

Checkbook Register

Final Report

Coursera Web Development Capstone Project

Assignment 4, 2016-06-12

Table of Contents

1	Introduction.....	3
1.1	Brief description of checkbook register usage.....	3
1.2	How this program helps.....	3
2	Design and Implementation.....	4
2.1	Converting from Design to Code.....	4
2.2	Choices and Modifications.....	4
2.3	Modules and Libraries.....	4
2.4	Screenshots.....	5
3	Conclusions.....	9
3.1	Results.....	9
3.2	Features and shortcomings.....	10
3.2.1	Features.....	10
3.2.2	Shortcomings.....	10
3.3	Hindsight.....	10
4	References.....	10
5	User Manual.....	11

1 Introduction

1.1 Brief description of checkbook register usage

When keeping a check register, new entries have not necessarily been processed by the bank yet. When an account statement is received from the bank, the account holder can then know that the bank has actually processed the transactions listed in the statement.

A responsibility of the account holder is to “reconcile” the check register with the bank statement. The first step is to mark in the register those transactions that have “cleared” the bank, i.e. appear on the statement. Next, create entries for transactions the bank has processed that are not yet in the register, and immediately mark those entries as cleared. The final step is to compare the bank's notion of your balance with your checkbook register's notion of the balance of cleared transactions; they should match.

Some transactions may never “go through” (be processed by the bank) due to a lost check or a misunderstanding. These entries show up with older dates and have not cleared. When a sufficient time has gone by that you're fairly certain these entries will never go through, they can be “written off”: cleared and an equal amount added back in to the register.

1.2 How this program helps

This program helps the user maintain an online check register that looks very much like a paper register, but has some automated features. The primary platform is a web browser so it will work from any computer (including Chromebook) or phone.

Features include

- user management with authentication
- automatic balance, totals of cleared and uncleared items
- track the month an item cleared-balance
- display just open items or all items
- as simple & intuitive as possible – to be used by non-savvy 83-year-olds
- web browser-based – no need for a cell phone or tablet
- entries stay where you put them

2 Design and Implementation

2.1 Converting from Design to Code

I created Loopback models directly from the database table descriptions from assignment 3, Architecture Design and Software Structure Report. That's the totality of the back-end.

The various services were implemented directly from the specification in assignment 3. Additions were made where necessary for sharing data between controllers.

The design of the user interface in assignment 2, UI Design and Prototyping Report, led directly to the front-end code.

2.2 Choices and Modifications

Early on, I decided to use a two-line table for most of the controls and indicators. This was chosen to minimize space for use on small screens, e.g. 5" tablet/phones. It allows a label (the table header) and a button or value (the single row of table body). One button in particular acts as both a control and an indicator: the display all or uncleared button toggles its label to indicate the current state.

Initially, two of the buttons were dropdown lists whose contents changed depending on user login state and account selected state. The user login dropdown select would have two items when not logged in, login and create user, or two different items when logged in, logged in user name (as an indicator) and log out. The account dropdown also had different options depending on whether or not an account was selected. This method worked great during initial development, which was using Firefox on a Linux desktop. However, as soon as I tried it on an Android device, I discovered that it just didn't work; then I also discovered it didn't work on desktop Chrome or Opera browsers. These discoveries necessitated converting the dropdown selections to modals. The modal method works on all browsers tested, desktop and Android.

I am using \$rootScope in several places to communicate between controllers. During development, I discovered that I could be using variables in services, instead, so some places use this method. There are some callbacks set up in \$rootScope or services so that when a particular variable is changed in one controller, the callback can do something needed in another controller that shares use of that variable. After the code was written, tested, and fully operational, I discovered a simpler, cleaner solution: using \$emit and \$on through a service. I have not yet rewritten the code to take advantage of \$emit/\$on for this assignment, but after submission, I do intend to clean it up.

2.3 Modules and Libraries

My modules, "checkbook", "checkbook.controllers", "checkbook.services", match the breakdown of modules taught in class and the MVC pattern. "checkbook.controllers" and "checkbook.services" are inserted into "checkbook", so they don't have to be sourced explicitly in index.html.

Bootstrap was chosen to facilitate use on smaller screens. It will be used more fully in later versions of

this application.

AngularJS is central to the entire design of this project. It facilitates easy use of the MVC pattern.

NodeJS is necessary to use Loopback.

