

# Fin Lit Quest Final Report

FinLit Quest aims to teach financial literacy through an interactive web game. This project meets TeamUp's need to engage young learners by using trading, bartering and currency mechanics to teach economic concepts in a familiar and engaging format. The stakeholders, including TeamUp and Dr. Ritchey guided us and provided advice to ensure that the game not only captivates the interest of the players but also provides financial knowledge through practical application within the game's trading and economic system.

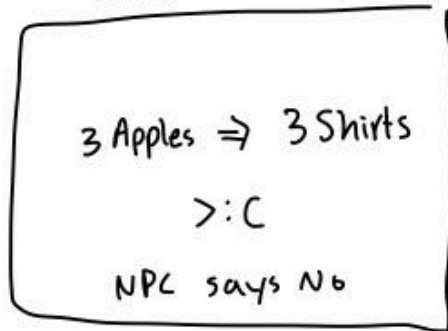
The initial project plan consisted of a simple game with limited interactivity focusing on basic trading between players and NPCs. However, as the project progressed across multiple sprints, we came up with significant improvements that were integrated based on feedback and reviews/retrospectives. The final product features a robust trading system where players can interact with NPCs with varying preferences and trade items using an evolving currency system. We introduced new elements like daily and weekly expenses, occupation-based NPC preferences, time effects/multipliers on gameplay and UI/UX improvements. To support these features, we developed an admin page, allowing users with admin access to make CRUD operations and effectively manage the game state and content. This evolution from the initial plan was driven by continuous stakeholder feedback, which highlighted the need for a more engaging and educationally rich environment to better meet the customer's educational goals.

## User Stories

### Sprint 1:

- Character System (2)
  - This was implemented, however it was broken up into many different pull requests over the course of its implementation. The extra complexity was largely a result of needing to create the entire infrastructure from scratch, as this was the first sprint.
- Deployment (1)
  - This was implemented, and went as expected.
- Inventory System (3)
  - This was implemented, but was split into several pull requests. Some pull requests were closed instead of merged as they were duplicates

## Bad Trade

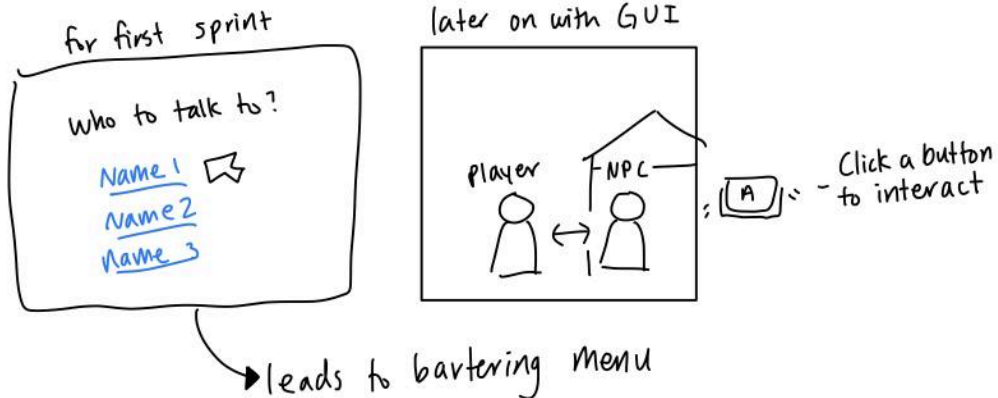


## Good Trade



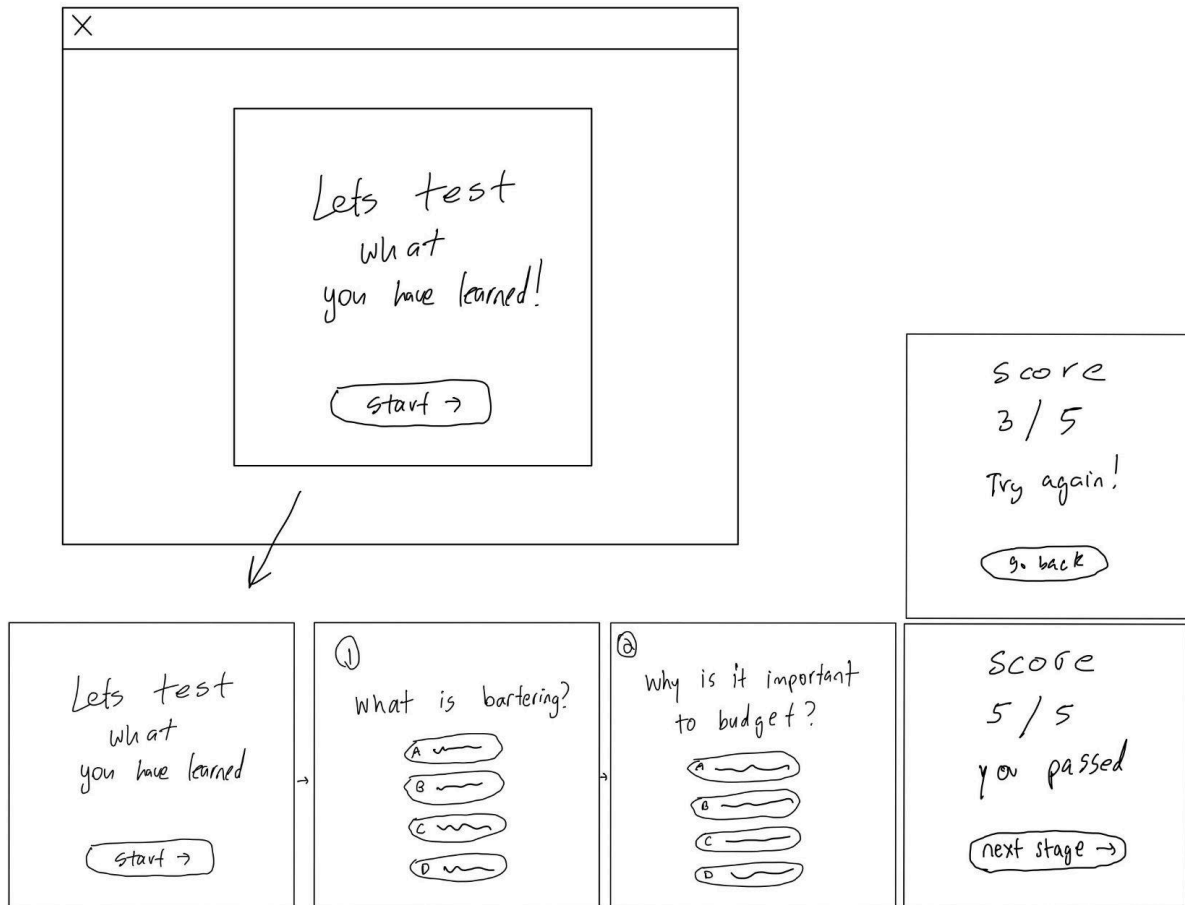
- Interaction System (2)

