Virtual Wallet: Final Report

(CSCE 606) Software Engineering

Texas A&M University

Spring 2023

Professor
Dr. Philip Ritchey

Virtual Wallet team
Chunkai Fu, Jackson Sanders, Keerthana Polkampally, Sambhav Khurana, Swetha Reddy

Client
Manik Taneja

Table Of Contents

Summary of the project as implemented

Main customer need

After several team hurdles with our customer, we identified the core needs of our customer. We needed to develop a personal finance management system that helps our customers manage their funds. Here are the expected functionalities of the app:

- The app should enable user registration and sign-in with ease.
- The app should enable users to add their credits to their profile.
- Customers should be able to add different currencies to their wallet so that they can manage their multi-currency funds flexibly.
- Our customer also wants to verify the identity of the registered users and only allow the app access to those who have provided valid identification details.
- Our customer also demands that the registered user should accept our terms of service before having access to their service.
- Our customer also wants to enable the users to change their password when they forgot it.
- It is also required that one user should be able to send money to another registered user.
- To manage the multi-currency interface simply, our customer also demands that we should show only one instance for each currency.
- To enable the user to use our app conveniently, the user should also be enabled to scan the QR code to pay.
- It is also required that the user should be able to claim the balance of all currencies at any time they want.
- The user should also be enabled to add and delete cards on file but they must have one card on file to use the multi-currency feature and make transactions.

How the application meets it (include who stakeholders are)

The stakeholders of this project are as follows: Customer: Manik Taneja; Project supervisor: Philip Ritchey, Teaching Assistant: S.M. Farabi Mahmud, Team members: Jackson Sanders , Keerthana Polkampally Sambhav Khurana, Swetha Reddy Sangem and Chunkai Fu.

To address the needs of our customer, we have adopted the Ruby on Rails framework and developed a virtual wallet app called “PayFast”. It allows the user to register accounts and create a virtual wallet to hold all their multi-currency balances and different types of cards. The users of the app can use the cash in their app to pay for commodities and also send cash to other users. The users can also add their credit cards to their wallet, pay with a QR code or transfer money from their credit card to any currency they want in the app. All conversions would be handled in the backend of our app. With the continuous efforts from our team members, we were able to deliver all the features to meet the customers needs mentioned above. The user simply needs to type the link to our website in their browser of choice, register an account and get started with the online finance management journey!

User Stories

For iteration 0, our team utilized a Google form to assign points to our initial set of user stories.

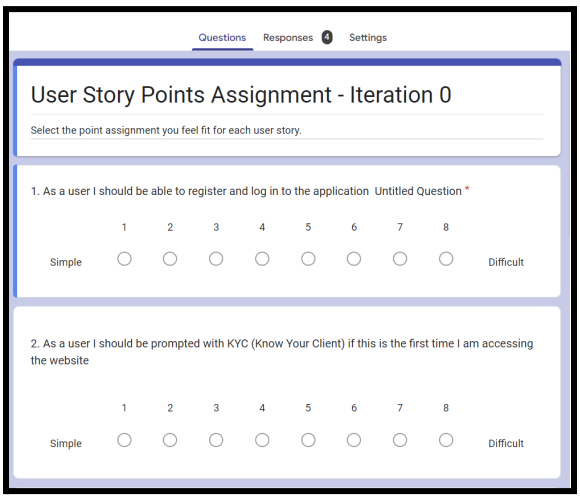


Figure 1. Iteration 0 User Story Points Assignment (Google Form snippet)

This method was successful in assigning points to our initial set of user stories, however, for future iterations we decided to simply vote on user stories at group meetings. Shifting this process from a web-based task to an in-person discussion made for a much more effective means of rapidly sharing valuable information.

