

Status Report 1

Group 2: Toffy Chen, Mingyang Li, Allen Zhang, Drew Beamer
March 21, 2024

Q 1 Introduction

1.1 Highlights

What was the plan for this iteration?

There are three sprints for this iteration.

First sprint - Cranky Frog 🐸 😡 (Feb.27 ~ Mar.12)

- Database schema brainstormed and set up
- Main pages (market page, trade status page, profile page) identified and designed
- Next.js environment set up and *MongoDB* connection established
- Customer feedback gathered for product backlog expansion (Mar.11)

Second sprint - Epic Buffalo 🐃 🏹 (Mar.12 ~ Mar.19)

- Navbar and market page implemented using *shadcn* UI library
- *Mongoose* utilized for database connections
- Server-side image compression and storage implemented
- Server action for post-validation and addition created
- Posting functionality Implemented, post information can be stored on the local database
- Customer review conducted for feedback collection (Mar.19)

Third sprint - Eloquent Tiger 🐅 (Ongoing, Mar.19 ~ Mar.26)

- Needed item categories and search filters identified from customer interviews (Mar.20)
- Authentication login implemented using *NextAuth.js*
- To expand potential customer base through outreach to professors
- To schedule a meeting with Davidson Technology & Innovation for email authentication
- To implement avatar change functionality for user login personalization

Highlight what the team accomplished.

- Figma design of all pages and frontend implementation of the market page and navbar.
- Database schema and post-storage
- Image compression and storage
- Login and log-out functionality
- On-time communication with primary users

1.2 Changes

Summarize any major changes since the proposal.

After talking to primary customers, we decided to separate the trade status from the personal profile page. We are going to create a simple drop-down personal profile and a trade status tab, which is clearer than putting all the things together.

Include each change's date, motivation, description, and implications.

Separate trade status page and profile page.

Date: February 27, 2024

Motivation: Trading is an important functionality of our designed system. Our primary customer believe that creating a separate page makes it clearer to the user, and allows more information about trade to be displayed. For us, it also helps create a clearer database schema.

Description: Our previous design put personal profiles and trade statuses together. Now we add a separate page for trade status which can be accessed from the navbar. The design of the trade status page will include trade history, current trade, etc.

Implication: We haven't implemented the trade status page. In the past sprints, we focused more on the market page. What we have already done is the implementation of every single post and the schema of the trades.

2 Customer-Iteration Goals

Describe the product backlog of this iteration.

User Story	Task	Description
As a user, I want to easily navigate through the website using a well-designed navbar, so that I can access different sections of the application seamlessly and efficiently.	Main pages and navbar design, UI implementation	Create clear navigation options that allow users to easily navigate.
As a user, I want the application to be accessible across different devices and screen sizes, so that I can use it conveniently whether I'm on a computer, tablet, or smartphone.	Responsive design (devices and screen sizes)	Allow users to use the web application and all functionalities on different devices.
As a user, I want to see my own profile so that I can edit my user name and see my overall ratings.	Profile drop-down design, Database schema	Create a profile drop-down for users to view and edit their profile details, such as username and avatar.
As a user, I want to review my post history so that I know what I traded in the past.	Post status page design, Database schema and integration	Create a page within the application where users can access their post history and review their past trades, including details such as item descriptions, trade partners, and timestamps.
As a user, I want to create a post for trading, specifying details such as the item I'm offering or seeking, the desired trade conditions, and any additional information so that I can connect with other users for	Post-error handling and validation, Post-creation form, Image compression and storage, Database schema and integration	Build a form/interface where users can input the necessary details of their trade offer or request.

trading purposes.		
As a user, I want to log in to my account securely, so that I can access my personalized trading dashboard and interact with other Davidson College users	Authentication flow implementation, Login interface, Login-error handling and validation	Implement secure user authentication functionality within the application that only allow Davidson College users.

Describe the customer's desired overall experience.

The customer wants a trading platform that's easy to use, safe, and personalized. They expect it to work well on any device and to be simple to navigate with clear instructions. They need strong protection for their account and data, as the platform is restricted to the Davidson community only. They also want regular updates and a way to give feedback to make the platform better. Plus, they want clear communication from the platform team and helpful support when they need it.

Describe the sprint backlog of this iteration.

First sprint - Cranky Frog

Database Setup and Schema Design

- Brainstormed and finalized database schema
- Set up a *MongoDB* environment on the local host.

User Interface Design

- Consult primary customers to brainstorm the main pages to be implemented on the web application.
- Designed all main pages (market page, trade status page, profile page), and navbar with both mobile and desktop versions on *Figma* after talking to primary customers (students at Davidson College).

