Jason Eide, Asif Zaman, Ethan Morris, Jose Hernandez

CSCE 3513 Software Engineering Spring 2015

Abstract

This document contains the report for the first iteration of a Personal Financial Management System

Project Deliverable

Iteration #2 Report

This document gives an overview of the second iteration of the Personal Financial Management System. In this report, we first discuss the delivered features/user stories (specified by the customer). The next section goes over the methodology used in building this software system, specifically principles of Extreme Programming (XP) that were utilized. Finally we outline the cases used for testing each of the user stories.

**User Stories**

In this iteration, the customer requested that the primary focus should be to implement the database. In terms of user stories, this is the “Secure Storage System” from the Requirement Specification document. Implementing the database was a major obstacle, as no members of our team had previous knowledge of databases. Currently we are still working out the details of the database. There have been some issues with the University’s hosting were it is not allowing us remote access to the database. These issues have been documented through correspondence with the CSCE Systems Support department to get these resolved (this documentation can be provided upon request). At the time of writing, we opted to mimic a database by storing the information on a local server (hosted on the laptop of one of our team members). This was achieved using MySQL workbench and MySQL connector. It is important to note that these two components must be installed on the host system for the current iteration to run properly. We will continue to work with CSCE Systems Support to get remote access working (and remove the local limitations).

In addition, we opted to implement a previously unmentioned user story. As we developed our system, we realized it would be beneficial to the user if there was a main overview tab for the program. Previously, the financial management system started in the Transactions tab. In this iteration, we designed an Overview tab that shows the user their income, expenses, and net income for a specified date range. It also displays a pie chart that gives the user an at-glance overview of their expenses for the specified date range.

For this iteration, our group spent a total of about 55 person-hours (Jason: 15 hours, Jose: 14 hours, Ethan: 16 hours, Asif: 10 hours). As none of us had experience with database design, we were unsure of how many hours it would take. Thus, we do not have an initial estimate to compare it to.

**Implementation of Extreme Programming Practices**

For this software system, we are employing Extreme Programming (XP) for development. This includes principles such as incremental planning, small releases, simple design, pair programming, collective ownership, sustainable pace, and on-site customer interaction. The majority of the XP practices in this iteration are the same as the previous iteration, as our team is employing XP for the entire development cycle.

1. **Incremental Planning** – Our requirements were created through the use of user stories, which were reviewed by the customer. The customer determined which stories should become development “tasks”.
2. **Small Releases** - The Expense/Income Manager is the main component and also provides the minimal useful set of functionality to the customer. Future releases will add functionality to the overall system.
3. **Simple Design** – Each individual component only executes the functionality for the user story that it solves.
4. **Pair Programming** – Approximately 12 of the 55 total person-hours on this iteration were through pair programming. Jason and Asif worked as a pair for 7 hours on the Overview component. Jose and Ethan worked as a pair on the database component for approximately 5 hours in addition to collaborating heavily online with one another.
5. **Collective Ownership** – Decisions concerning individual components or the overall project were made together as a group to solidify collective ownership. Finally, all code is available to all members of the group.
6. **Sustainable Pace** – Group members put in hours reflective of their overall workload which helped combat loss of code quality and production. Long programming sessions ended if productivity loss became noticeable.
7. **On-site Customer** – Early during this iteration cycle, the customer met directly with the entire team to discuss which system requirements should be implemented. Additionally, the customer is available for consultation at any time through email as well as multiple days per week in person (for this particular iteration, additional consultation was not necessary).

The biggest challenge for this iteration was implementing the database. Jose and Ethan took on the challenge of the database and ran into unforeseen issues in the execution. Our team was allotted server space on the University of Arkansas system; unfortunately the LAMP stack was not functioning properly on the University’s end for much of the design time. As such, our group opted to implement a local server for this iteration while we work with the University to sort out the issues on their end.

**Testing**

Testing for the second iteration included testing user input on the login page, calculations for the overview page, and database store and retrieve functionality.

Table 1: Test Cases for Login Dialog Box

|  |  |  |
| --- | --- | --- |
| **Functionality** | **Expected** | **Matches Expected?** |
| Cancel | Exit Program | Yes |
| Enter Email | Store Email | Yes |
| Enter Password | Store Password | Yes |

The login dialog box simply allows the user to enter the main system. The program will not continue until the user at least enters in an email address. If the user presses the cancel button, the program will automatically exit. Please note that the database validation is separate from the login testing—it will be covered later in this document. Additionally, the current iteration does not have error checking for valid email addresses and/or passwords. We are still testing the best option for a username (email vs. standard username) and plan on adding error checking for the next iteration (after the decision is made).