Following the generation of the first set of user stories and their points values, a storyboard for the virtual wallet was created on Figma. (A larger, more readable version is in Appendix 1B)

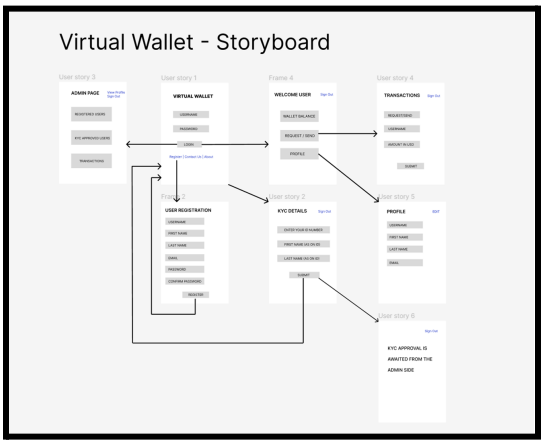


Figure 2. Figma Storyboard for Virtual Wallet (planned approach)

The full catalogue of User Stories can be found in Appendix 1A.

Team Roles

- **Scrum Master**
  - Chunkai Fu
- **Product Owner & Version Control Manager**
  - Sambhav Khurana
- **Implementation**
  - Keerthana Polkampally & Swetha Reddy
  - Chunkai Fu, Sambhav Khurana, & Jackson Sanders
- **Test Writing**
  - Swetha Reddy & Jackson Sanders & Keerthana Polkampally
- **Systems Modeling**
  - Jackson Sanders
- **Client**
  - Manik Taneja

Changes as Project Progressed

As our software engineering project began, our team was initially uncertain about how to effectively utilize agile development. However, as we learned and adapted to the methodology, we assigned key roles to ensure a streamlined workflow. Chunkai Fu took on the responsibilities of the Scrum Master, guiding the team through the agile process and managing the Scrum events. Sambhav Khurana filled the dual roles of Product Owner and Version Control Manager, prioritizing the product backlog and maintaining the integrity of our codebase. Implementation was carried out by Keerthana Polkampally and Chunkai Fu, who led the effort to develop and refine features. The routine workflow often saw Keerthana & Swetha pairing up to develop features in parallel with the trio of Chunkai, Sambhav, and Jackson. Swetha Reddy, Keerthana Polkampally and Jackson Sanders focused on Test Writing, ensuring the robustness of our software through comprehensive testing. Lastly, Jackson Sanders fronted Systems Modeling, consolidating the thoughts of the customer and the team as to how best to structure the wallet rails app. As a result of this role distribution, our team successfully transitioned into agile development and delivered the product.

Customer Meetings

The meeting notes from huddles with the customer can be found in Appendix 1C. Meeting notes are tagged with phases so that they can be traceable to the user stories for each phase (Appendix 1A).

Explain your BDD/TDD process

Our project had been attempted in a previous semester, however, the client requested that we only refer to the legacy code, not use it. The legacy project was not successfully deployed so our team opted not to refer to the code in the first place. With a fresh start, the project became a proving ground for learning and living the BDD/TDD processes.

Meetings with our customer elicited user stories which were then converted into cucumber & rspec tests. By employing TDD principles, we learned to write comprehensive test cases before implementing features, allowing for the identification of issues early on. None of our team members had worked with the TDD process before, so there was a large learning curve that went hand-in-hand with the process of delivering a deployed, functional product.

On top of TDD, our team also learned to utilize BDD techniques, which focused on the desired behaviour of the application from the perspective of the user. The definition of scenarios in a natural language format paired with the generation of storyboards & lofi UI mockups fostered a clear understanding of the user's requirements and expectations for the team members. Communication between team members increased drastically despite the fact that we found ourselves aligning with more rigidly defined roles as the semester progressed. As a result, our application evolved to meet user needs more efficiently, and we successfully implemented and iterated on features throughout the five phases of the project. The balanced combination of TDD and BDD enabled us to deliver a robust and user-centric application, fulfilling our customer's needs while maintaining a high level of quality and performance.

