORS

Users, Database system (Firebase)

FEATURE 5: POST/RESPOND TO REQUESTS

GOALS AND TASKS

An essential feature for the application, each user has the ability to "Holler" for help with a task and its should be visible to other users in the area. This should be visible on the Google Map display and others user should be able to preview and also request to fulfill the wish.

- Create Activity for Holler Requests
- Display "Holler" Request on Google Maps Display
- Allow for another user to respond via P2P messaging
- Record scoring posts for request and responses from users

ACTORS

User, Genie community

FEATURE 6: USER SCORING SYSTEM

GOALS AND TASKS

The scoring system applies a (karma) score or rank to users. A user's score must accurately display their reliability and trustworthiness to other users to promote a safe environment.

- Implement heuristic to determine a user's karma score
- Design interface for users to vote on other users' score
- Apply rank to user attribute

ACTORS

Users, Genie Community

FEATURE 7: FRONT END DESIGN

GOALS AND TASKS

Develop an intuitive and user-friendly interface that will aid the user in navigating through and using the app.

- Develop splash page and homescreen.
- Create xml page for user profile.
- Create xml screen for posting and replying to current posts.

ACTORS

GUI Builder

FEATURE 8: DATABASE INTEGRATION

GOALS AND TASKS

Database integration is intended to allow all user information and peer-to-peer to be saved on a Firebase database.

- Connect to database and confirm functionality
- Apply users and attributes to database entries
- Integrate peer-to-peer classes and methods to database

ACTORS

Peer-to-peer classes, User Information, Database

SPRINT SCHEDULE

SPRINT 1: 3/1/2016 – 3/14/2016

Task	Owner	Start	Finish	Dependencies
Develop splash page and homescreen.	Young-hoon Kim	3/1/2016	3/6/2016	N/A
Create xml page for user profile.	Young-hoon Kim	3/1/2016	3/14/2016	N/A
Create activity screen for posting and replying to current posts.	Young-hoon Kim	3/1/2016	3/14/2016	N/A
Verify Wifi is available & Set up Wifi connection permissions through application	Jonathan Buie	3/2/2016	3/8/2016	N/A
Send, receive and store user conversations	Jonathan Buie	3/8/2016	3/14/2016	Database access to store conversation history (Potential version 1.0 feature)
Connect to database and confirm	Peter Murphy	3/8/2016	3/14/2016	N/A

functionality				
Register user with database	Peter Murphy	3/14/2016	3/17/2016	Connect to database and confirm functionality
Apply users and attributes to database entries	Peter Murphy	3/14/2016	3/19/2016	Register User with database

SPRINT 2: 3/21/2016 – 4/44/2016

TASK	OWNER	START	FINISH	DEPENDENCIES
Connect with Google Maps API to send and receive data.	Young-hoon Kim	3/21/2016	4/4/2016	N/A
Acquire geographical information of user.	Young-hoon Kim	3/21/2016	4/4/2016	Accessible Database for other app users
Search and find other Genie users based on your location.	Young-hoon Kim	3/21/2016	4/4/2016	Accessible Database for other app user
Restrict search range depending on user's specific preferences	Young-hoon Kim	3/21/2016	4/4/2016	Accessible Database for other app user
Create a username and password for new profiles or delete existing profiles	Jonathan Buie	3/21/2016	3/28/2016	Authorization system
Integrate peer-to-peer classes and methods to database	Peter Murphy	3/21/2016	4/4/2016	Send, receive and store user conversations

Unauthorize user after logging out	Peter Murphy	3/21/2016	3/28/2016	Register user with database
Allow for another user to respond via P2P messaging	Jonathan Buie	3/28/2016	4/4/2016	P2P Messaging system completion
Create Activity for Holler Requests	Jonathan Buie	3/21/2016	4/4/2016	N/A
Text fields with predetermined tags for users interests and skills	Jonathan Buie	3/21/2016	3/28/2016	Database integration and list generation