- This was implemented, but was split into a few pull requests. A couple different designs needed to be created for this:



- End of phase Quiz (2)

- This did not get implemented, as we did not have time in the first sprint. It was never assigned a cucumber test



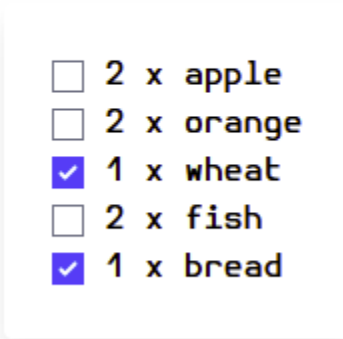
- Dialog System (3)
  - This was not implemented, and was not given a cucumber test. We decided to not add it into the first sprint and it was never re-added
- Budgeting (3)
  - This was not implemented, and was not given a cucumber test. We decided to not add it into the first sprint and it was never re-added
- Achievements (2)
  - This was not implemented, and was not given a cucumber test. We decided to not add it into the first sprint and it was never re-added
- Tips Feature (2)
  - This was not implemented, and was not given a cucumber test. We decided to not add it into the first sprint and it was never re-added

## Sprint 2:

- In world 2 onwards, the item values field should be available in player and nonplayer inventory page
  - This story was implemented, and went about as expected. It took significantly less time than expected to implement
- In the new and edit form for nonplayers, select items to accept and offer from dropdowns

- This story was implemented, but had to be split into two different pull requests for organization. It took very little time to implement
- In new/edit inventory entry, select item and character from list
  - This story was implemented, but had to be split into two different pull requests for organization. It took a significant amount of time to implement the tests
- Shopping list
  - This was implemented, and went as expected. Creating tests took much longer than the initial implementation

## Shopping lists



- ☐ 2 x apple
- ☐ 2 x orange
- ☒ 1 x wheat
- ☐ 2 x fish
- ☒ 1 x bread

- Advance to next world button
  - This was implemented, and went as expected.

Launch to next world if you satisfied all conditions!

Launch

Back

- Making a counter offer to an npc
  - This was implemented, and went as expected. However, the tests took a much longer time than the main implementation.

Score: ~

Barter:

item  give

#

↓

item  get

#

I give  ▾

Quantity I give

▴ ▾

I want  ▾

Quantity I want

▴ ▾

- Player accept an offer from a non-player
  - This was implemented, but took longer than expected. There were some weird issues with implementing the spec tests, and trading logic took a long time to create.

### Sprint 3:

- Occupational preference affects the prices they buy and sell for
  - This was implemented, and went as expected. The time taken was about the same as what was expected.

```

Trade with <name>
~~~~~
Bal: 0
Occupational Description: ~~~~~
~~~~~

Player inven:    <name> inven:
  ==              ==

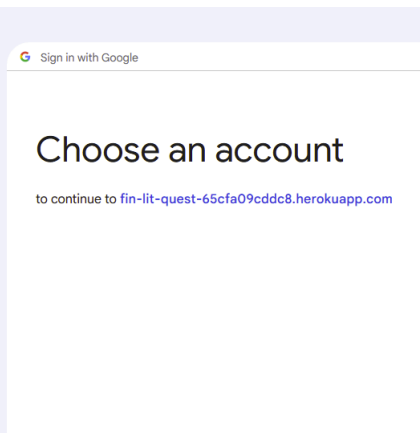
```

- Time advancement
  - This was implemented, and went as expected. The time taken was shorter than expected for the main implementation.
- Daily expenses
  - This was implemented, and went as expected. The time taken was slightly different from what was expected.
- Login and OmniAuth integration
  - This was implemented, and went as expected. The time taken for the login controller was much longer than expected.

