Third Party Aggregator Development for FDX API Integration at Citizens Bank

Project Team

Chelsea Chang, Management Engineering ‘23, cjchang@wpi.edu

Kevin Inger, Management Engineering ‘23, kninger@wpi.edu

Jonathan Metcalf, Computer Science ‘23, jemetcalf@wpi.edu

Jack Sullivan, Computer Science ‘23, jhsullivan@wpi.edu

Project Advisors

Professor Wilson Wong

Department of Computer Science

Professor Robert Sarnie

Department of Business

Abstract

Citizens Bank runs a Financial Data Exchange (FDX) API. Within this, OAuth enables third party applications to make API calls on consenting users' behalf. The research team developed a mock Personal Finance Management (PFM) application from the perspective of a third party which accesses the FDX development environment through OAuth authentication. The resulting PFM app will determine if Citizens Bank is ready to integrate with third party aggregators.

Executive Summary

The WPI team, a group of 4 students, was sponsored by Citizens Bank over the course of a seven week project with a high-level objective to determine the readiness of Citizens Bank to integrate its Application Programming Interfaces (APIs) with third party aggregators by developing a full-stack mock Personal Finance Management (PFM) application while implementing the Agile methodology for development. The WPI team's project was specifically designed for internal use at Citizens Bank. The value of this work was to allow Citizens Bank to test their APIs from the perspective of a third party aggregator. In banking, the aggregator is the organization that provides cumulative utility service, such as Plaid, which is a financial services company. In coding from the point-of-view of a third party, the WPI team intended to make real calls to the Citizens Bank APIs to test the efficacy of the current state, as well as identify the places where improvement could be made in order to achieve seamless integration.

With this goal in mind, the team began conducting research prior to the project timeline to aid the team's background understanding. Namely, the team outlined the difference between Open Banking and Banking as a Service. Albeit similar, open banking and banking as a service are not the same. Both open banking and banking as a service deal with banks connecting to non-banks through the means of API calls. However, the purposes are different. Within banking as a service, non-bank businesses integrate complete banking services into their own products. However, with the open banking model, non-bank businesses only use the bank's data for their own individual products. These are the third party providers.

Over the course of the 7 week project, the WPI team utilized numerous technical platforms. Most notably, the team used BitBucket for source code management, Jenkins for an automation server to enable deployment, OpenShift for deployment, Docker as a tool to deliver packages as containers, and Postman for sending test HTTP requests. Through the course of the project, the team learned what it is like to work in a heavily regulated financial institution. Despite the 7 week deadline and the project's congruent

goal to serve as an academic exercise, the team was treated as full-time employees of the bank, operating on the same schedule, attending company meetings, and integrating into the Citizens Bank work culture.

Acknowledgements

For our Major Qualifying Project, the research team had the privilege to be working with Citizens Bank as its project sponsor. We are thankful for the opportunity to work with all of the people at Citizens Bank and hope our contributions will have a lasting impact on the institution. We would like to extend our sincere thanks and appreciation to the following people, whose help and support allowed us to accomplish our research. First, we would like to thank our advisors, Professor Robert Sarnie and Professor Wilson Wong, who were by our side for the duration of our research and work with Citizens Bank. Their patience and support were unwavering and admirable. Additionally, thank you to all of our contacts at Citizens Bank, especially Jeffrey Collemer and Glenn Morin, who offered timely and consistent support and guidance. We are grateful that they offered us the opportunity to make a lasting impact on a prestigious organization. We are excited to see the continued and expanding success that Citizens Bank has in the community!

Table of Contents

List of Figures

List of Tables

Authorship

CC: Chelsea Chang, **KI:** Kevin Inger, **JM:** Jonathan Metcalf, **JS:** Jack Sullivan

Section	Main Author(s)	Main Editor(s)
Abstract	KI	JM, CC
Acknowledgements	KI	JM
Executive Summary	KI	JM
1.0 Introduction	KI	JM, CC
2.1 Company Background	KI	JM, CC
2.2 Difference Between Open Banking and Banking as a Service	KI	JM
2.3 Existing Use of External APIs	JM, CC, KI	KI
2.4 Technical Background	JM	KI, CC
2.5.1 OAuth Overview	JM, CC	KI, CC
2.5.2 Two-legged vs. Three-legged OAuth	JM, CC	KI, CC
2.5.3 OpenID Connect	JM	CC
2.5.4 Prior Authentication Methods	JM	KI
3.1 Agile Methodology	JM, CC	KI, CC
4. Citizens Bank Development Environment	KI, JM, JS	JM
4.1 Project Management Software	KI	JM, CC
4.2.1 Source Code Management	JS	JM, CC
4.2.2 Integrated Development Environment	JM	KI
4.2.3 Build & Deployment	JS	JM, CC
4.2.4 Testing	JM	KI, CC
5.1 Software Requirement Gathering	JM	KI, CC
5.2 Functional v. Nonfunctional Requirements	JM	KI

5.3 User Stories and Epics	CC, KI	JM
5.4.1 FDX Specification	JS	JM
6.1 Software Architecture Overview	JS	JM, CC
6.2 Software Security Framework	JM	KI
7. Software Development	JM	KI, CC
7.1 Sprint 1	KI, JM	JM, KI, CC
7.1.1 Sprint 1 Retrospective	KI, JM	JM, KI
7.2 Sprint 2	KI, JM	JM, KI
7.2.1 Sprint 2 Retrospective	KI, JM	JM, KI
7.3 Sprint 3	KI	JM, CC
7.3.1 Sprint 3 Retrospective	KI	JM, CC
7.4 Sprint 4	JM	CC
7.4.1 Sprint 4 Retrospective	CC, KI	JM
8.1 Risk Versus Reward	CC	KI, CC, JM
8.2 Risk Culture	CC	KI, CC, JM
8.3 Additional Risks	CC	KI, CC, JM
9. Assessment	KI	JM
9.1 Business Learnings	KI	JM, CC
9.2.1 Functionality vs Stability Risk Management Tradeoff	JS	JM
9.2.2 Enterprise Git Management and CI/CD	JS	JM, CC
9.2.3 Narrowing Down Problems	JM	CC
9.2.4 Operating in a Corporate Environment	JS, JM	JM
10. Future Work	JM	KI, CC
11. Conclusion	KI, CC	JM, CC

1. Introduction

As a sizable organization in the competitive financial services industry, Citizens Bank constantly seeks to upgrade and adapt to modern technologies. For a seven week project timeline, Citizens Bank brought on a group of 4 WPI students to assist in organizing and executing a proof of concept relating to open banking and API integration.

The goal of this project had two aspects to it, both of which required the team to work from the perspective of a third party aggregator. The first goal was to help Citizens Bank with its goal of publicly entering the open banking field by determining, simply, whether or not Citizens Bank is ready to integrate its API technologies with third party aggregators. The second goal was to build a tool in the form of a full-stack PFM application which would be a piece of software showing proof of concept of these technologies.

While working with Citizens Bank, the team was treated the same as full-time employees, having the complete experience of Citizens Bank leadership, operations, and culture while following the same stand-up, meeting, and ceremony schedules as other teams at the organization.

2. Background Research

2.1 Company Background

Citizens Bank has a long history of success, perseverance, and continuous adjustment to ever-changing times. First established nearly 200 years ago in 1828, Citizens Bank was founded as High Street Bank. After only a few decades, the bank soon reestablished itself as Citizens Savings Bank in 1871. By the year 1981, Citizens Bank had accrued a total of over $1B of assets, a major milestone. Just seven years later, in 1988, Royal Bank of Scotland (RBS) acquired Citizens Bank. By 2008, RBS was nationalized as a result of the 2008 Financial Crisis. That same year, Citizens Savings Bank's assets grew to $170B. By 2014, the company was ready for its Initial Public Offering on the New York Stock Exchange, now represented as Citizens Financial Group Inc. (hereby referred to as Citizens Bank in this paper). As of June 2022, Citizens Bank has grown to $226B+ in assets. It now stands as a recognizable organization, especially in the eastern side of the United States.

Citizens Bank now boasts over 1,200 regional branches throughout 14 states. It has been led by CEO Bruce Van Saun since 2013 and currently employs over 18,000 employees. It also supports a network of 3,300 ATMs across the U.S. and is currently headquartered in Rhode Island (*About Our Company*). Citizens is a publicly owned company, trading at $CFG. In the banking industry, it offers retail and commercial services to individuals and businesses, both large and small. The bank's main sources of business range from deposit products, mortgage and home equity lending, student loans, auto financing, credit cards, business loans, wealth management, and investment services. Of the bank's $226B+ of assets, current $159B (around 70.4%) comes in the form of deposits. Citizens Bank also supports a fairly even split between loans, with 48% being commercial loans and 52% being consumer loans.

A significantly impactful component of Citizens Bank is the culture which the organization aims to foster. Citizens Bank espouses the importance of having community values. The company lives by a credo, which states that customers should be treated the same way employees would want to be treated, employees will do what it takes to make the company the best place to work, and that employees will show that they care deeply about the community by acting ethically at all times (*About Our Company*). Citizens also choose to immerse themselves in the community, with programs in place to fight hunger, educate people on money management, and strengthen communities through the means it best can; economic advancement. Citizens seeks sponsorships that benefit customers, colleagues, and communities. It also supports two foundations, both of which provide grants for numerous different qualifying NGOs. A notable quote from the Citizens Bank website is “we are citizens helping citizens reach their potential”, exemplifying the care which the bank applies. Within the last year, Citizens donated $17M+ to supporting community programs 16.3M+ meals through a partner, Feeding America. Additionally, over 150,000 hours were spent by Citizens employees volunteering with over 2,000 organizations. The company supports paid volunteer time, separate from PTO, for employees and always matches employee donations (*About Our Company*).

2.2 Difference Between Open Banking and Banking as a Service

Open banking refers to the place where third party aggregators can access users’ financial data through APIs. Many of the third parties that do this are financial providers. In this situation, customers have the right to consent (or choose not to consent) to the data they are sharing, per legal regulations. Open banking is a popular infrastructure in Europe already. However, due to high levels of regulation, the concept is just starting to gain popularity in the United States. Open banking allows banks to work in a streamlined way with third party aggregators by allowing them to work with data-sharing agreements and access APIs. The goal of open banking is this streamlined process, in which it is easy for all parties to be productive while holding the integrity of the customers data.

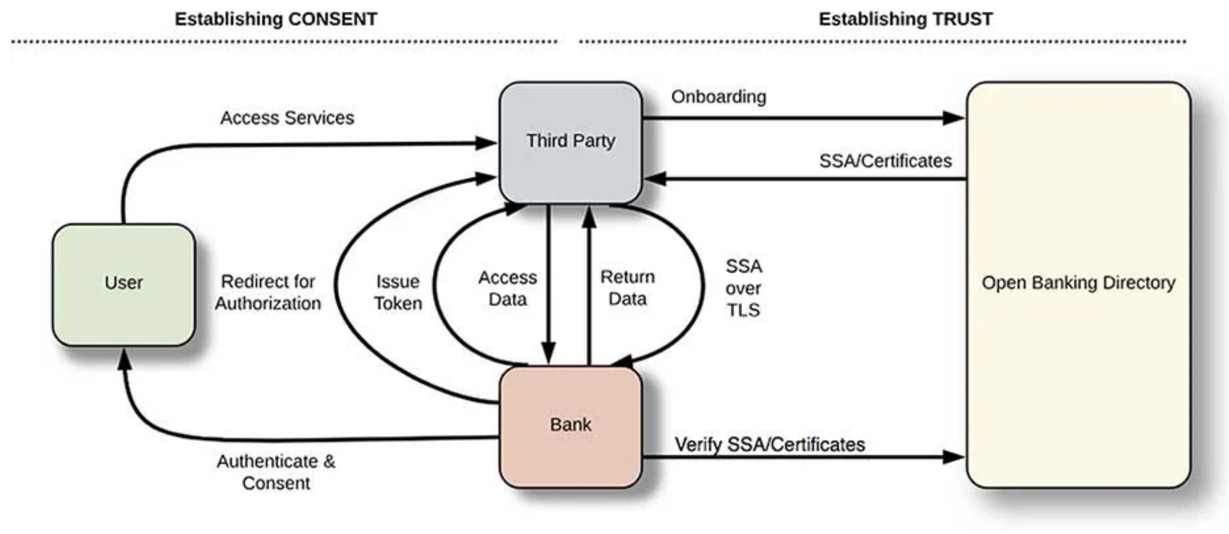


Figure 1: Open Banking Authentication Process (*Open Banking Technical*…)

Moreover, banking as a service refers to a model in which licensed banks integrate their digital banking services directly into the products of other non-bank businesses. In this case, a non-bank business can offer its customers digital banking services such as mobile bank accounts, debit cards, loans and payment services, etc. without needing to acquire a banking license of their own.

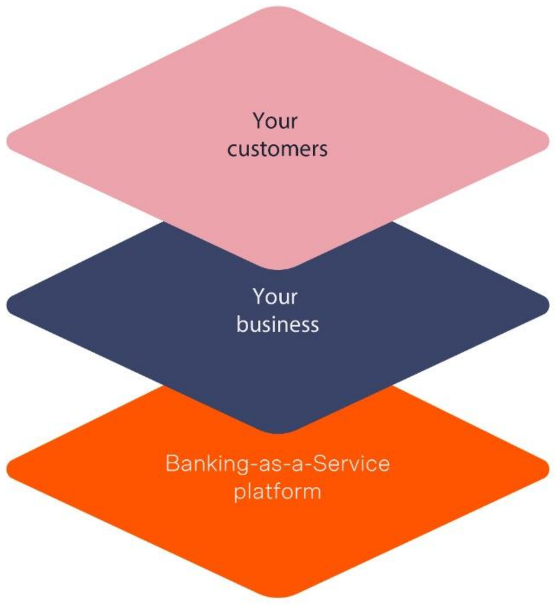
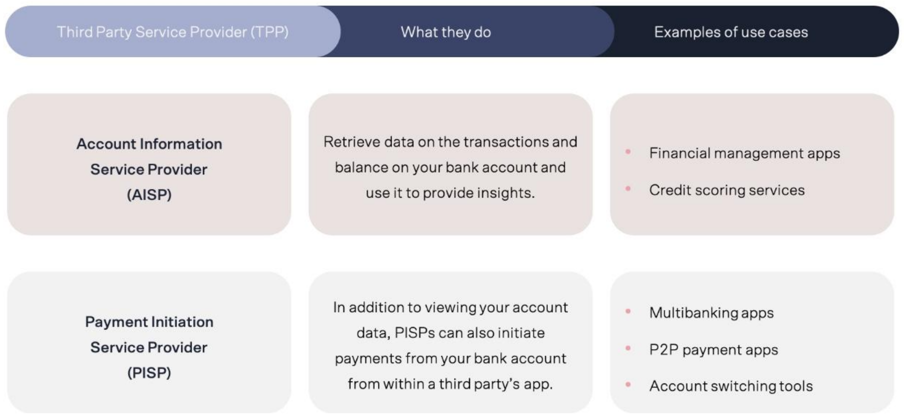


Figure 2: Banking as a Service in a Diagram (Finextra)

Open banking and banking as a service are not the same. Both open banking and banking as a service deal with banks connecting to non-banks through the means of API calls. However, the purposes are different. Within banking as a service, non-bank businesses integrate complete banking services into their own products. However, with the open banking model, non-bank businesses only use the bank's data for their own individual products. These are the third party providers. Examples of these differences can be seen below:



Third Party Service Provider (TPP)	What they do	Examples of use cases
Account Information Service Provider (AISP)	Retrieve data on the transactions and balance on your bank account and use it to provide insights.	• Financial management apps • Credit scoring services
Payment Initiation Service Provider (PISP)	In addition to viewing your account data, PISPs can also initiate payments from your bank account from within a third party's app.	• Multibanking apps • P2P payment apps • Account switching tools

Figure 3: Third Party Providers Use Case Examples (Turol)

2.3 Existing Use of External APIs

The needs of third party developers and external partners are accounted for when it comes to designing external APIs. External APIs allow a company greater third and fourth party services access — where fourth party refers to a service provider that a company does not directly have access to, but the company's vendors do — to benefit customers, gain developers, and company reach/functionality by exposing a route for external agents to request data from. This allows for greater data and service access to further the company, and allows for better launch points of new applications. Freelance developers may add value to the company without the commitment of development in the company. The publisher of an external API is able to capitalize on a large community of free agent developers. External APIs also are referred to as public APIs and open APIs.