SPRINT 3: 4/5/2016 – 4/17/2016

TASK	OWNER	START	FINISH	DEPENDENCIES
Display "Holler" Request on Google Maps Display	Young-Hoon Kim	4/5/2016	4/12/2016	Holler Activity completion
Record scoring posts for request and responses from users	JONATHAN BUIE	4/10/2016	4/17/2016	Scoring System
Generate a private chat window between two user	JONATHAN BUIE	4/10/2016	4/17/2016	P2P Messaging, User Profile Creation
Implement heuristic to determine a user's karma score	Peter Murphy	4/5/2016	4/17/2016	N/A
Allow user to request, accept or deny ability to communicate with another user	Jonathan Buie	4/10/2016	4/17/2016	P2P Messaging, "Holler" Activity
Design interface for users to vote on other users' score	Peter Murphy	4/5/2016	4/17/2016	N/A
Apply rank to user attribute	Peter Murphy	4/5/2016	4/17/2016	Design interface for users to vote on other users' score

ASSUMPTIONS

Many of the Assumptions of this project depend on the work we do with the iOS development team. As of now the development has been done in Native iOS code with is not efficiently available for use on Android. The mission of the technical lead is to develop an Native Android application where most of the shared code will be used in the backend with the database. The code generation that we will focus on involves the integration of the database with Android systems as well as the front-end development. We have a vision of the User Interface in the form of mockups created by Genie's marketing team. However, this doesn't provide any code generation, so the xml for the application layouts will need to be generated from scratch for the Android application.
Completion of all deliverables outlined in this plan assumes the following:

- Efficient and Effective creation of the backend database system using Firebase
- Project Use cases have been approved by Founder and technical leads

CONSTRAINTS

We have identified the following constraints on completing documentation activities for Genie:

- Access to technical Lead in Chicago with the expertise in necessary system requirements.
- Working with iOS development team time constraints on back-end deliverables.
- Ease-of- use of the developed firebase database system

TESTING OVERVIEW

The following sections outline the test plan for the development of Genie. The tests are designed to ensure that the software satisfies both the user and system requirements laid out by the management team and will be carried out at the feature level in the form of unit tests and the sprint level in the form of system tests. In other words, after each feature is validated and verified with its own individual test, they will be merged together to complete a sprint. The sprint will not be complete until it has passed all system testing in the form of use cases.

UNIT TESTING

Each feature built during the sprints is subject to Unit Testing in the form of a user story or object class test and not considered complete until it is documented and passed. Each unit test focuses on validation versus defect testing with the emphasis on a timely deployment of the minimum viable product (MVP).

COMPONENT TESTING

Component testing refers to the testing of a sprint and will be exclusively performed as use case tests. Component testing will commence after the features/tasks for that sprint have been tested and merged. Each component/sprint must pass at least one usage scenario.

SYSTEM/RELEASE TESTING

System/Release testing refers to the testing of the final MVP after the last sprint has been tested and verified. At this point, we will likely commit use-case testing. Since mobile applications are difficult to automate, we will likely move to user-testing to expose errors and improve usability.

OTHER TESTING

Other types of testing such as performance testing and beta testing will be likely ignored in favor of functionality and timeliness. These features are considered to be low-priority, meaning that we cannot guarantee full path or decision coverage for our testing.

BUG TRACKING

When a bug or error has been exposed by any test, they will be either fixed immediately or posted to the Issues Tracker on the Project github page. Each issue will sufficiently identify the problem and will be assigned to the owner of the offending module or class.

TEST PLAN

This plan is intended to be reference material for all members in the Android development team. It will be used as a starting framework for scheduling the test framework, and chartering the roles and responsibilities for each test.

FEATURE 1: AUTHENTICATION

TESTING REQUIREMENTS

- Registration saves user data to database
- User must input a password
- Logging out unauthorizes user on device
- There cannot be duplicate users

TESTING METHODOLOGY

- Register user Foo
  - Confirm password is required
- Log in as Foo with incorrect credentials
  - Should fail and throw “incorrect” error message
- Log in as Foo with password
- Log out
- Register user Foo
  - Should fail and throw “Duplicate User” error message

OWNER

Peter Murphy

FEATURE 2: GOOGLE MAPS INTERFACE

TESTING REQUIREMENTS

- The Google Maps Interface must be able to locate the current user and visually show the user’s current location.
- The user’s location information must be able to update periodically and reflect user’s most recent location.
- There must be a function to search and location other nearby users.