## Welcome to FinLit Quest!

To begin exploring, please first sign in with Google

Sign in with Google



- Add descriptions for different professions
  - This was implemented, and went as expected. The time taken was much less than what was expected.
- Merge the offer and counteroffer pages
  - This was implemented, and went as expected. The time taken was much higher than what was expected, and there were many more parts to the cucumber story that usual.

- Weekly expenses
  - This was implemented, and went as expected. The testing was quick to implement, as it did not require any new features.

main menu



- Track the logged in player
  - This was implemented, and went as expected, however, the cucumber test needed to be modified to work properly.
- Add consequences for failure
  - This was implemented and went as expected. The time estimates were also quite accurate.

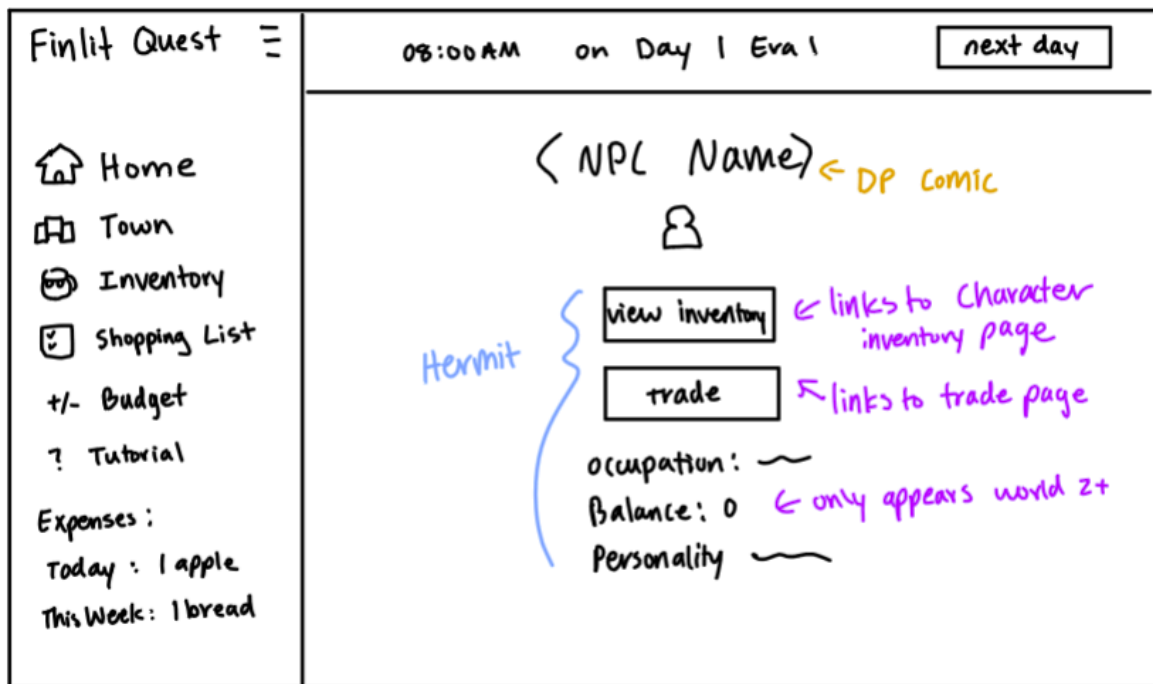
You do not have enough items to trade!

09:00 AM on Day 1, Era 1

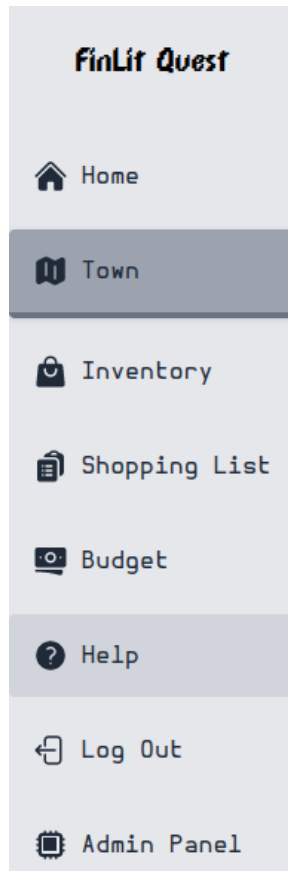
- Time Variance
  - This was implemented, but there were many issues that needed to be ironed out at the end of the implementation.

