Project Delivery Methodology (PDM)

TECHNICAL ARCHITECTURE

Commerce Bank

CommerceApp

VERSION: 1.0 REVISION DATE: 11/4/2020

Approval of the Technical Architecture indicates an understanding of the purpose and content described in this deliverable. By signing this deliverable, each individual agrees with the content contained in this deliverable.

Approver Name	**Title**	**Signature**	**Date**
Evan Wike	Project Manager	Evan Wike	11/8/20

Table of Contents

Section 1 DOCUMENT SCOPE **4**

Section 2 OVERALL TECHNICAL ARCHITECTURE **5**
2.1 System Architecture Context Diagram 5
2.2 System Architecture Model 6
2.2.1 Overall Architectural Considerations 6
2.3 System Architecture Component Definitions 7
2.3.1 System Architecture Components 7

Section 3 SYSTEM ARCHITECTURE DESIGN **8**
3.1 System Architecture 8
3.1.1 Overall Component Functions 8
3.1.2 List of Components 8
3.2 User Interface 8
3.2.2 Technical Considerations 9
3.2.3 Selected Product(s) 9
3.2.4 Selection Rationale 9
3.2.5 Architecture Risks 9
3.3 Database Management System 9
3.3.2 Technical Considerations 9
3.3.3 Selected Product(s) 9
3.3.4 Selection Rationale 9
3.3.5 Architecture Risks 10
3.4 Account Registration, Authentication, and Recovery 10
3.4.2 Technical Considerations 10
3.4.3 Selected Product(s) 10
3.4.4 Selection Rationale 10
3.4.5 Architecture Risks 10
3.5 CSV File Management 10
3.5.2 Technical Considerations 11
3.5.3 Selected Product(s) 11
3.5.4 Selection Rationale 11
3.5.5 Architecture Risks 11
3.6 Notification System(s) 11
3.6.2 Technical Considerations 11
3.6.3 Selected Product(s) 11
3.6.4 Selection Rationale 11
3.6.5 Architecture Risks 11

Section 4 System Construction Environment **12**
4.1 Development Environment 12
4.1.1 Developer Workstation Configuration 12
4.1.2 Supporting Development Infrastructure Configuration 13
4.2 QA Environment 14
4.2.1 QA Workstation Configuration 14
4.2.2 Supporting QA Infrastructure Configuration 14
4.3 Acceptance Environment 14
4.3.2 Supporting Acceptance Infrastructure Configuration 14

Section 1 DOCUMENT SCOPE

***Document Scope** describes the context and the goals of this document in a narrative.*

This document describes the Technical Architecture of the Commerce Bank Web Application System that satisfies business requirements as documented in the Requirements Document, 10/04/2020, and implements the functionality and satisfies the technical, operational, and transitional requirements described in the Project Plan, 10/11/2020.

The goal of the Technical Architecture document is to define the technologies, products, and techniques necessary to develop and support the system, as well as to ensure that the system components are compatible and comply with enterprise-wide standards and directions as defined by Commerce Bank.

This document will also:

Identify and explain the risks inherent in this Technical Architecture;

Define baseline sizing, archiving and performance requirements;