2.4 Technical Background

The team's project was coded in TypeScript, which is a programming language built on top of JavaScript that adds optional static typing. JavaScript's primary use is in web development, as it is used to make web pages interactive through client-side, and in some cases including our own, server-side code. A statically-typed language is a language where the types of variables (such as integers, strings, and so on) are known when the code is compiled. The benefit of adding static typing is that some errors can be caught at compile time instead of at run time, making it much easier for developers to debug. Since TypeScript is built on top of JavaScript, it compiles to JavaScript, so it can be used wherever JavaScript can be.

2.5 OAuth 2.0

2.5.1 OAuth Overview

OAuth is the primary open standard authorization framework. It allows third party applications to access resources owned by a given user that are stored in other applications on behalf of the user without the user having to supply their username and password to the third party. Examples of this include Plaid connecting your bank checking account to Venmo, or TikTok connecting to your Facebook profile to find friends. OAuth is incredibly beneficial for both clients and servers, because it greatly increases the security of users, as there are fewer points that could be compromised in order for their credentials to be stolen, as well as ensuring to the servers that they are sending these private resources to the correct user.

OAuth primarily uses JSON Web Tokens (JWTs) that are per-user and grant access to specific scopes that denote which resources the token can be used to access on behalf of the user. These tokens are preferred over username and password combinations for several reasons. This includes expiration dates on the tokens that force re-authentication after a certain period of time, the ease of revoking access compared to having to change a password, and the flexibility of scoping.

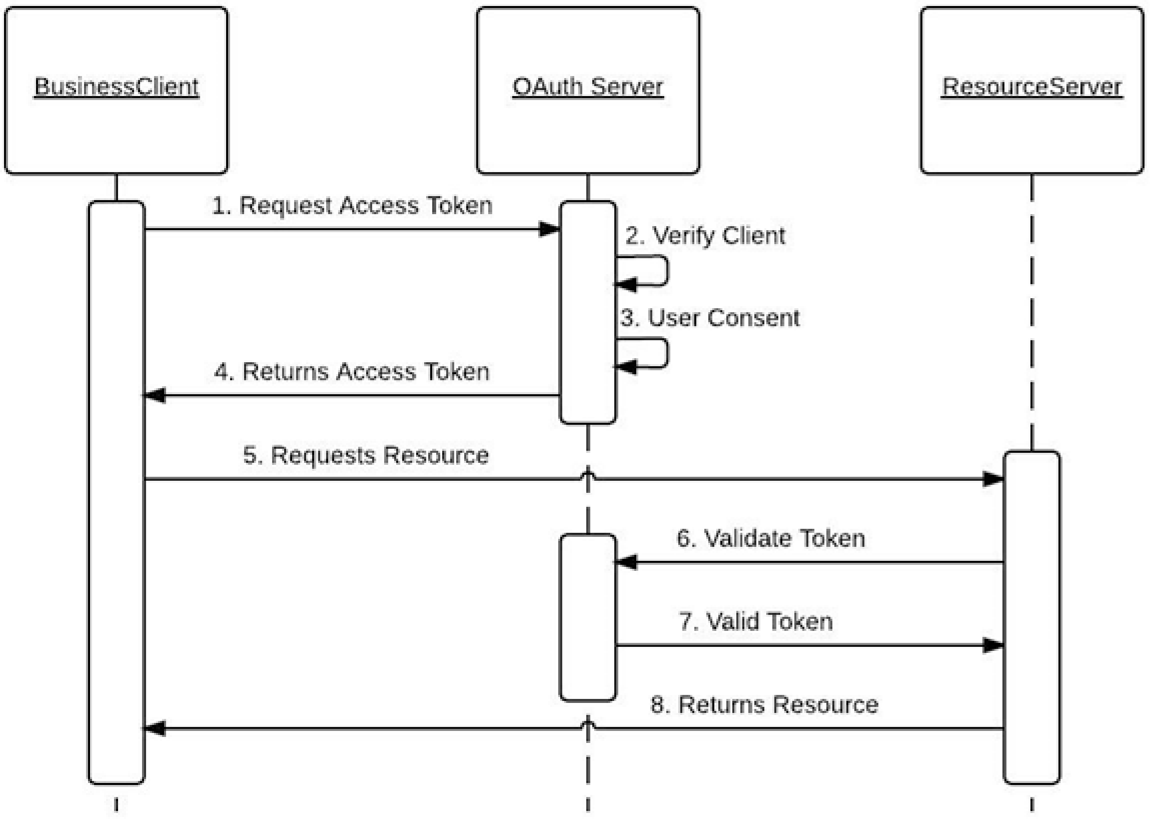


Figure 4: OAuth 2.0 Architecture Diagram

2.5.2 Two-legged vs. Three-legged OAuth

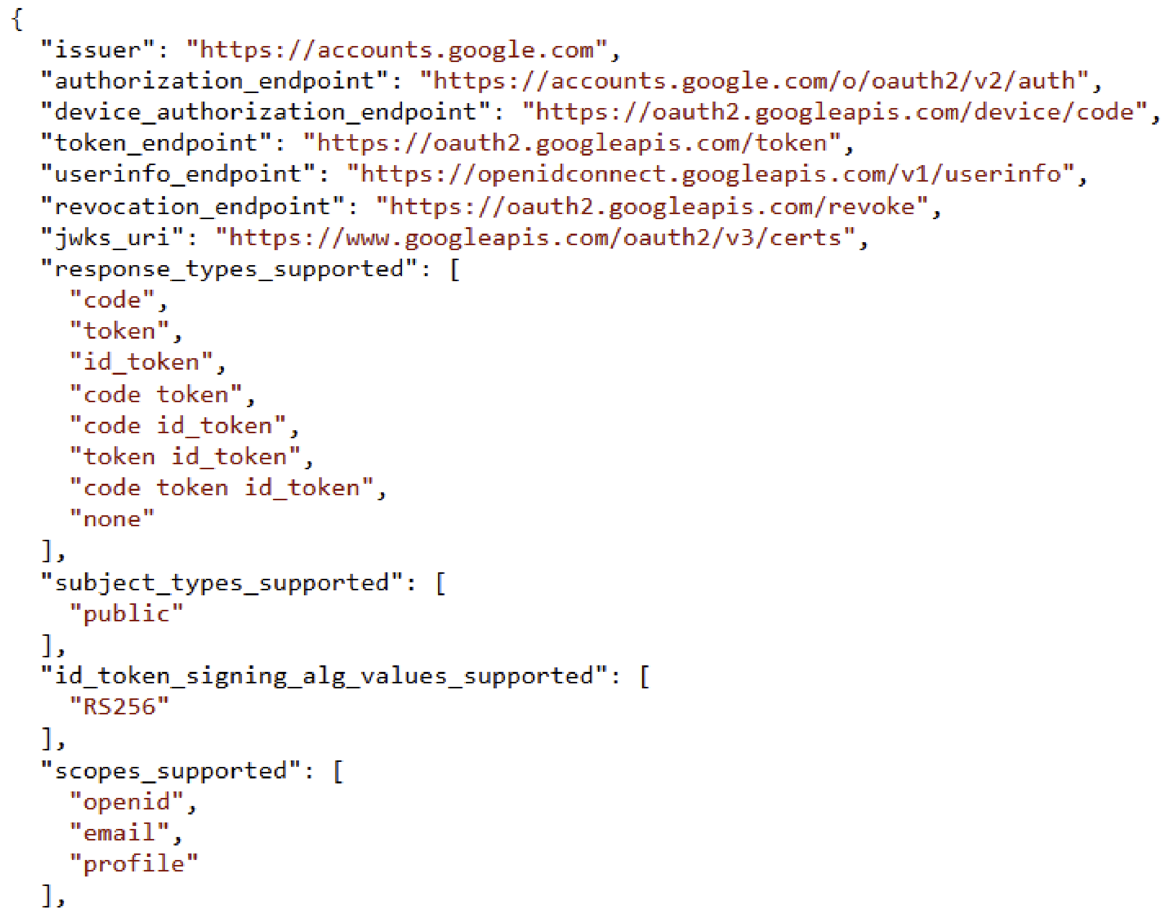
There are many different types of authorization flows that are specified in OAuth's standards, but the primary two in use at Citizens Bank and in our project are 2-legged OAuth and 3-legged OAuth. The difference between these two methods is that 3-legged OAuth involves a resource owner or end user, a third-party application, and the server, while 2-legged OAuth would be a request with a third-party application and the server, but no end user.

2-legged OAuth is utilized when a service account is needed for an application to use a service. This solely authenticates the application, but not the user, since there isn't a user in this version. In Citizens Bank’s case, an example of this would be to authenticate that Plaid is who they say they are when they request data, so that Plaid can pull any account data to provide, for example, specific account data to someone who requests it.

On the other hand, 3-legged OAuth is utilized when there is an end-user present, as the end user must authenticate to receive a token. This token authenticates the application as well as the user. In Citizens Bank’s case, these would be used to permit specific users to see their own accounts, but would not give them access to other accounts. One major benefit of 3-legged OAuth is that the users can directly give or revoke consent to Citizens Bank for sharing of data even if Plaid is still in the middle, since the token that Citizens Bank provides could be scoped to match the user’s desires.

2.5.3 OpenID Connect

OpenID Connect (OIDC) is an extension to the OAuth 2.0 protocol that is supported by many OAuth 2.0 providers, such as Google, Microsoft, and Okta. It defines a sign-in flow that allows users to authenticate and the application to obtain information about the user, such as the username or email. To support this extension, the server must provide a JSON object containing relevant information about OAuth for that site, such as the endpoints for authorization, token, and user information, as well as valid scopes and response types and other metadata. It is generally located at what is referred to as the wellKnown endpoint: “https://<server>/.well-known/openid-configuration”. Many client-side OAuth providers support this extension, allowing for developers to simply pass the client ID and secret while the library handles the flow through the wellKnown URL. For large OAuth providers like Google, the wellKnown is already known by the library, and for smaller OAuth providers, the endpoint is supplied by the developer in addition to the client ID and secret.



```
{
  "issuer": "https://accounts.google.com",
  "authorization_endpoint": "https://accounts.google.com/o/oauth2/v2/auth",
  "device_authorization_endpoint": "https://oauth2.googleapis.com/device/code",
  "token_endpoint": "https://oauth2.googleapis.com/token",
  "userinfo_endpoint": "https://openidconnect.googleapis.com/v1/userinfo",
  "revocation_endpoint": "https://oauth2.googleapis.com/revoke",
  "jwks_uri": "https://www.googleapis.com/oauth2/v3/certs",
  "response_types_supported": [
    "code",
    "token",
    "id_token",
    "code token",
    "code id_token",
    "token id_token",
    "code token id_token",
    "none"
  ],
  "subject_types_supported": [
    "public"
  ],
  "id_token_signing_alg_values_supported": [
    "RS256"
  ],
  "scopes_supported": [
    "openid",
    "email",
    "profile"
  ],
```

Figure 5: Example Part of Google's wellKnown JSON

2.5.4 Prior Authentication Methods

The main prior method of aggregators interfacing with Citizens Bank before these OAuth APIs were introduced was "screen-scraping", in which third party aggregators would ask their users to supply their credentials directly to them. To get the data, the third parties would open a browser window hidden to the user, input the credentials, then "scrape" the account and transaction information from the screen to display it to the user in the third party's app. This was incredibly insecure, as the third party had full access to the user's credentials and account. OAuth is a much more secure method for achieving the same results.

3. Methodology

3.1 Agile Methodology

Agile is a software development methodology where a project is broken down into manageable units, and team members take on specific roles with different responsibilities. The goal of Agile is a smoother and more efficient development process.

