Software Project Management Plan

Commerce Bank Project

February 28, 2017

Team Members

Lucy Kull
Caleb Hillhouse
Candice Johnson
David Jones
Cameron L'Ecuyer
Elliot Kieffer
Nick King

Team 3
University of Missouri - Kansa City

Document Control

Change History

Revision	Change Date	Description of changes
V1.0	2/28/2017	● Initial release
V2.0	5/1/17	● Added “A User can View their Budget” to section 3.1 ● Added iterations 2, 3, and 4 ● Added sprints 2, 3, and 4 ● Edited dates in section 4.1

Document Storage

This document is stored in the project’s GitHub repository at: https://github.com/cs451/Spring2017Team3/tree/master/Project%20Documents/spmp.doc.

Document Owner

Lucy Kull and Caleb Hillhouse are responsible for developing and maintaining this document.

Table of Contents

1 Overview

1.1 Purpose and Scope

The budgeting application will access spending habits saved in the application's database. User goal progress and budget data will be displayed to the user. No user information will be displayed to anyone outside of internal Commerce Bank access without the user's permission. The reason for this is the security and safety of the user's information.

1.2 Goals and Objectives

The three main goals of the product are:

1. Provide an easy way to set goals and track progress towards those goals.
2. Provide an easily understandable view of the user's financial information to empower their future financial decisions.
3. Improve the experience of financial planning with gamification elements to make finances fun.

1.3 Project Deliverables

The following items will be delivered to the customer on or before 5/1/2017

1. Source code for web application, code behind, and SQL server portions of the system
2. User's Guide
3. Test Plan
4. System Test Cases

1.4 Assumptions and Constraints

Assumptions:

1. Databases at Commerce Bank will be able to conform to the databases we design, or conform our code to fit their current databases, although the way they do this is unknown.
2. No one session will require pulling data from multiple users, unless said users are using a shared account.

Constraints:

1. Must be a web application built in .NET Framework.
2. Client side framework/libraries must be included (no external resources).
3. DB must be in SQL server 2012.
4. Responsive on mobile devices/smaller screens.
5. Tool should allow for multiple goals to be entered and possible goals created.
6. Daily screen should be easy to read and show goals created as well as how customer is doing so far.
7. Show example of what happens when user reaches goal and what happens when they don't reach their goal.
8. Gamification features when goals are met (like FitBit and Apple Watches).
9. Notifications when appropriate.

1.5 Schedule and Budget Summary

Activity	Start Date	End Date
Technical Prototype	02-20-2017	02-27-2017
Baseline Architecture	02-27-2017	03-15-2017
Working Demo	02-27-2017	03-20-2017
Working System Tests	04-03-2017	04-12-2017
User Guide	04-17-2017	04-26-2017

This project does not have a budget, as it is being developed for a grade and not for profit. All costs will come out of the developer's pocket and thus are not part of a budget.

1.6 Success Criteria

- Project can successfully manage budgets of two different users.
- Users can put in goals and budgets and the project will accurately keep track of these goals and budgets.
- All requirements and deadlines are met by May 1st, 2017.

1.7 Definitions

Application - shorthand for Budgeting Web Application

Controls – the individual elements of a user interface such as buttons and checkboxes.

Customer/Client – the institution to whom the project will be delivered. In this instance,

Commerce bank.

Budgeting Web Application - the project currently being worked on

May – adverb used to indicate an option. For example, "The system may be taken offline for up to one hour every evening for maintenance." Not used to express a requirement, but rather to specifically allow an option.

Product – what is being described here; the software system specified in this document.

Project – activities that will lead to the production of the product described here. Project issues are described in a separate project plan.

Scenario – one path through a use case

Shall – adverb used to indicate importance; indicates the requirement is mandatory. "Must" and "will" are synonyms for "shall".

Should – adverb used to indicate importance; indicates the requirement is desired but not mandatory.