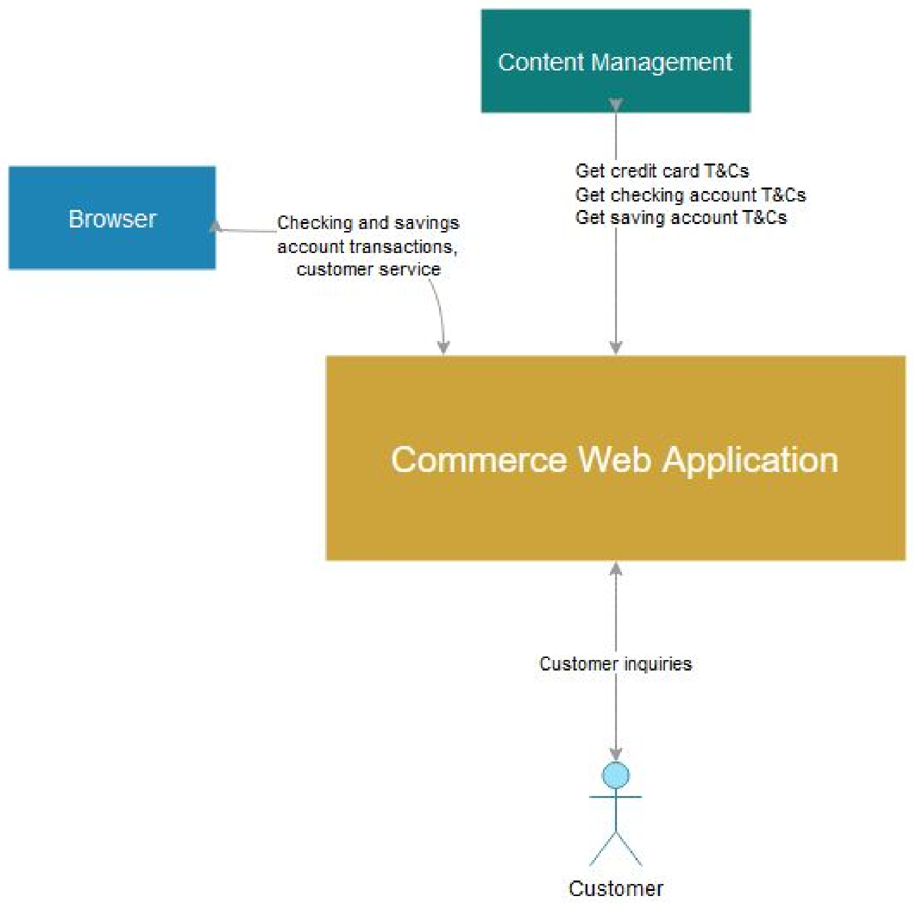
Identify the hardware and software specifications for the Development, Testing, QA and Production environments;

Define procedures for both data and code migration among the environments.

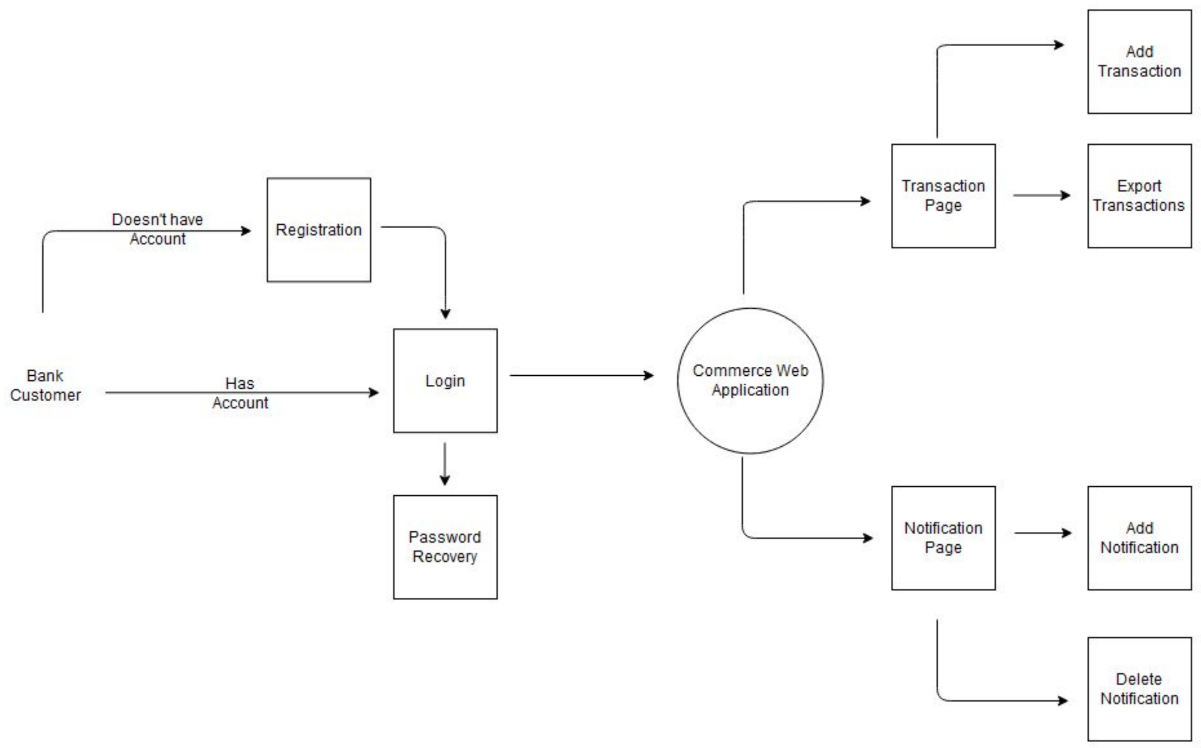
Section 2 OVERALL TECHNICAL ARCHITECTURE