Configuration Management Approach

- As a group of 5, we utilized Git as our Configuration Management tool for our project. We established 8 branches for different aspects of the project to facilitate development and ensure proper organization.
- With a consistent approach, we made weekly changes throughout the semester with no unexpected spikes in the project.
- Before each iteration, we divided the work and assigned each member to a specific branch to work independently. To ensure consistency in our coding style and practices, we established a set of coding guidelines and standards that everyone followed.
- After the completion of the work on the branch, we thoroughly tested the code before merging it into the main branch.
- We employed Git's tagging feature to mark important milestones and releases of the project.
- We also made sure to keep our repository organized and clean, deleting any branches that were no longer needed and keeping the main branch up-to-date with the latest changes. This approach ensures that our final code is efficient, stable, and can be readily deployed for customer use.

- In summary, our Configuration Management approach ensured that our Git repository was well-managed, and our collaborative efforts resulted in an efficient and functional product.

Issues

- **Ruby on Rails/SWE Learning Curve** - None of the members of our group had built an application using R.o.R. before this project. Only two members had some experience with web stacks and the disparity in version control knowledge was large heading into iteration 1. While at times it felt our knowledge of Ruby was trailing behind what was required to deliver an acceptable product, the team was ultimately able to become versed enough in the MVC framework to produce functional SaaS features.
- **Devise Gem** - Early in the project, the team came across a gem called the Devise Gem and its appealing authentication functionality. However, implementing such a gem so early in our development led the team to realize that handling the entire MVC structure with a purist approach would make our application much more scalable. Thus, we decided to take the time to start from scratch (several times) and get a much better hold on the convention necessary to meet the needs of our customer and *only* the needs of our customer.
- **Merging** - As is fundamental in the Agile lifecycle, our team encountered many points which involved the merging of features into a central repository. This repo would then be the one to be deployed on Heroku. Most memorably, when merging a version with reconstructed forms of KYC & SuperUser features, we faced many difficulties in getting the functionalities to line up with those of the preexisting wallet & balance features.
- **Admin feature** - Since we couldn't just introduce admin as a user with a boolean, we used the active admin feature but without any Devise logic. We've coded all the MVC for admin and had to sync that with the active admin. It was quite a task for the user variables to be used in admin and changes to be made to them in the database.
- **Deployment** - We had a lot of issues while deploying our application to Heroku. The app database pg issue was one of the first issues we encountered. Also, issue while deploying sign-in with the Google option and also while updating the KYC status by the admin. The app was crashing whenever there was an ambiguity in the routes or a certain part of the code is a predefined part. We will have to specifically write those parts for the deployment to not crash.
- **Parallel Coding** - After the registration and login, there were KYC and admin parts of the application which is a prerequisite for the rest of the application, so we had to divide the entire application coding to be done in parallel and then combine it. This gave us issues during merging.
- **Database** - Figuring out which entities should be connected, how the tables have to be connected in the database using foreign keys and the effect of updation of entities on all the tables in the database has been quite challenging.
- **Testing** - Testing the application has been quite a strenuous task as there were a lot of dependencies between models and hence, building the factory models, and redirection links have been difficult.

- **Security issues** - Some key issues were regarding the admin access only with the link to the page, transactions without actual users being approved by the admin, passwords being encrypted and card, and KYC license number details being fully visible to the user. These security risks have been handled by using necessary gems in case of password encoding.
- **Secret key storage** - Initially all keys and secrets are stored locally which is open to security risks. We figured out how to store all sensitive information on the AWS secret management system and store the master key and secret in Heroku run time.
- **Issues with the host for mailer** - Setting up the account to send user email notifications for resetting their password encounters some issues with some outdated features of App Pass from Google. We improved the security of using a host account by enabling multi-factor authentication and also storing the secrets on AWS mentioned above.
- **Local and production** - We always need to be aware of the fact that they are actually two “versions” of our app: local and online. It may work locally, but we almost always need to tweak the config or set ENV variables to make it work in production.
- **Jewel of the crown “MVC”** - Understanding how the models, views, controllers and routes work together is unfortunately a steep learning curve for us, but we pulled through it by spending a big chunk of our time understanding it while at the same time trial and error to finally understand and get what we want.
- **“What does this folder/file do?”** Issue - There are many directories and files pre-generated while creating a Saas app with rails. We eventually decided to not use scaffolding and instead try to configure things only when we need it. We ditched many versions until finally, we made a clean app structure where everything is there for a reason, which we know.
- **There are more…** - Learning the principle of MVC is only a start for our journey for building Saas apps in Ruby on Rails. We always encounter issues that we wouldn't expect, which prompts us to learn and execute. Constant learning has always been the theme and will always be a theme in our software engineering journey.