Environment Setup and Learning

- Set up the *next.js* environment
- Learned how to connect *Mongo* and *next.js*.

Customer Feedback and Backlog Expansion

- Gather feedback from primary customers on the designed pages.
- Discuss with customers to expand the product backlog for the next sprint based on their needs and suggestions.

Second sprint - Epic Buffalo

UI Implementations

- Implemented the navbar and market page using the *shadcn* UI library.

Database Integration

- Utilized *Mongoose* to build connections with the database.

Posting Functionality

- Implemented server-side image compression, storing images in 64-base strings.
- Created a server action that validates and adds posts to the database
- Create the posting form with the needed input fields

Customer Review and Feedback

- Talked to primary customers(students at Davidson College) to review the overall design and posting functionality.

Third sprint - Eloquent Tiger

Categories and Filter Functionality

- Met with primary customers (students at Davidson College) to identify needed item categories and search filters.

Authentication Flow Implementation

- To implement authentication login using NextAuth.js
- To schedule a meeting with Davidson Technology & Innovation about Davidson College email authentication

Customer Outreach and Engagement

- Expand potential customers by talking to professors at Davidson College

Other User Experience Enhancement

- To implement avatar change upon user login

Explain why you have selected the work items in the sprint backlog for this sprint (or iteration).

The work items in the sprint backlog are based on the following reasons:

1. **User needs:** The selected work items address the most critical user needs identified during the planning phase. For example, in the first sprint, tasks related to database setup and user interface design were prioritized to ensure a solid foundation for the application.
2. **Customer feedback and input:** Tasks were chosen based on feedback and input from primary customers (students at Davidson College). This ensured that the development efforts were aligned with user expectations and preferences.
3. **Dependency:** Work items were selected to manage dependencies effectively. For instance, in the second sprint, database integration was prioritized following the completion of the database setup tasks from the first sprint.
4. **Iterative Development:** The sprint backlog was structured to follow an iterative development approach, with tasks building upon each other to deliver incremental value to users.
5. **Technical Feasibility:** The selected work items were assessed for technical feasibility within the sprint timeframe. We consider factors such as available resources, team expertise, and the complexity of the tasks.

Overall, the main goal of the initial sprints is to establish a solid foundation for the project by laying out the main framework and design of the system. This approach will facilitate the implementation of user-driven functionality in the future and allow for greater flexibility to adapt and change based on customer feedback.

2.1 Use Cases

Write a use case for each main user goal for a primary or secondary customer. Show each use case's title, user goal, and full basic flow. Choose meaningful titles. Each use case should have at least one specific alternative flow and one bounded alternative flow.

Use Case 1

Name: Create Trade Post

Goal: A user creates a post for trading, specifying details such as the item they are offering or seeking, the desired trade conditions, and any additional information to facilitate trading with other users.

Actors: User, Trading Platform System

Basic Flow:

The user logs into the trading platform.

{Initiate Post Creation}

The user navigates to the 'Create Post' section of the platform.

{Enter Details}

The user fills out the form with details of the trade, including whether they are offering or seeking an item, descriptions of the item, and desired trade conditions.

{Submit Post}

The system validates the entered details and creates a new trade post.

{Confirm Post Creation}

The system displays a confirmation message to the user, indicating that the post has been successfully created.

{View Post}

The user views their post on the platform to ensure accuracy and visibility.

Alternative Flows:

Specific Alternative Flow:

At {Submit Post}, if the system detects missing or invalid details,

The system displays an error message detailing what needs to be corrected.

Resume the basic flow at {Enter Details}.

Bound Alternative Flow:

At any point between {Initiate Post Creation} and {Confirm Post Creation}, if the network connection is lost.

Display "Sorry, network connection is lost. Please try again."

Resume the basic flow at {Initiate Post Creation}.

Use Case 2

Name: Secure Account Login

Goal: To allow a user to securely log in to their personalized trading dashboard and interact with other Davidson College users.

Actors: User, Trading Platform System

Basic Flow:

The user navigates to the login page of the trading platform.

{Enter credentials}

The user enters their username and password into the respective fields.

{Request authentication}

The system verifies the entered credentials against the database.

{Authentication}

If the credentials are correct, the system grants access to the user trading dashboard.
{Close login}

Alternative Flows:

Specific Alternative Flow:

At {Authentication}, if the credentials do not match
Displays an error message "Invalid username or password."
Resume the basic flow at {Enter credentials}.

