CSCC01 Project Deliverable #5

Group: Leap C

Written by:
Luke Zhang, Evan Ng, Anthony Alaimo, Pravinthan Prabagaran, Chi Jian Hsu

Table of Contents

Product Backlog

Changes

During deliverable 5, after getting feedback from PickEasy in deliverable 4 we added many more user stories to our product backlog. Like deliverable 5, we still have two main users: restaurant staff and customers. Under restaurant staff users we have four categories of stories: achievements, rewards, my restaurant, and scan. Under customer users, we have five categories of stories: achievements, rewards, scan, and navigation. The details such as cost and priority of the user stories are listed below. Like deliverable 4, a priority of 3 denotes that user story is designated for sprint 3 and a priority of 4 denotes that user story is designated for sprint 4.

Backlog

Restaurant Staff

- Achievements
    - (Cost: 6) (Priority: 3) As a restaurant staff, I want to have achievements grouped together based on a template
- Rewards
    - (Cost: 6) (Priority: 4) As a restaurant staff, I want to be able to modify the weight of getting a random reward for a certain level.
- My Restaurant
    - (Cost: 4) (Priority: 3) As a new restaurant staff, clicking on save when first creating a restaurant should route you to Achievements page
- Scan
    - (Cost: 12) (Priority: 3) As a restaurant staff, I want to be able to scan a QR code to validate an achievement is completed for a customer
    - (Cost: 4) (Priority: 3) As a restaurant staff, I want to be able to see a history of my customer's previously scanned rewards and achievements.

Customer

- Achievements
    - (Cost: 12) (Priority: 3) As a customer, I want to be able to scan a distinctive QR code for each achievement to complete an achievement and obtain a ticket, for example, after I've ordered a coke, a staff member will scan a QR code for the achievement "purchase a soft drink".
    - (Cost: 6) (Priority: 3) As a customer, I want to have my achievements grouped by restaurant
- Rewards

    - (Cost: 10) (Priority: 4) As a customer, I want to be able to see the list of all rewards categorized by tiers for each restaurant
    - (Cost: 10) (Priority: 3) As a customer, I want to be able to spend my tickets to receive a random reward from my unlocked tiers, for example, if I am in the silver tier and I spend 5 tickets, then I receive a random reward from either the silver or bronze tiers.
    - (Cost: 8) (Priority: 3) As a customer, I want to be able to spend my tickets to level up my current tier, for example, if I am in the bronze tier and I spend 10 tickets, then I level up to the silver tier. And if I am in the silver tier and I spend 10 tickets, then I level up to the gold tier, and so on and so forth for the platinum and diamond tier
    - (Cost: 12) (Priority: 3) As a customer, I want to be able to scan a distinctive QR code for each reward to redeem a reward, for example, on my phone, I open the correlated QR code to the reward I've earned, and a staff member will scan my QR code for the reward "FREE Large Fries".
    - (Cost: 6) (Priority: 3) As a customer, I want to have my rewards grouped by restaurant and tier
- Scan
    - (Cost: 4) (Priority: 4) As a customer, I want to be able to see a history of my previously scanned rewards and achievements.
- Navigation
    - (Cost: 8) (Priority: 4) As a customer, I want to be able to move to other pages of a certain restaurant easily without having to search every time

Mobile Support

- (Cost: 25) (Priority: 4) As either a customer or restaurant staff, I want to be able to access and utilize the achievement and reward functionality on my phone and Mac

Release Plan

Our release plan has not changed from the last deliverable. We will continue to have sprints of two weeks, except for our last sprint which will only be one week due to the deadline cutting short our sprint. The following listed are the dates of our four sprints, where all the sprint dates are Saturdays

- 1st Sprint: June 20th - July 4th
  - Release for D3: June 29th
- 2nd Sprint: July 4th - July 18th
- 3rd Sprint: July 18th - Aug 1st
  - Release for D4: July 20th
- 4th Sprint: Aug 1st - Aug 9th
  - Release for D5: Aug 10th

Sprint Plan

As laid out in our release plan, this deliverable we took on sprint 3 and sprint 4, so we chose all user stories with priority 3 and 4 respectively. In our backlog, we've broken down each user story into tasks and given each of them an acceptance criterion. We've associated each task with a number so that it's clear who is assigned to which task and in our Assignees section, we've described who is doing what task. Also, note the two sprints are split up into two sections.

Backlog for Sprint 3

Restaurant Staff

- My Restaurant
    - (Cost: 4) (Priority: 3) As a new restaurant staff, clicking on save when first creating a restaurant should route you to Achievements page
        - Acceptance: Given I've finished inputting all my restaurant information, when I click save, I should be automatically routed to the Achievements page
        - Task
            - 1: Make save button should route to achievements page
- Scan
    - (Cost: 4) (Priority: 3) As a restaurant staff, I want to be able to see a history of my customers previously scanned rewards and achievements.
        - Acceptance: Given I scanned a reward/achievement when the scan is confirmed, then I should be able to see the previously scanned reward/achievement in a history list containing the time of scan, customer name, reward/achievement name, and the progress (only for achievement's).
        - Task
            - 2: Create history for previously scanned rewards/achievements
    - (Cost: 12) (Priority: 3) As a restaurant staff, I want to be able to scan a QR code to validate an achievement is completed for a customer
        - Acceptance: Given a customer's QR code for a completed achievement, when I scan that QR code, then I should have validated the completion of the achievement.
        - Task
            - 3: Validate the completion of achievements using QR scanning
- Achievements
    - (Cost: 6) (Priority: 3) As a restaurant staff, I want to have achievements grouped together based on template
        - Acceptance: Given a list of achievement templates, when I create an achievement, then I should have achievements with the same template grouped together.