Additional Gems/Tools

- *SimpleCov - Code Coverage Analysis Tool*
  - SimpleCov tracks code coverage for Ruby applications, helping developers identify areas needing more tests and maintain high-quality code.
- *UML Generator (Rails ERB) - Visual Representation of Rails Models*
  - This tool generates Unified Modeling Language (UML) diagrams for Rails applications, providing a visual representation of models and their relationships.
- *ActiveAdmin - Administration Framework*
  - ActiveAdmin is a Ruby on Rails framework that simplifies the creation of admin interfaces for managing application data.
- *Dotenv-rails - Environment Variable Management*
  - Dotenv-rails is a gem that manages environment variables, making it easy to securely store and access sensitive information in Rails applications.

- *OmniAuth - Authentication System*
    - OmniAuth is a flexible Ruby authentication system that supports multiple authentication strategies and simplifies user login implementation.
- *OmniAuth-Google-OAuth2 - Google Authentication Strategy*
    - This gem adds Google OAuth2 support to OmniAuth, allowing users to authenticate with their Google accounts.
- *OmniAuth-Rails-CSRF-Protection - CSRF Protection*
    - OmniAuth-Rails-CSRF-Protection is a gem that adds CSRF protection to OmniAuth authentication requests in Rails applications.
- *AWS SDK SecretsManager - Secure Data Management*
    - The aws-sdk-secretsmanager gem provides an interface to AWS Secrets Manager, enabling developers to securely manage and access sensitive data.
- *RQRCode - QR Code Generation*
    - RQRCode is a Ruby library for generating QR codes, allowing developers to create and customize QR codes for various purposes within their applications.
    - Used to generate a QR code when users access their cards

