ENG131 QualOpt Web App Development

Author: Kaimin Li

Supervisor: Dr. Kelly Blincoe

University of Auckland, Private Bag 92019, Auckland, New Zealand

Faculty of Engineering, ECE Department

Abstract

GitHub users are filing complaints to sites that host data on their profiles as a result of researchers sending out too many surveys using these sites. Many researchers do not give reasons for the number of emails they send out for a particular study, which is a main cause of the issue.

The purpose of the QualOpt web app is to create a platform for researchers to email surveys to GitHub user, as well as a way for unwillingly GitHub users to quickly and easily opt out of future requests.

This report highlights the design and implementation of the web app, the technologies used in the development and an evaluation on the progress of the web app.

Table of Contents

Introduction

Every year, university researchers send out thousands of online surveys via email to various GitHub users to obtain data for their studies. These email addresses are readily available from free online services such as GHTorrent. The GHTorrent site has received numerous complaints about receiving far too many requests for studies and warned their users to use the service in moderation. GitHub users that are frustrated at the number of surveys are asking for their profile details to be removed from the database [1]. In order to handle this issue GHTorrent has created a list for GitHub users who wishes to opt out. This list, however, only holds a very small number of users because the link to the list is difficult to find.

The goal of the QualOpt project is to create a web application for researchers and GitHub users to manage surveys. Researchers will use the application to send out survey invitations in moderation, whilst GitHub user may choose to opt out using a simple link in the emails they receive. This is to address the increasing number of surveys being sent out.

There are also secondary goals to this project, mainly features that will enhance the usability of the web app. This would include filters to selectively email certain groups of users, as well as filters for GitHub user to filter the types of surveys they receive.

Application Overview

The QualOpt web application is separated into two parts: the front-end user interface and the server side web service. The front-end client is designed as the primary user interface for researchers to operate. The server side web service is designed to receive requests from the front-end client and process those requests. Most of the functionality of the features are implemented as a part of the web service.

Front End Client

The front-end client is accessible through a web browser, and mostly consist of HTML and JavaScript. Its primary functionality is to send user form data to the web service for processing. The current front-end client has the following pages:

- A sign-up page for researchers to register an account.
- A login page for researcher accounts.
- A dashboard/main menu where the user can view a list of their studies.
- A 'create study' page where the user can create a study.
- A 'study summary' page where the user can make changes to a study.
- A 'draft email' page where the user can draft and send emails to selected participants.
- A 'participant list' page where the user can set filters and obtain a list of GitHub user emails to use as participants.

A similar interface is planned for GitHub users to use, where they will be able to configure the upper limit for the number of emails they receive, the type of studies they are willing to participate in, as well as other filters. The front-end is fairly lightweight as most of the processing will be done by the server.

Web Service

The web service is developed in Java, and consist of most of the functionality of the web app. The web service oversees receiving user entered data sent by the client and saving it in a MySQL database, processing user authentication, sending emails, managing studies and recording data. Currently, the list of implemented features is as follows:

- User sign up. This receives HTTP form data and stores it into the MySQL database. The email and password used for registration can then be used for login.
- Token based login. This receives login requests and checks if the email and password exists and match. A token is then generated and used for authentication for the remainder of the session.
- Create new study. A study is simply a profile of the survey that the researcher wishes to conduct. This will create one with the information entered by the user and save it in the database.
- Fetch a list of studies. This is tied to the user; whenever a user navigates to the main menu this will send over a list of their studies to display on the page.
- Set and change the current study. For the web service to know what information to use when sending emails, it must know which study the user is using. This keeps track of current study for the web service.
- Send emails. The web service will attempt to connect to the user's mail server using the credentials provided by the user to send out the emails.
- Unsubscribing. Each email sent will have a survey link as well as an unsubscribe link. The unsubscribe link will call the web service to save the GitHub user's email to the unsub table in the database.
- Recording data. The survey link of each of the emails will first direct to the web service, where information about the GitHub user as well as the number of times the survey is click can be recorded. The web service redirects to the survey after recording the data.

Designing the Web App

Designing the web app involved gathering requirements and drawing a paper prototype of the app. Primary specifications were obtained from the supervisor, with additional feedback given from an admin of the GHTorrent site. This built the list of requirements for the project.

