

CS 320 Course Project Final Report

for

Hot Logbook

Prepared by

Group Name: *Sasswords Puck*

|  |  |  |
| --- | --- | --- |
| Jesse Griffin | 11641870 | [jesse.w.griffin@wsu.edu](mailto:jesse.w.griffin@wsu.edu) |
| Patrick Kent | 11019238 | [patrick.kent@wsu.edu](mailto:patrick.kent@wsu.edu) |
| Pierson Cavulli | 11608370 | [pierson.carulli@wsu.edu](mailto:pierson.carulli@wsu.edu) |

|  |  |
| --- | --- |
|  |  |
| Date: | 13 December 2018 |
|  |  |
|  |  |
|  |  |

Contents

Contents ii

1 Introduction 1

1.1 Project Overview 1

1.2 Definitions, Acronyms and Abbreviations 1

1.3 References and Acknowledgments 1

2 Design 2

2.1 System Modeling 2

2.2 Interface Design 2

3 Implementation 3

3.1 Development Environment 3

3.2 Task Distribution 3

3.3 Challenges 3

4 Testing 4

4.1 Testing Plan 4

4.2 Tests for Functional Requirements 4

4.3 Tests for Non-functional Requirements 4

4.4 Hardware and Software Requirements 4

5 Analysis 5

6 Conclusion 6

Appendix A - Group Log 7

# 

# Introduction

## Project Overview

The Hot Logbook is an passoword manager that allows users to store their account information privately. The user will be able to keep track of their accounts, edit accounts, and remove accounts through our interface.

When the user wants to add an account, they will be given the option to use a randomly genearated password. This is a secure password that would be extremly hard to hack. The user will also be able to edit previously added accounts for cases where changing a password or user name is needed.

## Definitions, Acronyms and Abbreviations

App: Application

CSS: Cascading Style Sheets

CSRF: Cross-Site Request Forger

DB: Database

FAQ: Frequently Asked Questions

HTML: Hypertext Markup Language

JS: JavaScript

RFI/LFI: Remote/Local File Inclusion

SQL: Structured Query Language

SRS: Software Requirements Specifications

UI: User Interface

XSS: Cross-Site Scripting

### Naming Conventions

Variable and function names follow the Camel-case convention unless otherwise specified.

## References and Acknowledgments

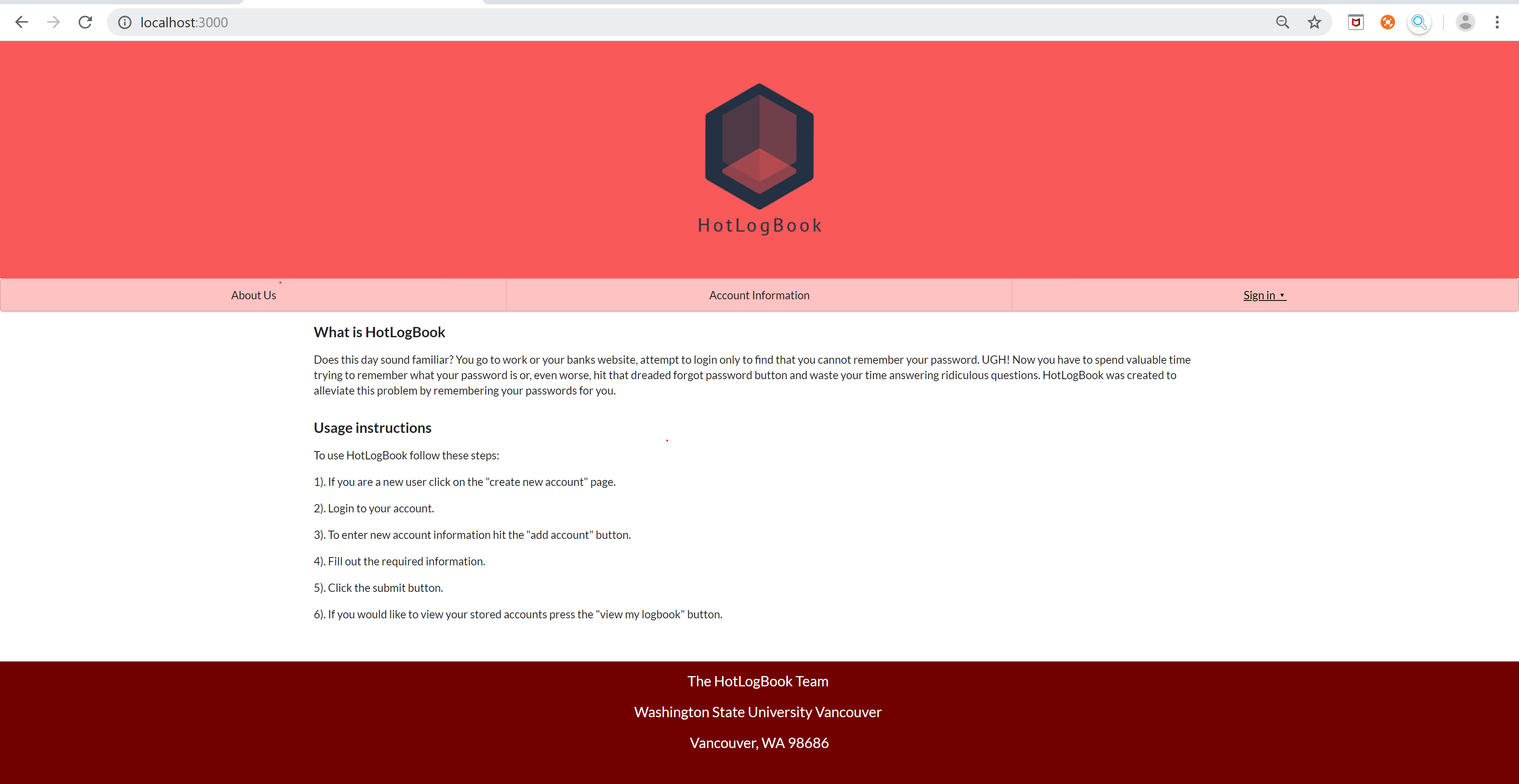
Our site would not be possible without the aid of Meteor and Semantic UI.