        - Task
            - 4: Group achievement's together based on template

Customer

- Achievements
    - (Cost: 12) (Priority: 3) As a customer, I want to be able to scan a distinctive QR code for each achievement to complete an achievement and obtain a ticket, for example, after I've ordered a coke, a staff member will scan a QR code for the achievement "purchase a soft drink".
        - Acceptance: Given I have 0 tickets in my customer account, when I get my QR code scanned for an achievement of 3 tickets, then I should have 3 tickets and the achievement should be marked completed
        - Tasks
            - 5: Generate QR code on the customer side.
                - Mark achievement complete, increase number of tickets
            - 6: Create QR code scanner on restaurant side
    - (Cost: 6) (Priority: 3) As a customer, I want to have my achievements grouped by restaurant
        - Acceptance: Given that I click on Achievements when on the home page, then I should have my achievements grouped together by restaurant
        - Acceptance: Given I'm on the home page, when I click on Achievements, then I should have my achievements grouped together by restaurant
        - Task
            - 7: Restaurant name as heading, under that will be the list of achievements
- Reward
    - (Cost: 12) (Priority: 3) As a customer, I want to be able to scan a distinctive QR code for each reward to redeem a reward, for example, on my phone I open the correlated QR code to the reward I've earned, and a staff member will scan my QR code for the reward "FREE Large Fries".
        - Acceptance: Given I have one reward in my customer account, when I get my QR code scanned for that corresponding reward, then I should the reward marked completed
        - Tasks
            - 8: Generate QR code on the customer side
            - 9: Create QR code scanner on restaurant side
    - (Cost: 10) (Priority: 3) As a customer, I want to be able to spend my tickets to receive a random reward from my unlocked tiers, for example, if I am in the silver tier and I spend 5 tickets, then I receive a random reward from either the silver or bronze tiers.
        - Acceptance: Given I have 5 tickets, am in Silver tier and a random reward costs 5 tickets, when I purchase a random reward then I should have 0 tickets, remain in Silver tier and a random reward from either the Bronze or Silver tiers

        - Tasks
            - 10: Create RNG animation to obtain a random reward
            - 11: Implement code to obtain random reward and subtract number of tickets
    - (Cost: 8) (Priority: 3) As a customer, I want to be able to spend my tickets to level up my current tier, for example, if I am in the bronze tier and I spend 10 tickets, then I level up to the silver tier.
        - Acceptance: Given I have 5 tickets, am in Gold tier and a level up costs 5 tickets, when I level up then I should have 0 tickets and be in Platinum tier
        - Task
            - 12: Create level up animation to the next level
            - 13: Implement code to now allow random rewards to unlock the rewards in the next tier's reward pool
    - (Cost: 6) (Priority: 3) As a customer, I want to have my rewards grouped by restaurant and tier
        - Acceptance: Given I'm on the home page, when that I click on Rewards when on the home page, then I should have my rewards grouped together by restaurant
        - Task
            - 14: Group rewards by restaurant and tier

Assignees

Luke
- Task 10
- Task 11

Evan
- Task 1
- Task 4

Anthony
- Task 12
- Task 13

Pravinthan
- Task 2
- Task 3
- Task 5
- Task 6
- Task 7

Chi Jian (Jeremy)
- Task 8
- Task 9
- Task 14

Backlog for Sprint 4

Restaurant Staff

- Rewards
    - (Cost: 6) (Priority: 4) As a restaurant staff, I want to be able to modify the weight of getting a random reward for a certain level.
        - Acceptance: Given I've created rewards when I assign these rewards to a level, then I should be able to change the weight of getting each reward at the given level.
        - Task
            - 1: Let restaurant staff modify the weight of random rewards
        - **Jeremy**

Customers

- Navigation
    - (P: 4) (C: 8) As a customer, I want to be able to move to other pages of a certain restaurant easily without having to search every time
        - Acceptance: Given I have found a restaurant on Rewards, when I click on the history or achievements button, I should be routed to that page with the given restaurant automatically searched and highlighted. And vice versa for achievements
        - Task
            - 2: Add navigation buttons to rewards/achievements pages
        - **Anthony**
- Rewards
    - (P: 4) (C: 10) As a customer, I want to be able to see the list of all rewards categorized by tiers for each restaurant
        - Acceptance: Given I am a customer on the rewards page, when I click on the reward pool button, then a list of all the rewards in each tier should be shown to me where the available rewards are highlighted and the unavailable rewards are greyed out
        - Task
            - 3: Display possible rewards customers can obtain
        - **Luke**
- Scan
    - (Cost: 4) (Priority: 4) As a customer, I want to be able to see a history of my previously scanned rewards and achievements.
        - Acceptance: Given I scanned a reward/achievement, when the scan is confirmed, I should be able to navigate to the history page where I can view my previous scanner reward/achievement. This will be displayed as a table depicting the restaurant, time of scan reward/achievement name, and the progress(only for achievement's).
        - Task

- 4: create scan history for customers previously scanned rewards/achievements

- **Evan**

Mobile System

- (P: 4) (C: 25) As either a customer or restaurant staff, I want to be able to access and utilize the achievement and reward functionality on my phone and Mac
  - Acceptance: Given I am accessing the website on my phone or Mac, when I go to any screen, then all the functionality should be available to me as if I were on my Windows computer
  - Task
    - 5: Make app usable on mobile devices
    - 6: Make app usable on Mac
  - **Pravinthan**