The design of the app was drawn using a flowcharting tool. The diagram consists of what each of the pages of the user interface would contain and how the interactions of the pages link up. Below is the design for the planned GitHub user interface:

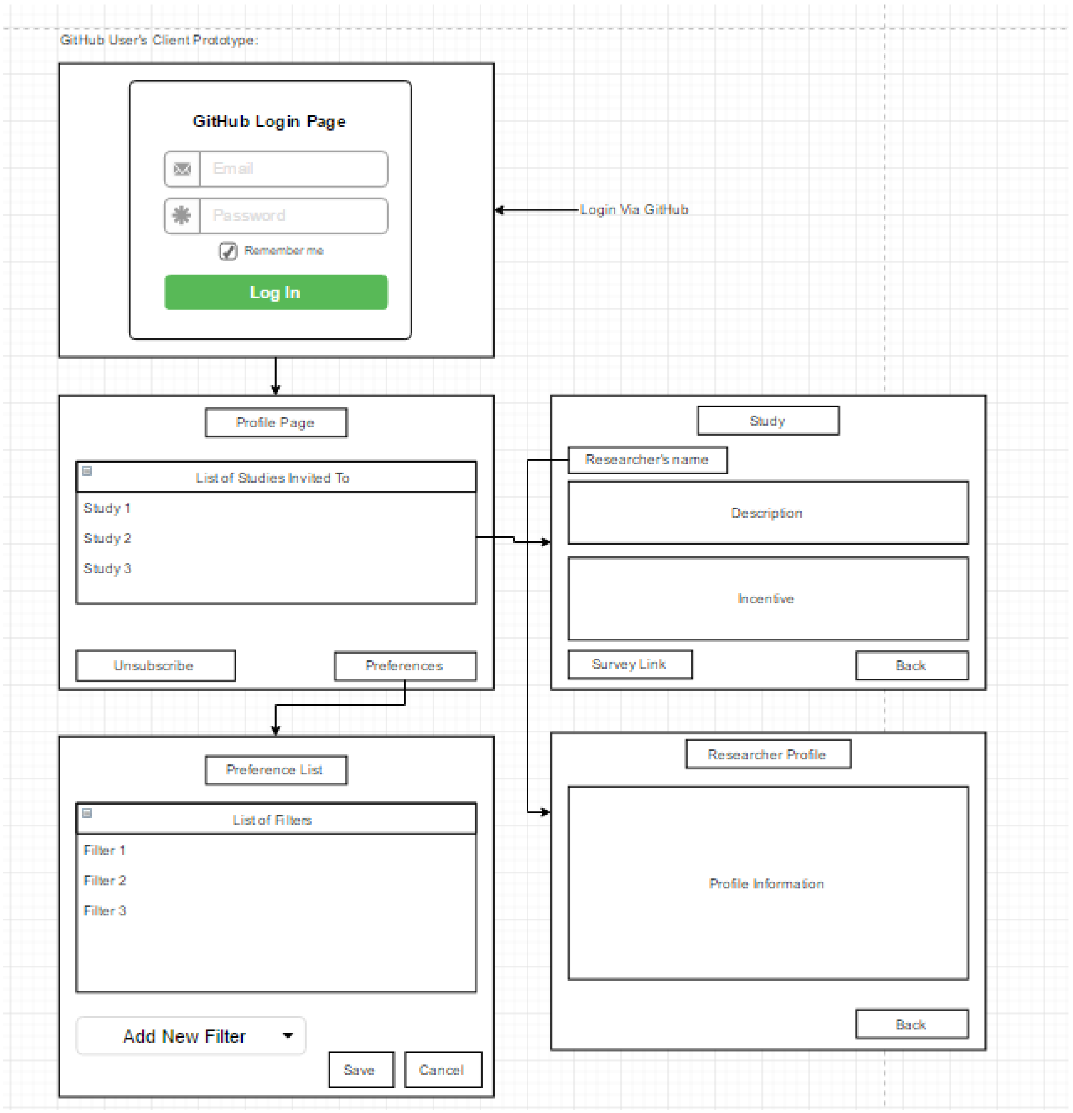

Figure 1, GitHub User's interface

The implemented researcher's user interface is as follows:

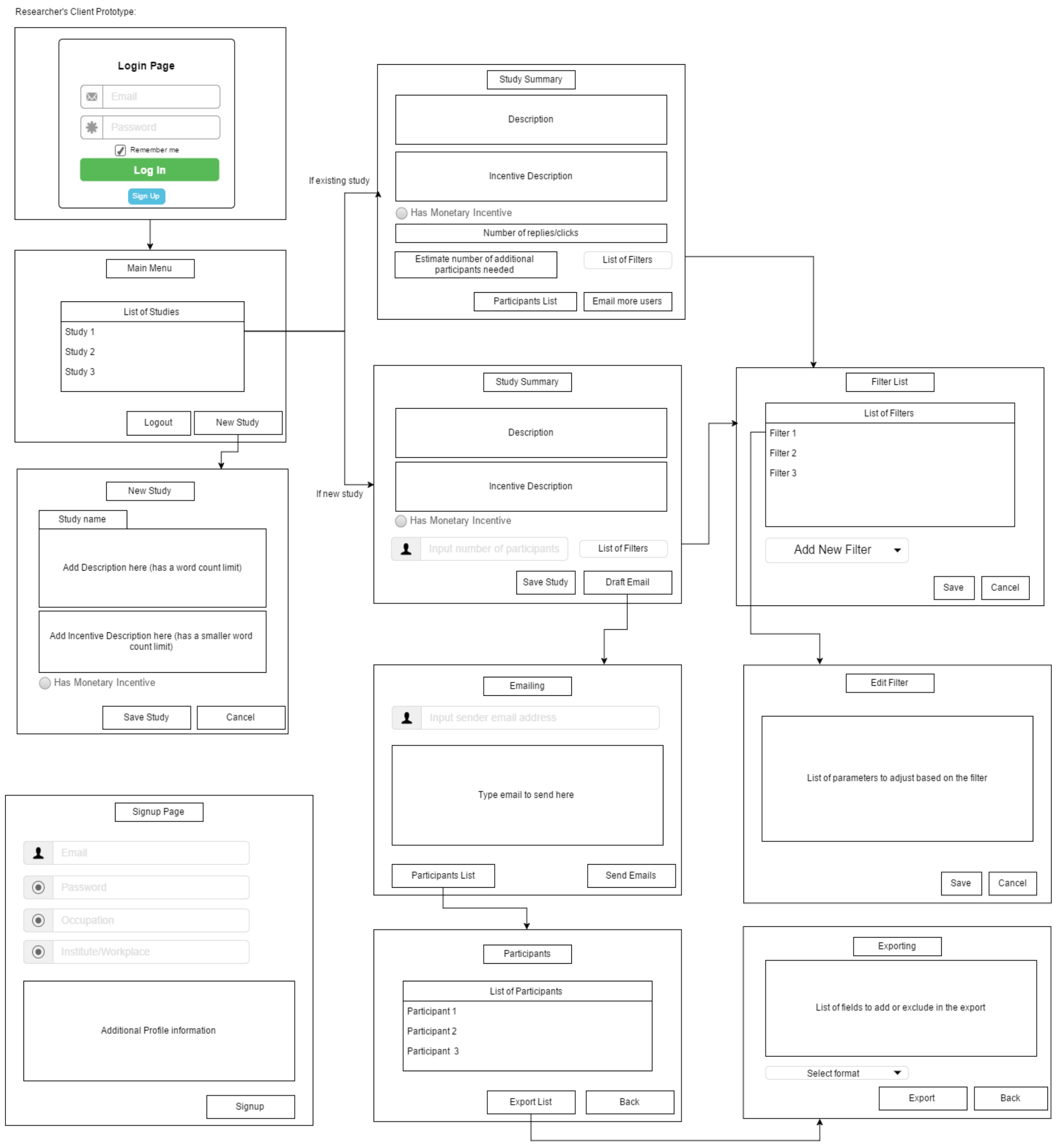

Figure 2, Researcher's interface

Feature Implementation

Front-end to back-end communication

The web app implements the client-server model, where the client handles visual and simple operations and sends request to the server whenever something needs to be processed. The jQuery library is used to perform Ajax calls to the server. This allows the client to configure a HTTP request by indicating the type of request (GET, POST, PUT, or DELETE), data type and content. Upon receiving a response from the server, either a success or error method will be called to either navigate to a new page, load more information to the page, or to report an error.