## Sprint 4:

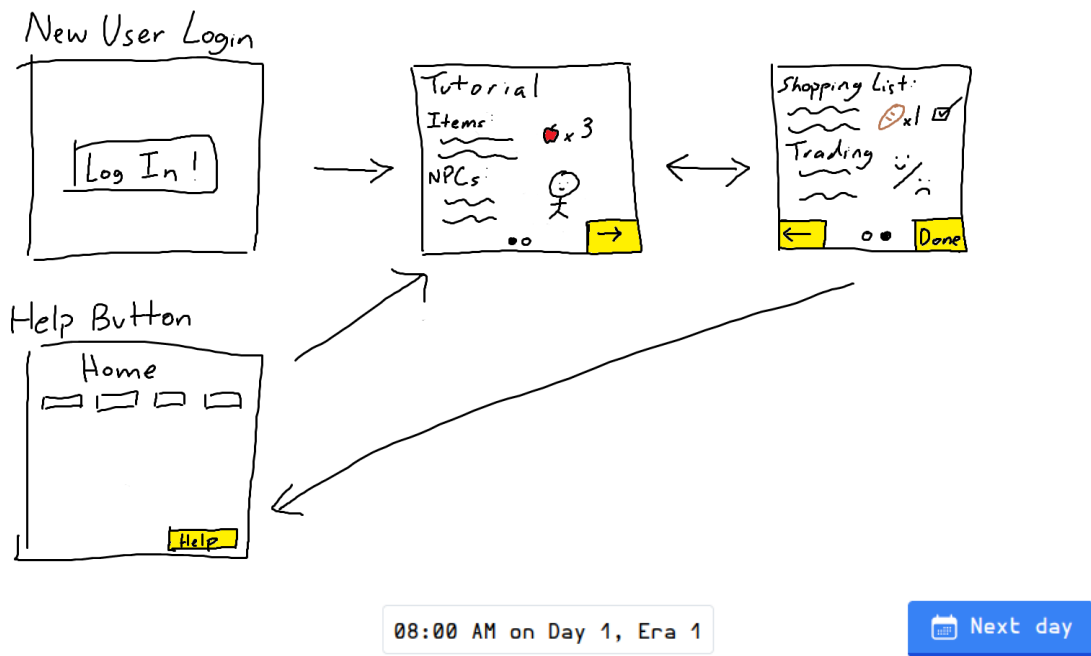
- Trade fails increase time of day
  - This was implemented, and took much less time than expected
- Adding a sidebar element
  - This was implemented, and took about the same amount of time as expected. It required a few UI designs to be added. Example:







- Add starter items when account is made
  - This was implemented, and went smoothly. It took about as long as was expected. The cucumber tests did need to be modified to work properly
- Get items every week after expenses (allowance)
  - This was implemented, and did not take very long to implement. Implementation went smoothly, but tests took a lot more time than the main implementation.
- Basic UI Rework
  - This was implemented, but was split into several pull requests for different pages. It took less time than expected, but still took a significant amount of time. There was a significant amount of UI design that went into this, with a full chore dedicated to creating the designs.
    - [Finlit\\_Quest\\_UI\\_Style\\_Guide.pdf](#)
- Add a tutorial
  - This was implemented, but was split into two different pull requests. There were many different steps, and even though it did not take as long as expected it still took a significant amount of time.



## Tutorial: Era 1

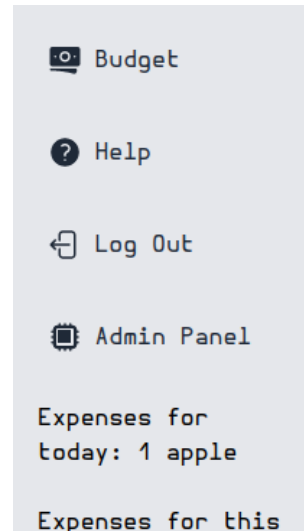
Welcome to FinLitQuest!

Back

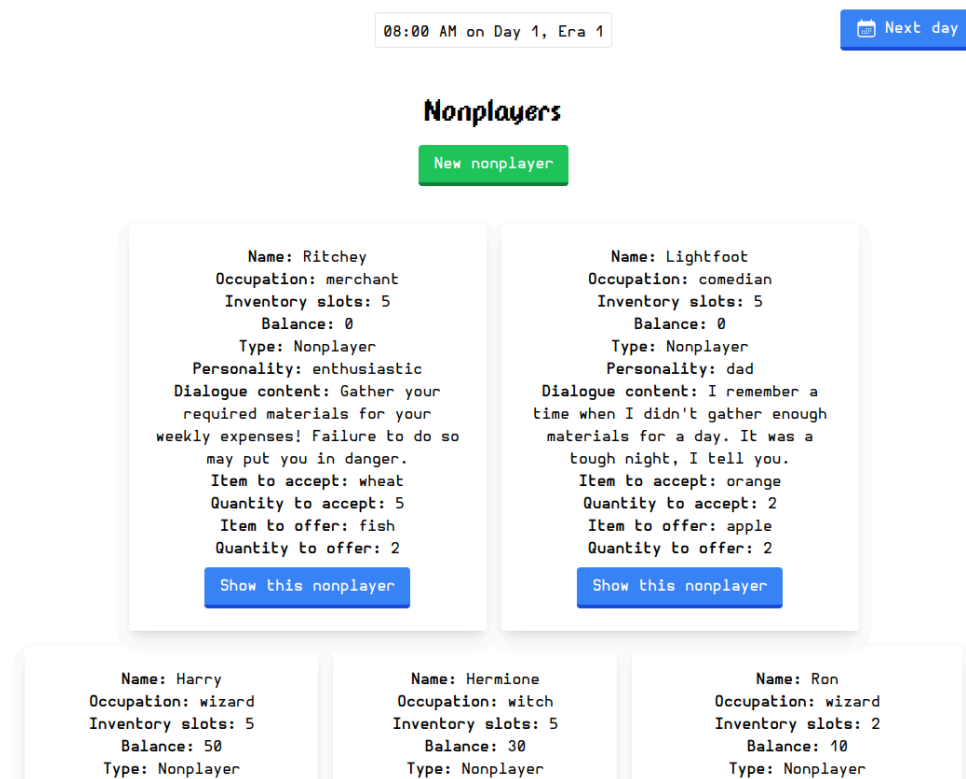
Next

### Sprint 5:

- Add a logout button
  - This was implemented, and went smoothly as expected.