Agile development centers around breaking a project into timed releases called iterations. Iterations are evenly spaced time periods made during sprints. During sprints, team members implement, refactor, or replace use-cases, which are the atomic unit of functionality of the app. Use-cases are listed tasks categorized into stories. Stories are categorized into epics, and each story begins in the sprint backlog, which is a list of stories that need to be done. Each user story is tracked and assigned to one or more team members, often with Agile software like Trello or Jira. Team members discuss their progress in daily meetings called scrums, which are quick check-ins where teammates discuss their progress and plans. The Agile philosophy is that this strategy of constantly breaking down and rebuilding an application is the most efficient way for a team to engineer a polished, optimal final product. A diagram of Agile methodology can be seen below:

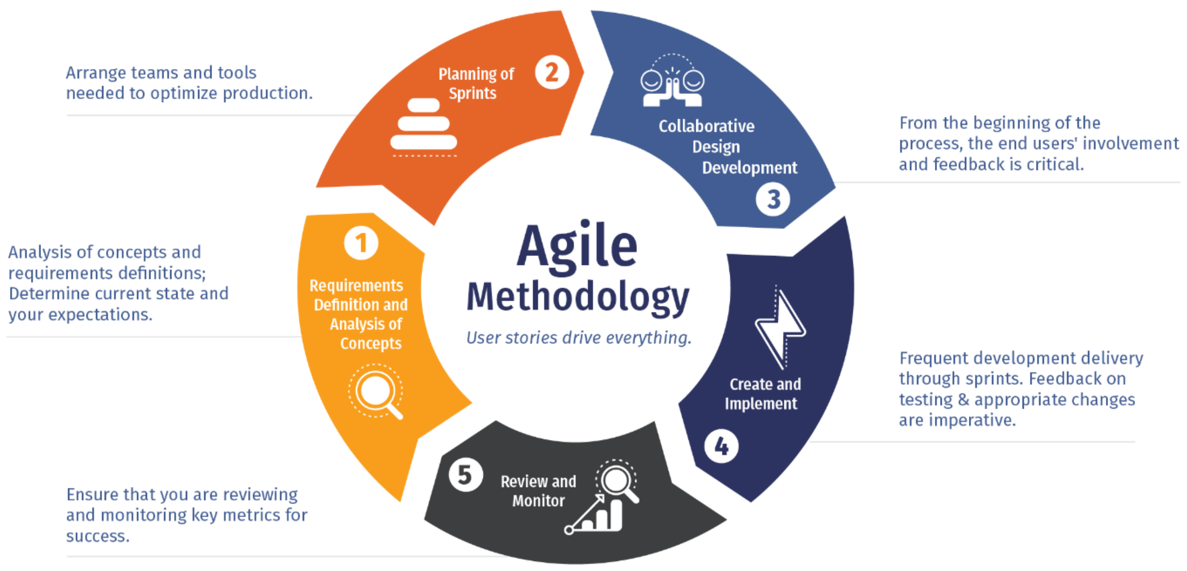

Arrange teams and tools needed to optimize production.
Planning of Sprints
2
Collaborative Design Development
3
From the beginning of the process, the end users' involvement and feedback is critical.
Analysis of concepts and requirements definitions; Determine current state and your expectations.
1
Requirements Definition and Analysis of Concepts
Agile Methodology
User stories drive everything.
Create and Implement
4
Frequent development delivery through sprints. Feedback on testing & appropriate changes are imperative.
Ensure that you are reviewing and monitoring key metrics for success.
5
Review and Monitor

Figure 6: Diagram of Agile Methodology (*What is Agile…)*

To keep the team moving along swiftly and provide more structure to the team, roles are assigned to people. The base role is the development team member, which does the work set forth by the product manager. They work together to manage the sprint backlog, participate in scrum, and contribute to the goal of the sprint. Several people within the team are given roles with additional responsibilities. These roles are the product owner, project manager, and scrum master.

The product owner's job is to remain focused on the overall value that the product can deliver and make sure that the project changes as the requirements change to maximize the value. This generally includes understanding the customer's needs, but also management, stakeholders, and other important people related to the project.The product owner achieves this primary responsibility through maintaining the sprint backlog, which is the overall list of stories that need to be completed, by making sure that the tasks that should be completed always reflect their vision to maximize the value of the product.

The project manager's job is less general, and focuses more on streamlining the daily operations of the project. They manage the team day-to-day to make sure that the project meets the requirements set by the product owner. To achieve this, they prioritize features and allocate tasks to the developers, as well as communicate with the stakeholders, product owner, and management about progress.

The scrum master is responsible for ensuring that scrum is being done correctly. They work with everyone to make sure that no one has any blockers and facilitates communication between the different roles. They also act as coaches to make sure that everyone can perform their job as effectively as possible. The scrum master is not specifically required to run the daily scrum — though this is possible, and even likely — but rather, making sure that they happen and that they are successful.

In addition to daily scrums, there are three other main ceremonies in Agile: sprint planning meetings, sprint review/backlog grooming, and the sprint retrospective. Sprint planning meetings occur at the beginning of the sprint, while sprint review/backlog grooming and the sprint retrospective occur at the end of the sprint.

Sprint planning meetings usually take around an hour and are aimed at estimating how much work can be done in the sprint. This is done through the use of story points, which are an abstract unit of measurement determining approximately how much effort it will take to complete a given task. Story points are not related with time, but rather relative to each other; for example, a story with 4 points versus a story with 2 points would be about twice as hard. Story points usually follow the Fibonacci sequence. During the meeting, each story is given a certain amount of story points based on each developer's input about its complexity and amount of work. From this, the team can determine how many user stories to take on, since they aim to take on a similar amount of points as they can complete in an average sprint.

Sprint review and backlog grooming are two separate ideas, but usually are lumped into one meeting that usually takes around an hour. Sprint review is an opportunity for developers on the team to do demos of the work they have completed to simultaneously increase morale from success and receive feedback from others in case changes should be made. Backlog grooming is focused around making sure that the backlog is clean and orderly for the next sprint, as well as being up to date with current requirements.

Finally, sprint retrospectives occur after every other meeting in a sprint and are a chance for the team to improve from sprint to sprint. Each person on the team provides feedback about what went well and what could be improved from the previous sprint to be taken into consideration for the next sprint. Rather than being a “blame game”, sprint retrospectives are a way to determine issues that need to be rectified.

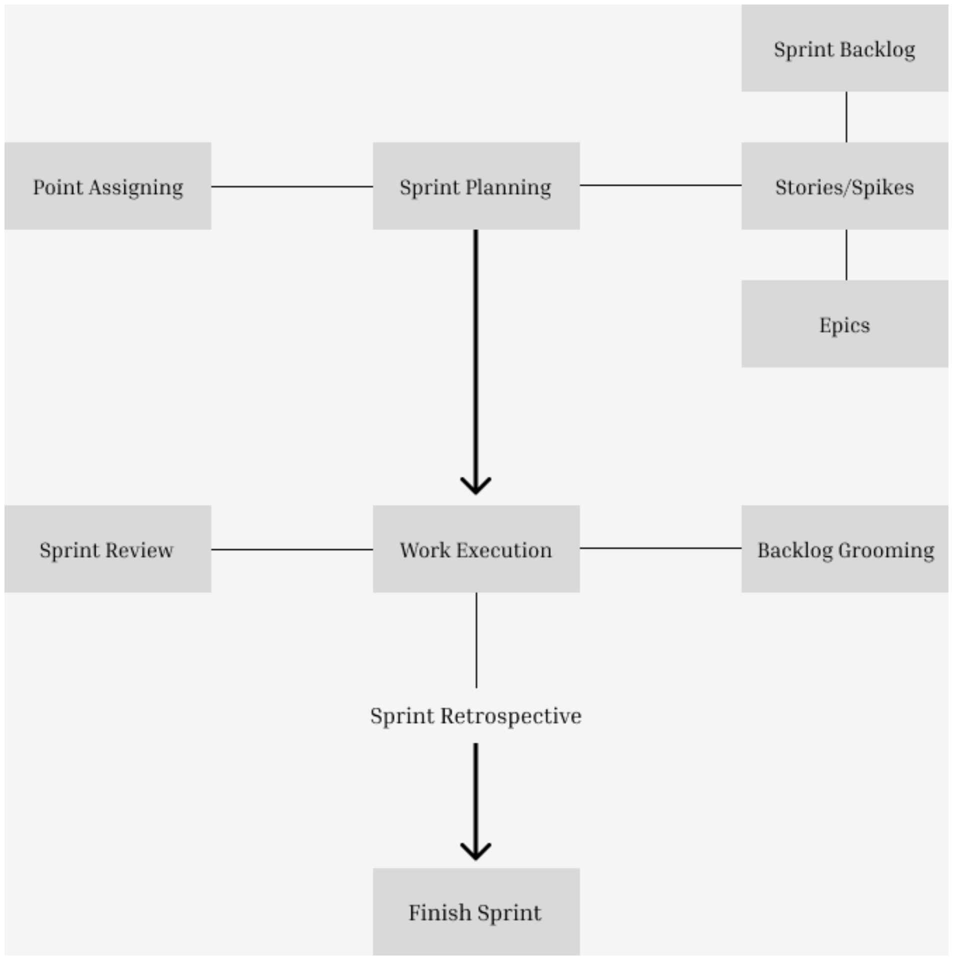


Figure 7: Scrum Process Flow

4. Citizens Bank Development Environment

Citizens Bank employs a suite of APIs across both the cloud and their on-site private network. We worked closely with a development environment within the Financial Data Exchange (FDX) API, which exists within the Citizens Network and follows the PaaS cloud computing model. The Citizens network is accessed via F5, an enterprise VPN service owned and operated by AWS. In their network, Citizens Bank self hosts nearly every tool in their project pipeline in order to comply with regulatory and general security protocols; this includes many of the tools described in the subsections below. The cloud services model can be seen below:

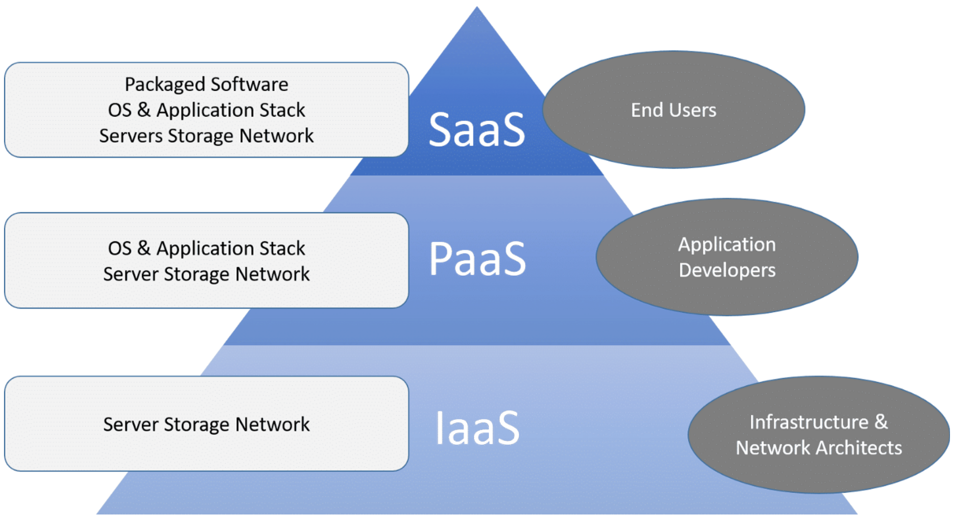


Figure 8: Cloud Services Model

Within the Citizens Bank network, there are multiple systems for managing user access to internal development tools. The majority of management access requests are handled by two systems: SailPoint, and MyAccessRequest. It is important to note that in this environment, “users” refer to Citizens Bank employees and contractors, and not third party clients. For the project team to get access to most of these tools, the team had to make access requests for each one.

4.1 Project Management Software

The project team has implemented use of Atlassian's Jira software as a tool which is used for organizing project tasks. The decision to use this was relatively easy, as it was a requirement set by Citizens Bank. Within this software, the team enables use of the Agile project methodology detailed in Chapter 3. The Jira software allows the team to create user stories associated with created epics, as well as specific technical tasks within each user story. Additionally, the team can use this software to create descriptions of each story, definitions of done, assigned points, and assigned people for each story. Jira is used for members of the team to communicate with each other regarding progress of each story and task, helping the team distinguish who is responsible for each technical task. At the end of each sprint, Jira provides team analytics including a burndown chart.

4.2 Development Environment & CI/CD

4.2.1 Source Code Management

BitBucket is an enterprise git repository management service similar to GitHub. BitBucket access is granted to Citizens Bank users via SailPoint. In addition to standard git features, BitBucket enterprise offers additional features such as better integration with enterprise Continuous Integration/Continuous Development (CI/CD) tools, integrity checks, and added security. CI/CD is a popular development method of automating much of the building and deploying process into a single pipeline where developers can simply push their code to the repository, and it will kick off processes to build, test, and deploy.

4.2.2 Integrated Development Environment

The project team used Visual Studio Code version 1.73.1 as an integrated development environment, which was installed by using Software Center. Both developers had prior experience using this software in previous projects. Visual Studio

Code integrates with git/Bitbucket, allowing the developers to easily push and pull changes.

4.2.3 Build & Deployment

Jenkins is a CI/CD tool similar to Travis CI. It integrates with BitBucket repositories and is configured using a “Jenkinsfile” in the root directory of the repository. Jenkins enables deployment, which is done to OpenShift.

OpenShift is an enterprise Kubernetes Distribution maintained by RedHat. Citizens Bank uses OpenShift to deploy many services, including the FDX development environment that our MQP is focused on. Kubernetes is a container orchestration platform that enables the deployment of services across multiple computers. Containers are instances of images (Citizens Bank uses Docker for this), which are stateless snapshots of linux operating systems with any additional required software installed.

Docker Enterprise is the licensed version of Docker, which is the standard container tool used with Kubernetes platforms like OpenShift. Docker images typically use a linux operating system, so running Docker on Windows requires either Windows Subsystem for Linux (WSL) or a virtual machine such as hyper-V.

4.2.4 Testing

Postman is a tool for sending HTTP requests that can be customized to a very high degree, including changing headers, the body, the type of request, and more. These show the HTTP requests and responses in an accessible user interface. The team used version 7.23.0 to make test API calls to internal Citizens Bank APIs.

5. Software Requirements

5.1 Software Requirement Gathering

The primary way that the project team gathered the initial software requirements was through discussions with our sponsors at Citizens Bank. The conclusion of these discussions were that at its core, the project should be a mock Personal Finance Management (PFM) app to test the new 3-legged OAuth APIs and the Financial Data Exchange APIs that Citizens Bank is currently developing. The project team was given more leeway for various enhancements to the project on top of that core idea. The team came up with a list of enhancements to discuss with the sponsors, which included testing the security of the APIs, the latency and other stats about the APIs, or additional functionality on top of the APIs. After bringing the list to the sponsor, the team concluded that if there was extra development time, adding functionality on top of the FDX APIs would be the most feasible given the time constraints. The project team discussed the current software requirements and made minor adjustments to the requirements as necessary in each sprint planning and backlog grooming meeting.

5.2 Functional v. Nonfunctional Requirements

As discussed in the section above, the main functional requirement was to create a mock PFM app to test the new OAuth/FDX APIs that Citizens Bank is developing to act as a third party aggregator. This includes using OAuth with Citizens Bank as an OAuth provider in order to get a JSON Web Token that could be sent alongside requests to the FDX APIs to act as authentication and authorization.

The non-functional requirements of the project were it being documented and running in a containerized environment. The sponsors emphasized the importance of it being well-documented, so future teams can see the work and use it as an example. Additionally, running the app in a container allows for it to be deployed within the Citizens Bank network. The importance of this was also emphasized by the sponsors, as future projects done by Citizens Bank within their network would encounter the same challenges that our app would, allowing us to act as the “canary in the coal mine”.