Heroku, Github Repo, & PivotalTracker

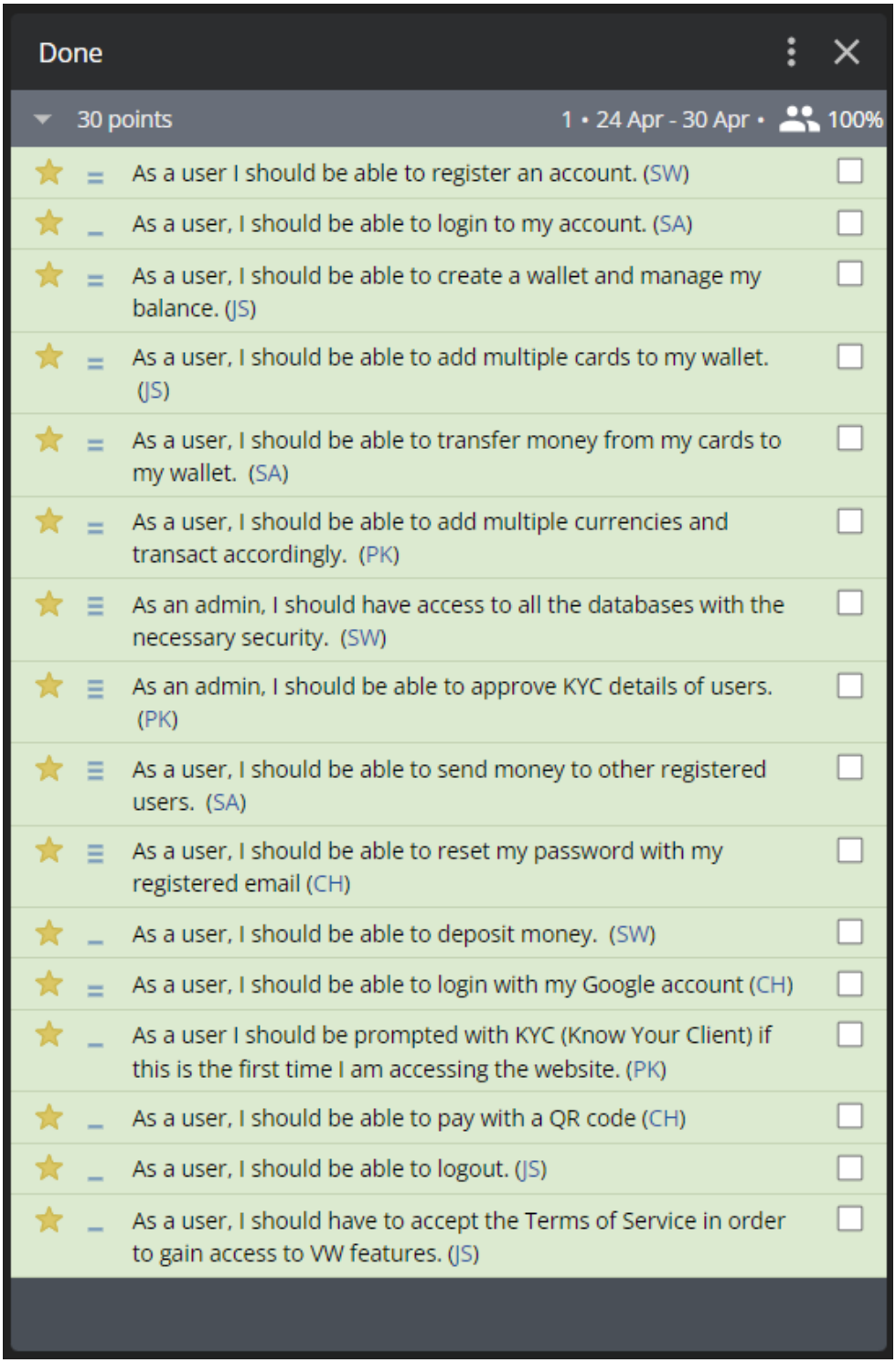
- https://bigwallet.herokuapp.com/home
- https://github.com/ChunkaiFu/wallet
- https://www.pivotaltracker.com/n/projects/2629557