Table 2: Test Cases for Overview Tab

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Functionality** | **Start Date** | **End Date** | **Expected Output (Total Income)** | **Expected Output (Total Expenditures)** | **Expected Output (Net)** | **Matches Expected?** |
| Calculate | 2/1/2015 | 3/4/2015 | $367.72 | -$98.10 | $269.62 | Yes |
| Calculate | 4/2/2015 | 5/2/2015 | $2,250.75 | -$1,974.93 | $275.82 | Yes |

The overview functionality was fairly straightforward to test as it simply calculates the total income, expenditures, and net income for a specified date range, using values from the transactions tab (that was tested in the last iteration). Additionally, in the overview tab, a pie chart is created with the breakdown of expenses for the given time period. As you can see in the table above, the values are calculated as expected. Additionally, the pie chart is set accordingly as well, which can be visually confirmed upon running the program.

Figure 1: Database Testing (Initial Login)

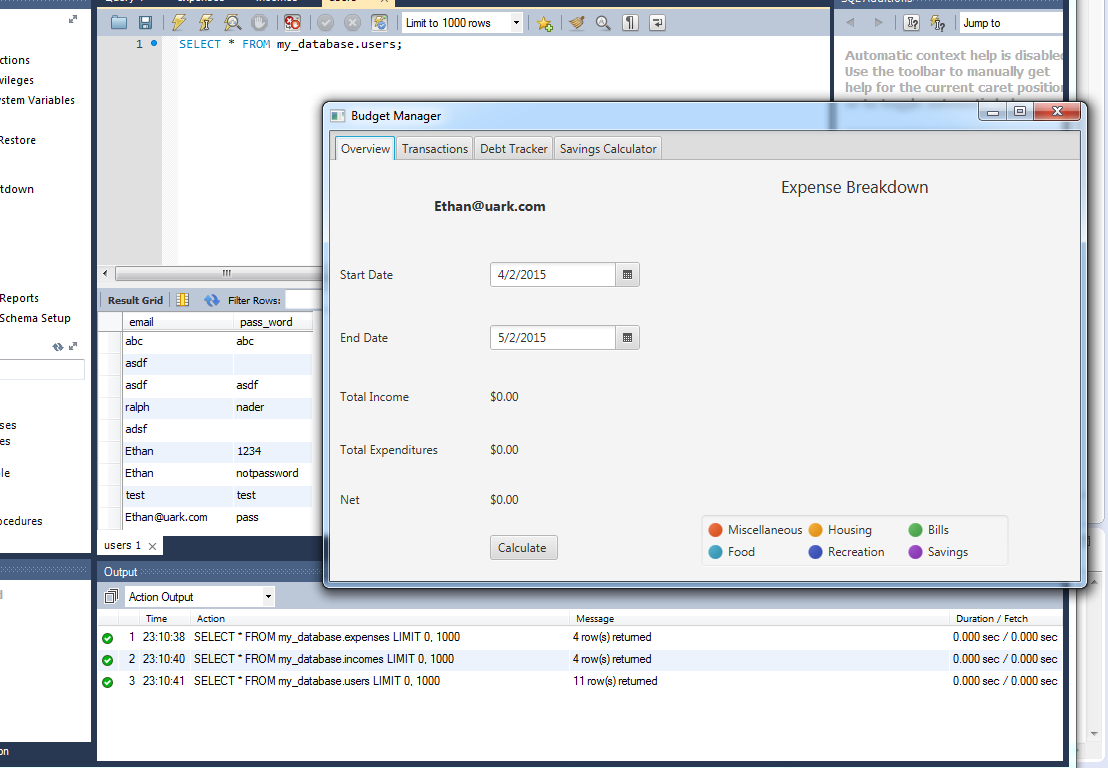


Figure 2: Database Testing (Adding Transaction Data)