5.3 User Stories and Epics

Sprint	User Story	Points	Type
EPIC: Full-stack PFM Application			
1	SPIKE - Basic PFM application setup / deployment	0	Spike
1	SPIKE - As a developer, get admin access	0	Spike
1	Jack Secondary Admin access	0	Impediment
1	As a user, I want to login and authenticate application	8	Story
1, 2, 3, 4	As a user, I want to login with username and password	8	Story
2, 3, 4	As a user, I want to open PFM application	8	Story
2, 3, 4	As a user, I want to see custom interface design	3	Story
2, 3, 4	As a user, I want to see checking balance for my account(s)	8	Story
3	Setting up OAuth	0	Impediment
4	As a user, I don’t want the PFM to store my credentials (authorize PFM app without providing the data)	3	Story
4	As a user, I want to see checking transactions for individual account	5	Story

4	As a user, I want to see savings balance for individual account	5	Story
4	As a user, I want to see savings transactions for individual account	5	Story
4	As a user, I want to see checking balance for combined and/or multiple accounts	3	Story
4	As a user, I want to see checking transactions for combined and/or multiple accounts	3	Story
4	As a user, I want to see savings balance for combined and/or multiple accounts	3	Story
4	As a user, I want to see savings transactions for combined and/or multiple accounts	3	Story

Table 1: User Stories and Epics

5.4 Use Cases

5.4.1 FDX Specification

Our application uses OAuth to authenticate requests to Citizens' FDX API. FDX (Financial Data Exchange) is an RESTful API standard for open banking. Our application uses two FDX endpoints in order to fetch data for users. Note that these endpoints are fetched server-side via getServerSideProps, and authenticated with the JWT acquired from signing in.

- `GET {baseUrl}/accounts`

This returns a paginated list of all accounts tied to the current user. For example:

```
{
  "page": { "nextOffset": "11", "totalElements": 490 },
  "links": { "next": { "href": "/accounts?offset=11&limit=10" } },
  "accounts": [
    {
      "depositAccount": {
        "accountId": "1956825110",
        "nickname": "Mcclain",
        "accountType": "SAVINGS",
        "status": "NORMAL",
        "accountNumberDisplay": "211070175",
        "balanceAsOf": "2022-12-01T18:01:32.102+00:00",
        "currentBalance": 529416583006.15
      },
      ...
    }
  ]
}
```

- `GET {baseUrl}/accounts/{accountId}/transactions`

This returns a paginated list of transactions associated with the selected account. For example:

```
{
  "page": { "nextOffset": "11", "totalElements": 31 },
  "links": { "next": { "href": "/transactions?offset=11&limit=10" } },
  "transactions": [
    {
      "depositTransaction": {
        "accountId": "1956825110",
        "transactionCategory": "Interest",
        "postedTimestamp": "2022-06-10",
        "description": "IOD INTEREST PAID",
        "debitCreditMemo": "CREDIT",
        "amount": "0.36",
        "availableBalance": 45882.11,
        "transactionStatus": "POSTED"
      },
      ...
    }
  ]
}
```

6. Design

6.1 Software Architecture Overview

Our application is a containerized full-stack web app deployed to OpenShift within the Citizens Bank network. Users can sign into Citizens Bank via a button on the frontend; with OAuth and a session managed by the NextAuth library, our application fetches data from the FDX API, the server renders pages, and then it delivers them to the client on request.

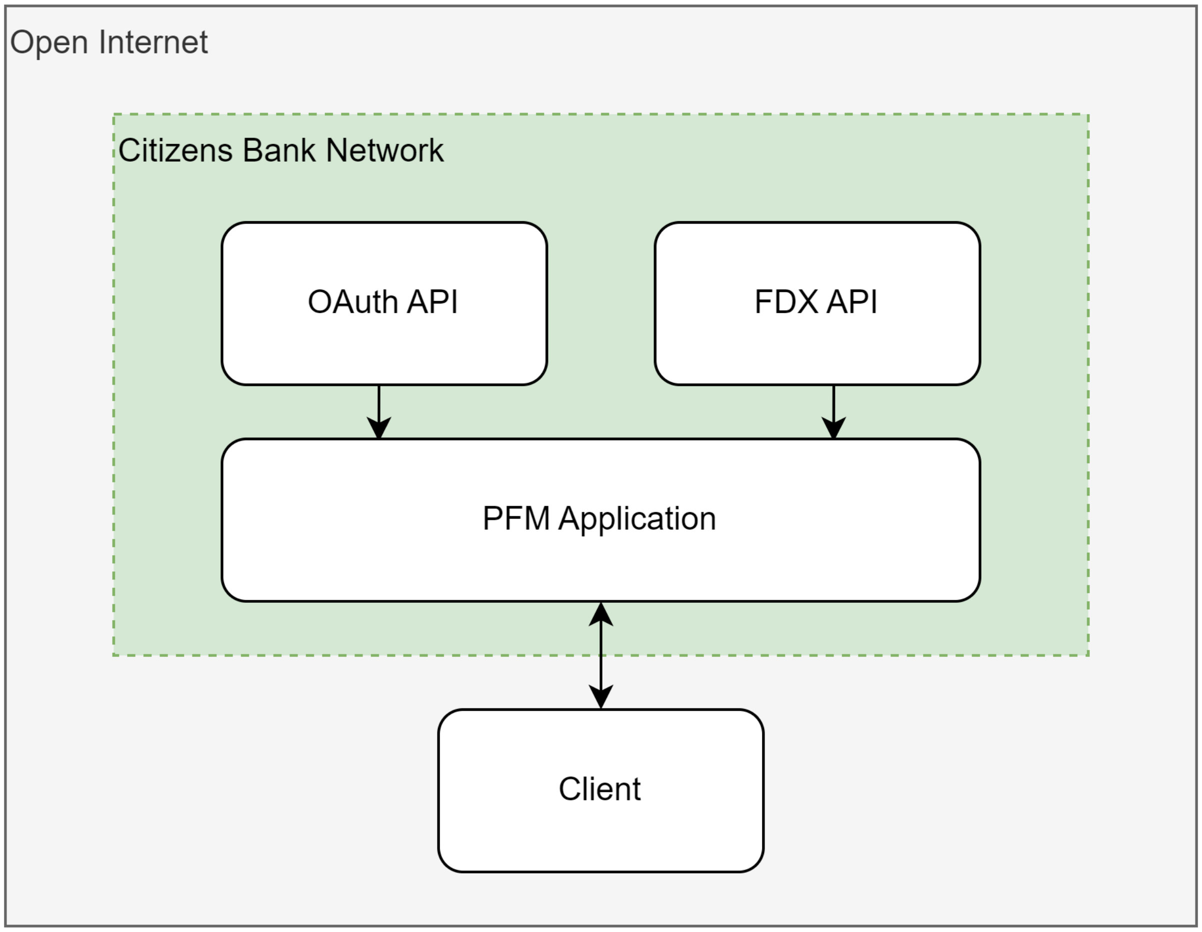


Figure 8: Citizens Bank Network Within the Open Internet

The PFM application is built with Next.js, which means that the exchanges between the client and the application have slightly more complexity than a “standard” full-stack solution. Firstly, our application makes use of server-side rendering wherever possible. Server-side rendering is accomplished in Next.js via the `getServersideProps` function, which gives developers a way to inject the results of server-side procedures into the page via React’s props API. Because Next.js allows asynchronous code in getServersideProps, we are able to actually do OAuth authentication while in the server-side rendering stage. This is further assisted by NextAuth, which is designed to work specifically with Nextjs and getServersideProps.

The application itself is divided into four main pages; note that a “page” is a special react component within NextJs that can be server-rendered via getServerSideProps:

- `/:` Landing Page

  The landing page is a static page that just summarizes the project and who we are. It has no functionality and does not use getServerSideProps.
- `/dashboard:` Dashboard Page*

  The dashboard page contains a card view of each bank account tied to the signed in user. Each card is assigned a random color that can be customized by the user. The data for the cards is fetched during the server render from the /accounts FDX endpoint. In some cases, multiple requests are necessary since /accounts returns a paginated response.
- `/transactions:` Aggregated Transactions Page*

  The transactions page is an aggregated table of all transactions across every account tied to the current user. It also includes interactive sorting by clicking on column headers in the table of transactions. The data for the transaction table is fetched during the server render from the /transactions FDX endpoint. Multiple requests from the app are necessary, since transactions have to be requested for each account and accumulated into one list.

*These pages are only accessible if the user has a valid session; if not, the user is redirected to the landing page.

6.2 Software Security Framework

Testing the OAuth APIs was one of the main functional requirements of the project, since we would need to authenticate in order to gain access to the FDX APIs. In order to do this, the team had to follow the flow as outlined in Figure 9.

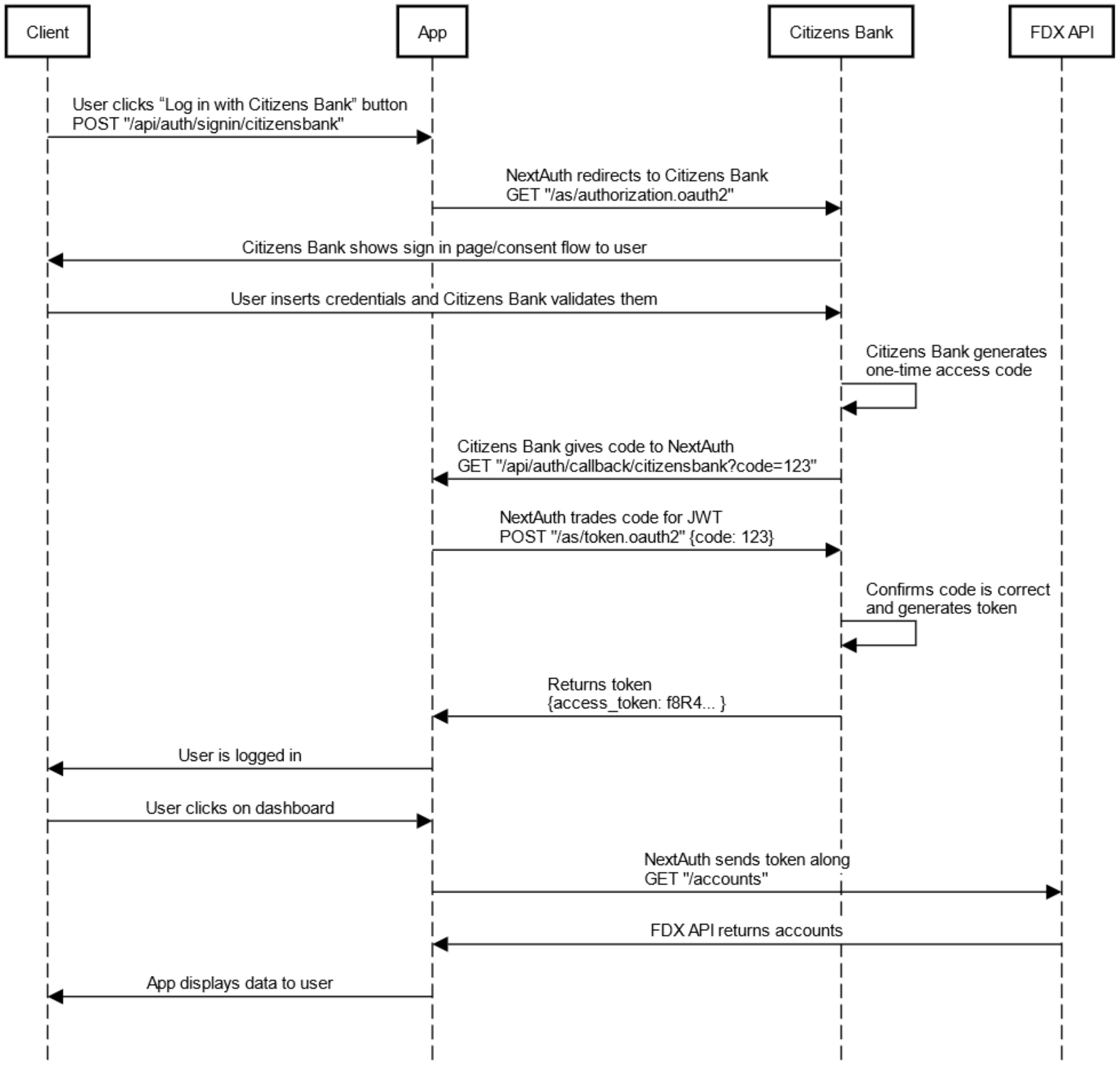

Client
App
Citizens Bank
FDX API
User clicks "Log in with Citizens Bank" button
POST "/api/auth/signin/citizensbank"
NextAuth redirects to Citizens Bank
GET "/as/authorization.oauth2"
Citizens Bank shows sign in page/consent flow to user
User inserts credentials and Citizens Bank validates them
Citizens Bank generates one-time access code
Citizens Bank gives code to NextAuth
GET "/api/auth/callback/citizensbank?code=123"
NextAuth trades code for JWT
POST "/as/token.oauth2" {code: 123}
Confirms code is correct and generates token
Returns token
{access_token: f8R4... }
User is logged in
User clicks on dashboard
NextAuth sends token along
GET "/accounts"
FDX API returns accounts
App displays data to user

Figure 10: Citizens Bank OAuth Flow

1. The client will click on the “Log in with Citizens Bank” button
    a. POST request is sent to /api/auth/signin/citizensbank
2. NextAuth will send a GET request to Citizens Bank’s authorization endpoint containing information like scopes and redirect URL
3. Citizens Bank displays their sign in page/consent flow
    a. The user enters their credentials and Citizens Bank validates them
4. Citizens Bank generates a one-time authentication code and calls GET /api/auth/callback/citizensbank?code=123
5. NextAuth calls Citizens Bank’s token endpoint with the code
6. Citizens Bank verifies the code is correct and generates a JWT that authenticates the user with the correct scopes, etc.
7. It returns the token to NextAuth, which then generates a session token and stores the session
8. The user is now logged in and any fetch requests to get data will have the token sent alongside it

7. Software Development

The project team, with input from our sponsors, decided to do one week sprints, which ran from Wednesday mornings to Tuesday evenings. Two sprints were extended to two weeks instead to make sure that those sprints were long enough to do sufficient development due to days off from holidays. As per the Agile guidelines, the team met in daily scrums to discuss their progress, their planned work, and any impediments preventing the team from completing the current stories. Additionally, the team held the remaining standard Agile meetings. The team hosted sprint planning meetings on the last day of every sprint to plan the next sprint. The team also held sprint retrospectives a day or so after the sprint ended, in which the team reflected on the previous sprint's positives and negatives to adjust processes in the current and future sprints. Finally, the team met for sprint grooming/review meetings in the middle of the current sprint for reviewing and improving the product backlog for the next sprint. During the sprint grooming meetings, the team also demoed their work from the previous sprint. The specific details of each sprint are discussed in the corresponding subsection below.

