**Software Project Management Plan**

**Commerce Bank Online Portal**

03/07/2020

**Team Members**

Aanchal Tiwari

Alper Erel

Michael Homer

Nikoloz Dzidzava

Od Purevsuren

Document Control

**Change History**

|  |  |  |
| --- | --- | --- |
| **Revision** | **Change Date** | **Description of changes** |
| V1.0 | 03/08/20 | Initial release |
|  |  |  |

**Document Owner**

Michael Homer**Table of Contents**

[**1**](#_heading=h.3znysh7) **OVERVIEW 4**

[**1.1**](#_heading=h.49x2ik5) **Purpose and Scope 4**

[**1.2**](#_heading=h.2p2csry) **Goals and Objectives 4**

[**1.3**](#_heading=h.3dy6vkm) **Project Deliverables 5**

[**1.4**](#_heading=h.1t3h5sf) **Assumptions and Constraints 5**

[**1.5**](#_heading=h.4d34og8) **Schedule and Budget Summary 6**

[**1.6**](#_heading=h.2s8eyo1) **Success Criteria 6**

[**1.7**](#_heading=h.17dp8vu) **Definitions 6**

[**1.8**](#_heading=h.3rdcrjn) **Evolution of the Project Plan 6**

[**2**](#_heading=h.26in1rg) **STARTUP PLAN 6**

[**2.1**](#_heading=h.lnxbz9) **Team Organization 6**

[**2.2**](#_heading=h.35nkun2) **Project Communications 7**

[**2.3**](#_heading=h.1ksv4uv) **Technical Process 7**

[**2.4**](#_heading=h.44sinio) **Tools 7**

[**3**](#_heading=h.2jxsxqh) **WORK PLAN 7**

[**3.1**](#_heading=h.z337ya) **Activities and Tasks 7**

[**3.2**](#_heading=h.3j2qqm3) **Release Plan 8**

[**3.3**](#_heading=h.1y810tw) **Iteration Plans 8**

[**3.4**](#_heading=h.4i7ojhp) **Budget 8**

[**4**](#_heading=h.2xcytpi) **CONTROL PLAN 8**

[**4.1**](#_heading=h.1ci93xb) **Monitoring and Control 8**

[**4.2**](#_heading=h.3whwml4) **Metrics Collection 8**

[**5**](#_heading=h.2bn6wsx) **SUPPORTING PROCESS PLANS 8**

[**5.1**](#_heading=h.qsh70q) **Risk Management Plan 8**

[**5.2**](#_heading=h.3as4poj) **Configuration Management Plan 8**

[**5.3**](#_heading=h.1pxezwc) **Verification and Validation Plan 9**

# **Overview**

## *Purpose and Scope*

Commerce Bank users will be able to access a web application that has a number of features: login page, built-in notifications regarding transactions, summary of transaction history, etc.

This application should be useful and easy to use for the average customer. Technical support should not be needed. Currently, there is no alternative at this time, we would be building from the ground up.

The web application will be able to access a database, notify the users of certain changes in their account, and be able to save data changes. The goal is to build the application which meets the criteria set by the Commerce Bank in the guidelines given and it is supposed to be as easy to use as possible.

While the motto is to make sure that the application is responsive and aesthetically pleasing to the users, things such as searching/filtering through the notifications, transactions, etc. may be outside the scope of the project.

## *Goals and Objectives*

Goals from req doc: The application should be easy to use and navigate. There should be not be any technical failure and application is expected:

1. Provide web interface for users’ bank accounts
2. Function in a simple and intuitive manner.
3. Provide the users with notifications/triggers when user-specified events occur on the account.
4. Provide a detailed dashboard for viewing all events that have been triggered
5. Provide an API for interaction between data and frontend.
6. Expose endpoints on the API for public use
7. Itemized transaction summary for detailed insight to account history

## *Project Deliverables*

1. Source code for both the web page and API backend of project
2. Unit tests of the code
3. Documentation of architecture
4. Test Plan
5. Admin Guide

## *Assumptions and Constraints*

Assumptions:

1. Assume that the user is vaguely familiar with online portals
2. Assume that users are using a relatively up-to-date browser with support for ES6
3. Assume development platform should be easy for developers to learn and work with

Constraints:

1. Must be a web app and work across common browsers
2. No external script/css references are allowed
3. At least one css framework must be used
4. Project must use a modern technology
5. A database must be used to store information