Assignees

Luke
- Task 3

Evan
- Task 4

Anthony
- Task 2

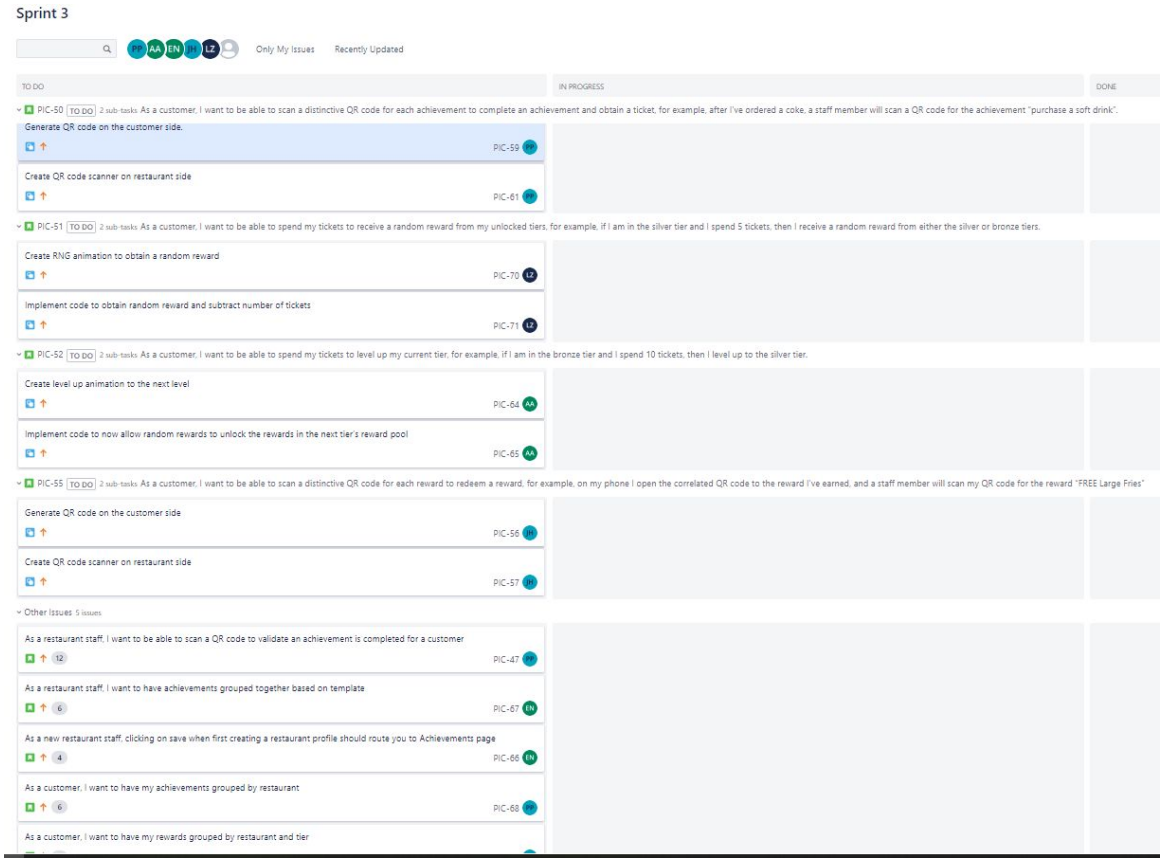
Pravinthan
- Task 5
- Task 6

Chi Jian (Jeremy)
- Task 1

Task Board

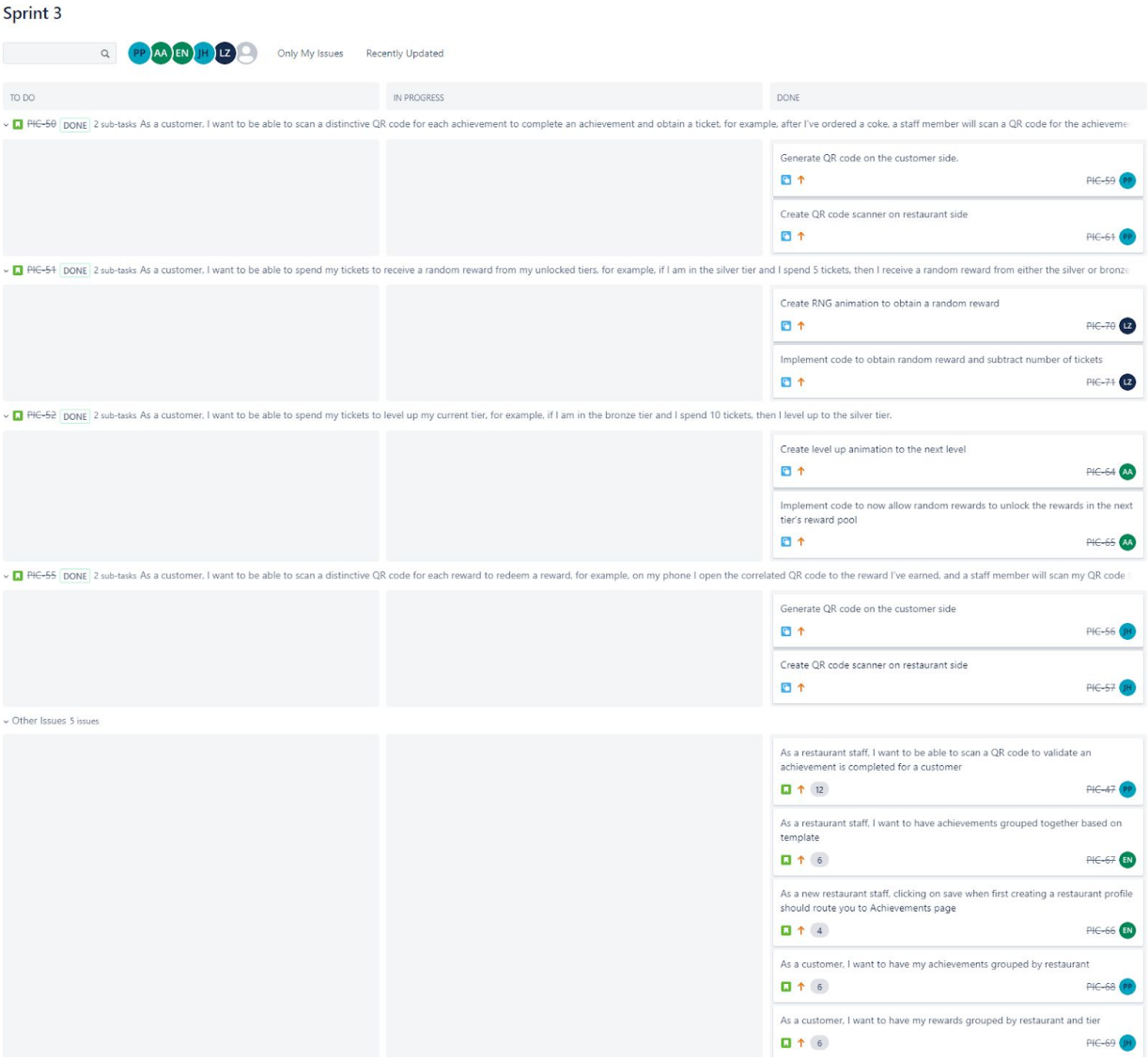
We were not aware that we were to take progressive snapshots, throughout the sprint, of our task board and burndown charts until it was too late, so we only have the beginning and ending snapshots of our sprints. Since we had two sprints, we have a total of 4 sprints for both the task board and burndown chart

Start of Sprint 3



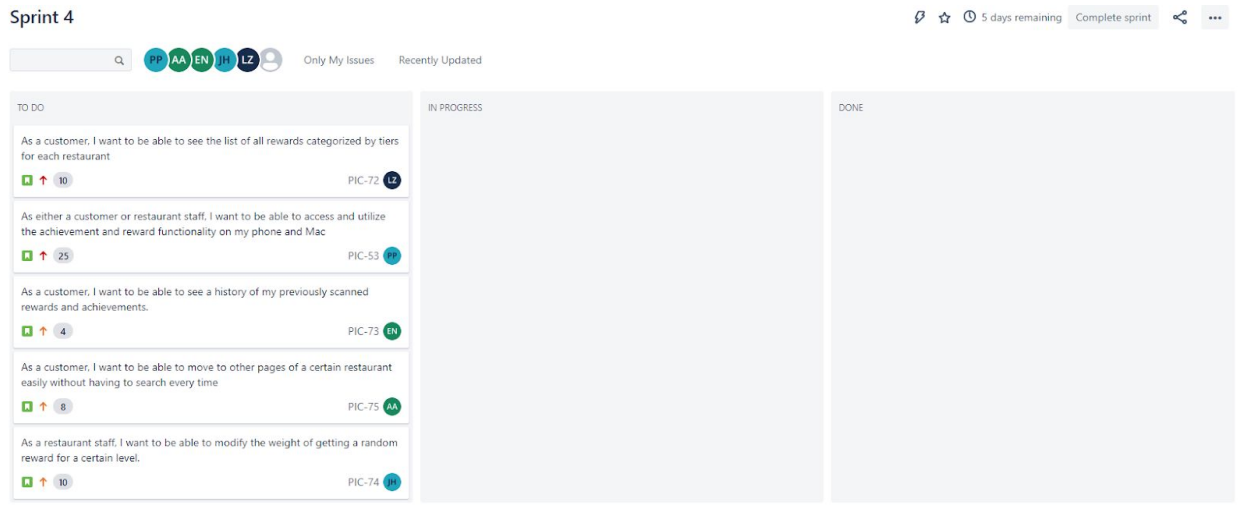
[Link to Start of Sprint 3 Task Board Snapshot]