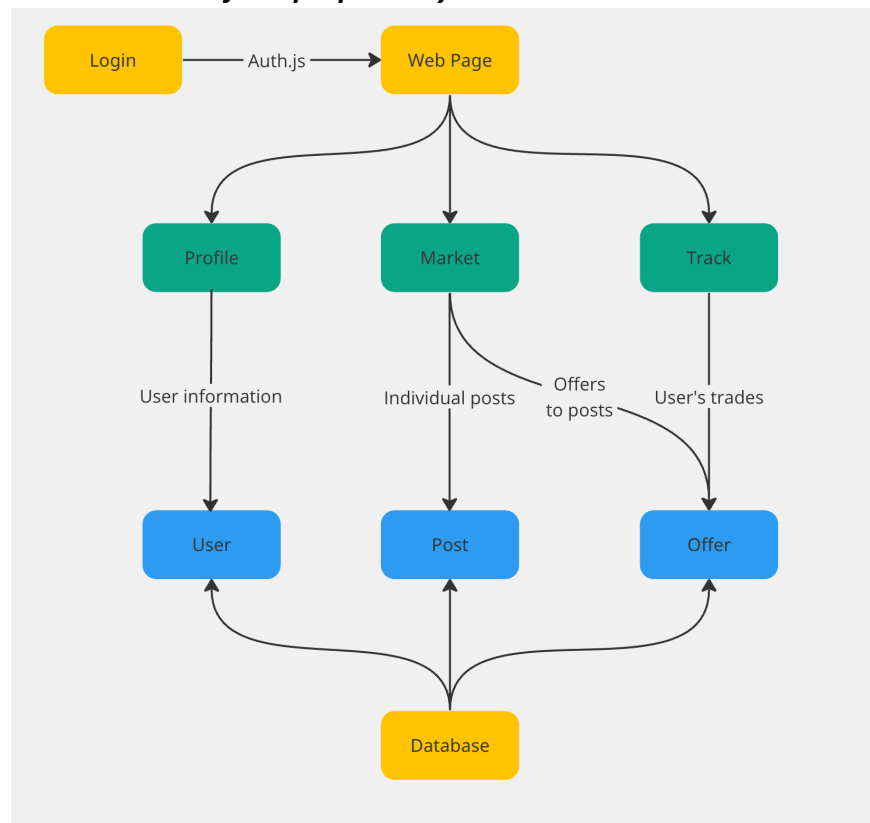
Bound Alternative Flow:

At {Request authentication}, if the system cannot reach the database due to a server issue or maintenance.

Displays "Service temporarily unavailable. Please try again."
Resume the basic flow at {Enter credentials}.

3 System Description

What are the main elements of the proposed system?



Frontend web page

There are three main components of our web page: user profile, market and status tracking. On the user profile page, users can see their current score, email, and user name. In the market, users can see all the posts from other users, and they can make offers for currently active posts. In the tracking page, users can see all the trades they are involved in and the status of the trades.

Backend database

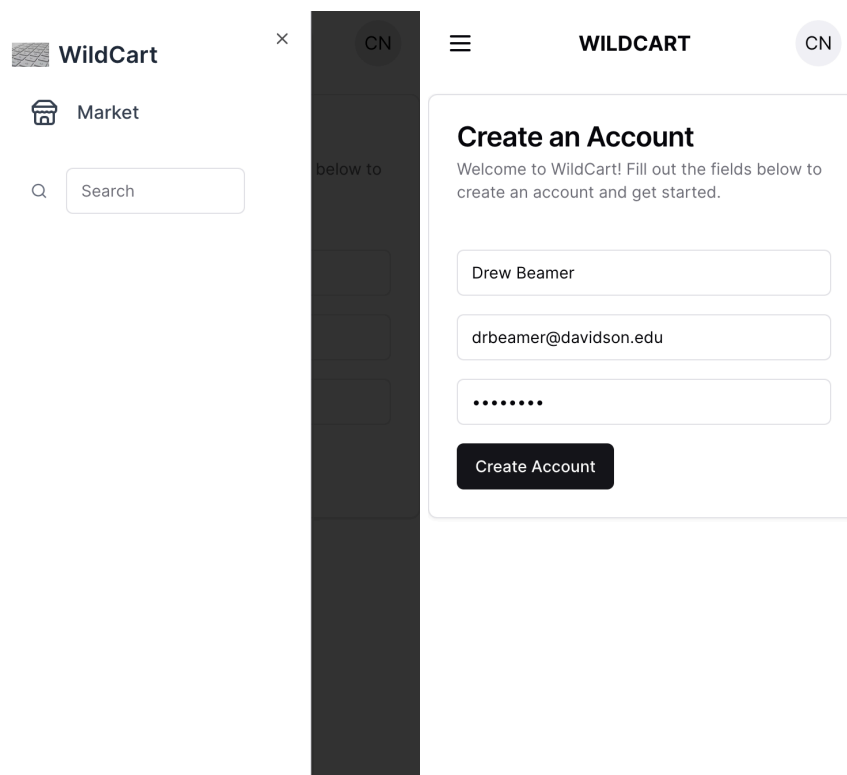
In our MongoDB instance, there will be three collections corresponding to three main components on the web page. The data will be fetched and saved through server-side functions. Currently, we deploy the database in our local host, but we will move it to Mongo Atlas in the following iterations