- Update the UI for Admin pages
  - This was implemented, but the time estimates were not quite what was expected. Some were longer, some were shorter.



- Unlimited items for npcs
  - This was implemented, and took less time than expected
- Improve the game over page
  - This was implemented, and took about the same amount of time as expected.

# Game Over!

You can't afford to pay your expenses, and you have no time left!

Restart

- Add pixel art backgrounds
  - This was implemented, and took much less time than expected

08:00 AM on Day 1, Era 1

Next day

Welcome to Town 1!



- Add more content to the database
  - This was implemented, and went smoothly.
- Introduction of currency
  - This was implemented, but was somewhat complex. There were many more cucumber tests than usual, and a branch was created that was never merged.
- Apply UI improvements
  - This was implemented, but was split into two different pull requests.

## Roles

### Sprint 1:

**Product Owner:** Nick Anaya

**Scrum Master:** Emmie Teng

**Coding Monkey:** Justin Abraham, Victor Pan, Griffith Thomas, Jackson Stone, Stella Yang

### **Sprint 2:**

**Product Owner:** Stella Yang

**Scrum Master:** Jackson Stone

**Coding Monkey:** Justin Abraham, Victor Pan, Griffith Thomas, Nick Anaya, Emmie Teng

### **Sprint 3:**

**Product Owner:** Griffith Thomas

**Scrum Master:** Victor Pan

**Coding Monkey:** Justin Abraham, Stella Yang, Jackson Stone, Nick Anaya, Emmie Teng

### **Sprint 4:**

**Product Owner:** Jackson Stone

**Scrum Master:** Justin Abraham

**Coding Monkey:** Victor Pan, Stella Yang, Griffith Thomas, Nick Anaya, Emmie Teng

### **Sprint 5:**

**Product Owner:** Justin Abraham

**Scrum Master:** Nick Anaya

**Coding Monkey:** Victor Pan, Stella Yang, Griffith Thomas, Jackson Stone, Emmie Teng

## **Sprints**

### **Sprint 1: 8 points**

- Deployed the application
- Setup database for characters, inventory
- Update CRUD changes
- Prettify user game flow
- Added an interaction system between user and database

### **Sprint 2: 12 points**

- Added more tests that were missing as well as refactoring them
- Added dropdowns to CRUD so that selection of things is easier
- Implemented shopping list, make counter offer, accepting offers, and advancement to next world (Bartering System and Phase Advancement)
- Implemented different views based on what attributes the player and non player has.

### **Sprint 3: 22 points**

- Implemented Login and OmniAuth
- Implemented time advancement for user as well as daily and weekly expenses
- Reworked the trading mechanism
- Added NPC preferences and descriptions

- Cleaned up various features

#### Sprint 4: 12 points

- Fixed consequences for failing a trade
- Created a UI style guide and updated the User Interface
- Added a sidebar for ease of navigation
- Implemented starting items for new players
- Protected admin routes
- Implemented tutorial

#### Sprint 5: 16 points

- Prepared application for handoff
- Added logout features
- Updated interface for admin pages
- Improved various user interface pages
- Added more content to the database
- Sent game to testers for feedback
- Completed introduction to currency

#### Points Breakdown Table

Name		Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Total
Justin Abraham	Stories	2	1	2	0	1	6
	Points	4	2	4	0	2	12
Nick Anaya	Stories	0	1	3	1	1	6
	Points	0	3	8	1	2	14
Victor Pan	Stories	1	1	0	1	1	4
	Points	3	2	0	3	2	10
Jackson Stone	Stories	2	0	2	0	1	5
	Points	3	0	5	0	3	11
Griffith Thomas	Stories	1	2	0	2	1	6
	Points	2	2	0	3	2	9
Emmie Teng	Stories	2	2	2	2	2	10
	Points	5	3	4	5	4	17

Stella Yang	Stories	2	0	3	1	4	10
	Points	5	0	7	2	7	21

## Meetings

- **01/05/2024:**
  - First meeting with the customer to figure out the scope of the project, understand what the customer wanted and figure out an initial gameplan on how to proceed. We asked lots of game defining questions like: what type of application, what the UI should look like, how some of the core features should work, what the MVP and first phase should look like, etc. We didn't demo anything at this point, just scoped out the project and made the pivotal tracker.
- **01/10/2024:**
  - After discussing the plans with Dr. Ritchey we can ask some more questions about the project so we explained our pivot points in this meeting. Instead of having a sprite based top-down game running on an engine we have to work with something with a database and server, and we mentioned the idea of SimpleMMO, which the customer liked. We explained what we are working on, which is generating the project foundation and setting up the repo and databases. We said that the shift from the original envision is a "curveball". SO real demo in this meeting, still the initial meetings, we explained what we were planning on doing though, since the sprint was ending soon.
- **01/16/2024:**
  - In this meeting, we talked about the current state and future direction of our project, especially focusing on the first sprint's MVP and the next sprint. At the end of the demo, the client provided positive feedback about this first sprint/MVP, considering the development had certain constraints and challenges. While talking about the upcoming sprint, we planned to focus on improving the current game phase and implementing a functional bartering system. The client showed a strong interest for us to integrate motivational elements for payer trade, this could be achieved through narrative-driven incentives or gameplay objectives. The client also stated that while these additions/implementations are exciting, they represent a challenge and should be incrementally addressed throughout the next sprints. One of our next goals will be to implement a gradual transition from bartering to a monetary system, with a strategic approach to avoid limiting future development choices. We also asked about project management for bug tracking, and the client recommended we stick with Pivotal Tracker.