2.1 System Architecture Context Diagram

Since we do not have a full understanding as to how Commerce Bank's system works, the System Architecture Context Diagram provided in this document will loosely show how the customer interacts with the web application and what they will be able to do with the web application. We can assume that customer information such as bank account number, address, etcetera are all stored in Commerce Bank's system.



2.2 System Architecture Model



2.2.1 Overall Architectural Considerations

*The **Overall Architectural Considerations** section defines how additional technical requirements have been addressed by the architecture. Representative items in this section may include:*

In the Overall Architectural Considerations section, we will address the following categories and how we have addressed them with the architecture that we are using.

- Security Strategy
    - To maintain customer security, the web application will not display banking account number or routing number. The primary focus of the web application is to allow users to track their transactions as well as create and manage notifications that will alert them to their specified terms when a transaction that meets the requirements is processed. Authentication is also being handled by Firebase, which will help ensure integrity of user accounts.
- Performance requirements
    - Since we are unsure what hardware customers will use to access the web application, we are doing our best to ensure that performance is the same for users across the board regardless of whether they are using a smartphone, a laptop, or a desktop computer to access it. This means that we are going for a

simplistic and minimalistic design that will not rely on outside elements such as images that could cause the web application to load slowly for users who have slower internet speeds.

- Data export
    - Transactions will be able to be exported as .CSV so that users can import them into their preferred spreadsheet program.

2.3 System Architecture Component Definitions

2.3.1 System Architecture Components

The **Architecture Component Definitions** section provides narrative describing and explaining each architecture component in the System Architecture Model, and identifies specific elements that comprise that component in this system. The following are examples of architecture components and elements:

Architecture Component	**Component Elements**
Commerce Web Application	DBMS
Registration	Creation of accounts, creation of accounts through integrated services (Facebook/Google), verification Emails
Login / Password Recovery	Password Reset
Transaction Page	Exporting Transactions as .CSV file, manually adding transactions
Notification Page	Adding and deleting of notifications

Section 3 SYSTEM ARCHITECTURE DESIGN

*The **System Architecture Design** section provides detailed descriptions of each product implementing architecture components, and explains the rationale for product selection.*

3.1 System Architecture

*For each **System Architecture Component** (identified in Section 2.3 above), the narrative describes specific **Component Functions**, requirements and other **Technical Considerations** that were used in the decision-making process, as well as any specific **Products** selected to implement this component. The **Selection Rationale** identifies any other products that may have been considered, and provides rationale for the decision. **Architecture Risks** identifies any potential risks associated with the architecture element.*

3.1.1 Overall Component Functions

*Aggregating the **Component Elements** (identified in Section 2.3 above), the necessary **Component Functions** may be determined and grouped as follows.*

- Performant, cross-platform UI
- DBMS to securely manage accounts and transactions
- Account registration, login, and password recovery via integrated services, Email
- Exporting data from transaction logs to CSV file
- Customizable mail, text, and/or push notifications