7.1 Sprint 1

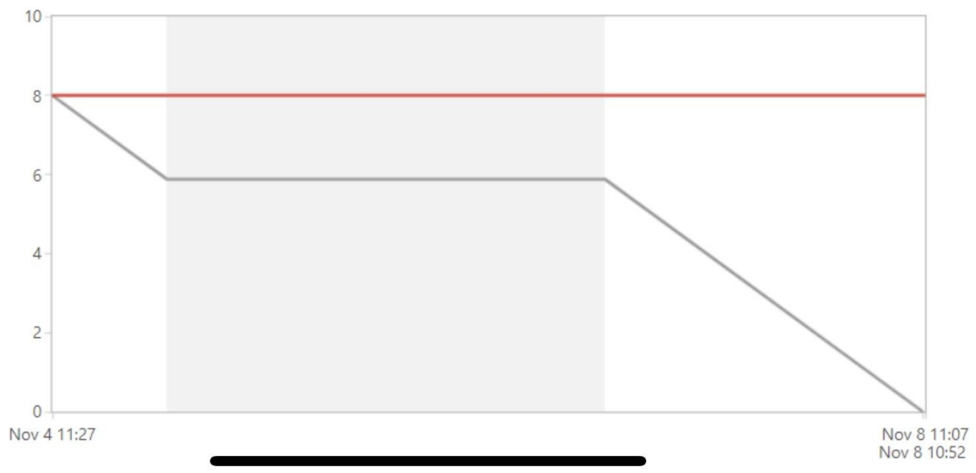


Figure 11: Sprint 1 Burndown Chart

Total Points: 0

During the first sprint, the project team focused on learning Citizens Bank's systems and structure, as well as requesting access for the various tools required to begin developing an app. The "zeroth" sprint, which was the first week of the term, was included in this section because both sprints had the same goals and spikes as the team worked to set up their environments. It took until halfway through the first sprint to begin development, as the team did not have access to BitBucket and one developer did not have local admin access, preventing them from fully setting up their development environment. The team was eventually able to resolve both of these issues while at the Citizens Bank campus in Johnston, RI on Thursday.

After the initial week and a half of almost zero progress, the team worked quickly once no longer impeded. After two days, the team had managed to fully deploy the NextjNs application to OpenShift such that it was available on a static URL within the Citizens Bank network. This meant that any pushes to the Bitbucket repository would trigger the Jenkins pipeline to build the docker image (including an optimized Next.js build of the site) and deploy it to OpenShift at our URL. At this point, the CI/CD pipeline and live deployment was fully set up and the team could finally start properly developing.

The team also determined that they would not be able to get a working database, seemingly due to strict regulations about data storage. The team decided to simply use a memory store for persistent data, meaning that if the app is restarted, data will be lost. Otherwise, the team communicated with its stakeholders and sponsors to determine the goals of the UI/UXD. The team decided that a minimalistic approach to the UI with user-focused mobility and intuitive use would be ideal, deciding on a dashboard with all account information (deposits/spending/transfers) from all banks in chronological order with a separate page for transactions. The team debated features such as a search bar and different types of filters, and decided to prioritize them determined by user interaction.

7.1.1 Sprint 1 Retrospective

Over the course of the two initial sprints, the team was finally able to get over multiple weeks-long spikes which prevented the developers from beginning programming. Additionally, throughout the sprints, the team found that they did communicate well with each other and the sponsor, making sure that everyone was on the same page. However, the team also found that during meetings, they would stray away from the point of the meeting to focus on other points. To solve this, the team decided to make sure that the meeting titles always reflected the primary topic of the meeting, and if other topics needed to be discussed, they were scheduled separately or discussed after the meeting was over. Finally, the team also agreed they had to leverage their findings in their Citizens Bank Confluence page to provide thorough documentation for future teams.

7.2 Sprint 2

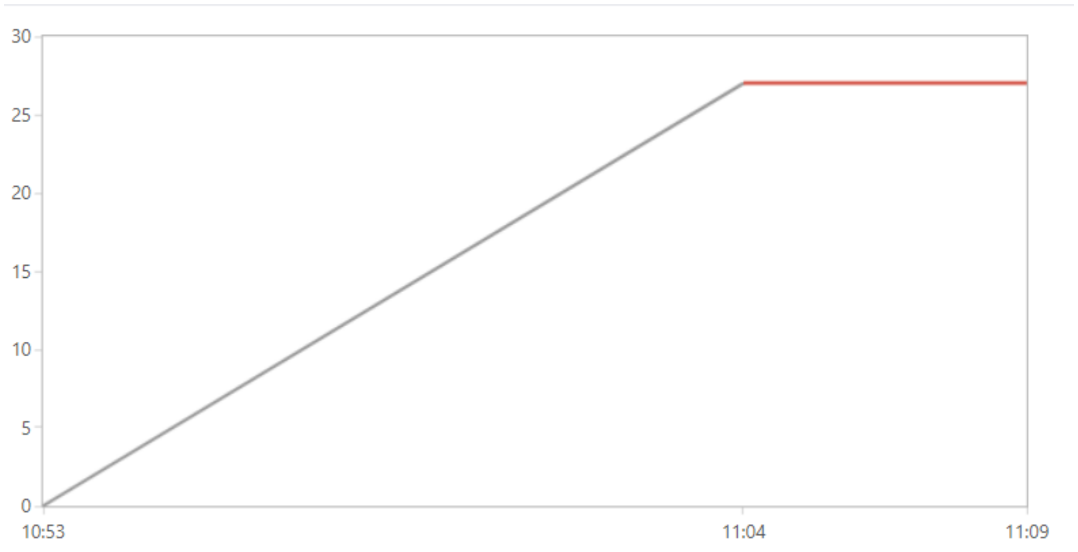


Figure 12: Sprint 2 Burndown Chart

Total Points: 0

The team had a productive week, despite the score of 0 points. There was a great deal of progress made on both the technical and non-technical ends, particularly on the user interface in this sprint. The team has developed a visually functional user interface on the front end equipped with a home landing page with a non-functional login button, a dashboard tab showing customizable cards with balances and running totals, and a transactions page in development. The team could still not use OAuth by this point, a problem which they worked with Citizens on extensively. Because of this, the user interface was tested displaying mock data during this sprint, rather than pulling real data from the FDX APIs.

7.2.1 Sprint 2 Retrospective

Despite scoring 0 points, the team was pleased with its progress. A great deal of work was still completed and while no stories were closed this week, the team set itself up for success. Jonathan's diligence on OAuth work was particularly impressive. Additionally, Jack's contributions to the UI was admirable, as it helped the team to start to visualize an end in sight. However, the team still agreed that it needed to improve in regards to documentation. The team completed a lot of the academic paper during this sprint, but still needed to update the Confluence page for Citizens Bank.

7.3 Sprint 3

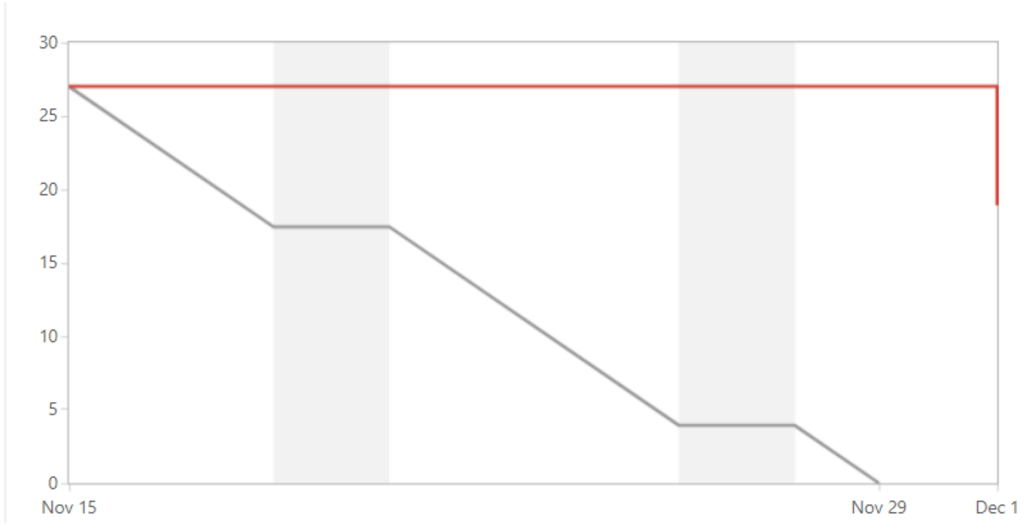


Figure 13: Sprint 3 Burndown Chart

Total Points: 0

The team had yet another successful week despite another 0 point sprint. A large portion of this project revolves around identifying the readiness of Citizens Bank to begin integrating with third party aggregators. This sprint’s work was integral in the team's conclusion: Citizens Bank at this time is unable to fully support integration with external third party aggregators. However, they could support individual integration on a case-by-case basis. This conclusion was reached because the team was able to make progress, but only with significant help from resources internal to Citizens Bank that third parties might not necessarily have access to.

In implementing OAuth, the team’s work needed to be treated like an external URL. The team faced an issue when making fetch requests from the app, so it resorted to working with Citizen Bank’s Platform as a Service (PaaS) team. While the team considered numerous times to switch to another OAuth provider, it was decided that this would not make a difference because the problem was somewhere in OpenShift blocking access to making fetch requests, so regardless of the provider, the same problem would be encountered. If the team could access OAuth APIs from the public, it would not be facing this issue.

While troubleshooting the numerous issues, the team received a great deal of advice. Some Citizens Bank employees were suggesting use of a hardcoded mock identity provider (IDP) setup, and others suggested attempting to integrate with Citizen Bank's implementation of OAuth, Ping. Although it was not yet ready at Citizens Bank, attempting to use it would better align with the second goal of the project and better prepare the app for future work after Ping was ready. However, both of these suggestions would face roadblocks. The primary issue with the mock IDP setup would be that it was not the actual OAuth server, so using it would not help Citizens Bank gauge whether they were ready to integrate with third party aggregators. Additionally, due to its hardcoded state, the team would likely not have sufficient time in its short scope to get this fully completed. On the other hand, the issue with the Ping solution is simply that it was not yet ready at Citizens Bank; the tokens that it provided would not act as authorization for the FDX APIs, while the mock tokens would. The issues the team faced played a role in Citizens Bank hosting an internal Sandbox FDX API Alignment meeting to discuss shortcomings and what work still needed to be done to make this ready. Additionally, the team created its final plan to complete the project, which was to continue coding its deliverable tool as far as it was able to. This would offer Citizens Bank the ability to seamlessly integrate real data once the work it needed to do was complete, which consisted of integrating their OAuth tokens into the FDX APIs. In order to do this, the team created two separate branches, one focusing on the real OAuth server, and the other focusing on the mocked one. Since the mocked server could provide tokens that would let the app pull data from the FDX APIs, that branch would be the primary branch for continuing frontend work, as the team could work on correctly displaying actual data. On the other hand, the other branch would serve as the base for continued work on getting OAuth to work properly, so when it is finally ready, the app could be switched over to using OAuth instead.

7.3.1 Sprint 3 Retrospective

The team agreed that what went well this sprint was continuous progress, despite being unable to close out any of our user stories. The team had a major breakthrough in coming to a conclusion on the project's main goal, assessing whether Citizens Bank is ready for third party aggregator integration through the use of open banking APIs. While it was determined that Citizens Bank was not yet ready, the feedback was highly valuable for them. While it would have been great to deliver the full-stack tool the team had hoped it would be able to, Citizens Bank is satisfied with the team's work and thankful for the proof of concept determination the team completed. Nonetheless, the team has learned, and hopes to improve on in the future, a great deal about story writing, estimating, and scope. The repeated 0 point sprints do not reflect the amount of effort the team has dedicated to this work. The team recognized that an area for improvement would be writing stories with smaller scores and simplified acceptance criteria, which would allow the team for improved velocity that more accurately reflects the work it did.

7.4 Sprint 4

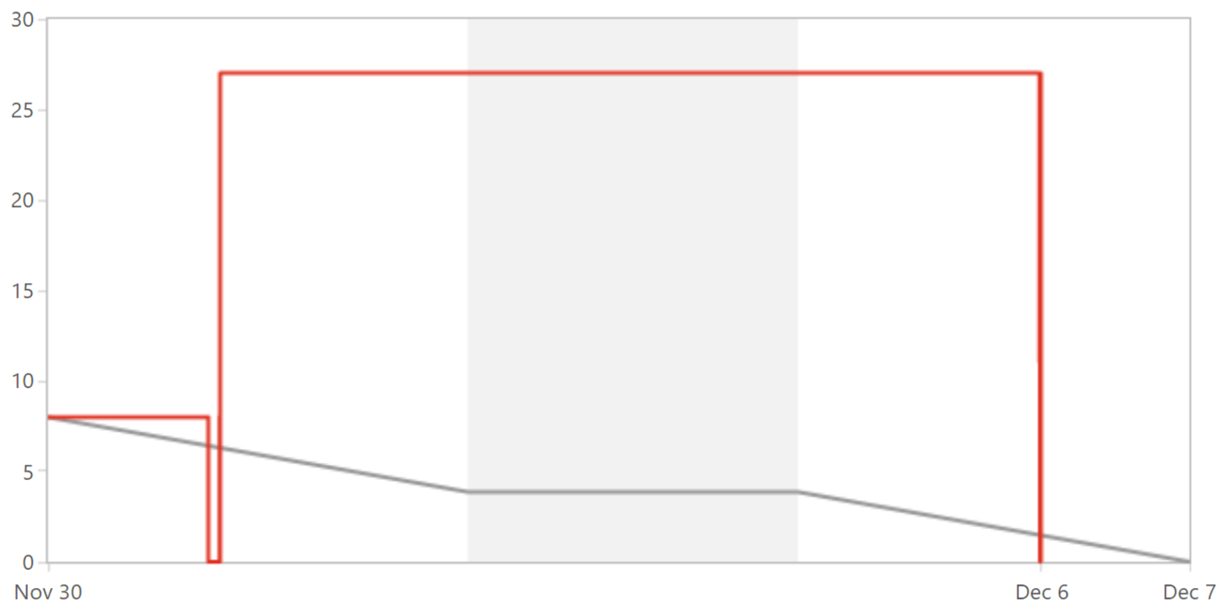


Figure 14: Sprint 4 Burndown Chart

Total Points: 57

In this sprint, the team was finally able to close many of the stories that had been kept open since the initial sprint. This was due to the major impediment of not being able to make fetch requests being resolved, allowing the team to finish many of these tasks that had been otherwise completed.

The team ended up merging the two branches into one as the same error was occurring for the branch still in the old environment. To fix this, the branch in the new working environment was updated to be able to swap between the versions with a single line, which was an environment variable pointing to the location of the wellKnown endpoint. Additionally, the transactions FDX API was completed towards the end of this sprint, allowing the app to pull data from the FDX endpoint. Previously, the team had created a script that would generate mock transaction data upon page load to display the UI as it would look eventually as a temporary workaround until the API was working.