- **02/01/2024:**

- During this meeting, we discussed the state of our sprint, demonstrating the stories which were complete and explaining the final two stories which would be completed later that day and the next day. The client was very happy with our progress, but also pointed out some core gameplay features which we should consider incorporating into the next sprint. This primarily consisted of adding more mechanics beyond the shopping list into the game such as expenses to motivate the player to continue to try and increase their wealth. We discussed how we would implement this, including using NPCs as teaching mechanics to the player, allowing the player to learn from them as examples and having direct dialogue teaching to the player. Furthermore, we discussed how OAuth would be a priority for us going forward, as during this sprint we started thinking about how we would incorporate authentication to store progress and decided it would be better to add this feature as soon as possible. We also discussed our direction of primarily creating a game engine first and then adding the content second, which the client agreed is a direction that is good and we should continue, citing his experience working with large game studios and his knowledge that this is how they generally produce new games. Furthermore, this allows the game to be much more extensible by other game designers and maintainers later on with the core functionality already in place. Finally, we discussed what we learned from this sprint, the main two points being: we should strive to maintain consistent progress instead of completing work in the first or latter half of the sprint while prioritizing blocking tasks, and us having finished most of the work of the sprint in the first half shows that we can increase our velocity going forward in an effort to maintain more consistent progress.

- **02/08/2024:**

- This meeting was a sprint planning meeting with the client where we discussed what should be implemented for the upcoming sprint. There were no demos that took place during this meeting since it was building off of the demo we showed in the end-of-sprint meeting previously. From this meeting we goal some new stories to work on: Create the OAuth system to allow players to login and keep track of an active player, Create recurring expenses (similar to food or rent) and this introduced time advancements and a turn-based system, Enhance the trading system to have character variance, time variance, and bias, ADd consequences for expenses, and then stretch goal of adding phase two features of currency and banking.

- **02/22/2024:**

- In this meeting with the client, we discussed what story points we have finished so far. This included our login system, profession preferences along with descriptions that hint at what the NPC prefers, time advance along with expenses, and a better looking trade page. We demonstrated these features to the client and he was satisfied with the result. We also updated him on the remaining tasks which included the consequences for failure and price variance based on time. After the demo, we



discussed some concerns we had on features we already implemented. First, we discussed whether or not there should be a deadline for advancing to the next phase/level. The client said that this should be a long term goal, but that some sort of incentive to move on would be good. Allowing the player to try again after failing would be convenient, but we should not reward players that stumble through the game mindlessly. Next, we discussed whether or not unsuccessful trades should take time. The client said that there should be a downside to unsuccessful trades, but we should randomize it to prevent players from memorizing and taking advantage of the system. Lastly, we discussed how NPC inventories should be handled. The client stated that gameplay should be more important than realism. He presented NPCs having unlimited items for some items or implementing supply and demand. After discussing our current sprint and concerns, we moved on to what he would like to see for the next sprint. The client stated that he wanted to focus on the introduction of currency next. In addition to this, he wanted to see a tutorial. This would have content relating to the time value of money, assets, and liabilities, but would need to have a balance of mechanics and text. He also introduced the idea of debt and the consequences of being in debt. Lastly, we talked about how we should approach the last few sprints of the project. He proposed that we could either improve the UI in one sprint or combine feature sprints with some UI stories. Either way, the client expressed that he wanted to see a focus on the UI of the game.

- **02/29/2024:**
  - In this meeting we showcased some of the trade consequences and new account generation of starter items. And showed the UI changes that we made, and what the progress of the sprint was going so far. This meeting was really short, as it was mostly just an update and a check-in. The main thing that the client stated from this meeting was that he wanted all the features to be minimally styled.
- **03/22/2024:**
  - This meeting we showcased the UI changes that we made, and discussed some of the future changes that should happen for the project. The client mentioned that since it is towards the end of the project, we shouldn't focus on any large features, but instead polish the features we currently have. The main focus for now was to make the user experience a lot better, and start looking into a video that shows the flow of the game experience and UI.
- **03/28/2024:**
  - In this meeting with the client, we discussed the story points that we had finished in the sprint. These story points included: creation of the tutorial for the player, updates to the trading time changes and adding failure case, added starter items when making a new account, weekly allowance, UI overhaul, and a sidebar element to the web application - this also included a multitude of chores that we did to clean up the codebase and player experience which was increasing out code coverage, protecting the admin routes, and creating a style guide which the UI would follow for

consistency. We showed the new features and he seemed pleased with how things worked and liked how the UI turned out in comparison to what it was before. One feature that we weren't able to finish in this sprint was the currency exchange, which he said was okay since we were on the brink of getting it in. He made sure to highlight that we shouldn't add any huge features for this last sprint because it is coming to the end and we didn't want to have any buggy solutions. But since we were almost done with the story and it would be an effortless merge, it was fine. For sprint 5, we would be dropping the bank system since it would be too large of a feature to add at this point. Instead he told us to focus on refining what we have right now, making it more presentable and working on the hand-off to the Teamup group. This included documentation for deployment. For this upcoming sprint, The client said to focus on the user experience and improve the look and feel of the game. He mentioned how the game has really evolved from what he was expecting from a sprite game to a web application game which he said was interesting. This is also the time to get some minimal user feedback to make improvements to the game experience. For the next sprint, we'll focus on the handoff, user experience and refining what we have now.