End of Sprint 3



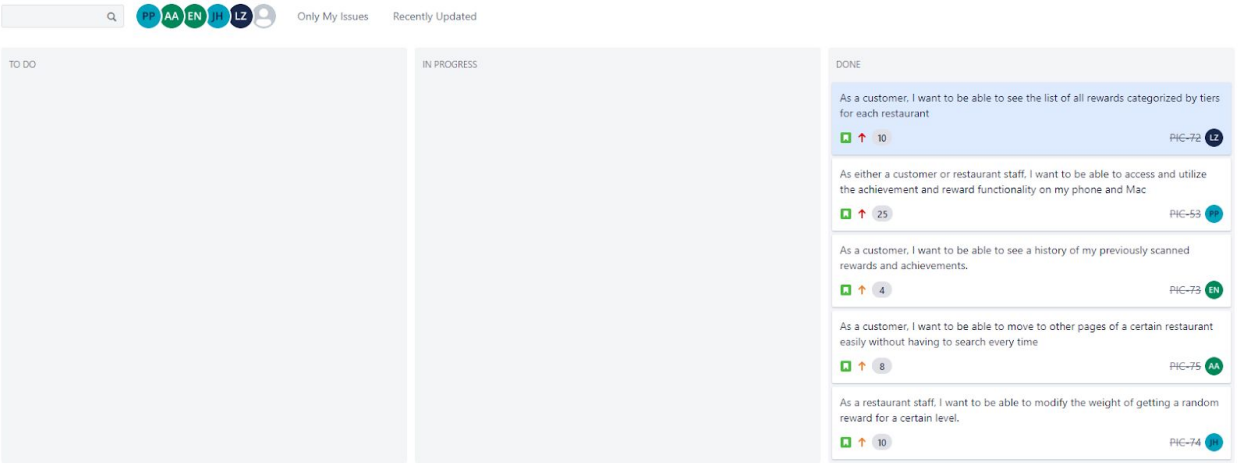
[Link to End of Sprint 3 Task Board Snapshot]

Start of Sprint 4



[Link to Start of Sprint 4 Task Board Snapshot]

End of Sprint 4

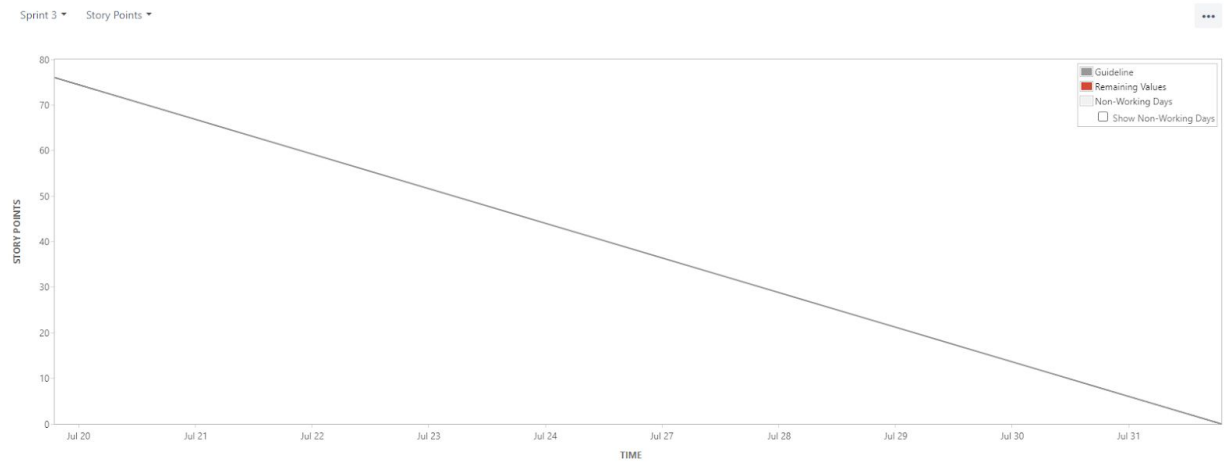


[Link to End of Sprint 4 Task Board Snapshot]

Burn Down Chart

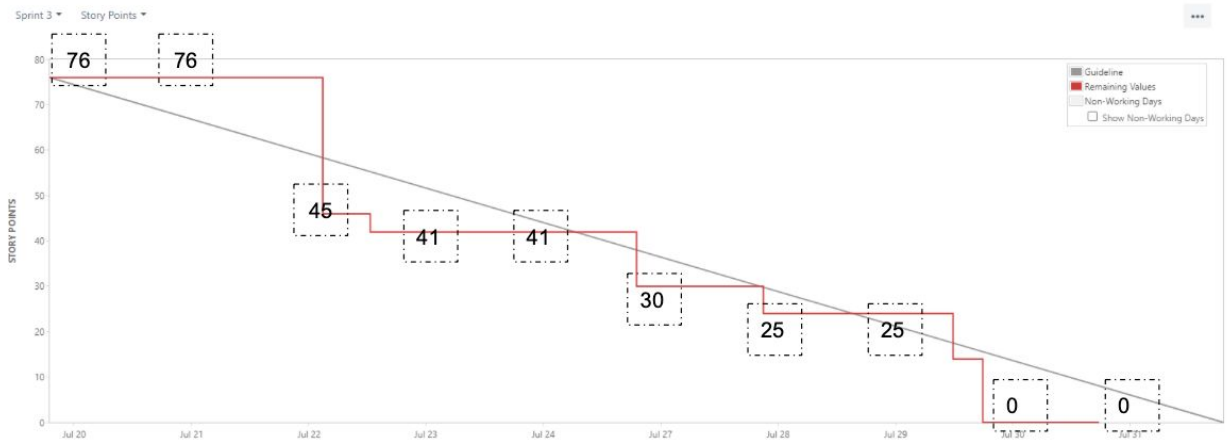
As above, we were not aware that we were to take progressive snapshots of our task board and burndown charts until it was too late, so we only have the beginning and ending snapshots of our sprints. Since we had two sprints, we have a total of 4 sprints for both the task board and burndown chart