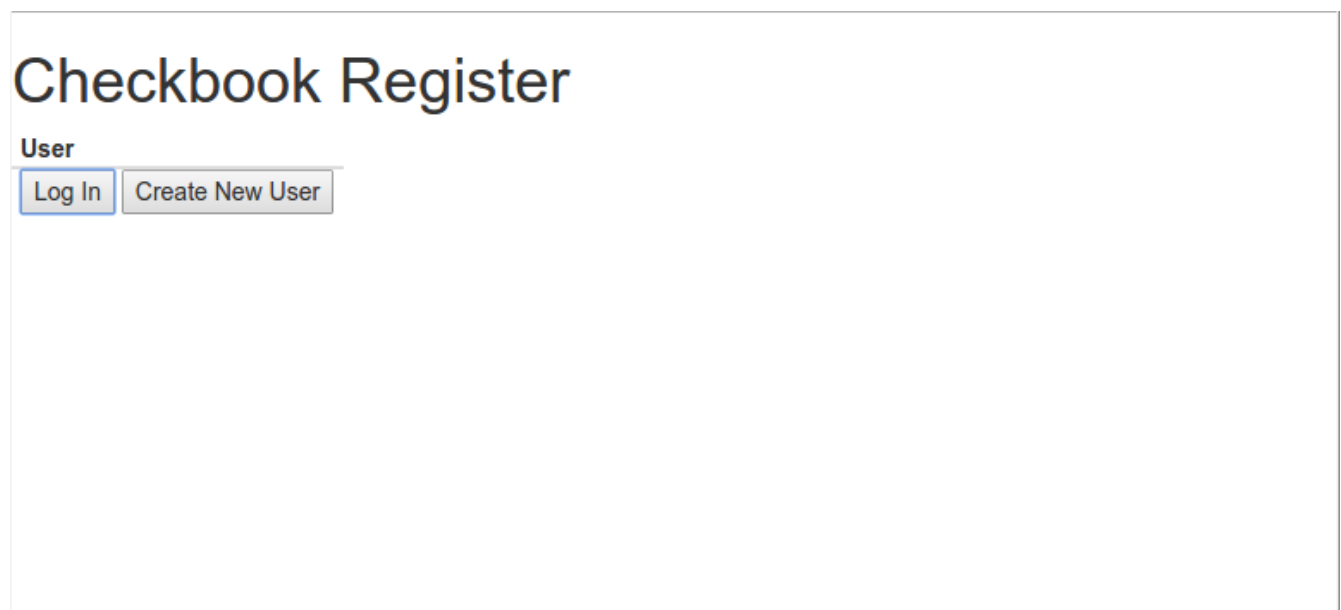
Loopback, along with Mongoose and MongoDB, are a very easy (almost trivial) framework for creating the back-end, including user authentication, authorization, and a web interface. For my project, I didn't even add any code to the back-end that Loopback created, just used the database table models.

I considered using a Python back-end because Python runs on more systems than NodeJS, and I'm more familiar with Python. However, for the purposes of this class, I didn't want to take the time to learn the Python back-end modules, and it would have made peer evaluation more difficult.