## Schedule and Budget Summary

Milestone dates:

* + 03/16/20 End Iteration 2: Basic API
  + 04/06/20 End Iteration 3: API functionality complete and Front end start
  + 04/20/20 End Iteration 4: API Security Continued work for Front end
  + 05/04/20 End Iteration 5: Deliver Project with stretch goals

Budget:

* + Salaries: $68,625
  + AWS: $0
  + Domain Name: $5-20

## *Success Criteria*

* All High-Priority and at least 80% of Medium Priority requirements are meet and delivered by May 4
* No more than one major defect per 1000 lines of code written
* Unauthorized users are unable to retrieve data from API and Web page
* Basic security practices are put into place (CSRF tokens, User Cookie sessions, user permissions, password salting)

## *Definitions*

**User** – the person or persons who will interact with the Commerce Bank application.

**Developer** – the person or organization developing the system/application, also sometimes called the supplier in the project.

**Architect** - Creator of program structure and overall data interactions

**Client** – the person or organization for which this Commerce Bank application is being built.

**AWS** - Amazon Web Services. Used for platform, database, and web hosting

**SVC -** Software Version Control. Used to track changes in source code as well as documentation over the course of the project

**Closeout** - Ending of iteration or project and its associated tasks and emitted documentation

## *Evolution of the Project Plan*

At the beginning of each iteration, new updates will be made to the project plan to include the tasks up the upcoming iteration. Also included will be an assessment of any new risks found and their respective route for managing them. At closeout, our plan will be updated with the effort the iteration took to better help estimates for future iterations.

# **Startup Plan**

## *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. Project management will be a shared role with each team member having equal say in the project

Programmers (3): Programmers are primarily responsible for coding and unit testing modules. They are also expected to take part in architecture planning and review meetings.

Architect: The architect is responsible for engineering the overarching structure of the system as well as orchestrating the interactions between data models.

UX Designer: UX designer is responsible for making sure the user experience is consistent and coherent. This includes not only design of the UI but also analyzing what tools and features may be beneficial to the user and how they are presented.

## *Project Communications*

Most information will be gathered through research done by individuals in the team. Requirements will be sourced from the client as well as other “hidden” requirements gathered through consulting with developers.

This information will generally be distributed in the form of a document that everyone on the team has access to. At the beginning of each iteration a new control document will be opened up as a github issue. This issue with contain the necessary requirements and task for the current information as well as other supporting material. The issue will open for comments so that team members may discuss concerns or clarify aspects of the document.

## *Technical Process*

For development of each feature, developers are required to create passing unit tests for all functional requirements based code in there tasks. It is expected that previously written tests must also pass, otherwise it is up to the developer to address any issues are code they may have broken in the process. Changes to code as well as the addition of new features must first be approved by two members of the team prior to being commited. Any questions, comments or concerns must also be addressed prior to commit.

## *Tools*

General:

* Version Control: Source code and written documentation/artifacts will be stored in the teams github repository
* Defect tracking: Any defects will be logged as a new issue via github

Front end:

* Programming Language - Javascript
* Framework: React
* Automated testing: Jest/React Testing Library
* Dependency Management: NPM

Backend:

* Programming Language - Ruby
* Framework: Rails
* Dependency Management: we will be using the bundler gem with versioned lockfiles to manage ruby/rails dependencies
* Automated Testing: Rspec

# **Work Plan**

## *Activities and Tasks*

A work breakdown structure is an excellent tool for identifying a complete list of tasks.

Depending on the needs of the project, some or all of the following attributes will be recorded for each task:

* Task name
* Task Description
* Owner
* Effort estimate
* Actual effort
* Planned start and stop dates
* Actual start and stop dates
* Dependencies among other tasks
* Data model
* Inter-model interactions

## *Release Plan*

Milestones:

* 03/16/20
  + Essential models and backed API created
  + Prototype app completed
* 04/06/20
  + API work completed
  + Basic login and dashboard via UI
* 04/20/20
  + Enhance security between backend API and frontend UI
  + Completed UI for labeling transactions, Creating triggers and viewing trigger history
  + Downloadable summary of dashboard is available
* 05/04/20
  + resolve bugs
  + Stretch work
  + Deliver finished API and Frontend UI

Release Schedule:

A release will be made at the end of each iteration as described in each of the milestones laid out above. At every release, a tag version will be created on the repo for release and historical purposes.