Start of Sprint 3



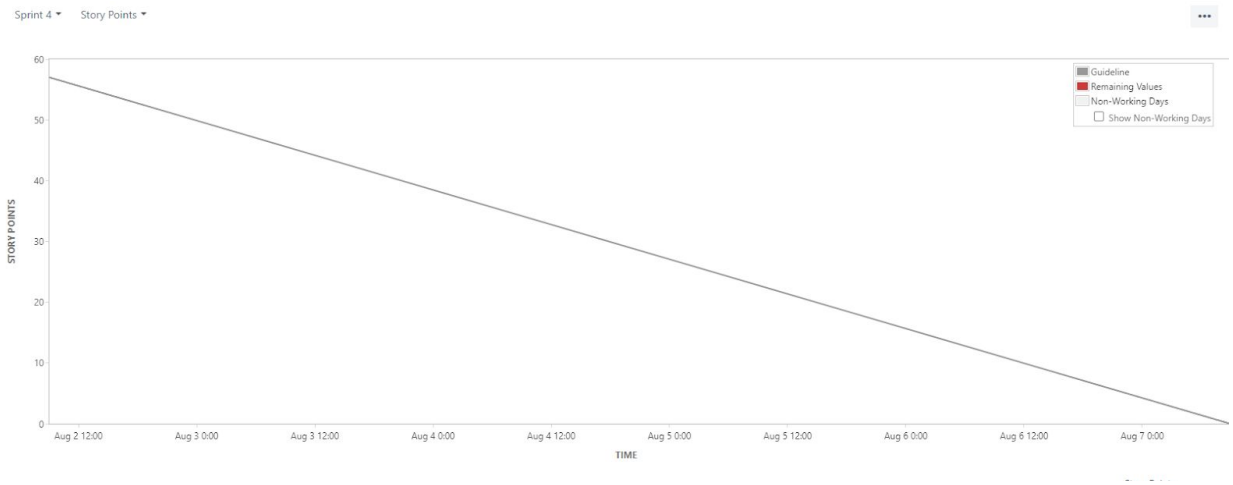
[Link to Start of Sprint 3 Burn Down Chart Snapshot]

End of Sprint 3



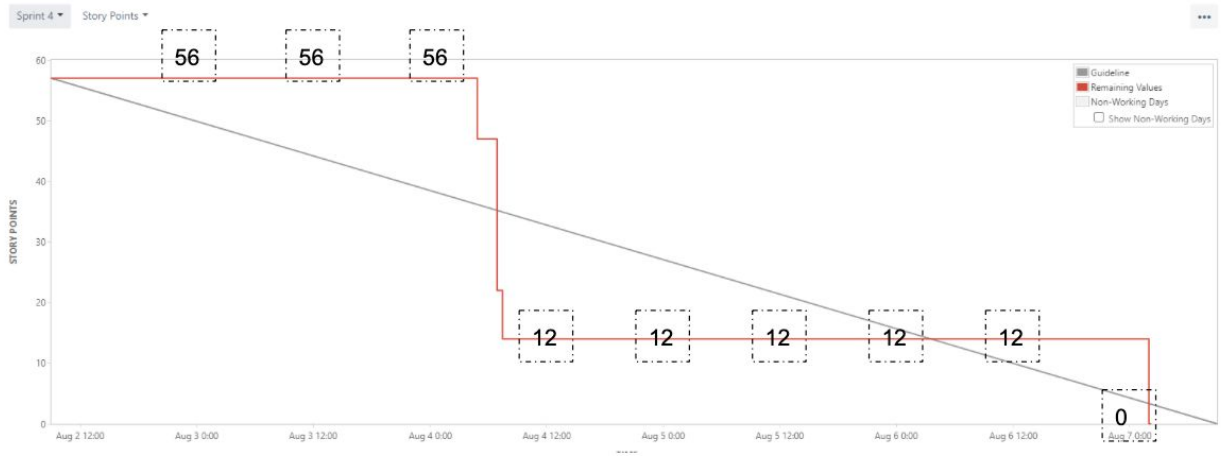
[Link to End of Sprint 3 Burn Down Chart Snapshot]

Start of Sprint 4



[Link to Start of Sprint 4 Burn Down Chart Snapshot]

End of Sprint 4



[Link to End of Sprint 4 Burn Down Chart Snapshot]

Architecture

Changes

For this deliverable, we did not change as much as the previous deliverable. We added and combined components and further simplified the diagram. We made changes on the Client Application side as well as the Server side.

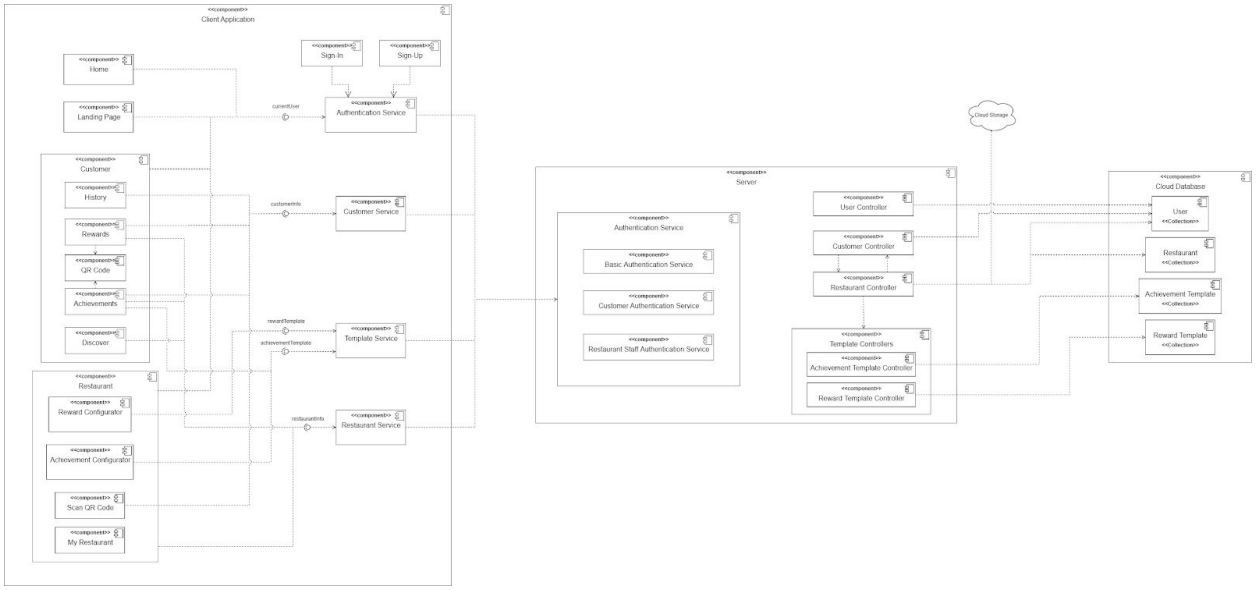
Client Application Changes

On the Client Application side, we added History and QR Code Components to the Customer Component to match our new features. We split the Redeem Component into Achievements and Rewards Components to maximize cohesion and better allow customers to understand what each feature entails. We also realized that our User Service Component was only being used as a way to get customer information, so we merged it into the Customer Service Component, which allowed us to decouple our design even more and maximize cohesion by having Customer Service Component solely provide the customer details. We also realized that the Customer and Restaurant Home Components were very similar, so we merged them into one main Home Component.