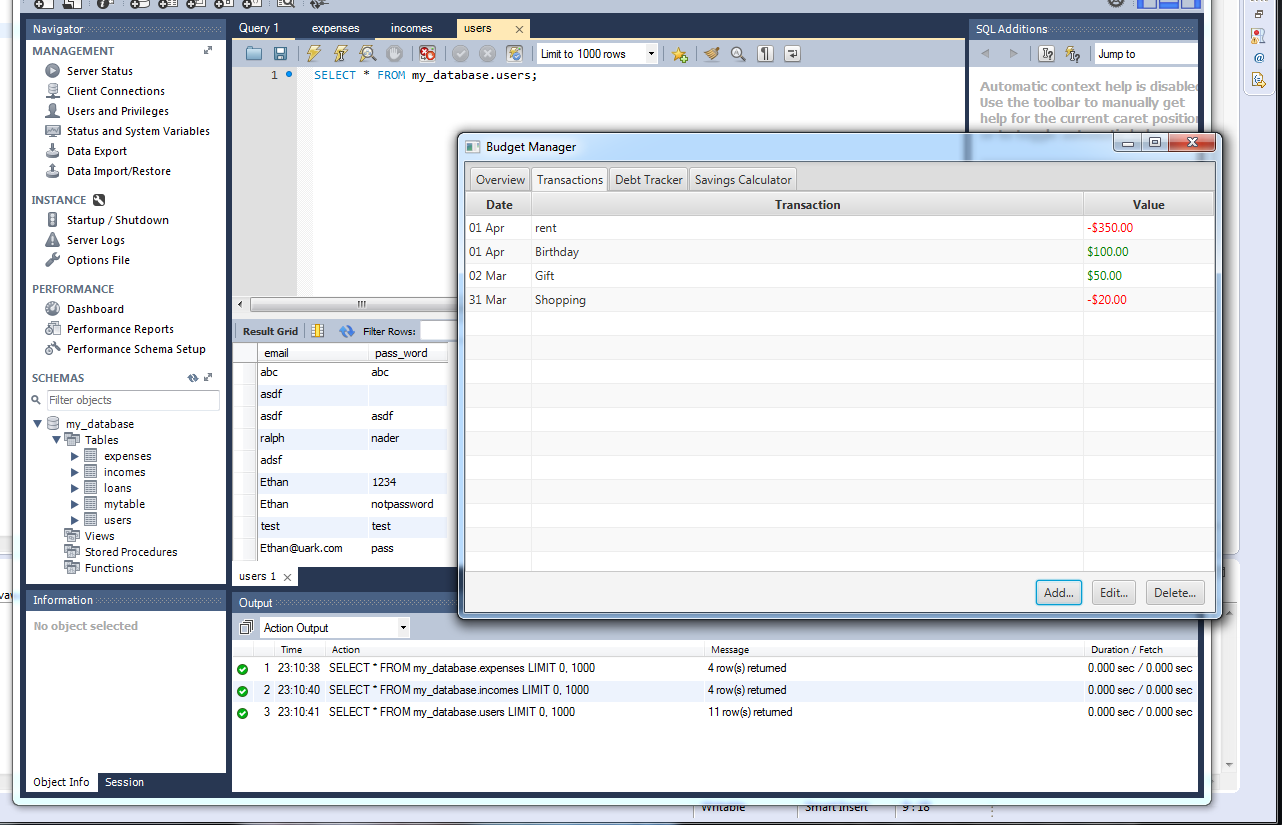


Figure 3: Database Testing (Retrieving and Calculating Overview Data)

