CSCC01 Project Deliverable #4

Group: Leap C

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

Table of Contents

Product Backlog

Changes

During deliverable 4, after deploying our first prototype and getting feedback from PickEasy, we realized we had to make some changes to the user stories in our product backlog to be focused more on the requirements given: a unique and gamified loyalty and rewards system. So we've done a major overhaul of our user stories to focus on two main users: restaurant staff and customers. In addition, each user has most of its user stories split into two major features: achievements and rewards. There are a few user stories not under these major categories, for example landing page, profile, and mobile support. The story point allocation method has stayed the same with the heuristic of developer hours. Lastly, the priorities are set with the idea that priority 2 user stories will be completed on sprint 2, priority 3 user stories will be completed on sprint 3, and so forth. So the lower the number, the higher the priority, where priority 1 is the highest priority and priority 4 is the lowest.

Backlog

Landing Page

- (Cost: 8) (Priority: 2) As a user, I want an easy way to distinguish between the restaurant staff and customer accounts.

Restaurant

- Achievements
    - (Cost: 4) (Priority: 2) As a restaurant staff, I want to be able set the number of tickets it takes for a customer to move to the next level or obtain a random reward
    - (Cost: 12) (Priority: 2) As a restaurant staff, I want to be able to create achievements from a template so that I save time by choosing premade achievement structures and only filling out the important details for example, "Order <item> <number> times"
    - (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
- Rewards
    - (Cost: 12) (Priority: 2) As a restaurant staff, I want to be able to create rewards from a template so that I save time by choosing premade reward structures and only filling out the important details for example, "Free <drink item>"
    - (Cost: 6) (Priority: 2) As a restaurant staff, I want to be able to create rewards only available at a certain level for example, "Free <meal item>" is a reward that would only be available to customers at the Diamond level
- Profile

    - (Cost: 6) (Priority: 2) As Jeff, a restaurant owner, I want to be able to associate/link my account with my restaurant so that I can update and make changes to my restaurant's page.

Customer

- Achievements
    - (Cost: 8) (Priority: 2) As a customer, I want to be able to display a list of achievements, a list of rewards, my tier level, and number of stamps obtained for each restaurant
    - (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".
- Rewards
    - (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

Mobile Support

- (Cost: 20) (Priority: 4) As both a customer or restaurant staff, I want to be able to access and utilize the achievement and reward functionality on my phone

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 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 8th
  - Release for D5: Aug 10th

Sprint Plan

Since this sprint is sprint 2, we chose all user stories with priority 2. In our backlog, we've broken down each user story into tasks. We've associated each task with a number so that it's clear who is assigned to which task. In our assignees, we've described who is doing what task

Backlog

Landing Page

- (Cost: 8) (Priority: 2) As a user, I want an easy way to distinguish between the restaurant staff and customer accounts.
    - Acceptance Criteria: Given I enter the PickEasy web app for the first time, when I load the landing page, then I should be prompted by two buttons to two different user home page routes: restaurant staff and customer
    - Tasks:
        - 1: Create new landing page (to navigate to PickEasy or PickEasy for Restaurants)
        - 2: Create restaurant dashboard UI
        - 3: Create restaurant sign up/in screen and modify consumer sign up/in screen

Restaurant

- Achievements
    - (Cost: 12) (Priority: 2) As a restaurant staff, I want to be able to create achievements from a template so that I save time by choosing premade achievement structures and only filling out the important details for example, "Order <item> <number> times"
        - Acceptance Criteria: Given a specific achievement template in my list of achievement templates is "Order <item> <number> times", when I click the template, then I should be prompted by three inputs: two inputs for item and number and one input for number of tickets to award
        - Tasks
            - 4: Create front end for restaurant's achievements
            - 5: Create back end for restaurant's achievements
    - (Cost: 4) (Priority: 2) As a restaurant staff, I want to be able set the number of tickets it takes for a customer to move to the next level or obtain a random reward
        - Acceptance Criteria: Given I have not set the number of tickets to level up, when I set the ticket threshold to five tickets, then it should save my choice
        - Tasks
            - 6: Create Number of Tickets Input