3.1.2 List of Components

*Each of the **System Architecture Components** will be discussed in the following subsections.*

- User Interface
- Database Management System
- Account Registration, Authentication, and Recovery
- CSV File Management
- Notification System(s)

3.2 User Interface

3.2.1 Component Functions

- Client-side access point to web-based commerce application
- Registration/Login/Recovery Page
- Dashboard, Transaction, and Notification Pages

3.2.2 Technical Considerations

- User-friendly (as defined in Requirements Document)
- Cross-platform (Desktop, mobile, etc)
- Modular and decoupled from backend
- Scalable to many concurrent users
- Ensures security and user privacy

3.2.3 Selected Product(s)

Angular - frontend web framework

3.2.4 Selection Rationale

We considered Angular, React, and Vue, any of which could have fulfilled our needs. The reason we chose Angular was due to the experience of our team with Javascript, CSS, and HTML.

3.2.5 Architecture Risks

The main risk is the threat to the privacy of the user and their information. By keeping the UI decoupled from the backend and using proper authentication, we ensure that users may not access each other's information, or alter records they should not.

3.3 Database Management System

3.3.1 Component Functions

- Maintain records of balances, transactions, accounts, etc

3.3.2 Technical Considerations

- Secure and reliable
- Scalable to many concurrent users and large databases
- Redundancy and backups

3.3.3 Selected Product(s)

Prototyping: Google Firebase - web services and app development platform
Release: MySQL Server - DBMS

3.3.4 Selection Rationale

For the purpose of vertical prototypes we have elected to use Google Firebase due to the ease of learning and speed of deployment. For the final product we intend to design and deploy a MySQL database so that it may eventually be deployed on secure private servers. MySQL was an easy choice due to its current prevalence in industry.

3.3.5 Architecture Risks

The DBMS poses perhaps the largest architecture risk to our application, since it will maintain all records for balances, transactions, and accounts. It is of utmost importance that all data remain inaccessible to those who are not expressly authorized by Commerce Bank.

3.4 Account Registration, Authentication, and Recovery

3.4.1 Component Functions

- Creation of accounts through integrated services (Facebook/Google), verification Emails
- Password recovery
- Access website or reach cooldown after X number of failed logins

3.4.2 Technical Considerations

- User-friendly (as defined in Requirements Document)
- Maintains user privacy and account security

3.4.3 Selected Product(s)

Google Firebase - web services and app development platform

3.4.4 Selection Rationale

Google Firebase was an easy choice due to the ease of learning, ease of deployment, and the reliable reputation of Google's web services. Amazon Web Services were also considered, but we felt that this would impose a steeper learning curve for our team.

3.4.5 Architecture Risks

The obvious architecture risk involving this component is the access of accounts by unauthorized users via fraudulent / stolen credentials, or bypassing login altogether. Thankfully, Google's authentication services handle most of the difficult security vulnerabilities for us.

3.5 CSV File Management

3.5.1 Component Functions

- Exporting a selected date range from the transaction log to a CSV file

3.5.2 Technical Considerations

- Proper file format for Excel
- Modules already exist for this task

3.5.3 Selected Product(s)

No product has been selected as of this writing.

3.5.4 Selection Rationale

We will prefer CSV modules which are lightweight, reliable, easy to learn and read, and are set to receive long-term maintenance.

3.5.5 Architecture Risks

The main architecture risk here is that the module may fail, disabling the save feature.

3.6 Notification System(s)

3.6.1 Component Functions

- Email, Text and/or Push notifications

3.6.2 Technical Considerations

- Maintainability and scalability
- Modules already exist for this task

3.6.3 Selected Product(s)

No product has been selected as of this writing. Viable options must be explored.

3.6.4 Selection Rationale

We will prefer notification systems that are easily integrated into our application and are set to receive long-term maintenance.

3.6.5 Architecture Risks

The main architecture risk is that the module will fail, disabling all notifications. This could prevent users of the application from receiving important warnings about their balances and transactions.