- **04/11/2024:**
  - This was the final meeting we had with the customer and we showcased the entire application and the brand new features that we had. This included the updated UI, the navbar and discussed the currency implementation. We also stated how we updated the README to make it easier to deploy the application and how we had sent out a user testing form to get more opinions on the game. We basically showcased the features of the games in totality. After all the features were set, the client wanted us to send screenshots that showcase the visually good things about the application and post them in the discord for the product page.

## BDD/TDD Process

During development, we aimed to have a fully test-driven development approach. Every story was also behavior-driven, starting with a cucumber specification. Before implementing a solution to a cucumber feature, RSpec tests were created that covered the necessary code to fulfill the requirements. Only once both the cucumber and RSpec tests were complete did we write actual code. The cucumber specification defines the minimum requirements for a feature to be "done". Then, the RSpec tests define the code that needs to exist in order to satisfy the behavior.

While this went smoothly in most cases, it had some issues with too large or too small of features. In the development of small features, where the amount of code written is very small or the solution to the problem is known beforehand, much more time was spent on writing up the tests than actually building the feature. In addition, some of these small tests ended up being somewhat redundant,

testing properties that are guaranteed by the framework itself. For large features, the number of cucumber steps or edge cases had a tendency to grow exponentially, leading to tests that technically met coverage requirements but did not actually cover all edge cases. In other cases, the feature being added, such as additional styling, was not directly testable in a meaningful way by the test suite.

Overall the largest difficulty when implementing this test-driven development was when the behavior of a feature was not fully specified. While a story and cucumber feature file helped to clarify the intent of a new feature, it often did not rigorously specify how a feature should work, leading to assumptions or additional questions needing to be asked later in development. In addition, the team had trouble getting started as it was very difficult to imagine every piece of code that needed to exist prior to giving it a try.

## Configuration Management

We managed the project mainly through Github and Pivotal Tracker. Github hosted the repository where all our code was and we used Pivotal Tracker to keep track of the sprints, stories and bugs. Pivotal Tracker has an integration with Github so we were able to link each story with the branches that pertained to a story - making management and tracking easier. On Github, workflows were set up so that every time a change was made, it wouldn't break the main branch. These workflows helped with deployment to Heroku, running tests and checking Rubocop and Rubycritic. We also had verified commits when changes were made on Github. We never had sprints that were directly spikes, but we did have tasks that tended towards a research focus to aid with a story. In total we had 38 branches made and 5 releases, each release coming after the end of a sprint.

## Production Release Issues

In the beginning of the sprint, we also came across an error of the heroku instance repeatedly re-seeding itself, causing there to be duplicate versions of all entities. This was resolved by changing the configurations of the deployment to run `db:seed:replant`

There was also an error of our assets not appearing on the live deployment, and this was fixed by running `assets:precompile`

While deploying the application to Heroku, the team that verified our project came across an error that said: ActiveSupport::MessageEncryptor::InvalidMessage

```
remote:
remote:   Please report issues, and support the project! Thanks, [7eter 1-[. 13o1ing
remote:
remote:   Bundle completed (37.27s)
remote:   Cleaning up the bundler cache.
remote: ----> Detecting rake tasks
remote: ----> Preparing app for Rails asset pipeline
remote: Running: rake assets:precompile
remote: rake aborted!
remote: ActiveSupport::MessageEncryptor::InvalidMessage: ActiveSupport::MessageEncryptor::InvalidMessage
```

This was due to ruby on rails requiring the assets to be encrypted and have a key to decrypt them. This was fixed by setting the key as an environmental variable.

```
heroku config:set RAILS_MASTER_KEY=<your master key>
```

## Tools and Libraries

### Platforms:

- GitHub:
  - Robust platform for version control and collaboration. It helps teams manage projects, track changes, and collaborate on code in a distributed environment.
  - Can be complex for beginners. Additionally, for private repositories and advanced features, costs can accumulate for teams needing more resources or security.
- Heroku:
  - Cloud platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud. It supports several programming languages and simplifies app deployment, scaling, and management. Heroku's integration with GitHub and automatic deployment features make it particularly appealing for rapid development cycles.
  - Can become expensive as applications scale due to its pricing structure based on dynos (application containers)
- Pivotal Tracker:
  - Project management tool designed specifically for software development teams. It emphasizes agile methodologies and provides tools for planning, tracking, and iterating throughout the software development lifecycle. Its story-based approach helps teams maintain focus and momentum.
  - Steep learning curve for those not familiar with agile methodologies and may not be as flexible as other project management tools
- Slack:
  - Communication platform that supports real-time messaging, file sharing, and powerful integrations with numerous other tools like GitHub, Heroku, and many others
  - Can lead to information overload with its persistent notifications and multitude of channels and messages
- Google Meets:
  - Video communication service that is integrated with google suite apps such as Calendar, Drive, and Gmail, making scheduling and document sharing seamless
  - Can experience technical issues such as poor video quality and connectivity problems, especially with inadequate internet connections

### Code Quality:

- CodeClimate:

- Automatically reviews code for maintainability, generates test coverage reports, and identifies code smells. It integrates well with GitHub, providing visibility into code quality directly in pull requests.
- The metrics can be too rigid, flagging complex but necessary code as issues. It might not always align perfectly with all project requirements or coding styles.
- SimpleCov:
  - Provides a detailed report of which lines of your code are being executed to help ensure thorough testing.
  - Doesn't ensure good testing as it doesn't measure the quality of tests — only that the lines of code are executed during testing
- RubyCritic:
  - Pretty good feedback
  - Too stern and picky on a lot of aspects that are generated in the code and is disagreed with by the developer team
- Rubocop:
  - Static code analyzer and formatter for Ruby. It enforces many of the guidelines outlined in the community Ruby style guide.
  - Can be quite strict with style enforcement, which can be frustrating. Configuring it to match your project's specific style preferences requires effort.