Server Changes

Since we removed the User Service Component on the Client Application side, we decoupled our User Controller from the Authentication Service as well as both the Customer and Restaurant Controllers. We merged code from a previously unlisted Authentication Controller into the User Controller as well to maximize its cohesion.

Diagram



[Link to Architecture Diagram]

Explanations and Dependencies

Client Application

Home Component

The Home Component is responsible for allowing users to sign in or sign up as a customer or as restaurant staff. Once signed in, this component provides access to the other main components (depending on whether the user is a customer or a restaurant staff) through tabs and address routing.

Landing Page Component

The Landing Page Component is responsible for routing users to either the Customer application or the Restaurant Staff application. It depends on the Authentication Service Component to automatically route users if they are already signed in.

Customer Component

The Customer Component is decoupled from the rest of the client application and contains five subcomponents (History, QR code, Rewards, Achievements, and Discover). This component and all its subcomponents depend on the Authentication Service Component to ensure users are signed in as customers and to retrieve basic user information.

History Component

The History Component is responsible for showing the customer's scanned achievement and reward history. It depends on the Customer Service Component to receive the necessary information.

QR Code Component

The QR Code Component is responsible for showing the restaurant staff the customer's achievement QR Code and the redeemed reward QR Code.

Achievements Component

The Achievements Component is responsible for allowing customers to activate achievements and allowing restaurant staff to scan achievements in the form of QR codes. This component also shows customers the achievements they have accumulated from different restaurants. It depends on the QR Code Component, Customer Service Component, Restaurant Service Component and Template Service Component to receive the necessary information.

Rewards Component

The Rewards Component is responsible for allowing customers to level up and roll for rewards as well as allowing restaurant staff to redeem rewards in the form of QR codes. This component also shows customers the rewards they have earned from different restaurants. It depends on the QR Code Component, Customer Service Component and Restaurant Service Component to receive the necessary information.

Discover Component

The Discover Component is responsible for displaying the list of restaurants to the user as well as links to the specific restaurant's achievements and rewards. This component depends on the Restaurant Service Component.

Restaurant Component

The Restaurant Component is decoupled from the rest of the client application and contains four subcomponents (My Restaurant, Achievement Configurator, Reward Configurator, and Scan QR Code). This component and all its subcomponents depend on the Authentication Service Component and Restaurant Service Component to ensure users are signed in as restaurant staff and to retrieve basic user and restaurant information.

Reward Configurator Component

The Reward Configurator Component is responsible for allowing restaurant staff to view and manage the restaurant's rewards. Restaurant staff can choose a reward template and configure the template variables and reward tier. Restaurant staff can also modify the weighted

probability for each reward tier. This component depends on the Template Service Component to retrieve the reward templates from the back-end.

Achievement Configurator Component

The Achievement Configurator Component is responsible for allowing restaurant staff to view and manage the restaurant's achievements. Restaurant staff can choose an achievement template and configure the template variables and the number of tickets to award on completion. Restaurant staff can also modify the number of tickets it takes to level up and roll for a reward. This component depends on the Template Service Component to retrieve the achievement templates from the back-end.

My Restaurant Component

My Restaurant Component is responsible for displaying and providing editable information about the restaurant such as its image, name, description, cuisine, and cost.

Scan QR Code Component

The Scan QR Code Component is responsible for retrieving data from customers' QR codes as well as showing a history of recent scans. This component depends on the Customer Service Component to update individual customer's achievement progress and to mark rewards as redeemed.

Sign-In Component

The Sign-In Component is responsible for acquiring the username and password inputs from the user and passing it to the Authentication Service Component to verify and authenticate the user, thus signing them in.

Sign-Up Component

The Sign-Up Component is responsible for acquiring the first name, last name, username and passwords inputs from the user and passing it to the Authentication Service Component to verify and create an account. After successfully doing so, the user is automatically signed in.

Authentication Service Component

The Authentication Service Component communicates with the Server's Authentication Service Component to obtain and verify user authentication information.

Customer Service Component

The Customer Service Component is responsible for retrieving and updating customer information such as achievement progress and obtained rewards. This component depends on the Server's Authentication Service Component to ensure authorization when accessing relevant information.

Template Service Component

The Template Service Component is responsible for retrieving reward and achievement templates from the back-end. This component depends on the Server's Authentication Service Component to ensure authorization when accessing relevant information.

Restaurant Service Component

The Restaurant Service Component is responsible for retrieving and updating restaurants' information such as image, name, description, cuisine, cost, achievements, and rewards. This component depends on the Server's Authentication Service Component to ensure authorization when accessing relevant information.

Server

Authentication Service Component

The Authentication Service Component contains three subcomponents (Basic Authentication, Customer Authentication, Restaurant Staff Authentication). It ensures users are verified when attempting to retrieve or update information.

Basic Authentication Service Component

The Basic Authentication Service Component ensures users are signed in when retrieving and updating information.

Customer Authentication Service Component

The Customer Authentication Service Component ensures users are signed in **as customers** when retrieving and updating information.

Restaurant Staff Authentication Service Component

The Restaurant Authentication Service Component ensures users are signed in **as restaurant staff** when retrieving and updating information.

User Controller Component

The User Controller Component is responsible for signing up and signing in as a customer or restaurant staff. This component depends on the Cloud Database's User Collection to retrieve and update user information.

Customer Controller Component

The Customer Controller Component is responsible for retrieving and updating customer-specific information such as achievement progress and obtained rewards. This component depends on the Cloud Database's User Collection to retrieve and update user

information. This component also depends on the Restaurant Controller Component to retrieve and verify information about restaurants.

Restaurant Controller Component

The Restaurant Controller Component is responsible for retrieving and updating restaurant-specific information such as image, name, description, cuisine, cost, achievements, and rewards. This component depends on the Cloud Database's User Collection to retrieve and update user information. This component also depends on the Cloud Database's Restaurant Collection to retrieve and update restaurant information. This component also depends on Cloud Storage to upload and retrieve restaurant images.

