Built with Firebase

Muninn

Muninn Matchmaking

12/06/2019

Ylewi Adrong-Ayun
Daniel Fyre
Mark He
Christian Villanueva

University of North Carolina at Greensboro
Ike Quigley

Table of Contents

3. Functional Requirements
    n. Primary
    o. Secondary
4. Technical Requirements
    p. Operating Systems and Compatibility
    q. Interface Requirements
        vi. User Interface
        vii. Hardware Interface
        viii. Software Requirement
        ix. Communications Interface
5. Non-Functional Requirements
    r. Performance Requirements
    s. Safety and Recovery Requirements
    t. Security Requirements
    u. Policy Requirements
    v. Software Quality Attributes
        x. Availability
        xi. Correctness
        xii. Maintainability
        xiii. Reusability
        xiv. Portability
    w. Process Requirements

xv. Development Process Used

xvi. Time Constraints

xvii. Cost and Delivery Date

INTRODUCTION

a. **Purpose and Intended Audience**

Muninn Matchmaking is an application made for connecting users across the gaming community. Muninn Matchmaking helps users seek out gamers who share similar interests. Through user created profile and tags, our algorithm matches people whom are the most compatibility to the user. Once introduced, users can communicate with Muninn Matching's in-app chat system so that they can plan their next gaming session.
Our intended audience are those seeking to find gaming friends across the internet with likewise interests. Our app is only available to those in the country of United States of America.

c. **Definition/Jargons**

In old Norse mythology, the name "*Muninn*" derives from "*Huginn and Muninn*" who are two helping spirits in the form of ravens that scours the world in order to see and hear information for Odin.

d. **Project Scope**

The scope of this project is an app-based system that supports the video gaming market directly to its community and does not include any advertising for any specific products nor account billing. Changes to the database are expected as well as the messaging API.

e. **Technical Challenges**

Issues with Github. Resorted to creating a new repository entirely. Varied Sign In Options.

f. **References**

1. Mr. Ike Quigley - (Department of Computer Science) - University of North Carolina at Greensboro
2. Android Studio - (developer.android.com/studio)
3. GitHub - (github.com)
4. Firebase - (firebase.google.com)
5. Applozic's Messaging API - (/www.applozic.com)
6. Evan Yeung - Dota Matching

II. OVERALL DESCRIPTION

g. **Project Features**

This project contains several features some that are completed others that are a work in progress which may or may not make it to the final product.

- Welcome Page - The first user interface page that branches with the login and registration options.
- Login Page - Provides email and password login for returning users.
- Sign Up Page - Requires the user to input minimum information needed for email and password sign up.
- Set Profile Page - Tasks the user to input the necessary information needed in order to save the user into the database and proceed to the Home Page. Necessary information requires minimal data from the user such city and state.
- Home Page - The controller for fragmented views; a bottom navigation bar that contains all these views.
- Matches Page - Allows the user to be matched based on their tags which are compared to other user's tags. During this process of comparing tags, a count is taken to be used to formulate a percentage based on how much the user's tags are related. (Returns true or false depending if the users have a 66% match or more)
- Applozi API - provides the users a form of online communication with the matched user ranging from an implemented global chat to person-to-person.
- Firebase Database - The database that we read and write user information to.

h. **User Characteristics and Classes**

i. **Operating Environment**

Android Studio is the official integrated development environment (IDE) for Google's Android projects and is based on JetBrain's IntellJ IDEA software, a commercial Java IDE with tools specifically designed for Android application development.

j. **Design Implementation Constraints**

Android Studio

k. **Assumptions and Dependencies**

III. FUNCTIONAL REQUIREMENTS

l. **Primary**

A stable internet connections due to the requirement of the Firebase database as well as the use of the messaging API.

m. **Secondary**

Requires the user to enter personal information such as email address, city, or state.

IV. TECHNICAL REQUIREMENTS

n. **Operating Systems and Compatibility**

Only compatible with devices that runs on the Android-based system.

o. **Interface Requirements**

i. **User Interface**

May need the user's permission to read and write the necessary files needed to run the application.

ii. **Hardware Interface**

Runs only Android-based smartphone and tablet device with resistive or surface capacitive touch screen models.

Device is recommended to have at least 30 MB of free space to download this application.

iii. **Software Requirement**

Requires Android version 4.0 (Ice Cream Sandwich - API 14) or newer. The most updated Android Studio SDK.

iv. **Communication Interface**

English (U.S.) Keyboard.

V. NON-FUNCTIONAL REQUIREMENTS

p. **Performance Requirements**

q. **Safety and Recovery Requirements**

Allows user to send a verification email if password was forgotten and have them log into their email to reset it.

r. **Security Requirements**

The application requires a valid email address from the user. The database authenticates users and provides password protected feature access to profile creation and reset password..

s. **Policy Requirements**

Private project owned by UNCG.

t. **Software Quality Attributes**

i. **Availability Correctness**

ii. **Maintainability**

The gradle inside of Android Studio needs to constantly be Updated to stay up to date. AppLozic will also need to be updated to keep the messaging API up to date and working. Firebase will also need to be maintained for storage reasons as well as any updates to Firebase itself.

iii. **Reusability**

The messaging API from AppLozic is reusable. There is a method to handle the account creation and sign-in for the AppLozic servers. This method is called using two lines of code in the register method, making the API very easy to modify and remove. Launching the API is also one line of code, which can be attached to a button, as done in our app.

The user interface can also be reusable as a template. All Strings and colors are stored in their own files respectively, making it very easy to change the values of String and colors.

iv. **Portability**

Applicable for portable smartphone devices with stable connection.

u. **Process Requirements**

i. **Development Process Used**

We used the Scrum Sprint Cycle of the Agile Methodology.

ii. **Time Constraints**

The time constraint for this project was an estimate of 105 days or approximately four (4) months starting from August 27th, 2019 - December 10th, 2019.

iii. **Cost and Delivery Date**

No monetary value was involved in the making of this project. Everything used were either open source or databases.

Firebase provides a free trial for full access for an entire year. Some Firebase resources include a database, servers and support from Google.