#### Development:

- Rails:
  - Powerful framework for building web applications quickly and with less code. It follows "convention over configuration" which reduces the amount of decision-making and setup for developers.
  - Too heavy for smaller projects, and its "magic" can obscure what's happening, making debugging difficult for beginners.
- Rspec:
  - Very well defined testing framework
  - Can easily be bloated if not well written
- Puma:
  - Fast and concurrent web server for Ruby/Rails applications. It's designed for speed and concurrency.
  - Configuring Puma could be complex if used beyond development
- OmniAuth:
  - Flexible authentication system for web applications. It allows developers to integrate multiple authentication providers with minimal effort.
  - May be difficult to work with
- Capybara:
  - Simulates how a real user would interact with your application
  - Can be a little flaky and may not conclusively test for the behavior
- Tailwind CSS:

- Allows for rapid UI development. It is highly customizable and helps in building responsive designs with ease.
- Can make our html.erb bloated, harder to maintain, and breaks existing CSS

## Repository Contents and Deployment

The repository contains all of the information and tools necessary to create a development environment locally. First, Ruby version 3.2.2 must be installed. All project dependencies can be installed through bundle, which is included in the bin folder of the repository. In order to setup authentication with the local development environment, Google OAuth credentials must be provided. These can be added by first deleting the config/credentials.yml.enc file, and then creating a new one along with a new master key with `EDITOR=nano rails credentials:edit`. For more information about setting up authentication, see the Auth section of the readme in the repository. After this, the database can be seeded by following the instructions in the readme and the local development environment is ready.

The repository also contains tests, both unit and behavioral, which can be found in the spec and features folders respectively. Unit tests can be run with `bundle exec rspec`, and behavioral tests can be run with `bundle exec rails cucumber`. Additionally, this project is also configured to use rubocop and rubycritic.

Instructions for deployment to Heroku are provided in the readme. Deployment is generally done via CICD with Github Actions, however can be done manually by pushing to Heroku's git repository associated with the project. Deploying this project on Heroku also requires it be paired with a Postgres database addon. One important note is that the `RAILS_MASTER_KEY` and `SECRET_KEY_BASE` environment variables must be set on Heroku before the first deployment.

All of the core code for the project is located in the app directory. Inside of this directory, the most relevant directories are controllers, models, and views. Controllers contain all of the logic for handling requests and rendering views, along with handling authentication. The views directory contains templates, specifically embedded ruby (ERB) templates, that define the content of pages. Finally, models contains active record representations of all tables in the database.

One final item of note is that this project makes use of Tailwind utility classes. This functionality is provided through the 'tailwindcss-rails' gem, included in the gemfile, that manages compiling the tailwind utility classes into css. As these classes need to be compiled, there is a script located at bin/dev included that will manage listening to changes and recompiling during development automatically. To compile manually, run `bin/rails tailwindcss:build`. This is a good step to have in a continuous deployment pipeline.

## User Feedback

For the final week of the sprint we had sent out a survey and testing announcement to get user testing for our game. We did get lots of traction to the website with roughly 43 new people creating accounts and attempting the game. Unfortunately, while we did get a lot of traffic to the website, we weren't able to turn that traffic into feedback through the survey with < 5 people actually submitting a response in the survey. This tells us that there is room for improvement in either our survey potentially being too daunting to complete, or getting more reach in our initial user testing so that more people try out our game. From the responses that we did receive:

- The experience results that they mentioned said that we did a moderately good job for the UI and game experience.
- The design and visuals are really modern and sleek. The font selection, animated sprites, and overall feel of the site is beautiful.
- They went into the game thinking it would be a top-down game like pokemon, when in actuality it is a click-through game. They wanted to be able to walk around. This could be a pivot point for the game to go into. We were constrained by the class and time, but it is possible to implement something similar.
- Wished the expenses and shopping list was permanently on the sidebar, it was annoying to keep clicking to it. The expenses and budget has been permanently added to a fixed sidebar, but the shopping list on the sidebar is something that can be implemented in the future.
- There is a lot of whitespace that could be filled up

## Project Links

**Deployment:** <https://fin-lit-quest-65cfa09cddc8.herokuapp.com/>

**Github:** <https://github.com/teamup-apps-for-good/fin-lit-quest>

**Project Tracker:** <https://www.pivotaltracker.com/n/projects/2687724>

**Slack:** <https://hagglerdev.slack.com>

## Presentation and Demo

**Presentation Slides:**  Final Presentation

**Presentation Video:** <https://youtu.be/qLFzQBggDzM>

- For captions, make sure to select the english captions on Youtube!

**Demo Video:** <https://youtu.be/ZeVTqQh4rdI>