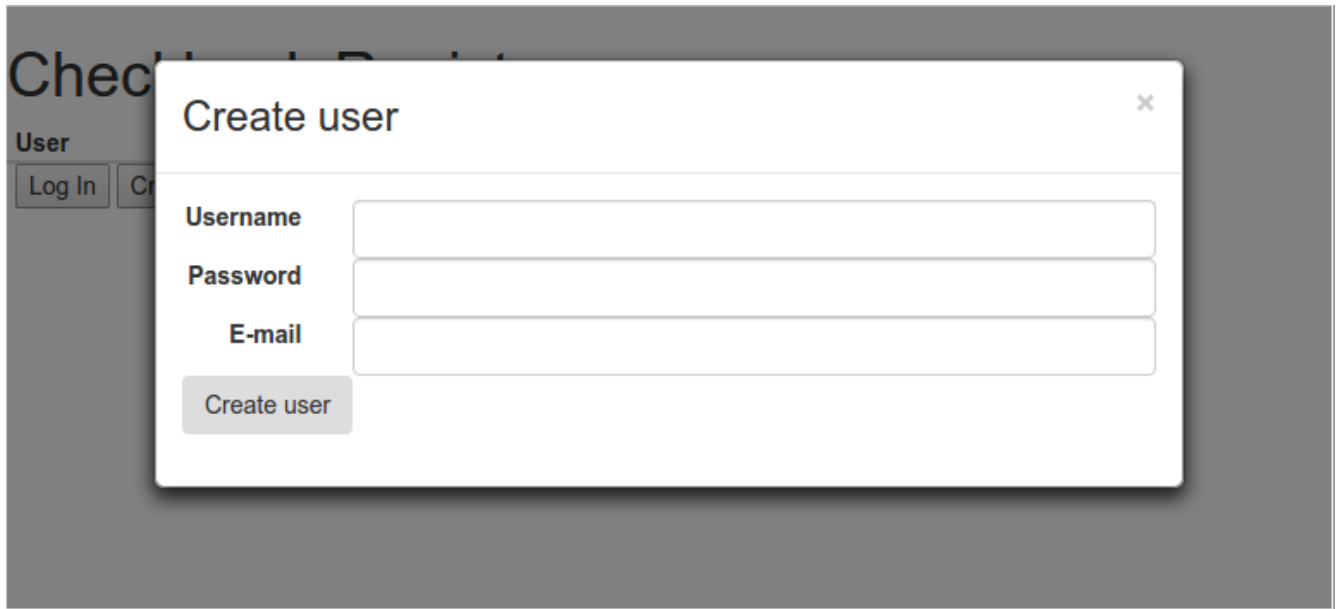
2.4 Screenshots

See chapter 5, page 11 for information on what these screenshots mean.

Initial page load:



Create-new-user button has been clicked:



A screenshot of a web application showing a modal dialog titled "Create user" with a close button (X) in the top right corner. The modal is overlaid on a background that includes a "User" section with "Log In" and "Create user" buttons. The "Create user" modal contains three input fields: "Username", "Password", and "E-mail". Below these fields is a "Create user" button.

Create user [X]

Username

Password

E-mail

Create user

Log-in button has been clicked:



A screenshot of a web application showing a modal dialog titled "Log in" with a close button (X) in the top right corner. The modal is overlaid on a background that includes a "User" section with "Log In" and "Create user" buttons. The "Log in" modal contains two input fields: "Username" and "Password". Below these fields is a "Login" button.

Log in [X]

Username

Password

Login

Logged in, no accounts have been created yet:

Checkbook Register

User

Account

z

Log Out

Create

Select account. Accounts x, y, and z have already been created:

Checkbook Register

User

x

Log Out

Line	Check						
1						00	
2	1234					03	
3	1234					03	
4	1235	June 4	Gni		-\$34.00	\$9,883.03	

Select account

x

z

y

Create Account

Checkbook Register

User

x

Log Out

Line	Check						
1						00	
2	1234					03	
3	1234					03	
4	1235	June 4	Gni		-\$34.00	\$9,883.03	

Create account

Account name

Create account

Account selected, already has entries:

Checkbook Register

User

Account

Clear Code

Balance

Cleared

Display

x

Log Out

x

X ▾

\$9,883.03

\$10,000.00

All

Add Entry

Line	Check No	Date	Name	Description	Cleared	Debit	Deposit	Balance
1		beginning of time	initial balance		x		\$10,000.00	\$10,000.00
2	1234	June 2	ABC Corp	3 widgets		-\$29.97		\$9,970.03
3	1234	June 4	Def			-\$53.00		\$9,917.03
4	1235	June 4	Ghi			-\$34.00		\$9,883.03

Clear-code dropdown:

Checkbook Register

User

Account

Clear Code

Balance

Cleared

Display

x

Log Out

x

X ▾

\$9,883.03

\$10,000.00

All

Add Entry

Line	Check No	Date	Name	Description	Cleared	Debit	Deposit	Balance
1		begin	time	initial balance	x		\$10,000.00	\$10,000.00
2	1234	June	ABC Corp	3 widgets		-\$29.97		\$9,970.03
3	1234	June	Def			-\$53.00		\$9,917.03
4	1235	June	Ghi			-\$34.00		\$9,883.03

Display uncleared only. Notice line 1 is not displayed, as it has already been cleared:

Checkbook Register

User Log Out Account Clear Code Balance \$9,883.03 Cleared \$10,000.00 Display

Line	Check No	Date	Name	Description	Cleared	Debit	Deposit	Balance
2	1234	June 2	ABC Corp	3 widgets		-\$29.97		\$9,970.03
3	1234	June 4	Def			-\$53.00		\$9,917.03
4	1235	June 4	Ghi			-\$34.00		\$9,883.03

Create register entry:

Checkbook Register

User Log Out Account Clear Code Balance \$9,883.03 Cleared \$10,000.00 Display

Line	Check No	Date	Name	Description	Cleared	Debit	Deposit	Balance
1								
2	1234							
3	1234							
4	1235							

Create register entry

Check No Date

Name

Description

Debit Deposit Cleared

3 Conclusions

3.1 Results

This class project resulted in a working checkbook register manager that I will be able to use with my parents' checking accounts. I need to increase security (by setting up record ownership in MongoDB and using https) and implement editing before it is usable.