Token based Login

Token based login systems works by sending the client a token representing its session with the server if the client can authenticate with the server. When the user attempts to login, the server checks its credentials to make sure that they are valid. The server then generates a random string of characters known as a token and sends it to the client. After logging in, whenever the user access a feature where a login is needed, the client sends a request to the server with the token in its HTTP header. The server will first check to make sure the token is correct before proceeding with the request. Since the server keeps track of a list of tokens, multiple sessions can all communicate with the server without interfering with each other. Here is a diagram that describes this:

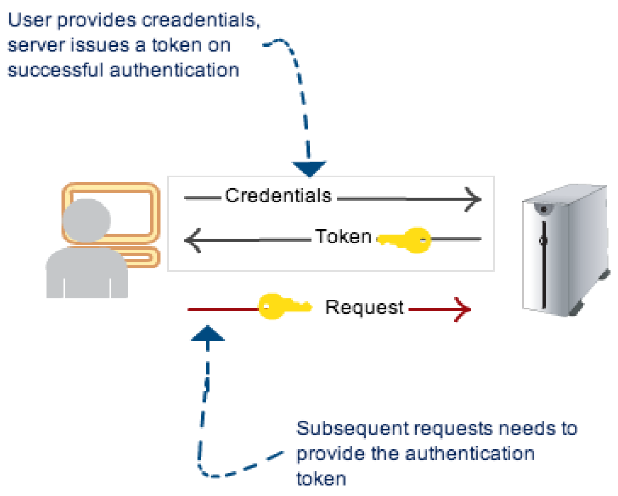
User provides creadentials, server issues a token on successful authentication
Credentials
Token
Request
Subsequent requests needs to provide the authentication token

Figure 3, Token Based Authentication

Data Storage

User profiles, studies and unsubscribed GitHub users are all stored in a MySQL database under their respective tables. The server makes a connection to the database instance upon requested, then parses a SQL statement to the database to be executed. The database either stores a new entry, change the values of an existing entry, or replies with a list of entries.

The user profiles table stores user emails, passwords, general information such as which institution they work for as well as the address to their mail server. All that information will be provided by the user upon signing up. The studies table stores the user ID as the foreign key, name and description, as well as whether the study has monetary incentive. This table is frequently queried by the client to retrieve a user’s studies when the main menu is accessed. The unsubscribe table is updated when a GitHub user clicks the unsubscribe link, and stores that users email address.

The database does not keep data on GitHub users. This information is readily available on sites like GHTorrent and can be accessed by querying their services. Keeping a full database of GitHub user information requires constant updating and large disk capacities and is therefore out of scope.

Web Service Framework

The framework used to for the implementation of the web service is JAX-RS. This is a Java API for creating REST web services; it uses annotations to simplify the implementation.

Each feature of the web service is implemented within a method (for example, login is a method on its own), then annotated with the appropriate tags depending on the type of request it receives. The path of the method, type of content produced and type of content received are also defines using annotations. Furthermore, parameters such as query parameters or form data can be extracted using annotations.

Evaluation and Future Work

The core features of the QualOpt web app have been implemented, however due to the nature of the app (emailing hundreds of GitHub users) a few of the feature were not tested properly. One of the major setbacks to the project was querying GHTorrent for GitHub data. The service required SSH tunneling which must be approved by the GHTorrent admins beforehand. This slowed down the filter implementations. Future work on this project will need a reliable source of GitHub data.

The QualOpt project is in the prototype stage and is therefore unfinished. Many additional features can still be implemented to increase the usability of the software:

- Additional filters for thee researcher client. This includes filtering based on commit count, projects, time and more.
- A user interface for GitHub users to configure the amount and type of surveys they receive. Similar filters may be used.
- Password and user information encryption. Currently most of the information sent to and from the server are plain text or form data which can be a security concern if the app is deployed for practical use. Encryption is essential pass the prototype phase since privacy sensitive information is core to the functionality of the app.
- Automated data collection is implemented on a basic level (click count of surveys), and can be expanded to record more about the user.

Due to time constraint issues, the project did not receive formal usability feedback.

References

1. *GHTorrent FAQ.* Retrieved February 14, 2017 from http://ghtorrent.org/faq.html