TESTING METHODOLOGY

- User Story Test (Features must satisfy this case to pass testing)
  - User opens up map on app and sees that his/her current location is shown visually on the map.
  - The user is curious what other Genie users are nearby and selects an option to search for nearby users and selects the search range.
  - The app searches within the user specified range and updates the map with nearby users, if any are found. If none are found, a message should come up to notify the user that no users were found.

OWNER

Young-hoon Kim

FEATURE 3: PEER-TO-PEER MESSAGING

TESTING REQUIREMENTS

- Must verify Wifi is available & set up Wifi connection permissions on current device

    - Notify user if Wifi unavailable
- Show user the list of available users that can help with Genie request
- Generate a private chat window between two users
- Allow messages to be sent and received between users
    - Store Conversations in the database
- View previous conversations between users in old requests until closed

TESTING METHODOLOGY

User Testing

- Send message from TestUser0 to TestUser1
    - Confirm it was sent and saved successfully
- Send message from TestUser1 to TestUser0
    - Confirm it was sent and saved successfully
- Close application and re-open the conversation between TestUser0 and TestUser1
    - Confirm that the conversation is still visible

OWNER

Jonathan Buie

FEATURE 4: USER PROFILES

TESTING REQUIREMENTS

- User successfully creates as a profile with name and password
- Prompt user to select a variety of “tags” that indicate their interests and store in database
- Ensure Users Karma Score is displayed based on information in database

TESTING METHODOLOGY

Use Case Test

- Register User “temp” with password “temp”
- Pick 5 “tags” and verify they are displayed in the User Profile view page
    - Sports, math, computer science, dancing, cats

Class Test

- With temp User, manually display initial karma score
    - Should be 0 with new profile
- Manually give negative review which should make Karma score negative
    - Verify in the User Profile View Page
- Increase Karma Score 5 times
    - Verify each change in the User Profile View Page

OWNER

Jonathan Buie

FEATURE 5: POST/RESPOND TO REQUESTS

TESTING REQUIREMENTS

- Activity for Holler Requests posts to all user with similar Tags
- Click a Holler Request and open options for direct conversation

- Record scoring posts for request and responses from users

TESTING METHODOLOGY

Use-case Testing

- testUser0 creates a Holler request with Tags “computer science”
- testUser0 has option to message testUser2 due to similar tag interests
- testUser1 should see no recent posts with similar tags
- testUser2 should see 1 new post in the main screen
- Close app and confirm state is preserved

OWNER

Jonathan Buie

FEATURE 6: USER SCORING SYSTEM

TESTING REQUIREMENTS

- User must be able to give others a rating
- User’s rating must be saved and updated on database
- User’s karma score must be appropriately shown based off previous ratings

TESTING METHODOLOGY

Class Test:

- Rate user “Foo” 100 times with fixed rating pattern
- Confirm that this rating matches the expected result

Use Case Test:

- Rate user 5 times with rating system on application
- Confirm that the rating is saved and the karma score is shown appropriately.

OWNER

Peter Murphy

FEATURE 7: FRONT END DESIGN

TESTING REQUIREMENTS

- A splash page must be visible before entering the home screen.
- All features of the app must have an appropriate visual screen.
- Text and images must be properly displayed without distortion or crowding.
- Navigation of the app must be both functional and intuitive.

TESTING METHODOLOGY

Black Box Testing

- Functionality of User Interface will be compared to that of the original mockup design.
- All interactive buttons will be tested to check for functionality.
- Visual components and layouts will be judged subjectively for readability.

OWNER
Young-hoon Kim

FEATURE 8: DATABASE INTEGRATION

TESTING REQUIREMENTS
- Users and attributes must be saved to database
- Conversations must be saved on database

TESTING METHODOLOGY
Use Case Testing
- Create new user and Save settings
- Send messages between two test users
- Set user location
- Confirm that preceding information is saved on Firebase Server

OWNER
Peter Murphy

TEST SCHEDULE

Testing Phase	Features Under Testing	Testing Period
Sprint 1	• Provide Splash Screen and Homepage • User Profiles • Authenticate User	3/18-3/19
Sprint 2	• Google Maps Interface • Peer to Peer Messaging • Post/Respond to Requests	4/2-4/4
Sprint 3	• User Scoring System • Database Integration	4/17-4/26
System Test	• Any outstanding features that need to be tested • Complete System Testing	Ongoing