The team spent much time working on editing and finalizing the paper as well, in this sprint. Part of the way through the sprint, the final draft of the paper was due for editing.

7.4.1 Sprint 4 Retrospective

The team closed out a total of 12 stories during this week's retrospective and last scrum. There were still some technical adjustments to be made regarding both the real Ping server and the mocked one, but the team was satisfied with the progress. There was progress in the team regarding delegating and managing final pushes to get the academic paper finished, and reaching completion in terms of quality. The team planned to finalize our presentation next, along with a live demo included for both the sponsors and advisors. The team also agreed that there could be some better communication in terms of seeking help internally in the team, and more continuous documentation updates. Another issue was that many of the stories relied on each other for continuation and project build up, so instead what could have been more helpful would be predicting stories individually from each other.

8. Business & Project Risk Management

8.1 Risk Versus Reward

To determine what risks are present in a given project or release, finding the risks and having to manage the risks are both crucial to the integrity of the final product. Risks are based on the specific environmental control we have, keeping an inventory of controls across the environment, and screening for potential risks before the launch. Due to Citizens Bank being one large operation held together by many smaller processes and regulations, our team specifically zeroed in on smaller risks that were specifically applicable to our app, alone.

When our team looks at the largest risk presented as a whole in our project, it is worth noting that a business risk presented is regarding our ability to affirm if Citizen Banks' technology is compatible with third party aggregators. Ultimately, the team was not able to produce the deliverable that was originally planned due to complications surrounding limitations with Citizens Bank. The team had to prioritize a tight timeline, accounting for time needed to gain authorization for tools, platform, and software needed, and turnaround time to redirect the project for roadblocks due to Citizens Bank jurisdictions. On the other hand, the reward considering our project revolves around bringing forth a deliverable that can be demoed and display functionality to both the sponsor, the advisors, and WPI. The whole experience regarding building the product consisted of managing and working in an industry with the user in mind, accounting for the needs and expertise of the Citizens Bank sponsor, and receiving technical and management consulting for the advisors at WPI. This provided a real-world corporate work experience for the team, troubleshooting management experience for the advisors, and exposed situations in which the sponsors became aware of the limitations that caused the group to have to work around them.

8.2 Risk Culture

Citizens Bank Level 1 Risks	Citizens Bank Level 2 Risks (Specific Categories of Level 1)
1. Credit Risk	Changes in Macro-Economic Environment, Counterparty/Settlement, Credit Concentration, Credit Mitigation, Hard Underwriting, Lending
2. Compliance Risk	Anti-Money Laundering/Sanctions, Bank Operations, Conduct and Ethical Behavior, Customer Harm, Deposits, Lending
3. Interest Rate Risk	Interest Rate Risk in the Banking Book
4. Liquidity Risk	Contingent Liquidity Risk, Operating Liquidity Risk
5. Operational Risk	Business Continuity, Change, Data, Financial Reporting, Legal, Model & Artificial Intelligence/Machine Learning, Non-Financial Reporting, People, Transaction Processing, Property Services, Third-Party Management, Boundary Events
6. Price Risk	Commodities Risk, FX Risk, Interest Rate Risk, Mortgage Servicing Options Risk, Securities Risk, Traded Credit Risk
7. Reputational Risk	Corporate Responsibility, External, Internal
8. Security/Fraud/Financial Crimes Risk	Fraud, Information Security/Cyber, Physical Security
9. Strategic Risk	Execution of the Strategic Plan, Strategic Decision Making, Strategy Governance
10. Technology Risk	Applications, Asset Management, Technology Infrastructure

Table 2: Risk Culture at Citizens Bank

At Citizens Bank, there are ten categories of risks accounted for: credit risk, compliance risk, interest rate risk, liquidity risk, operational risk, price risk, reputational risk, security/fraud/financial crimes risk, strategic risk, and technology risk. At a large, highly structured organization, all risks must be foreseen and recorded to ensure issues can be avoided, stopped in the process, or to create a plan for resolution.

Citizens Bank has to consider compliance, operational, strategic, technology, financial crime, and reputational risk as main risks in affiliation with our team. The compliance risk specifically involves accounting our group for our conduct and ethical behavior. Our Citizens Bank sponsors had to comply with internal policies and regulatory authorities, which is why we were not given access to personal customer data to test our app on, and just created our own numbers and name for the user. The deliberate or inadvertent exposure to confidential private-side/material non-public information would have been a violation of customer data protection.

The operational risks include both people risks and third-party management risks. Citizens Bank assumes the risk if our group were to fail to meet the sponsor's needs, and fall short of expectations. As taking us on as a third party group, Citizens Bank expects that we manage our own risks, and meet regulatory and company governance oversight requirements. Due to the nature of the randomized placement of the group and creation of our group, Citizens Bank also could have potentially encountered an operational risk of talent and skill set in people. This would happen if there was a failure to ensure proper employee talent and skill sets.

The strategic risk could involve either poor governance or poor strategic planning. If the sponsor failed to generally shape the actions of what was expected of the group or had poor direction in a vision for a project, this would impact the value of the group's operations. When the governance and oversight of a group is not adequate, this presents huge risks and leaves vulnerabilities as it turns into a matter of internal limitations.

The technology risk in our project was associated with potential deficiencies in design, performance, and supporting infrastructure. Application maintenance and application design is crucial in scrums, sprints, and retrospectives. This serves as a foundation to monitoring progress, process, and collaboration. System availability is another technological infrastructure risk regarding deficiencies from availability and performance. The failure to design or implement a system to satisfy the group's deliverable objective, and adequate capacity to satisfy business needs is a system design related risk.

Financial crimes are unfortunately a risk with any Citizens Bank team with access to personal data or infrastructure access. The goal of information security is to reduce risk of unauthorized or misused accessing, disrupting, modifying, or destroying data in a digital or physical manner. Citizens Bank has the risk of the team improperly identifying, prioritizing, remediating, or mitigating software vulnerabilities from insufficient information/cyber vulnerability management. There is also always a risk associated with the sponsors if they gave us personal info of customers, which would be an unauthorized logical access and authentication to internal systems.

Lastly, Citizens Bank faces a few external reputational risks with working with our group. Reputational risk represents negative impacts resulting from harm to the bank's reputation from unpopular business decisions, strategic direction, or perceived commitment to social and environmental causes. This can be either internal, e.g. employees, or external, e.g. customers and investors. Because our group consists of contractors for Citizens Bank, our associated risks would be internally inclined. If anyone in our group was not happy with our experience with working for Citizens Bank, this would be a risk of low employee satisfaction potentially from inadequate people management, adverse corporate/sponsor's decision, or perceived inadequate social responsibility.

8.3 Additional Risks

Our group faced risks of our own regarding the environment at Citizens Bank. In terms of technology risk, the technology infrastructure was not always adequate. For instance, we have had multiple setbacks due to not being able to access specific applications needed to move forward in coding. This was due to the high security at

Citizens Bank. Inconveniences included things such as admin access being taken away, long wait times for admin access, multiple request forms for specific access, and restricted internet use at Citizens Bank and the software on their applications. This technology infrastructure risk came from deficiencies in accessibility to the needed tools to do the project. This caused a time and business disruption for our team due to inadequate system availability. This also goes hand in hand with a reputational risk. The incidents of technological setbacks and inconveniences caused an internal reputational risk within the group by negative perception and association with the banking industry.

The operational risks specific to our group included people management risk and key person risk. For instance, in key personnel risk, if there was an overreliance on a select personnel to execute most or a large amount of the vital functions of the group project, that could become a later issue in terms of succession and all group members being able to prove adequate knowledge of concept. The people management risk involves a failure to manage performance, development, and capacity. The team ideally should function as one to move the progress forward, as well as have each person self regulate to prevent hierarchy. We are our own liability as a third party contractor to Citizens Bank, having to effectively manage our own risks.

9. Assessment

Over the course of the project, the team had two main goals, the first being to simply determine if Citizens Bank is ready to integrate this technology with third party aggregators, and the second being to develop a tool to prove this. At a high level, the team was able to complete the first goal by showing the proof of concept is not yet ready for the use of third party aggregators. Additionally, the team did not provide the tool it planned to in its entirety due to Citizens Bank limitations, but it did develop a tool as far as it was able to given the circumstances.

9.1 Business Learnings

Being sponsored by Citizens Bank was a unique experience for the team. The group was treated like full-time employees (and decided to take on the team name of "Airbus Pod"), so its exposures were plentiful. The team learned a great deal about Citizens Bank as a business through its leadership, industry, and culture.

Citizens Bank has a leadership that motivates and mentors its employees. On numerous occasions, whether within its own work team or by observing other teams, the team saw leaders asking probing questions to team members. The style was not authoritative, but rather encouraging of people to ask questions, ask for help, and challenge the norm. Despite this approach, there were still times where teams were still confused about goals, plans of action, etc. While the business owners acknowledged skilled technical experts, the leadership made clear and diplomatic distinctions between technical and business value to drive decision making. A notable and reiterated quote from one of the team's sponsors was, "How do we make money?" This question was often asked when debating technical value versus business value, as an unambiguous reminder that technical development must support customer needs and wants. In the future, our project team's business-focused students aim to be more confident in stepping into decisions to address business value, despite the technical influences, and hope to model these conversations like the similar discussions they experienced at Citizens Bank.

The team learned quickly and persistently that banking as an industry, and particularly Citizens Bank as a fairly large organization in said market, moves slowly and cautiously. Banking is heavily regulated, and understandably so. The team promptly learned that the industry is constantly dealing with customer data and financial information, which contributes to the numerous levels of security protocols that had to be addressed to complete the project. The team faced challenges at times regarding this, as often, it felt some standard Citizens Bank practices were trivial yet tedious. Examples of a non-value-added and bottleneck control was the strict process of filing for multiple accesses needed for tools to build the project, and having access taken away at seemingly random times. Citizens Bank encountered technical challenges in furthering the team's project or accepting alternative ideas to accomplish goals.

The team's work reached a point where it was up-to-date with existing Citizens Bank technology. The obstacles of prolonged time and reduced productivity due to technical restrictions left the team wishing it could have had more clarity surrounding protocols, and more importantly how to properly overcome them.

The Citizens Bank work culture is a compelling one, as the team experienced the crossroads of qualified and intelligent management with plentiful blockers. The team was impressed with the passion many of the employees exhibited and the willingness of both leadership and others to help with its abundant questions and trials. It was interesting for the team to function from the perspective of a third party aggregator. It was able to highlight what went well and what needed improvement for Citizens Bank to integrate real third party aggregators. The team was, in some ways, a group of consultants testing a proof of concept. While it was disappointing to see this proof of concept not be successful for the time being, the team provided Citizens Bank service with value information. Additionally, it was valuable for the team's business students to learn technical terms and functions on the fly. This exposure was new for both of them, from APIs to Agile methodology to OAuth, etc. All aspects considered, the team agrees that working with Citizens Bank was a hugely beneficial experience, and likely a glimpse of operations in any largely regulated industry, and an opportunity for which the team members are grateful.

9.2 Technical Learnings

9.2.1 Functionality vs Stability Risk Management Tradeoff

Within any software ecosystem there is an inherent tradeoff between functionality and stability. Using newer software gives developers access to improved functionality, but newer software is inherently more vulnerable to security risks. “Bleeding edge” is a colloquial term for software at the far end of this spectrum: the newest, and most untested software. The advantage of staying behind the curve and using older software is a higher guarantee of stability and security. Since Citizens Bank wants its system to be as secure as possible, it tends to the stability end of the spectrum whenever it encounters this tradeoff. For example, Citizens Bank uses a tool called Nexus to maintain their own repositories of docker images, npm packages, helm charts, and other types of package managers. Working within this environment showed us real world examples of how risk can be mitigated within an organization by preferring stable software over bleeding edge functionality.

9.2.2 Enterprise Git Management and CI/CD

For individual projects, a single git repository and an accompanying live environment can be enough for a software team’s entire codebase. However, large organizations like Citizens Bank that oversee hundreds or even thousands of projects need a more robust way of organizing projects and deployments. Citizens Bank uses an enterprise BitBucket instance that is hosted within their network for their entire codebase. BitBucket repositories are assigned to pods, and can be linked to other internal tools like Jira. BitBucket also works seamlessly with Jenkins, which is a CI/CD pipeline; repositories containing deployable services simply need to include a configured Jenkinsfile in the root directory, and the Jenkins pipeline is automatically triggered on each commit. BitBucket and Jenkins were therefore integral to the development of our project, which was a good learning experience since similar setups are commonplace in almost all organizations that develop software.

9.2.3 Narrowing Down Problems

In developing OAuth, the team faced a serious impediment lasting several weeks when deploying the app to OpenShift. While attempting to set up NextAuth to send an authorization request to the endpoint located within the Citizens Bank internal network, the application was throwing an SSL “wrong version” error. Knowing that other deployments could make network requests, the team assumed that the problem lay within NextAuth or Next.js and an incorrect configuration somewhere, and dug into this with extensive research. Unfortunately, the team made no headway over the next couple days, and connected with the sponsors to receive help from Citizens Bank. Numerous different solutions were proposed by different groups within Citizens Bank, and despite the team having tried all of them, the problem was still not resolved. This included providing a repository of working code to check the team’s code against, which contained no meaningful differences in that regard. After some advice from the sponsor about providing the simplest version of the code that had the problem to these groups, the team took a step back and tried to narrow down the problem as much as possible. The team created a basic Express server that solely made network fetch requests to some endpoints within the Citizens Bank network when run. Interestingly, these requests all failed with the same error, proving that the problem was not with NextAuth or our application, but rather related to the handling of the deployment by Citizens Bank. We reached out to the Platform as a Service (PaaS) team to solve this, but they were still unable to resolve the issue. With help from one of our mentors, we redeployed our app to a separate environment after he noticed that some applications in the development environment could make network requests, while some could not. After deploying to the separate environment, our issue was fully resolved and we were able to get past this roadblock. The team quickly learned that it is extremely important to narrow down the problem as quickly as possible to ensure that when asking people for help, they are able to focus solely on the problem and how to solve it instead of being overwhelmed with irrelevant code. This is especially true because the people helping have less of an understanding of the application itself, but are more proficient in solving any given problem.

9.2.4 Operating in a Corporate Environment

Much of our technical and non-technical learnings from this MQP can be generalized as simply adjusting to the work environment of a major corporation. This applies to technical development - i.e., the use of enterprise-grade tools like Jenkins, BitBucket, and OpenShift - and even more so to the social dynamics of the company. Because we are generally all used to working in small teams on academic projects, being a small part of a larger dynamic was perhaps the “root” adjustment that our team had to make, in the sense that all the previously discussed learnings stem from this point. The developers on the team had to adjust how they communicated with the numerous different teams they contacted to ensure that they got the responses they needed. With so many different teams, it was easy for a request to “fall through the cracks", so the team quickly learned to follow up whenever necessary.