Template Controller Component

The Template Controller Component is responsible for retrieving all the templates based on type (Achievement or Reward) from the Cloud Database.

Achievement Template Controller Component

The Achievement Template Controller Component is responsible for retrieving all the Achievement templates from the Cloud Database. This component depends on the Cloud Database's Achievement Template Collection.

Reward Template Controller Component

The Reward Template Controller Component is responsible for retrieving all the Reward templates from the Cloud Database. This component depends on the Cloud Database's Reward Template Collection.

Cloud Database

User Collection

The User Collection contains information about users such as their username, first name, last name, achievement progress, obtained rewards, logs, and type of user (Customer or Restaurant Staff).

Restaurant Collection

The Restaurant Collection contains information about restaurants such as image metadata, name, description, cuisine, cost, logs, achievements, rewards, and reward weights.

Achievement Template Collection

The Achievement Template Collection contains all the achievement templates and their relevant variables and required tickets.

Reward Template Collection

The Reward Template Collection contains all the reward templates and their relevant variables and tier level.

Cloud Storage

The Cloud Storage stores restaurants' images.

Minimal Coupling, Maximal Cohesion

When designing the overall application, we tried to ensure it was as loosely coupled as possible but also maximally cohesive.

Minimizing Coupling

On the Client Application side, creating services allowed us to reuse HTTP/S calls when trying to retrieve or update information by injecting the services into any given component. This also had the positive side effect of making it easier to change components in isolation, create Mock Services for unit testing, and reuse components for other parts of the application. For example, we reused the Sign-In and Sign-Up Components for the Customer Home and Restaurant Home Components.

On the Server side, decoupling the Authentication Services allowed us to reuse the authentication methods for each API endpoint. We also decoupled the Controller Components so that we can change methods in each controller independently without issue.

On the Database side, we decoupled the collections to allow ourselves to better understand where each database request is being handled.

Maximizing Cohesion

On the Client Application side, using Angular allowed us to separate components with ease. Separating components such as the Customer subcomponents and Restaurant subcomponents allowed us to individually construct our template files (HTML), styling (CSS), and component logic (TypeScript) while maximizing cohesion. We were able to focus on making sure that any given component contained code that was only the intent of that component. For example, our Reward Configurator Component is maximally cohesive because it only contains code that allows the viewing and configuration of rewards and nothing else.

On the Server side, we separated our code into routes, controllers and models. In doing so, we were able to have multiple files of code, each focusing on its assigned task. Each model file only held the schema for its respective collection (e.g. models/user.js held the User schema), and each controller file only held its respective business logic.

On the Database side, we separated data into different collections for Users, Restaurants, Achievement Templates, and Reward Templates. In doing so, we were able to access only the relevant data to any given request.

Unit Testing

Documentation

Our documentation guide to unit tests can be found here.

Testing Strategy

The Test Approach

Our testing process is intuitive and designed to be easy to follow. We only test for components that are testable (Most components are testable but some do not have enough parts to test such as QR Code Component). We put our unit tests in unit test suites so that we can easily locate suites and add more unit tests as we desire. We test components independently in order to debug them quickly.

Test Environment

We run the automated unit tests in a headless Chromium browser. We perform our manual testing in our development environment, where we test on Chrome, Safari, and other browsers on Windows and macOS. Our hardware varies from low- to high-spec across each team member as well.

Tools for Testing

Our automated testing configuration uses Karma which handles the process of creating HTML files, opening browsers, running tests, and returning the results of those tests. Our automated testing configuration also uses Jasmine. An explanation of Jasmine can be found here.

Risk Analysis

We define each and every unit test with a “Risk Rating”. We define Risk Rating here.

Location of Unit Tests

The unit test suites are located in the *.spec.ts files, where the components we created tests for include: App, Header, Navigation Component, Home, Landing Page, Sign In, Sign Up, My Restaurant, Scan QR Code, Achievement Configurator, Achievements, Rewards, Reward Configurator, Discover. Pathing to all these *.spec.ts unit test suites can easily be found by going into “pick-easy/src/app/components” and from here you can path to the desired test file. In

each of the unit test suites, we have written a detailed comment explaining each individual unit test.

For example, let's take a look at the Achievement Configurator Component. Near the bottom of the test suite you can see there are four unit tests: the component is created, type of ticketsPickerInput should be a number and greater than 1, clicking the achievement template actually adds the template, and clicking remove should actually remove the achievement template. This procedure can be done for each of the features implemented in Deliverable 5 listed above to locate the specific unit tests implemented during the verification process.

The following is a list of all the unit test suite links for Deliverable 5:

- App Component
- Header Component
- Navigation Component
- Home Component
- Landing Page Component
- Sign In Component
- Sign Up Component
- My Restaurant Component
- Scan QR Code Component
- Achievement Configurator Component
- Achievements Component
- Rewards Component
- Reward Configurator Component
- Discover Component

Implementation

Modularity

Since we used Angular for the client-side, it was easy for us to modularize our code. Angular forces you to group specific code (i.e. TypeScript, HTML, CSS files) together and calls that group of files “components”. We reused components wherever possible and injected services and tokens as a form of modularity. Furthermore, TypeScript allows us to create types, classes, and interfaces, which makes it easy for us to reuse them across our entire code. On the server-side, we modularized our controllers and models as a way to decouple our code.

Minimal Coupling, Maximal Cohesion

We listed how we minimized coupling and maximized cohesion here.

Maintainable

From the start, we used the SOLID principles to design and implement our code, which in turn made it more maintainable for us. In terms of the Single-responsibility principle, developing in Angular made it so that each file is it’s own class and maintains its own responsibilities. In terms of the Open-Closed Principle, TypeScript allowed us to extend classes without modifying the class itself. In terms of the Liskov Substitution Principle, TypeScript also allowed us to use polymorphism and allowed us to substitute derived types for base types. In terms of the Interface Segregation Principle, we implemented many specific interfaces without being forced to implement methods that are not used. In terms of the Dependency Inversion Principle, we tried to ensure that high- and low-level modules depend on as many abstractions as possible.

Designing For Extension

Since we follow the SOLID principles in terms of our code design and implementation, our design is more extensible and allows us to add or modify components while minimizing impact to existing components. An example of this can be seen throughout the deliverables, where it was easy for us to add and modify features according to the client’s requirements.