Link to presentation/demo video

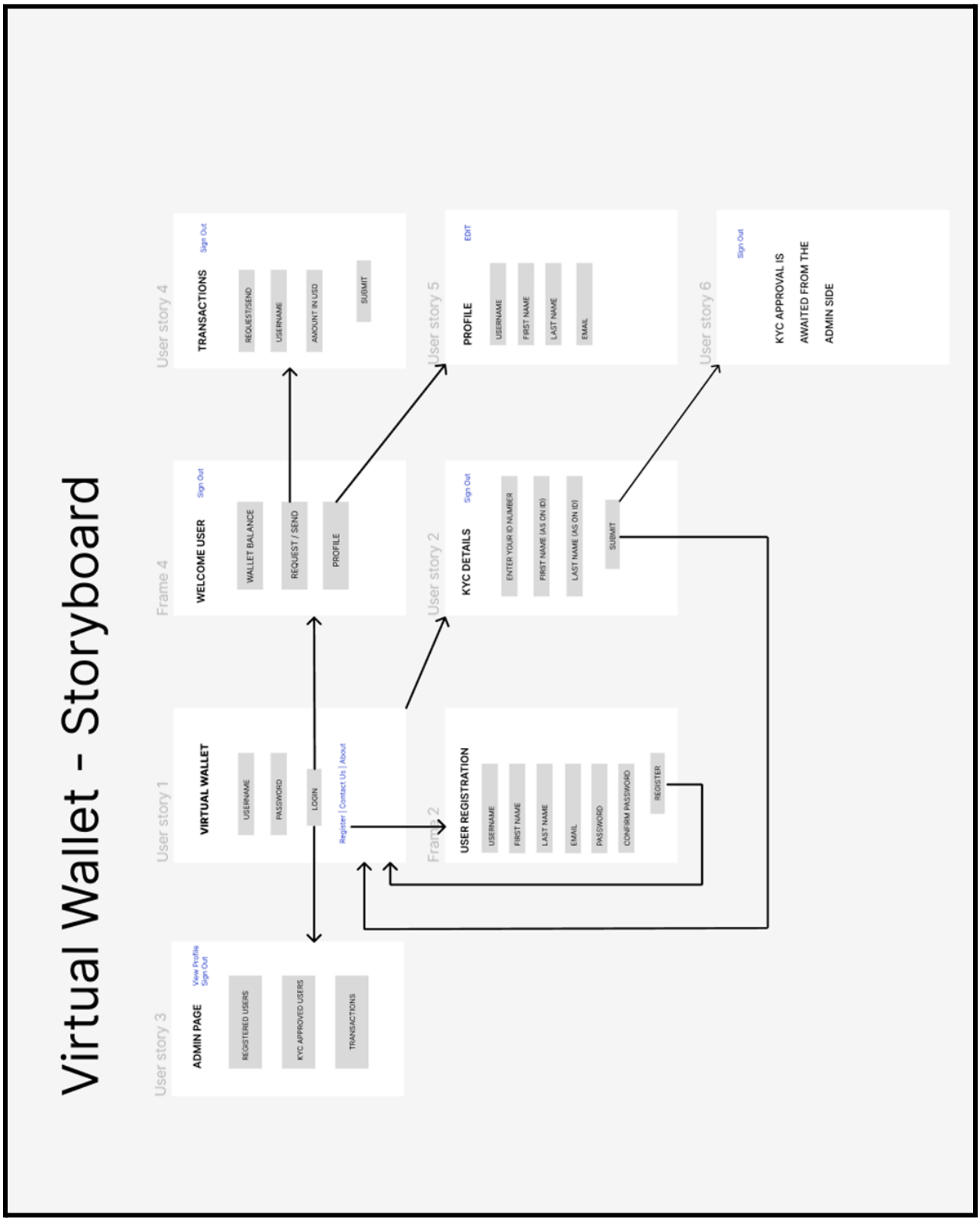
- https://www.youtube.com/watch?v=RcYXTyiOQ-U&ab_channel=ChunkaiFu

Appendix 1

1A - User Story Catalog with Assignments



1B - Virtual Wallet Storyboard



1C - Customer Meetings

Phase	Meeting Date	Notes
1	2/22/2023	Customer viewed our current progress and helped us with issues that we were facing in development.
1	2/24/2023	Demo of the application was given to the user. The customer was impressed with the work. All the implementations for the Login and Register are as expected by the Customer. The customer believes that we are on the right track and expects to see other user stories implemented for the next iteration.
2	3/9/2023	Customer viewed our current progress and helped us with issues that we were facing in development.
2	3/10/2023	Demo of the application with updated features was given to the user. Now, the program supports the login of a superuser through the addition of a new gem. The UML Use Case model was reviewed as well and future feature implementation has been given a more streamlined trajectory.
3	3/29/2023	Customer viewed our current progress and helped us with issues that we were facing in development. Issues with the KYC forms and deployment have been addressed. We have made the changes in the user stories and updated it in the pivotal tracker.
4	4/14/2023	Customer viewed our current progress and helped us with issues that we were facing in development. Most notably, KYC has been deployed along with a renovated MVC framework for wallet & card functionality. The customer said everything looked good but we need to focus on deployment.
5		No meetings for this phase, customer pleased with the progress.
Final	5/5/2023	Presented the final Heroku site to class, met with thunderous applause. Client very pleased with the final product.

1D - UML Diagram