Use case – describes a goal-oriented interaction between the system and an actor. A use case may define several variants called scenarios that result in different paths through the use case and usually different outcomes.

User - a Commerce Bank customer using the system

1.8 Evolution of the Project Plan

At the start of an iteration the project plan will be updated to include a schedule of tasks for the upcoming iteration as well as the estimated time and effort each task requires. At the end of an iteration this document will be updated to show the actual time and effort spent on each task.

Risk mitigation efforts will be continually evaluated. Severe risks and their mitigation plans will be added to the project plan as soon as they are identified.

2 Startup Plan

2.1 Team Organization

Project Manager:	The project manager is responsible for creating the project plan (with input from those doing the work), managing risks, running the weekly team meeting and providing monthly status reports to senior management.
Programmers (2):	Programmers are primarily responsible for coding and unit testing modules. They are also expected to take part in architecture planning and review meetings.
Database Admin (2):	Database Administrators are responsible for creating and defining the necessary database. In addition to database management, they have also been given the task of gamification. They will work out the details of how the app will take advantage of gamification. They are expected to take part in architecture planning and review meetings.
User Interface (2):	The User Interface team is responsible for UI Design and ensuring that the server and database provide the necessary tools and data to deliver requirements. The User Interface team often works closely with the Programmers in development. They are expected to take part in architecture planning and review meetings.

2.2 Project Communications

Team members will communicate with each other through a mobile application named Asana, a desktop application called Discord, Email, or in person.

2.3 Technical Process

The team will use the Scrum software development methodology. This team in particular has decided to dedicate one week per iteration. That is, for each iteration planned by senior management, the team will have two iterations. Having more iterations means more frequent review meetings. Our team has agreed that, in cases where it's possible for people to work ahead, they are allowed to do so. The idea is that this will help prevent schedule conflicts from hindering the development process.

2.4 Tools

- Programming Languages: ASP.NET, Javascript, CSS, HTML
- Build Tools: Visual Studio, Microsoft SQL Server Management Studio,

Notepad++
- Version Control: Github repositories
- Defect Tracking: Whenever a defect is found, it is cataloged and stored in the Repository ReadMe.
- Automated Testing: NUnit Testing

3 Work Plan

3.1 Activities and Tasks

A User can View their Goals
- Create User Interface elements that will allow the user to see a visual representation of their goals.

A User can View and Track their Progress
- Create User Interface elements that will allow the user to see a visual representation of how far the user has come and how far the user has yet to go towards the completion of their goal(s).

A User can Complete a Goal
- Use gamification to reward the user for completing their goals.

A User can not Complete a Goal
- Make a way for a user to delete or change a goal that they cannot complete and apply any gamification that might ensue from these actions.

A User can View their Budgets
- Create a bar graph that compares their budgets to the amount of money they have spent.

3.2 Release Plan

A more detailed release plan will be located in the Release Plan document located in the GitHub repository.

Iteration 0 - 2/13/17

This iteration included creating mockups for User Interface and gamification as well as creating basic functions for switching between to users. Creating the mockups were easy, however creating the basic functions for switching between two users was mildly difficult due to our lack of experience with the ASP.NET programming language.

Iteration 0.5 - 2/15/17

This iteration was a smaller iteration focused mainly improving both the gamification and ER diagrams as well as working on the database itself. These tasks were tedious, but not difficult.

Iteration 1 - 2/20/17

This iteration will focus mainly on allowing a user to create a goal. So far this task has been difficult to complete due to inexperience with the ASP.NET and JavaScript programming languages. The main issue was finding a way to have a JavaScript file communicate with a function that was created in a .NET file.

Iteration 2 - 3/6/17

This iteration was all about allowing a user to view their spending and create a budget based on what they see.

Iteration 3 - 3/20/17

In this iteration, we mainly focused on finishing the User Interface components that would allow a user to create and edit budgets and goals. We also added a summary page that held a graph showing the user's spending over the past month. On the budget page, we added a graph comparing budgeted money versus money spent.