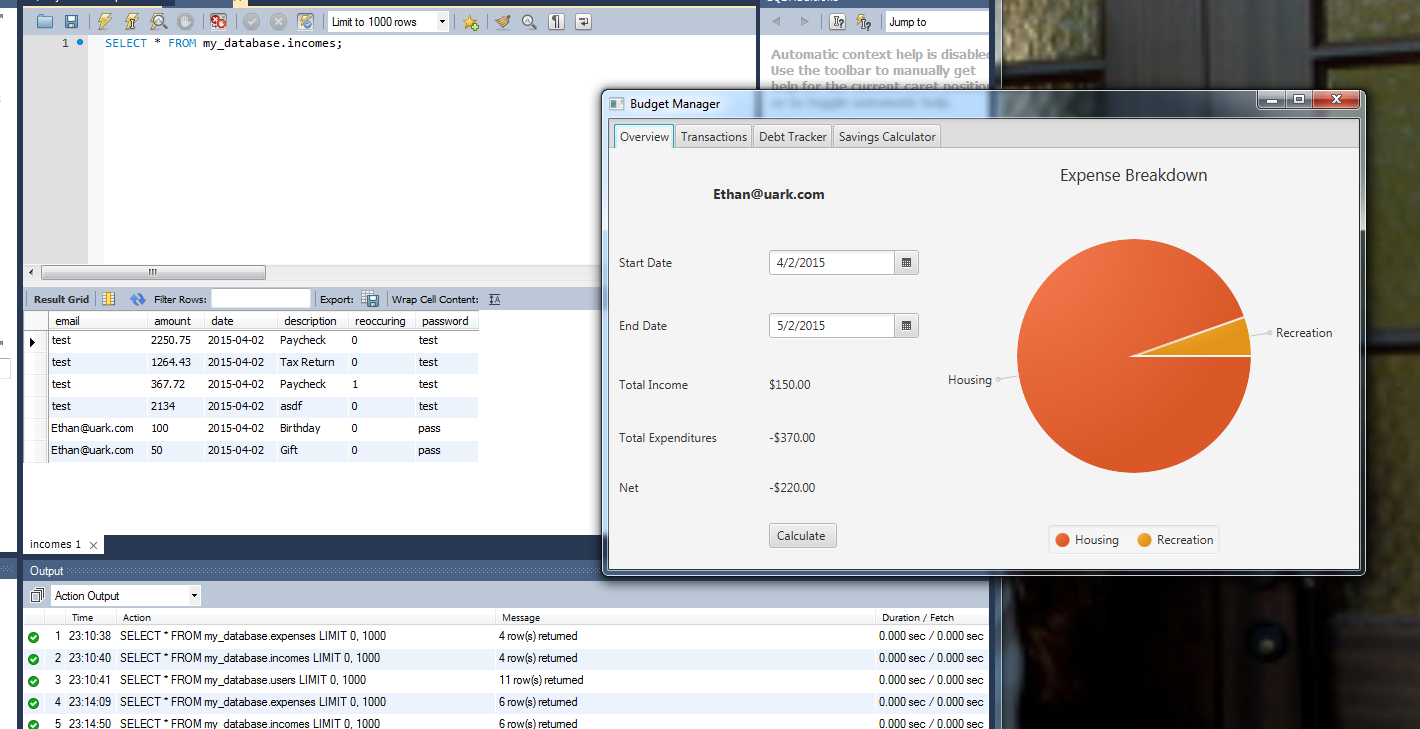
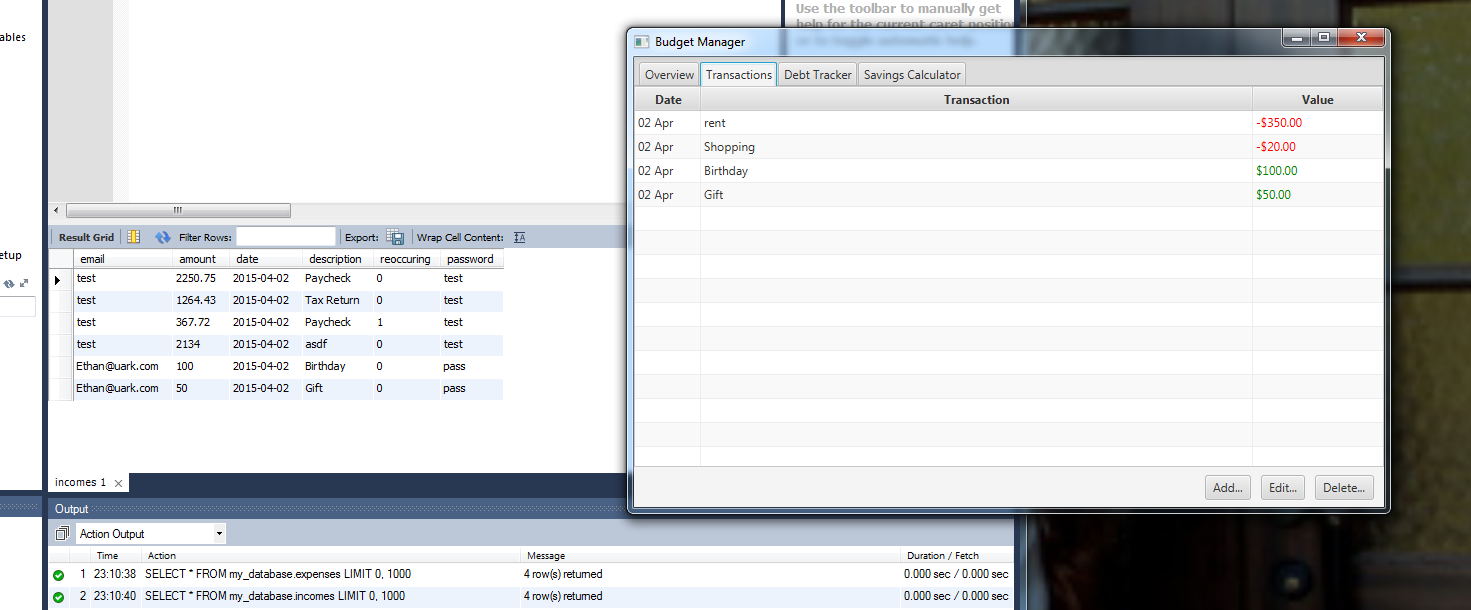


Figure 4: Database Testing (Retrieving Transaction Data)



In testing the database, we simply had to make sure that the program correctly created, retrieved, and stored user accounts along with the corresponding data. To test this, we first entered an email address (which is a proxy for the username) and password. If the username did not match an existing entry in the database, then a new account with the username and password was automatically created. Next was to create sample data in the transactions tab. Finally, upon exiting the program, the sample data should be stored in the database along linked to the particular username. To test if this process was done correctly, we simply had to login again and use the same login information. Upon logging in and recognizing a previously stored username and password, the program retrieved the information from the database. As reflected in the series of screenshots above, the testing was successful as it met the expected results.