Section 4 System Construction Environment

4.1 Development Environment

4.1.1 Developer Workstation Configuration

- A network connection is only required to work with Firebase -- though, dummy values may be used.
- The only directory required for development is the project root, commerce-app.
- Use of the WebStorm IDE is encouraged but not required.
- Components are built from commerce-app/src/app/{component-name}.
- An up-to-date version of Node is required.
- Software, such as BrowserStack, will be required for Cross-Browser and Cross-Device testing.

The following dependencies are required for development:

- @angular-devkit/build-angular: ~0.1001.6,
- @angular/cli: ~10.1.6,
- @angular/compiler-cli: ~10.1.5,
- @types/node: ^12.11.1,
- @types/jasmine: ~3.5.0,
- @types/jasminewd2: ~2.0.3,
- codelyzer: ^6.0.0,
- jasmine-core: ~3.6.0,
- jasmine-spec-reporter: ~5.0.0,
- karma: ~5.0.0,
- karma-chrome-launcher: ~3.1.0,
- karma-coverage-istanbul-reporter: ~3.0.2,
- karma-jasmine: ~4.0.0,
- karma-jasmine-html-reporter: ^1.5.0,
- protractor: ~7.0.0,
- ts-node: ~8.3.0,
- tslint: ~6.1.0,
- typescript: ~4.0.2,
- @angular-devkit/architect: >= 0.900 < 0.1100,
- firebase-tools: ^8.0.0,
- fuzzy: ^0.1.3,
- inquirer: ^6.2.2,
- inquirer-autocomplete-prompt: ^1.0.1,
- open: ^7.0.3

The following dependencies are required for both development and at runtime.

- @angular/animations: ~10.1.5,
- @angular/cdk: ^10.2.6,
- @angular/common: ~10.1.5,
- @angular/compiler: ~10.1.5,
- @angular/core: ~10.1.5,
- @angular/fire: ^6.0.4,
- @angular/forms: ~10.1.5,
- @angular/material: ^10.2.6,
- @angular/platform-browser: ~10.1.5,
- @angular/platform-browser-dynamic: ~10.1.5,
- @angular/router: ~10.1.5,
- angular-fire-schematics: ^1.0.0,
- firebase: ^7.0.0 || ^8.0.0,
- rxjs: ~6.6.0,
- tslib: ^2.0.0,
- zone.js: ~0.10.2

Developers should use the following editor configuration:
[*]

- charset = utf-8
- indent_style = space
- indent_size = 2
- insert_final_newline = true
- trim_trailing_whitespace = true

[*.ts]

- quote_type = single

[*.md]

- max_line_length = off
- trim_trailing_whitespace = false

4.1.2 Supporting Development Infrastructure Configuration

- Development server: http://localhost:4200/
- Unit Testing: ‘ng test’
- E2E Testing: ‘ng e2e’

4.2 QA Environment

4.2.1 QA Workstation Configuration

Hardware for QA would be more abundant compared to the Developer and Acceptance Workstations. The role of the QA is to ensure that the application can run smoothly on a wide range of devices (computers, phones, tablets) that run on different versions of different operating systems (Windows, MacOS, Linux, Android, etc.). Networking would be required to connect to Firebase to test different values that could cause problems inside the application. Webstorm IDE is only necessary if QA plans to write test code.

4.2.2 Supporting QA Infrastructure Configuration

- Test or Development server: http://localhost:4200/
- Android Emulator
- Different Web browsers (Firefox, Chrome, Internet Explorer, Edge, Safari)
- Unit testing: 'ng test'

4.3 Acceptance Environment

4.3.1 Acceptance Workstation Configuration

The application will be hosted on GitHub Pages for acceptance testing. No additional configuration is necessary since it is a Node project. The 'angular-cli-ghpages' NPM package will allow for direct deployment to GitHub Pages. Another option is Heroku.

Software, such as BrowserStack, will be required to demonstrate the responsiveness of the application across various browsers and devices.

4.3.2 Supporting Acceptance Infrastructure Configuration

No additional acceptance infrastructure is required.