System Validation

The four requirements that PickEasy laid out are a gamified way of completing achievements, achievements should be fully customizable/configured by the restaurant owners, easy way to redeem/verify coupons, and link to POS systems like Square/Shopify so we can automate collecting achievements. After discussing with PickEasy we learned that the loyalty system with achievements and coupons was a higher priority requirement than the POS system with Shopify. Therefore in our sprints, we prioritized these tasks first and in the end, we completed the first three requirements. We did not complete the fourth requirement of linking to POS systems. This, however, was okay in the client's eyes because they would like to have it, but understand due to time constraints it may not be feasible. All in all, we believe we have completed all major requirements of PickEasy and are confident in this from our validation meetings.

Our validation process consists of attending the weekly Thursday meetings with PickEasy, but during these past three weeks due to many miscommunications with the PickEasy team giving us no notice they would not attend the Thursday meetings, we were only able to obtain one piece of feedback through email. We waited in three 1-hour Zoom meetings and messaged the PickEasy team, but all three times no one showed up, so we resorted to email feedback. It is unfortunate that the validation process went this way because we would have greatly benefited from weekly communications, but nonetheless, we were still able to get valuable feedback.

8/5/20 Minutes: the following are meeting notes taken on 8/5/20 with Max with the project demoed on an August 3rd commit, 526f499. A 5-minute demo video was sent to Max in an email and his response was in an email

- Questions and Answers (Bold is **Max's answe**r)
    - Feedback on reward templates
        - **None. Love it.**
    - Feedback on achievement templates
        - **Once again, none. love it!**
    - Is mobile functionality necessary?
        - **It is recommended but not necessary.**
    - Is Shopify POS necessary?
        - **No. This is good to have but if you can't figure it out in the limited amount of time given, leave it.**
    - How is the gamified loyalty system we designed?
        - **I think it's great. You guys have followed the KISS rule and we think that is extremely important**
- Main Takeaways
    - Our two main features of rewards and achievements of the loyalty system have been validated

Overview

Project Overview

In the beginning, we mainly focused on design and planning for our project. With deliverable 1, we mainly focused on team logistics, team roles, and general introductions related to the project. We started off by creating our team name and getting to know the strengths of each team member. With this information, we divided the roles of each team member accordingly. As we progressed in deliverable 2, we continued to focus on team roles as well as focusing on project scenarios and personas. We began going to meetings with our client pick-easy, so we could learn about the application they wanted us to create. With this information, we started creating personas and user stories to use as a plan for designing our project. After we finished with deliverable 2, we set up our project backlog, had our sprint plan, and began development. In deliverable 3 we began by developing the basic building blocks for the pick-easy app. We with the sign-in page and home pages as well as basic functionality. Nearing the end of deliverable 3, we created a basic recreation of the pick-easy app. At the beginning of deliverable 4 and after a meeting with the pick-easy team, we shifted our attention from designing a replica app to designing completely original gamification of achievements and rewards. Throughout deliverable 4 we implemented creating and generating achievements and rewards, as well as the ability to activate and track progression via QR code scanning. By the end of deliverable 4, we were able to finish the majority of the app's features. Finally, in deliverable 5, we continued to wrap up any features left on our product backlog and then shifted our focus to quality assurance and documentation. From there, we contacted pick-easy one final time to see if they had any feedback left for us. After the meeting and receiving their seal of approval, the actual development of the product had concluded.

Project Velocity

- What was your estimated project velocity?
    - Sprint 3
        - Estimated Project Velocity: 76 story points for 2 week
        - The estimated project velocity is based on the initial amount of story points we had in our burndown chart, which is calculated by the sum of story points for each team member's task. Our burndown chart can be found in the Deliverable5 folder in our Github repository.
        - Actual Project Velocity: 76 story points in 1 week and 6 days.
    - Sprint 4
        - Estimated Project Velocity: 57 story points for 1 week
        - The estimated project velocity is based on the initial amount of story points we had in our burndown chart, which is calculated by the sum of

        - story points for each team member's task. Our burndown chart can be found in the Deliverable5 folder in our Github repository.
        - Actual Project Velocity: 57 story points in 6 days
- How did your project velocity change with each deliverable?
    - As our team progressed through each deliverable, the workload we could handle in each sprint increased. During sprint 1 (Deliverable 3), our team only had 12 stories assigned because team members were familiarizing themselves with the software used to create our website. The number of tasks assigned was few and most of the focus was on organizing and creating a base for our project. Moving into sprint 2 (Deliverable 4), our story points assigned totalled 56 for 2 weeks, as the team became more comfortable using the development tools. Going into sprint 3 and 4 (Deliverable 5), the story points assigned totalled 76 for 2 weeks and 57 for 1 week respectively. As the deliverables progressed, the project organization improved, and as a result project velocity continued to increase.

Project Plan

In general, for deliverable 4 and 5, we stuck to our initial plans with very little deviation. During deliverable 3 we had to redesign our product backlog such that our user stories better fit the client's requirements. Once we did that, all future deliverables were completed according to the plan. We also redesigned the architecture, by including more subcomponents, removing unnecessary components, and correctly connecting dependencies between them. Since then, we've been using the same design for our architecture when implementing our product. Although we kept the same plan of what we were implementing because we needed to implement new features like a QR code scanner we needed to update our architecture for Deliverable 5 with this new component on both the restaurant staff and customer side. Most importantly, compared to the beginning of the project verse now, we completely redesigned our initial implementation of the pick-easy web app. Initially, we had a focus on building a replica of the pick-easy app. Based on the feedback of the pick-easy team, we realized we weren't satisfying the client's requirements, which resulted in the Deliverable 3 redesign. After better understanding the client's requirements, our focus turned to the gamification of the achievements and rewards. Once we made this adjustment, we were able to follow the remaining plans with little to no change.

Deliverable Progression and End Result

Deliverables 1 and 2 work progression was focused mainly on our project's design and organizing how work would be done. Deliverable 3 work progression had a bit of a halt, as a lot of redesigning was needed. Most of Deliverable 3 was focused on the basic functionality of our application. Deliverable 4 had a shift in focus to features such as gamification of rewards and achievements. Sprint 3 of Deliverable 5 also focused on gamification as well as mobile features. Sprint 4 of Deliverable 5 was solely for the quality of life improvements such as navigation, categorization, documentation, and visual improvements within our application. In general, the

final design of the product we have differs quite a bit from what we initially had in mind. Originally, we had our eyes set on redesigning and replicating the pick-easy app. After talking to our client and completely re-doing our user stories and architecture, we ended up with a final product that personifies what the initial intention of the project was (gamification of the achievements and rewards system). Compared to the beginning, deliverable 5 summarizes and displays the intention and vision of what the client wanted and is the amalgamation of all our work from all the previous deliverables. It is evident by our project velocity increasing after each deliverable that we were understanding the product better and how each other worked best. Overall, we are happy as a team with our final result and the progress we made each deliverable.