10. Future Work

As Citizens Bank continues to refine its implementation of OAuth and the FDX APIs, more work could be done on our app to reflect these changes. The clear first avenue of future work would be to completely connect Citizen Bank's OAuth to the FDX APIs. By the end of our project, we had two versions of OAuth for different scenarios. One version was aimed at using the real version, even though it did not enable access to the FDX APIs by the time our project was over. The other version used a mocked OAuth server so that the app could get test account information to continue development even though OAuth was not complete. After Citizens Bank connects OAuth and FDX on their end — so that tokens generated by their OAuth would work as authentication in their FDX APIs — as a result of the mutual development between the real and mock OAuth servers, it would be a relatively quick change to fully embrace it, though no doubt there would be small configuration issues that would need resolving in the app.

Another avenue of future work would rely on Citizens Bank as well as they continue to implement different areas of the FDX and OAuth specifications. For example, the userinfo endpoint listed in the wellKnown currently does not return information about the user as it is not implemented yet. Therefore, once Citizens Bank implements this endpoint, our application could make a request to it to get data about the user such as the username, the full name of the user, and their email, so the application can display it in the user interface.

There are also different areas in the FDX specification that Citizens Bank is currently not developing. If Citizens Bank did eventually end up implementing these, then the application could reflect these changes by adding new tabs to the application that show the new data. Specifically, Citizens Bank only implements the /accounts and /accounts/{accountId}/transactions endpoints. Everything else from the FDX spec is not implemented, such as endpoints for rewards programs, credit card benefits, account statements, and even images of deposited checks. As these API features are implemented, our work could serve as a basis for future integration testing, since all these endpoints are authenticated by the same JWT already obtained from our OAuth flow.

11. Conclusion

For the project team, the goal of this project had two important steps. First was to develop a full-stack PFM application for open banking through the use of APIs. The team completed this as far as possible. The team's second goal was to make a clear distinction regarding the proof of concept of Citizens Bank's readiness to enter the open banking market through the use of APIs. Due to the team not achieving the result we had planned for, the outcome determined is that Citizens Bank is not ready to enter this field at this time. The application was coded as far as limitations would allow and is set up for seamless integration once the necessary technology is available at Citizens Bank.

The company has plans to continue the development of this technology. The research team completed its time with Citizens Bank as the organization began to expand its bandwidth and develop an action plan to move forward and make progress with. Despite the short timeline, the WPI team made valuable findings for Citizens Bank which will aid in the company's future technological advancement.

References

"About-Fdx - Our Mission." *Home*, https://financialdataexchange.org/FDX/FDX/About/About-FDX.aspx?hkey=dffb9a93-fc7d-4f65-840c-f2cfbe7fe8a6.

"About Postman." Postman API Platform, https://www.postman.com/company/about-postman/.

About Our Company | Citizens Financial Group, Inc.. Citizens Bank, 2022, https://www.citizensbank.com/about-us/our-company/overview.aspx.

Atlassian. "Bitbucket Overview." *Bitbucket*, https://bitbucket.org/product/guides/getting-started/overview.

Atlassian. "Jira: Issue & Project Tracking Software." *Atlassian*, https://www.atlassian.com/software/jira.

"Big-IP Services." *F5*, https://www.f5.com/products/big-ip-services.

Calvello, Mara. "Agile Meetings: A Comprehensive Guide for Leaders." *Fellow.app*, 6 Feb. 2021, https://fellow.app/blog/meetings/agile-meetings-comprehensive-guide-for-leaders/.

Catania, Patrick J., and Nancy Keefer. "The Marketplace." *Amazon*, Board of Trade, 1987, https://aws.amazon.com/marketplace/seller-profile?id=74d946f0-fa54-4d9f-99e8-f F3bd8eb2745.

Chai, Wesley, et al. "What Is Paas? Platform as a Service Definition and Guide." *SearchCloudComputing*, TechTarget, 7 Feb. 2022, https://www.techtarget.com/searchcloudcomputing/definition/Platform-as-a-Servic e-Paas.

Finextra. "What the Hell Is Banking as a Service? and What Is It Not?" *Finextra Research*, 30 Mar. 2021, https://www.finextra.com/blogposting/20099/what-the-hell-is-banking-as-a-service-and-what-is-it-not.

Flindt, Mikkel. "2 Legged vs 3 Legged Oauth." *Lekkimworld.com*, 12 Mar. 2020, https://lekkimworld.com/2020/03/12/2-legged-vs-3-legged-oauth/.

"JavaScript with Syntax for Types." *What Is TypeScript?*, TypeScript, https://www.typescriptlang.org/.

"Jenkins." *Jenkins*, https://www.jenkins.io/.

"JSON Web Tokens Introduction." *JWT.io*, Okta, https://jwt.io/introduction.

"Kubernetes Overview." *Kubernetes*, https://kubernetes.io/docs/concepts/overview/.

Microsoft. "Visual Studio Code." *Microsoft*, https://code.visualstudio.com/.

"OpenID Connect." *OpenID*, https://openid.net/connect/.

"Open Banking and Financial Apis: How to Integrate Your Company with the Digital Financial Ecosystem." *AltexSoft*, 20 Nov. 2019, https://www.altexsoft.com/blog/engineering/open-banking-and-financial-apis-how-to-integrate-your-company-into-digital-financial-ecosystem/.

Open Banking Technical Solution Guide. ForgeRock, https://www.forgerock.com/industries/financial-services/open-banking/UK-Spec.

"Our History." – *Citizens Bank*, Citizens Financial Group, Inc., https://investor.citizensbank.com/about-us/our-company/our-history.aspx?source=post_page.

"Red Hat OpenShift Makes Container Orchestration Easier." *Makes Container Orchestration Easier*, https://www.redhat.com/en/technologies/cloud-computing/openshift.

Sunil, Usha. "Project Manager vs Product Owner: Key Differences." *KnowledgeHut*, 8 June 2015, https://www.knowledgehut.com/blog/agile/project-manager-vs-product-owner.

Turol, Sophia. "Technical Challenges to Address When Integrating Banking as a Service." *Altoros*, 26 Aug. 2021, https://www.altoros.com/blog/technical-challenges-to-address-when-integrating-banking-as-a-service/.

"TypeScript Introduction." *Typescript Introduction*, W3Schools, https://www.w3schools.com/typescript/typescript_intro.php.

"TypeScript - Overview." *Learn TypeScript*, Tutorials Point, https://www.tutorialspoint.com/typescript/typescript_overview.htm.

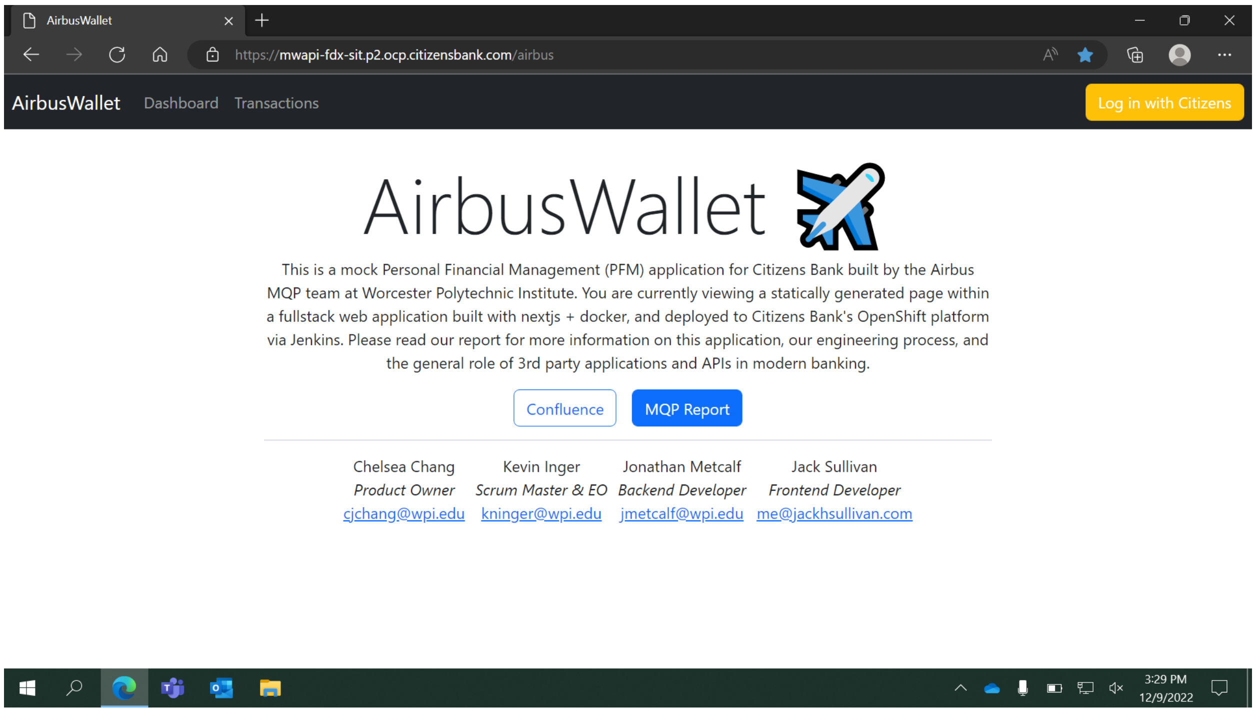
What Is Agile Methodology? Benefits of Using Agile. Nvisia, https://www.nvisia.com/insights/agile-methodology.

"What Is an API?" *Red Hat*, 2 June 2022, https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces.

Appendix A

Application Screenshots

Landing Page

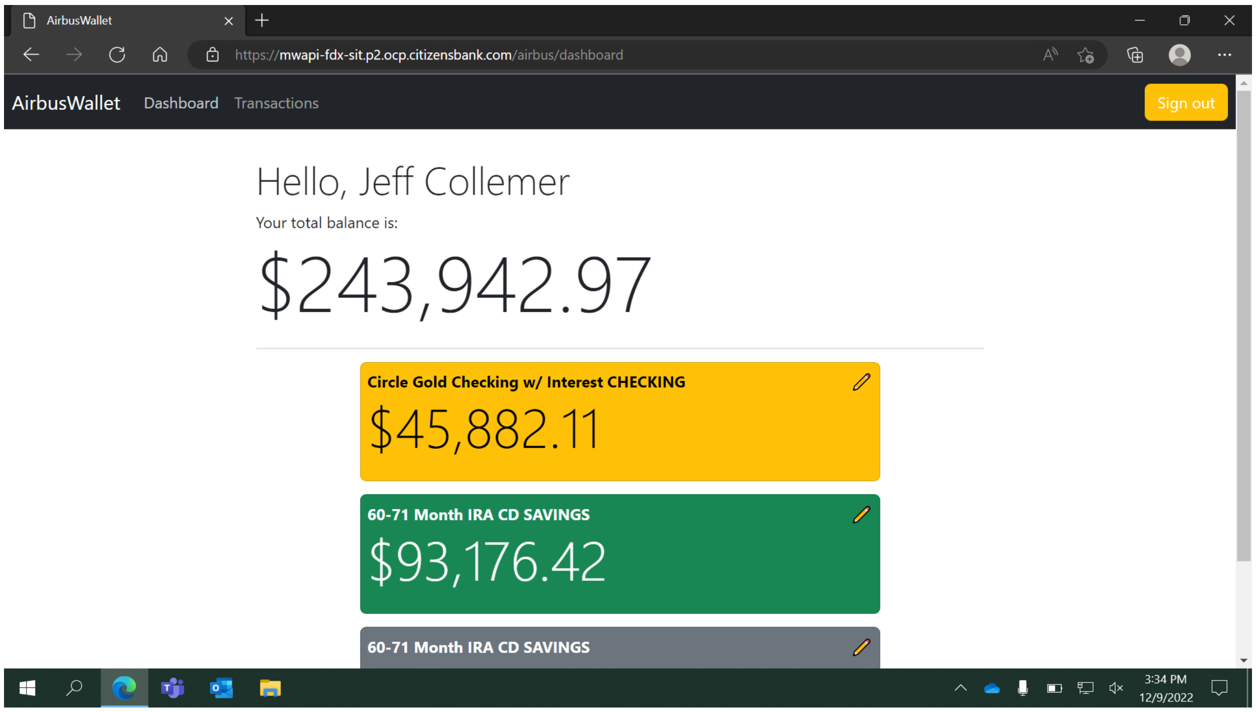


AirbusWallet

This is a mock Personal Financial Management (PFM) application for Citizens Bank built by the Airbus MQP team at Worcester Polytechnic Institute. You are currently viewing a statically generated page within a fullstack web application built with nextjs + docker, and deployed to Citizens Bank's OpenShift platform via Jenkins. Please read our report for more information on this application, our engineering process, and the general role of 3rd party applications and APIs in modern banking.