# Design

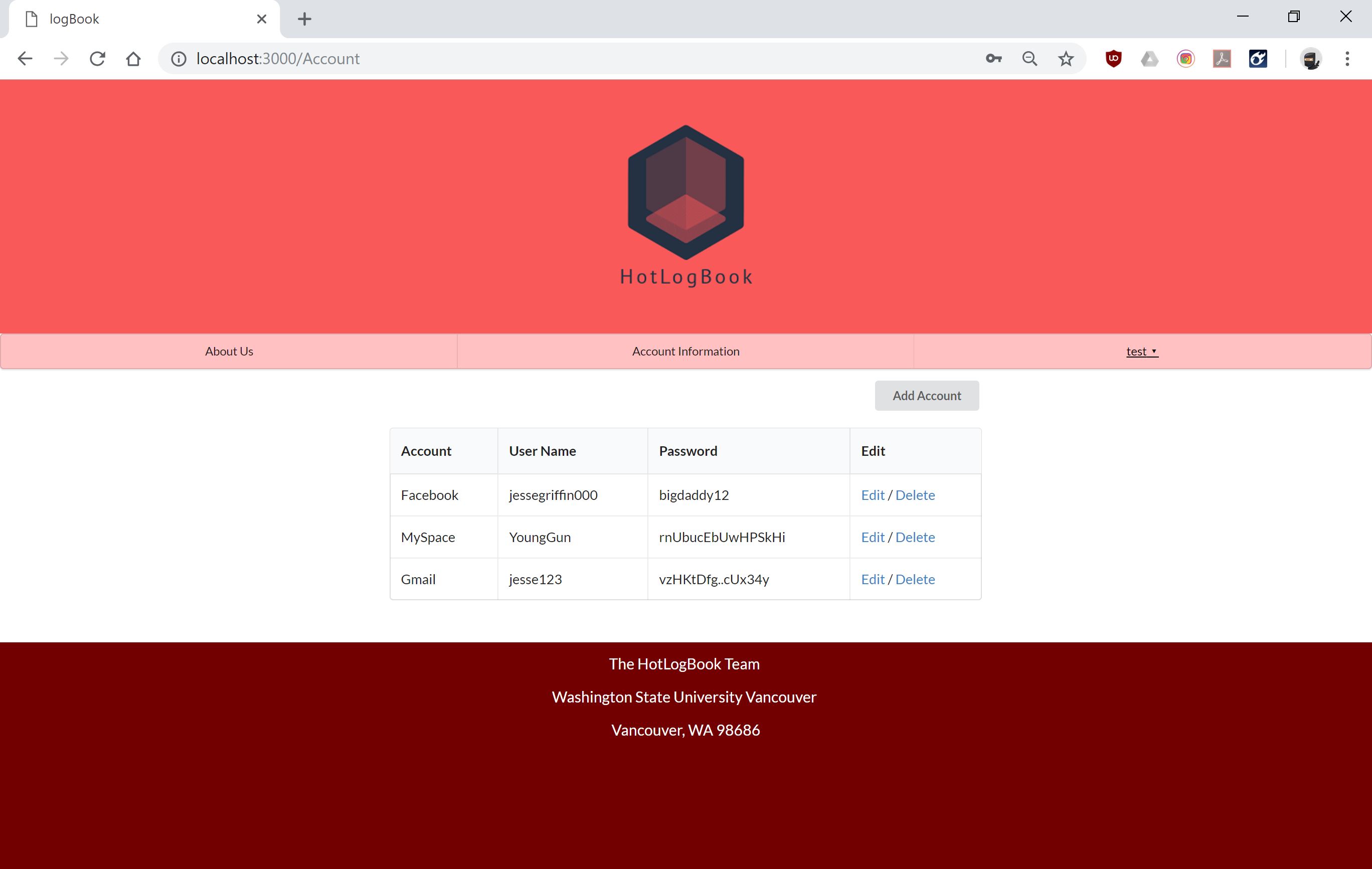
## System Modeling

## Interface Design

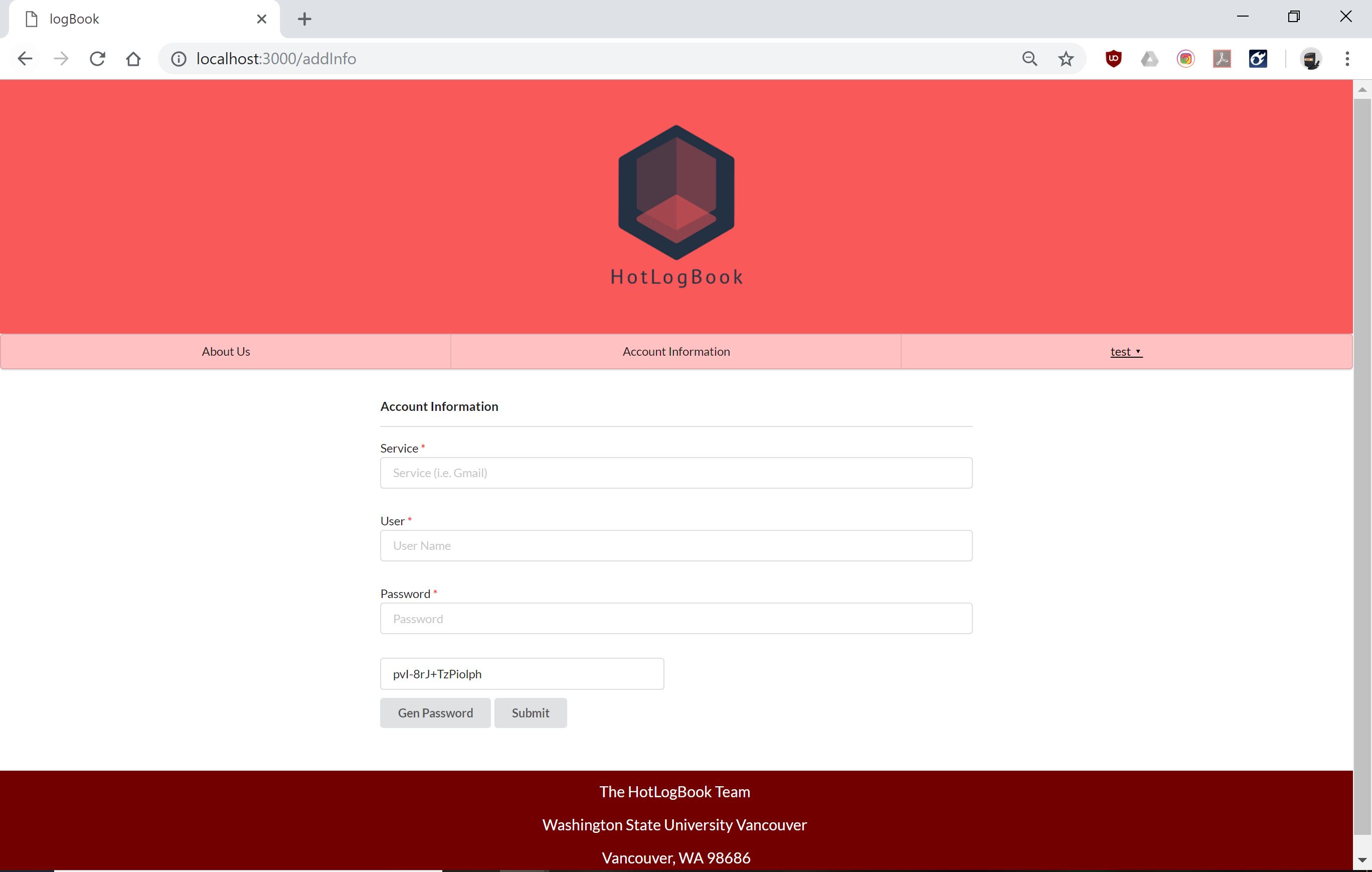
The following pages are final implementations of the page/UI designs laid out in the original project SRS document. General design mimicked assignment 7 with the dynamic template page layouts.