Login

We employ Auth.js for email authentication in the user login process. It will also connect to our database and save user's information after registration.

4 Current Status

4.1 Screenshots



Mobile Navbar and Account Creation Screen

Method

Trade ↺

Item Information

Title*

Robot

Category*

Electronics

Approximate Value

5000

Picture*

Choose File Kitty Media C4B 7752.JPG

Condition*

New

Description

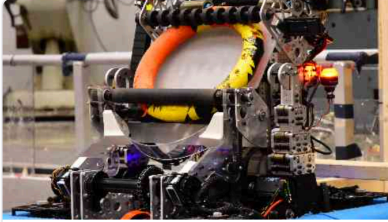
A new robot

WILDCART

CN

Market

Create Post



Robot

DB Drew Beamer Level 40

A new robot

Offer

Details

Create a Post Form and Market Page

4.2 Tests

List the tests you performed for this iteration's implemented parts.

- We tested the connection of our database and frontend add-post functionality by creating mock posts and checking the changes in the database.
- We tested the adaptability of our frontend design by inspecting the web page in different screen sizes in Chrome.
- We tested authentication by utilizing incorrect credentials for the sign in and verified that we were only able to start a session when a user had the correct credentials
 - Further, we verified that the user's credentials were no longer available after signing out
- We validated that only users who were signed in were able to create posts and that the post's seller_id field was the correct ID stored for the user in MongoDB

From the acceptance tests in the proposal, explain your plan to test them technically. In other words, how would you automatically test your acceptance tests? What are the inputs to the tests and outputs from the tests? How would you determine whether the test was passed or failed?

In the future, we plan to use a combination of [Jest](#) and [Husky](#) to do automated unit tests. The test will be initialized whenever a push is attempted.

- Given a user is logged into the platform, when they navigate to their account dashboard, then they should see a list of all offerings made to them.
 - **Input:** Mock user account and login action

- **Output:** A list of posts fetches from the database
- The success of the test will be based on whether or not the output data are valid
- Given a user is browsing items, when they view a product listing, then they should see the estimated condition of the item clearly stated with pictures.
 - **Input:** A post
 - **Output:** A boolean represents whether there is an estimated condition and clear pictures in the post
 - The success of the test will be based on the boolean condition and whether or not it matches the real condition
- Given a user is interacting with another user, when they view the profile, then they should see their trustworthiness rating and reviews from previous trades.
 - **Input:** Another user's profile
 - **Output:** A boolean represents whether there is a rating displayed
 - The success of the test will be based on the boolean condition and whether or not it matches the real condition

5 Project Management

Date	Change Made
2024-02-29	Initialize <i>Next.js</i> Application (Drew)
2024-03-12	Add and configure <i>shadcn</i> UI (Drew)
2024-03-18	Add database schema with <i>Mongoose</i> (Jerry)
2024-03-18	Add dev environment setup instructions (Drew)
2024-03-18	Create PostCard template component (Drew)
2024-03-19	Add Navbar for Desktop and Mobile (Allen)
2024-03-19	Add create Post server action and data validation (Jerry)
2024-03-19	Add create Post form UI and link to server action (Drew)
2024-03-21 (in progress)	Add <i>NextAuth.js</i> authentication (Toffy / Drew)

Sprint when Merged

Sprint 1 (Cranky Frog) - Green

Sprint 2 (Epic Buffalo) - Brown

Sprint 3 (Eloquent Tiger) - Orange

6 Reviews and Retrospective

What went well?

- Our group has had great communication
- Very few blockers; those that we have had have been easily dealt with
- We feel on the same page about plans for UI
- Collaboration when pair programming has been effective

What didn't go well?