Iteration 4 - 4/10/17

For this iteration we added additional functionality to the page containing a user's transactions. We added the ability for a user to categorize, search for, and sort transactions connected to their account.

3.3 Iteration Plans

Sprint 0

- Design login screen
- Create mockups for User Interface
- Create a basic ER diagram
- Create a general mockup for gamification
- Design basic login function
- Design basic logout function

Sprint 0.5

- Import data into database
- Improve upon gamification diagrams
- Create a basic database abstraction layer

- Improve upon ER diagram

Sprint 1

- A user can create a goal

Sprint 2

- View Spending Report
- Set a Budget

Sprint 3

- Create a landing page
- Make a Budgeted Money vs. Spent Money histogram
- Make a Monthly Spending pie chart
- Create and Edit Budgets
- Create and Edit Goals

Sprint 4

- Create a Search Function for Transactions
- Allow a User to Sort Transactions
- Allow a User to Categorize Transactions

4 Control Plan

4.1 Monitoring and Control

Weekly – Team meeting. Project participants report status, progress and potential problems.

Biweekly – Iteration meeting. Every two weeks, the velocity of previous iteration discussed and used to estimate expected progress in the next iteration. Goals for the next iteration are discussed and finalized.

2/11/2017 – Requirements Elicitation: The team meets with the buyer and discusses the project constraints and expectations.

2/27/2017 – Technical Prototype: Technical prototype demonstrating basic .NET functionality has been understood.

3/13/2017 - Requirements Review: Formal review of the project requirements document identifying our progress and last minute work to be done before mid-term product display

3/20/2017 – Project Status: Client is shown the current state of the project. Feedback is given and used to redirect the project if needed.

3/27/2017 - Feedback and Design Review: The team will respond and adjust to feedback that was provided from the Client in the review and plan future sprints accordingly

4/10/2017 - Project display and integration: The team will examine all moving pieces as a group and discuss possible improvements, bugs, and current technical risks still present as the project comes to an end

5/1/2017 - Final Group Meeting: Review the project and finalize presentation strategies and specifics as well as fix any bugs currently present in code

5/5/2017 – Final Presentation: The application is completed and demonstrated to the Client

4.2 Project Measurements

Phase	Measurement	Source
Release Planning	Record effort estimates and priorities for product features	Mgr
Iteration Planning	Record effort estimates for scheduled tasks Update effort estimates for product features Update estimated dates in release plan	Mgr
Iteration Closeout	Record actual effort for scheduled tasks Record actual effort for product features	Mgr/Pgr
System Test	Record time taken for task completion Record feedback from testers Compare results to previous tests to measure improvement	QA
Project Closeout	Reflect on overall task completion outcomes Compare project success criteria with outcomes Calculate average difference between estimated task completion time and actual time to complete tasks	Mgr

5 Supporting Process Plans

5.1 Risk Management Plan

Risk	Risk Probability	Risk Consequences	Risk Plan
Changing Requirements	.6	-Redundant/unuseable work -Constrained Time	Regularly check in with Commerce Bank to ensure requirements are well understood

			and stable. Detail iteration plans to allow for specific work. Possibly rework completed features to accommodate new requirements. Possibly rework schedule to allow for new time constraints.
Losing Key Personnel	.1	-Work reallocation -Possible feature reduction	Rework teams so key elements are still covered and on task. Simplify design and/or architecture. Reduce feature complexity if needed.
Server Issues	.4	-Setbacks in working prototyping -Unuseable application	Have backup functions/files to build local server. Include regular testing between various elements of application to ensure early identification of server errors.
UI users cannot resolve conflicts between general HTML/CSS and ASP.NET	.2	-Inelegant UI creation -Unuseable design -Simplified architecture	Simplify design. Allow time for UI developers to learn to resolve conflicts. Rework schedule to accommodate time change.
Core Code developers cannot	.1	-Feature reduction -More JS reliance	Allow time for developers to