Chelsea Chang	Kevin Inger	Jonathan Metcalf	Jack Sullivan
Product Owner	*Scrum Master & EO*	*Backend Developer*	*Frontend Developer*
cjchang@wpi.edu	kninger@wpi.edu	jmetcalf@wpi.edu	me@jackhsullivan.com

Dashboard Page



AirbusWallet

https://mwapi-fdx-sit.p2.ocp.citizensbank.com/airbus/dashboard

AirbusWallet Dashboard Transactions Sign out

Hello, Jeff Collemer

Your total balance is:

$243,942.97

Circle Gold Checking w/ Interest CHECKING

$45,882.11

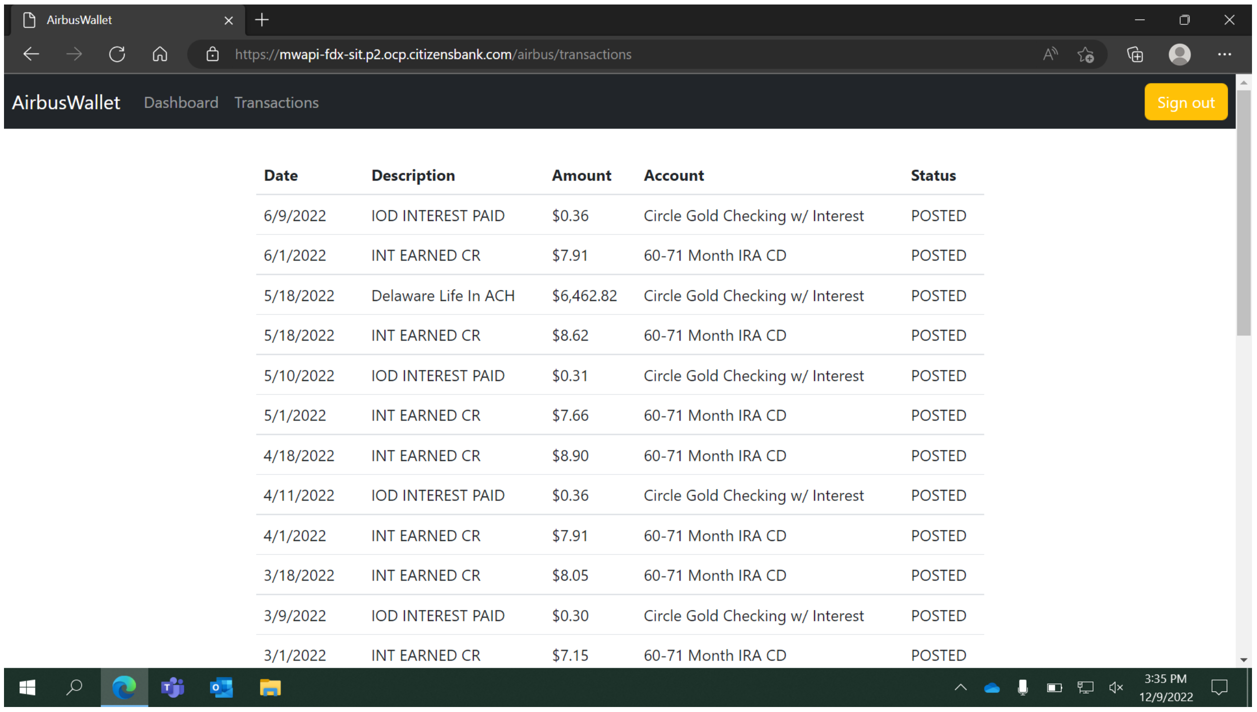
60-71 Month IRA CD SAVINGS

$93,176.42

60-71 Month IRA CD SAVINGS

3:34 PM 12/9/2022

Transactions Page



AirbusWallet

https://mwapi-fdx-sit.p2.ocp.citizensbank.com/airbus/transactions

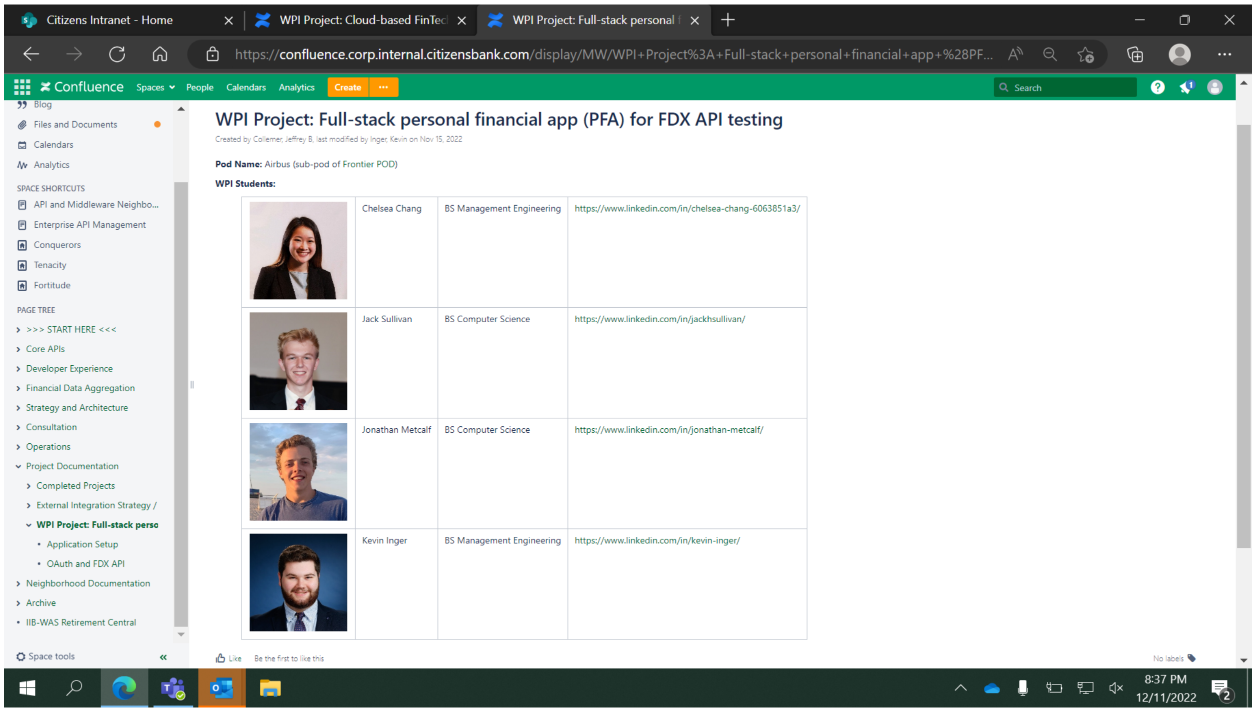
AirbusWallet Dashboard Transactions Sign out

Date	Description	Amount	Account	Status
6/9/2022	IOD INTEREST PAID	$0.36	Circle Gold Checking w/ Interest	POSTED
6/1/2022	INT EARNED CR	$7.91	60-71 Month IRA CD	POSTED
5/18/2022	Delaware Life In ACH	$6,462.82	Circle Gold Checking w/ Interest	POSTED
5/18/2022	INT EARNED CR	$8.62	60-71 Month IRA CD	POSTED
5/10/2022	IOD INTEREST PAID	$0.31	Circle Gold Checking w/ Interest	POSTED
5/1/2022	INT EARNED CR	$7.66	60-71 Month IRA CD	POSTED
4/18/2022	INT EARNED CR	$8.90	60-71 Month IRA CD	POSTED
4/11/2022	IOD INTEREST PAID	$0.36	Circle Gold Checking w/ Interest	POSTED
4/1/2022	INT EARNED CR	$7.91	60-71 Month IRA CD	POSTED
3/18/2022	INT EARNED CR	$8.05	60-71 Month IRA CD	POSTED
3/9/2022	IOD INTEREST PAID	$0.30	Circle Gold Checking w/ Interest	POSTED
3/1/2022	INT EARNED CR	$7.15	60-71 Month IRA CD	POSTED

3:35 PM 12/9/2022

Appendix B

Confluence Screenshots



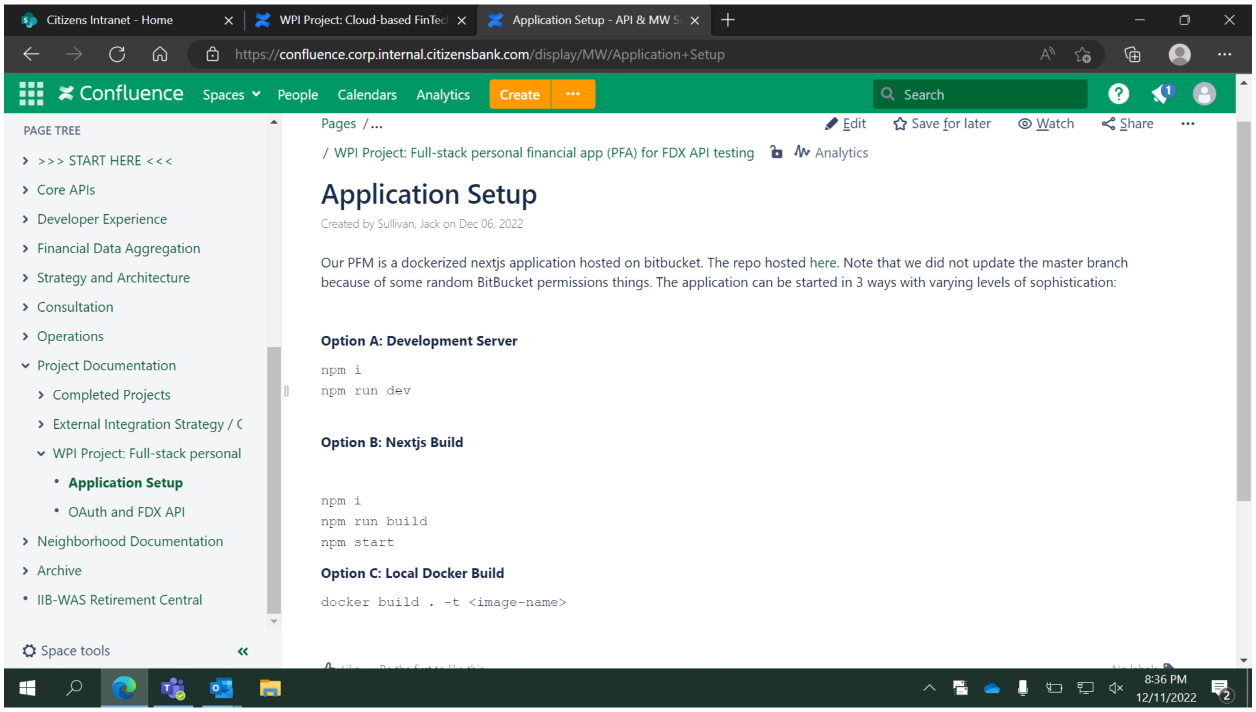
WPI Project: Full-stack personal financial app (PFA) for FDX API testing

Created by Collemer, Jeffrey B, last modified by Inger, Kevin on Nov 15, 2022

Pod Name: Airbus (sub-pod of Frontier POD)

WPI Students:

	Chelsea Chang	BS Management Engineering	https://www.linkedin.com/in/chelsea-chang-6063851a3/
	Jack Sullivan	BS Computer Science	https://www.linkedin.com/in/jackhsullivan/
	Jonathan Metcalf	BS Computer Science	https://www.linkedin.com/in/jonathan-metcalf/
	Kevin Inger	BS Management Engineering	https://www.linkedin.com/in/kevin-inger/



Application Setup

Created by Sullivan, Jack on Dec 06, 2022

Our PFM is a dockerized nextjs application hosted on bitbucket. The repo hosted here. Note that we did not update the master branch because of some random BitBucket permissions things. The application can be started in 3 ways with varying levels of sophistication:

Option A: Development Server

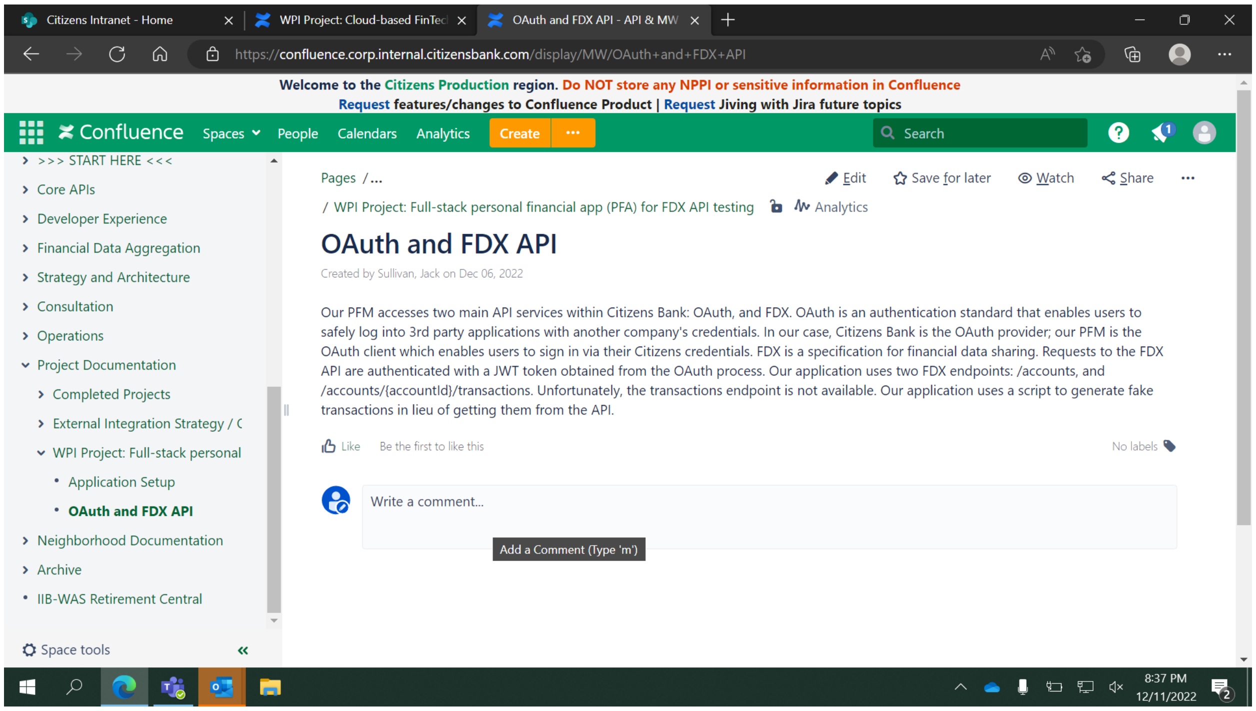
```
npm i
npm run dev
```

Option B: Nextjs Build

```
npm i
npm run build
npm start
```

Option C: Local Docker Build

```
docker build . -t <image-name>
```


Citizens Intranet - Home
WPI Project: Cloud-based FinTech
OAuth and FDX API - API & MW
https://confluence.corp.internal.citizensbank.com/display/MW/OAuth+and+FDX+API
Welcome to the Citizens Production region. Do NOT store any NPPI or sensitive information in Confluence
Request features/changes to Confluence Product | Request Jiving with Jira future topics
Confluence
Spaces
People
Calendars
Analytics
Create
Search
>>> START HERE <<<
Core APIs
Developer Experience
Financial Data Aggregation
Strategy and Architecture
Consultation
Operations
Project Documentation
Completed Projects
External Integration Strategy / C
WPI Project: Full-stack personal
Application Setup
OAuth and FDX API
Neighborhood Documentation
Archive
IIB-WAS Retirement Central
Space tools
Pages / ...
Edit
Save for later
Watch
Share
/ WPI Project: Full-stack personal financial app (PFA) for FDX API testing
Analytics
Like
Be the first to like this
No labels
Write a comment...
Add a Comment (Type 'm')
8:37 PM
12/11/2022

OAuth and FDX API

Created by Sullivan, Jack on Dec 06, 2022

Our PFM accesses two main API services within Citizens Bank: OAuth, and FDX. OAuth is an authentication standard that enables users to safely log into 3rd party applications with another company's credentials. In our case, Citizens Bank is the OAuth provider; our PFM is the OAuth client which enables users to sign in via their Citizens credentials. FDX is a specification for financial data sharing. Requests to the FDX API are authenticated with a JWT token obtained from the OAuth process. Our application uses two FDX endpoints: /accounts, and /accounts/{accountId}/transactions. Unfortunately, the transactions endpoint is not available. Our application uses a script to generate fake transactions in lieu of getting them from the API.