- We have been inconsistent about using the daily stand-up bot we have set up in our shared Discord server
- Not enough clarity on tickets before starting the sprint: we need to better define acceptance criteria and user stories ahead of time
- We are lacking a well-defined product backlog/user story backlog at the moment

For the goals that were not met, what were the issues?

- We have met most of our goals so far, however, some tickets have bled over into the following sprints largely due to inaccurate pointing/assessment of the learning curve of some of the technologies we are using

How do you plan to overcome the issues?

- We believe that the learning curve issues will improve in the long-run as we gain familiarity with the tech stack
- In the short term we are pairing up more to try to work through issues together
- We also plan to dedicate more time to more clearly defining tickets and user stories ahead of sprint review and planning

What do you plan to do differently in the next iteration?

- Have more interaction and feedback from users
- Pair up more frequently on challenging tickets
- Dedicate time to making and defining user stories and their respective acceptance criteria

7 Team Management

What were the team roles for this iteration? What did each team member contribute?

- We have maintained the roles we started with, where Toffy is our Product Manager, Drew is the Scrum Master, and Allen and Jerry are developers
- Toffy is currently working on setting up the *Auth.js*, and has been working on meeting with users to develop user stories and goals for each sprint. Toffy has also worked on several Figma designs for key components of the application's UI.
- Drew has worked on organizational tasks such as setting up the repository and Jira, along with various full-stack user stories. Drew also designed the market "Postcard" component on Figma.

- Jerry has largely worked on the backend, including defining the schema for our *MongoDB* database and server actions for creating a post.
- Allen has worked on frontend, and created the UI navbar we are using on the site. Allen also worked on the design of the navbar, and also trading history page with *Figma*.

What were the challenges regarding team management, e.g., regular meetings, etc.?

- The largest challenge regarding team management so far has been the asynchronous daily stand-up and a low participation rate in them.

What are the plans to overcome the challenges?

- We plan to be more proactive about the daily stand-up, via reminders to those who have not filled it out yet in Discord.

If you were the third party who knows very well about your team, what suggestions would you give to your team?

- Clarify Roles and Responsibilities: While it's clear who is responsible for what, ensure that every team member fully understands not just their own roles but also those of their teammates. This cross-understanding can foster better collaboration and mutual support, especially when someone is facing challenges.
- Skill Development: Given the learning curve with the new technologies, allocate time for team members to share knowledge and skills. For example, you could have short training sessions or 'tech talks' where one member educates the rest on an aspect of the technology they're more familiar with.

8 Goals for the Next Iteration

Write the next iteration's product log. Write the next iteration's sprint log.

Product Log

Post Management

- Display current posts on the user's profile page

Trade Browsing

- Enable filtering on the market page

Profile Management

- Create profile page

Sprint Log

Post Management

- Display current posts on the user's profile page
 - Develop backend API endpoints to retrieve the user's posts based on their profile ID.
 - Integrate the backend functionality with the user's profile page on the frontend to display the retrieved posts.

Trade Browsing

- Enable filtering on the market page
 - Enhance the market page frontend to include filtering options, such as dropdown menus or checkboxes for selecting criteria like item category or trade conditions.

- Implement backend functionality to filter posts based on user-selected criteria, using database queries or filters.
- Integrate the frontend filtering interface with the backend functionality to dynamically update the displayed posts based on the user's selections.
- Develop backend functionality to perform keyword searches across post titles and descriptions.

Profile Management

- Create profile page
 - Implement the profile drop-down frontend interface, including sections for displaying user information

Other than the issues discussed in Section 6, i.e., Review and Retrospective, what potential challenges do you see in the next iteration?

Briefly explain how your team would overcome each of the mentioned challenges.

- **Complexity in Implementing New Features:** As we add more features, the complexity of the system increases. This could lead to longer development times and potential integration issues.
Strategy: Implement code reviews and have a robust testing process. Consider using feature flags to manage the release of new functionalities, allowing for easier rollback and testing.
- **User Feedback Integration:** With plans to have more interaction and feedback from users, the challenge will be to efficiently integrate this feedback into your development process.
Strategy: Set up a structured process for collecting, reviewing, and prioritizing user feedback. Ensure there's a clear path from feedback to actionable development tasks.
- **Ensuring Quality with Rapid Development:** The need to deliver multiple features could compromise quality if not managed well.
Strategy: Maintain a balance between development speed and quality. Invest time in automated testing and continuous integration to catch issues early.
- **Technical Debt:** Rapid development can lead to accumulating technical debt, which could slow down future development.
Strategy: Allocate time for refactoring and addressing technical debt within each sprint. Encourage developers to flag areas of the codebase that need improvement.