The web page works well from a desktop web browser and also from tablets/phones.

I gained much skill and experience in Bootstrap, AngularJS, Javascript, Mongoose, and Loopback.

3.2 Features and shortcomings

3.2.1 Features

- Data is stored “in the cloud” immediately.
- There is support for multiple users with authentication, even multiple users simultaneously.
- Each user can have multiple accounts.
- It replaces a stand-alone program that won't run on Chromebook and only stores data locally with a web application that runs anywhere and stores data in the cloud.

3.2.2 Shortcomings

- The user interface is not pretty nor fancy.
- The capability of editing and deleting entries, accounts, and users is not yet implemented.
- It doesn't scale well to a large number of entries.
- Encryption (https) is not used for server communication.
- All data on server can be accessed by any authorized user. Each database item needs to have an owner.

3.3 Hindsight

I should have started with Ionic and built a real, hybrid Android application. However, the web page works quite well with small (tablet, phone) screens.

I would like to learn more about putting client code in the Loopback framework, although the little I did read about it indicates that it is not really supported.

Too late during development, I learned about using `$emit` and `$on` to communicate between controllers. Knowing this at the beginning would have simplified and cleaned up my code. I will change the code after the assignment is submitted.

Because of writing the code over time, my function and variable naming scheme is inconsistent, making it more difficult to maintain the code. I will rename and restructure after submission.

4 References

A working server running this code can be found at <http://towanda.dsl.frii.com:9000>. It is on a slow Internet link, so have patience, especially for the first run while your browser is caching the various

frameworks; speed is reasonable after the initial load. Feel free to create a user and accounts and play with it.

Alternately, you can download the code to your own server and follow the brief README to run it yourself. The code can be found at <https://github.com/LFenske/CheckbookRegister.git>.

[Python Checkbook Manager](#) is an outdated, stand-alone application that was the impetus for writing this application. It has many good features that I have incorporated into this project, but it is missing some features that I want.

Also, look at any paper checkbook register for an idea of what this application is for.

Some of the technologies in use are

- [Bootstrap](#)
- [AngularJS](#)
- [NodeJS](#)
- [MongoDB](#)
- [Mongoose](#)
- [Loopback](#)
- [Ionic](#)
- [Cordova](#)

5 User Manual

- Use any browser (computer or phone) to navigate to <http://towanda.dsl.frii.com:9000>. First use may take a while to load. Subsequent uses should load faster because of browser cache.
- See section 2.4, page 5 for screenshots of all the following operations.
- If this is your first time, create a user.
 - Click on the “Create New User” button.
 - Fill out the form.
- Login in with your user name by clicking the “Log In” button.
- Once you are logged in, you can log out again by clicking the “Log Out” button next to your user name.
- If you have no accounts, you must create at least one by clicking the “Create” button under the “Account” heading, then click the “Create Account” button and fill out the form. After creating an account, you can create another account or select an existing account.

- If you already have one or more accounts, and want to work with a different account or create a new one, click the button with your currently selected account and select a different one.
- Once you are logged in and have selected an account, you will see Clear Code, Balance, Cleared, Display, Add Entry, and your check register for this account.
- “Clear Code” is the text that will be used to mark a register entry as cleared. This is typically the month number of the bank statement where an entry appears. The clear code is changed through the drop-down list.
- “Balance” is the total of the Debit and Deposit fields, described later.
- “Cleared” is the total of the Debit and Deposit fields in entries that have been marked as cleared.
- “Display” has a toggle button to switch between displaying all entries and only those entries that have not been marked as cleared, i.e. entries that are outstanding.
- “Add Entry” will bring up a form for creating a new register entry. Your first entry should be the initial balance.
- The check register area has several columns, most of which are self-explanatory if you've ever used a check register.
 - “Line” is automatically generated and not settable.
 - “Check No” is the (optional) check number.
 - Clicking on an entry in the “Cleared” column will toggle that entry between cleared and not cleared. Marking an entry as cleared will display the current “Clear Code” and include its debit or deposit amount in the “Cleared” balance above. A common practice is to write the bank statement month number when marking a check as cleared.
 - Debits are entries that are subtracted from your account balance, like checks, ATM withdrawals, debit card transactions, bank fees.
 - Deposits are entries that are added to your account balance, like deposits, interest, transfers in to your account.
- Results of all operations are immediately saved.
- There is as yet no way to delete or change anything, except items in the “Cleared” column.