**Home/About Page**

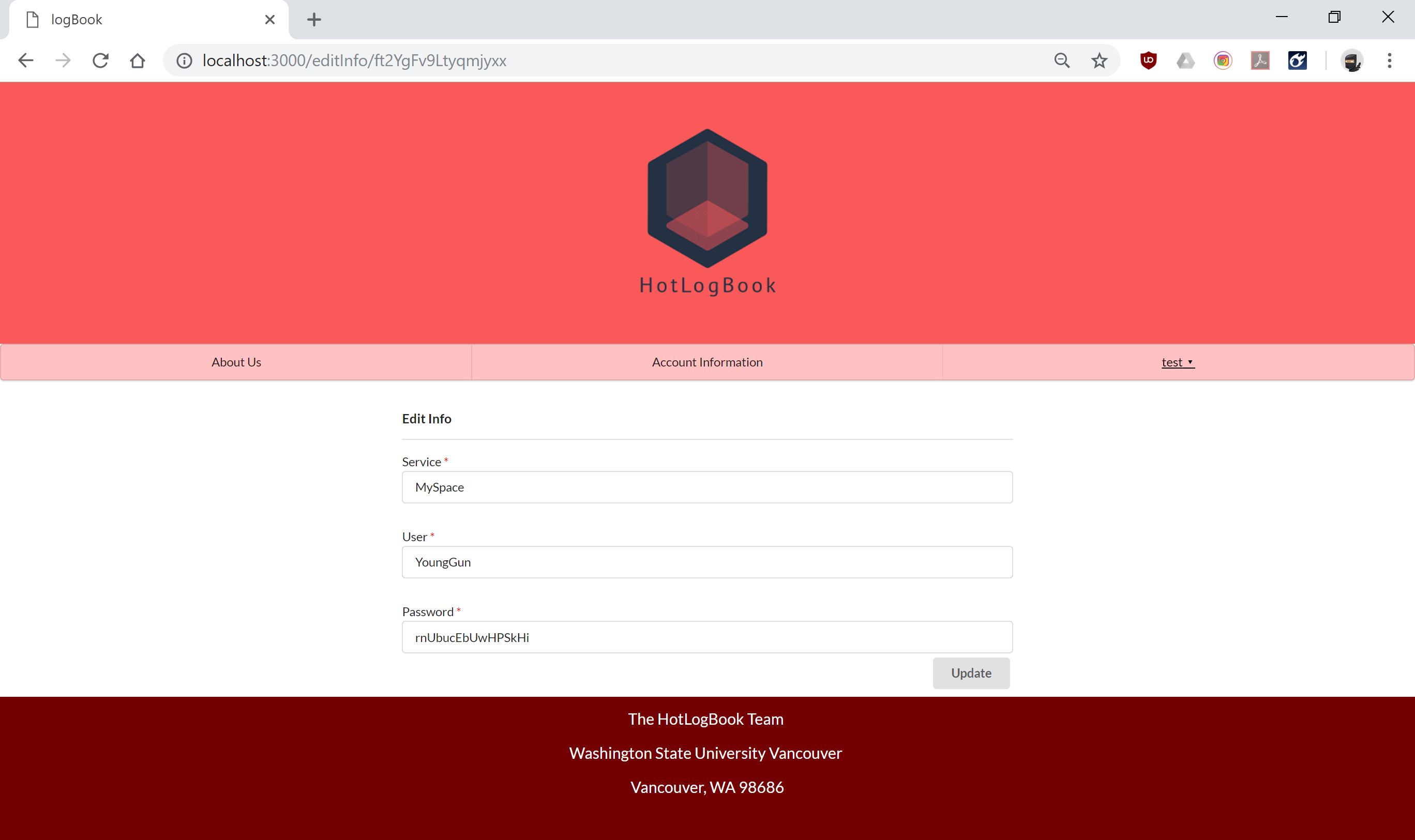
**Account Page**



**Add Account Page**



**Edit Account**



# Implementation

## Development Environment

Tools and Languages used to develop the HotLogBook:

* JavaScript
* HTML/CSS
* Meteor
* IntelliJ
* SemanticUI
* Mongo Database

## Task Distribution

*Jesse Griffin*: Created logo design for project. Also implemented sign in, add page, edit page, and schema inspired from Assignment 7.