use ASP.NET		-Lower client satisfaction	learn ASP.NET. Possibly find tutorials and/or classes for developers. Reduce number or complexity of features to allow for schedule change
Project is more complex than anticipated	.9	-Feature reduction -Greater time needed -Architecture needs refactoring	Rework architecture to accommodate new information. Possibly reduce number or complexity of features to allow time to bring required features to the standard expected.
Gamification is not implementable	.6	-More time needed -Possible lower client satisfaction	Simplify gamification design. Possibly move resources from more advanced features to gamification development.
One or more developer loses all technology	.4	-Lower productivity -More time needed	Allow affected developer to work off lab computer and/or partner computer. If data is lost as well, take time to rework or recover data. Possibly remove excess features if time is an issue

Github crashes and deletes all of our data	.001	-Loss of progress -More time needed	Keep local copies of work. If needed, move to work based on local versions
Version of ASP.NET and/or MySQL used by our team is not consistent with Commerce Bank	.7	-Application cannot be integrated into existing system -Lower client satisfaction	If there is time rework the current code to fit client specifications. If not, add comments and directions for maintenance developers
Commerce Bank no longer wants this project	.1	-All work is irrelevant	Accept our fate, move on to next project
Poor time management leads to scramble to complete project	.8	-Heavy workload -Quality loss -Added stress	Larger Sprint Loads, Eliminate excessive features,

5.2 Configuration Management Plan

1. All work products will be stored in a centralized GitHub repository.
2. The naming convention for documents will be: title.suffix where title is a mnemonic that reflects the function of the document and 'suffix' is the standard/normal suffix for the document type. For example, the requirements document created as a Microsoft Word document might be labeled: Requirements.doc.
3. All project (work products) items (documents, source code, test cases, program data, test data, etc) will be stored in the GitHub repository but not all will be under change control (subject to formal change control procedures). Only the system requirements and project plan will be baselined and under configuration control.

4. Items that are subject to change control will be considered baselined after a group review at the end of the life cycle phase during which they are created. Baselined here means that the product has undergone a formal review and can only be changed through the prescribed change control procedures.
5. The change control procedure once a product is baselined is: (1) anyone wanting to make a change to a baselined item will bring it up in meetings to the rest of the group describing the change, reason for the change, expected impact, and timeline for integrating the change. (2) If no one objects to the change, the change will be accepted. If anyone does object to the change, the reason for objecting will be discussed and everyone will be provided the opportunity to voice their opinion. At the end of the meeting a democratic vote will be held to decide whether or not the change should be allowed.
6. Including a change history with all documents is encouraged but only required for baselined documents. The change history should be at the front of the work item and include: (1) the name of the person making the change, (2) brief description of what has changed, (3) reason for the change, and (4) the date the change was integrated.

5.3 Verification and Validation Plan

To verify and validate the quality of the project, team members are required to work within the project contained on GitHub, and are expected to follow specific instructions provided in the ReadMe on how to do this properly. This ensures that the project remains unified during the course of the project and any issues that are presented during the course of development are addressed promptly. Team members are additionally held accountable for work during weekly meetings and sprint reviews.

5.4 Product Acceptance Plan

In order for the product to be acceptable, it must first be readable and portable to allow for environmental transition and differing server infrastructure. Features implemented must be compatible with the .NET framework and be easily implemented on the client side with little to no changes by the product owner.

The Product must meet the goals as outlined in the project requirements document in order to be deemed of acceptable product quality and functionality.

Objectively measurable criteria:

-Web application in .NET framework
-DB must be in SQL Server 2012

-Must be responsive to mobile devices/smaller screens
-Allow for multiple goals to be created of different types
-Daily screen should show progress on goals
-Create multiple customers, with multiple goals
-Gamification features in parallel with goals and budgeting