**iBudget**

**Software Requirements**

**Specification**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Approvals: | L. Assayah | \_\_\_\_\_\_\_\_\_\_\_\_\_ | date: | \_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | C. Leung | \_\_\_\_\_\_\_\_\_\_\_\_\_ | date: | \_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | Q. Pham | \_\_\_\_\_\_\_\_\_\_\_\_\_ | date: | \_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | V. Velev | \_\_\_\_\_\_\_\_\_\_\_\_\_ | date: | \_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | J. Reimels | \_\_\_\_\_\_\_\_\_\_\_\_\_ | date: | \_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Author** | **Version** | **Reason** |
| 2/20/12 | V.Velev | 1.0 | First Draft |
|  |  |  |  |

# Introduction

## Purpose

This document provides all of the requirements for the *iBudget* project.

## Scope

This document covers the design for release 1.0 of *iBudget*.

## Definitions, acronyms, and abbreviations

## References

## Overview

# Overall Description

## Product Perspective

*iBudget* is intended to address the need for a budgeting and personal finance tool that does not require a secure connection to a financial institution or one’s login credentials to that institution. It is targeting a market that is conscious of internet safety and aware of the risks associated with providing sensitive information to an online service. *iBudget’s* appeal is that itwill provide peace of mind to its clients without sacrificing any of the functionality offered by its competitors.

* + 1. **Concept of operations**
    2. **User interface concepts**
    3. **Hardware interfaces**
    4. **Software interfaces**
    5. **Communication interfaces**
    6. **Memory constraints**
    7. **Operations**
    8. **Site adaptation requirements**

## Product Functions

## User characteristics

## Constraints

## Assumptions and dependencies

## Apportioning of requirements

# Specific Requirements

3.1 Functional Requirements

**3.1.1 Create an account**

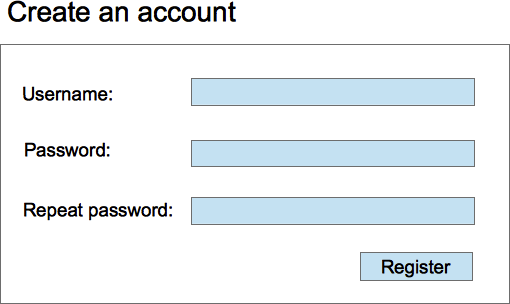
The user has to first sign up to get an account and be able to log in the next time he will access the website.

Actions:

* Click on the ~~“Create account”~~ “Register” hyperlink. The user is then brought to the corresponding webpage.
* Fill a form with the fields:
* Username (must be at least 3 characters and unique)
* ~~Email address (must be valid)~~
* Password (must be at least 8 characters, include at least 1 upper case character, 1 lower case character, 1 number and 1 symbol)
* Password Confirmation
* ~~Address (optional)~~
* ~~Zip code (optional)~~
* ~~State (optional)~~
* ~~Country (optional)~~
* Click on the ~~“Submit”~~ “Register” button

Results:

* If the user failed to fill the form correctly, the form is displayed again with the reason for the error in the corresponding field (not enough characters, username already used…)
* If the user succeeded then a message saying “Your account has been created” is displayed. The user is then redirected to the homepage and directly logged in. ~~A confirmation email is also sent to the user email address to confirm that he has signed up.~~



**3.1.2 Login**

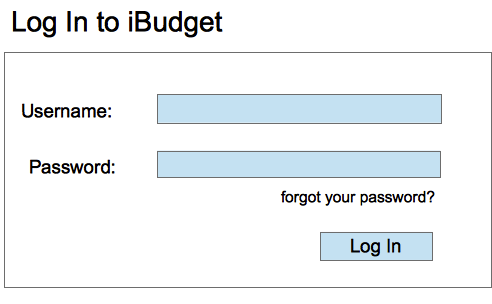
The user has to login to use the functionalities the website.

Actions:

* The user must click on the “log in” hyperlink
* ~~A pop-in window is displayed~~ The user is taken to the log in page where he has to fill out the username and password fields
* Then he must click on the ~~“Connect”~~ “Log In” button

Results:

* If the user failed to log in, the window is still displayed and an error message appears at the top of the form saying “Wrong username or password”
* If the user succeeded, the window disappears and the user is logged in in the homepage.



**3.1.3 Change password**

The user can change his password anytime when he is logged in.

Actions:

* The user clicks on the “Change password” hyperlink.
* He has to fill the fields:
* Old Password
* New Password (must be at least 8 characters, include at least 1 upper case character, 1 lower case character, 1 number and 1 symbol.)
* Confirm new password
* He clicks on the “Ok” button

Results:

* If the user failed to fill the form correctly, the form is displayed again with the reason for the error in the corresponding field (not enough characters)
* If the user succeeded then a message saying “Your password has been changed” is displayed. The user is then redirected to the homepage.

**3.1.4 Get a new password if the user forgets it**

Sometimes the user can forget his password. The application can provide a way to get a new password quickly.

Actions:

* The user clicks on the “forgot password” hyperlink.
* A message saying if the user wants to have a new password is displayed. If the user clicks on the “Yes” button, an email is sent to the user’s email address.
* The user has to open the email and click on the link.
* He is redirected to a webpage that allows him to type a new password in a text field. He has to type the password twice to confirm it.
* He clicks on the “Ok” button. A confirmation email is then sent to the user telling him that his password has indeed been changed.