## Iteration Plans

Each iteration will be comprised of user stories represented as Issues on Github. Issues will have three statuses: Todo, In-progress, Done. These stories will be determined, assigned and pointed prior to the beginning of each iteration. When iteration is finished the amount of time and effort it took to complete the story will be recorded for future planning purposes.

See Validation and Verification Plan section for details on acceptance for each story in an iteration.

## *Budget*

It is expected that each iteration will cost the following:

* Salaries: $6,862.5 (5 developers @ $30.50/hr. Average 45 hours/week)

AWS fees: $0. AWS offers a free tier that we are able to take advantage of for development purposes

Estimated $13,725 per iteration

# **Control Plan**

## *Monitoring and Control*

**Weekly** - Mondays and SaturdaysTeam meeting. Members report statuses,

issues, and continued planning for next iteration.

**End of iteration** - Regression meeting. Go over what when well and what didn’t go so well. Create a plan to prevent same issues from happening again in future iterations

**Beginning of Iterations** - Go over planned work for the iteration and discuss architecture with members. Use this time for questions and concerns over the direction for the iteration

## *Project Measurements*

|  |  |  |
| --- | --- | --- |
| **Phase** | **Measurement** | **Source** |
| Release Planning | Record effort estimates for product features  Determine features for each iteration/release cycle  Consult with architects for estimates | Mgr |
| Iteration Planning | Record effort estimates for scheduled tasks  Update effort estimates for product features  Update estimated dates in release plan | Mgr |
| Iteration Tasks | Design and refine unit tests for all production code  Validate that all previous tests are still passing  Validate all functional requirements were implemented | Mgr/Pgr/Code Reviewer |
| Iteration Closeout | Record actual effort for scheduled tasks  Record actual effort for product features  Record LOC count for modules written | Mgr/Pgr |
| System Test | Record the rate at which errors are found. | QA |
| Project Closeout | Archive project documents  Verify all Features were met | Mgr |
| Ongoing | Record defects found from integration testing  Assign each defect to one of the following categories: blocker, critical, major, minor or trivial. Keep track of the state of each defect: open, assigned, fixed, closed. | Mgr/Pgr/QA |

# **Supporting Process Plans**

## *Risk Management Plan*

Risks:

* User Security
  + Priority: Medium
  + actions:
    - Cookie sessions
    - Hashed passwords
    - https (maybe)
    - CSRF tokens on form submissions
    - No PII in url path
* Transaction Race Condition
  + Priority: High
  + Actions:
    - Establish locks on an account when a transaction is processing
* SQL Injection
  + Priority: Low
  + Actions: Transfer
    - Use Rails built-in DSL for db access within models to prevent SQL Injection

## *Configuration Management Plan*

Configuration management plans for this document and other baselined work products including review procedures and change management procedures.

1. All deliverables and artifacts will be stored in a centralized Github repository running remotely
2. All documentation/deliverables for each iteration will be stored under their respective iteration directory within the repo. Example: iteration 3 documentation will be found under documentation/deliverables/iteration 3
3. Anytime a baselined document is changed it is required that the change history section of the documentation be updated and recorded who changed it. Git will be responsible for representing *what* in the document was changed
4. Any changes to documentation must be followed with a reason as to why the change was made
5. All changes to documentation must be approved by all team members in the same manner as code reviews.
6. If revisions, additions or deletions are requested by a member on a review, these requests do not necessarily have to take place, but they must be addressed and the commenter must agree to the outcome of the request prior to its merger.

## *Verification and Validation Plan*

In order to verify quality in our software, the following steps will be taken

* All user stories *must* have an associated test suite created that directly tests the code that was written
* All new code commits *must* be reviewed by all members of the team and must have at least two approvals to be merged into the master branch
* Developer must submit proof that the code written addresses all AC in the story
* All previous tests *must* pass. If tests do not pass the source of the failure must be identified. Unit tests must never be changed unless if requirements for the project have changed

## Product Acceptance Plan

* Entirety of project should be completed before 05/04/20
* 30% code coverage with unit tests (all passing)
* All Highest-priority requirements are implemented
* When each iteration is complete, each user story must abide by the rules established in the above “Verification and Validation Plan” section.