*Patrick Kent*: Implementation of individual user privacy and user accounts. Tasked with implementing a delete function (didn’t succeed). Project editor (text and code submissions).

*Pierson Cavulli*: Started project and created file structure for project, edited the add account form (originally created by Jesse) to allow the results to be saved to Mongo, created/managed routes.js. Also created home page and many CSS contributions.

## Challenges

The challenges our group experienced were implementing individual user accounts and adding the delete feature for our app. The way we solved this problem was completing meteor tutorials and learning how schemas and user id’s are stored.

Other problems we ran into were GitHub related. We spent a whole team session on learning how to work together with git without creating merge conflicts. While GitHub was a hinderance in the beginning of our project, it ended up being helpful towards the end by allowing us to implement effective version control.

# Testing

## Testing Plan

* The genertate password function
* The links in the project
* Browser support (test which browsers work best with our application)

We plan to have all testing done by 12/13/2018

## Tests for Functional Requirements

We implemented the black box testing method. Consequently, our program was tested both by the Sasswords Puck team and random end users. The main funstionalities of the website were heavily tested both by the team and the end users who used our software. Any bugs reported have since been fixed. The HotLogBook passed the last test, which was performed on 12/9/2018. Here is a short list of the functionalites that have been tested:

* The generate password function was tested to ensure that a random password was generated on each click.
* To make sure the routes.js file was correclty routing the users requests all the links on our site were tested.

## Tests for Non-functional Requirements

The HotLogBook application works well in Chrome, and Firefox; unfortunatly, the font-fire effect does not work in Internet Explorer or Microsoft Edge. While the font-fire does not function in Microsoft Edge the functionality of the HotLogBook is unaltered. The Sasswords Puck team would recommend not using Internet Explorer because the edit account option does not work. A list of other non-functional tests that were performed include:

* Speed test: does the application hang for long periods of time
* CSS test: does each page render properly

## Hardware and Software Requirements

### Minimum Hardware Requirements

CPU: Intel 3rd generation/AMD 2013 or newer

Memory: 4 GB

Graphics: Onboard/Integrated Intel/AMD

# Analysis

## Project Hours

**Jesse Griffin**

Milestone 1: 6 hours

Milestone 2: 4 hours

Milestone 3: 16-20 hours

**Patrick Kent:**

Milestone 1: 5 hours

Milestone 2: 4 hours

Milestone 3: 20 hours

**Pierson Carulli:**

Milestone 1: 4 hours

Milestone 2: 3 hours

Milestone 3: 25-30 hours

Milestone 1 took a while because we were dividing out the work and talking about how we can work together. It was also where we developed the idea of our project.

Milestone 2 was difficult for how abstract it was. Hot LogBook is fairly simple in its scope: a web app that can recommend passwords for you and store them in a private user database. Designing the UML diagrams and class diagrams for something for which we didn’t really have any custom functions for proved challenging. That challenge was overcome by being mindful of overthinking, and following the K.I.S.S. model (Keep It Simple, Stupid). Less time was spent on this stage than the other two milestones.

Milestone 3 took the longest, being the implementation stage of the project. The majority of time was spent on discovering and solving new bugs, in addition to learning new data structures with MongoDB and the Meteor application infrastructure. Significant time was spent on attempting to implement a delete function, which one would think is as simple as insert, but from our knowledge of trees we know isn’t so simple.

# Conclusion

Another year, another project under the belt for some members of this team. Another lesson learned, carried over from previous projects by all members, is time management. It helps to have meetings planned out well in advance

Appendix A - Group Log

Team Sasswords Puck had extremely limited time to meet during the semester. We were lucky to get an hour to work productively in person every two weeks. Because of this, we talked at least once a week over skype to go over goals and expectations. Work was often performed individually, with group notices sent out to avoid merge conflicts in git. For the first three meetings in the semester, group minutes were kept and an agenda was written up, but offset work schedules meant actual meetings became harder to organize, so agendas and minutes were no longer tracked.