Results:

* The user has a new password and so he can log in in the website with the new password and can access his data without any difficulties.

**3.1.5 Log out**

The user can log out of the application before leaving the website or if he wants to log in with another username.

Actions:

* In every page of the website, the user can click on the “Log out” hyperlink

Result:

* The user is logged out and redirected to the homepage of the website.

**3.1.6 Upload CSV files**

The user can upload his bank files in CSV format in order to visualize them the way he wants afterwards.

Actions:

* The user clicks on the “Upload CSV file” hyperlink on the navigation bar. The corresponding webpage is then displayed.
* The user clicks on the “Upload file” button.
* The user can then choose the file he wants to upload on his computer and then clicks “OK”

Results:

* If the user failed, (wrong format, file can’t be read…), an error message is displayed and the user is invited to try again with uploading the file or another file.
* If the user succeeded, the data has been successfully uploaded and the user is asked if he wants to upload another file or not. If not, he is redirected to the homepage.

**3.1.7 Edit User profile**

The user can access his profile to see his information and can edit ~~them~~ it at will.

Actions:

* The user clicks on the “Profile” hyperlink in the navigation bar.
* The webpage is displayed and all the fields that the user can modify are organized in editable text fields. The user can modify information like city, zip code, state, email address etc…
* When the user is done editing, he clicks on the “Save” button.
* If he doesn’t want to edit after all, he can click on the “Cancel” button

Results:

* If the user clicked on “Save” and if there is no errors, he is redirected to the homepage and his profile has successfully been edited.
* If the user clicked on “Cancel”, A confirmation message is displayed in a pop-up window: “Are you sure you want to cancel?” If the user clicks “Yes”, he is redirected to the homepage and his information has not been changed. If he clicks on “No”, the window disappears and he can continue editing.

**3.1.8 Create custom categories**

The custom categories enable the user to associate data the way he wants and not be limited by the default categories.

Actions:

* The user clicks on the “Create Category” hyperlink on any page of the website.
* The webpage is displayed and the user can fill the text field with the name he chose.
* If he wants to add several names at a time, he can click on the “+” button. Another text field will appear below every time he clicks on the button.
* When the user is done with the names, he clicks on the “Save” button.

Results:

* The categories are saved and can be used later when the user wants to map data with categories

**3.1.9 Create custom financial institute mapping for data import**

The user can map the data he imported with any of the categories proposed or that he created.

**3.1.10 Set preferences**

**3.1.11 Change preferences**

**3.1.12 Delete an account**

The user can delete his account from the application.

Actions:

* The user clicks on the “Profile” hyperlink on the navigation bar of the website.
* Then he has to click on the “Delete Account” button.

Results: A confirmation message is displayed

* If the user clicks on “Yes”, a message saying “Your account has been successfully deleted” is displayed.
* If the user clicks on “No”, the account is not deleted and the user is still on the profile webpage.

3.2 Non-functional requirements

**3.2.1 Consistent look for different Internet browsers.**

The application should have the same look for all browsers. There can be differences related to the specific ways a browser can display some items.

~~The supported browsers are:~~ We currently support the latest versions of the following browsers:

* Mozilla Firefox
* Internet Explorer
* Safari
* Google Chrome

**3.2.2 Enforce strong password**

When the user chooses his password, he has to respect the following:

* at least 8 characters long
* at least 1 upper case character
* at least 1 lower case character
* at least 1 number
* at least 1 symbol

**3.2.3 End of session**

The application should end the session if the idle lasts more than 15 minutes. The user is then logged off automatically. All the data ~~he did~~ not saved will be lost ~~when he logs in again~~.

**3.2.4 Built-in reports**

The application should display any built-in report in less than 60 seconds.

**3.2.5 Clarity**

The application’s front-end should be clear and user-friendly. There will be a navigation bar at the top of the screen for easy navigation of the site.

**3.2.6 Encryption**

The user’s password must be single way encrypted.

**3.2.7 SSL**

Secure Sockets Layer (SSL) will be used to protect the privacy of all user-related data and its integrity

**3.2.8 Module functionality**

The Data Access layer will be abstracted from the core application code. Factory and keys and controllers will also be used to ensure modularity of the core code.

**3.2.9 Cookies**

The application should be able to remember ~~the user’s password using cookies~~ logged in users by using cookies.

**3.2.10 Portability**

The system will be cross-platform compatible and the only ~~condition~~ requirement for users ~~to use it~~ is access to the internet and a web browser.

**3.2.11 Maintainability**

Changing the look and feel of the application can be easily done just by switching the style sheet (ibudget.css). Also by abstracting out common element such as site-wide header and footer, making a change to them makes it a trivial task. Moreover, modular design has been incorporated throughout the whole system, so future requests for enhancements can be easily accommodated.

**3.2.12 Performance**

The application should load and display the pages in less than 10 seconds on an idle system with no network congestion. In addition, any built-in reports should be displayed in less than 60 seconds as taking longer than that will have a serious impact on user experience.