- Rewards
    - (Cost: 12) (Priority: 2) As a restaurant staff, I want to be able to create rewards from a template so that I save time by choosing premade reward structures and only filling out the important details for example, “Free <drink item>”
        - Acceptance Criteria: Given a specific reward template in my list of reward templates, “Free <drink item>”, when I click the template, then I should be prompted by one input for the name of the drink item
        - Tasks
            - 7: Create restaurant reward configurator in the front-end
            - 8: Create restaurant reward configurator in the back-end
    - (Cost: 6) (Priority: 2) As a restaurant staff, I want to be able to create rewards only available at a certain level for example, “Free <meal item>” is a reward that would only be available to customers at the Diamond level
        - Acceptance Criteria: Given the reward tiers bronze, silver, gold, platinum, diamond, when I click on the bronze tier, then I should be prompted to add reward under that bronze tier.
        - Tasks
            - 9: Create reward templates
- Profile
    - (Cost: 6) (Priority: 2) As Jeff, a restaurant owner, I want to be able to associate/link my account with my restaurant so that I can update and make changes to my restaurant’s page.
        - Acceptance Criteria: Given I’m creating the profile for my restaurant, when I click on a specific field on my restaurant card, then I should be prompted with an input to modify that field. For example, when I click on the cuisine field, then I should be prompted with a list of all possible cuisines, where I then can choose chinese as my restaurant’s cuisine
        - Tasks
            - 10: Create front end for restaurant profile
            - 11: Create back end for restaurant profile

Customer

- Achievements
    - (Cost: 8) (Priority: 2) As a customer, I want to be able to display a list of achievements, a list of rewards, my tier level, and number of stamps obtained for each restaurant
        - Acceptance Criteria: Given I’m searching for KFC’s reward info, when I click KFC’s card in the Discover page, then a dialog box should pop up and display the following info: name, photo, rating, price, tier level, number of tickets earned, list of achievements, and list of rewards
        - Tasks

- 12: Merge My Picks into Discover and add more details to the Discover page

Assignees

Luke
- Task 12

Evan
- Task 1
- Task 11

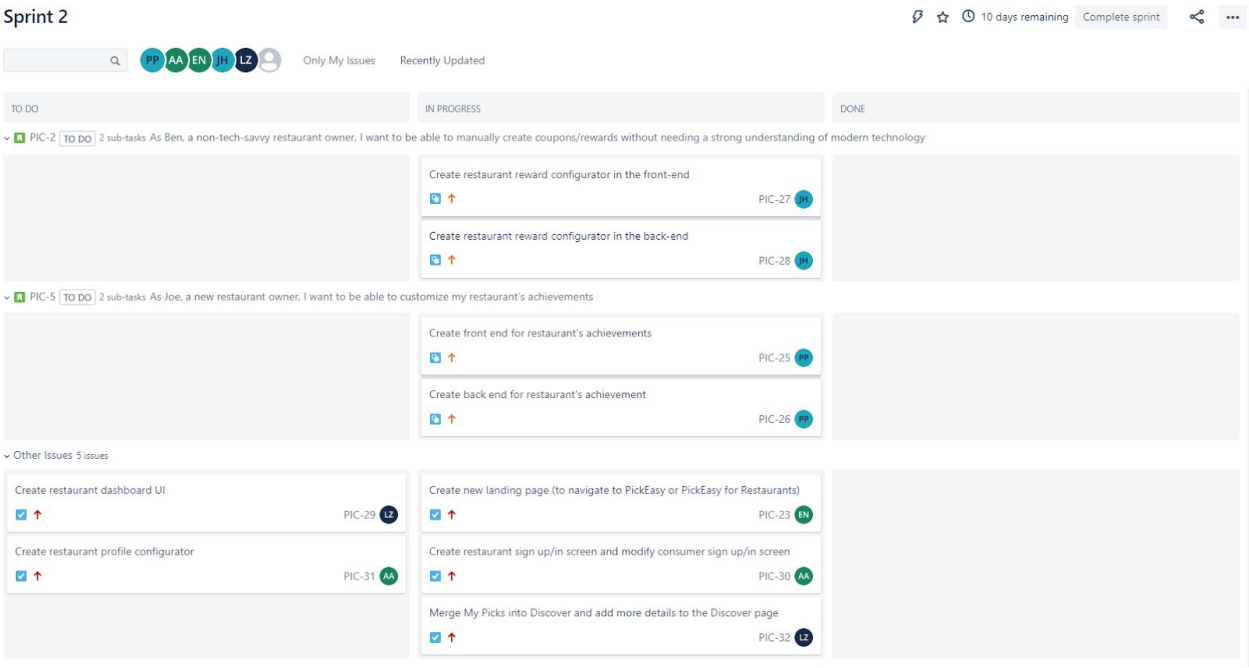
Anthony
- Task 2
- Task 3
- Task 10

Pravinthan
- Task 4
- Task 5
- Task 6

Chi Jian (Jeremy)
- Task 7
- Task 8
- Task 9

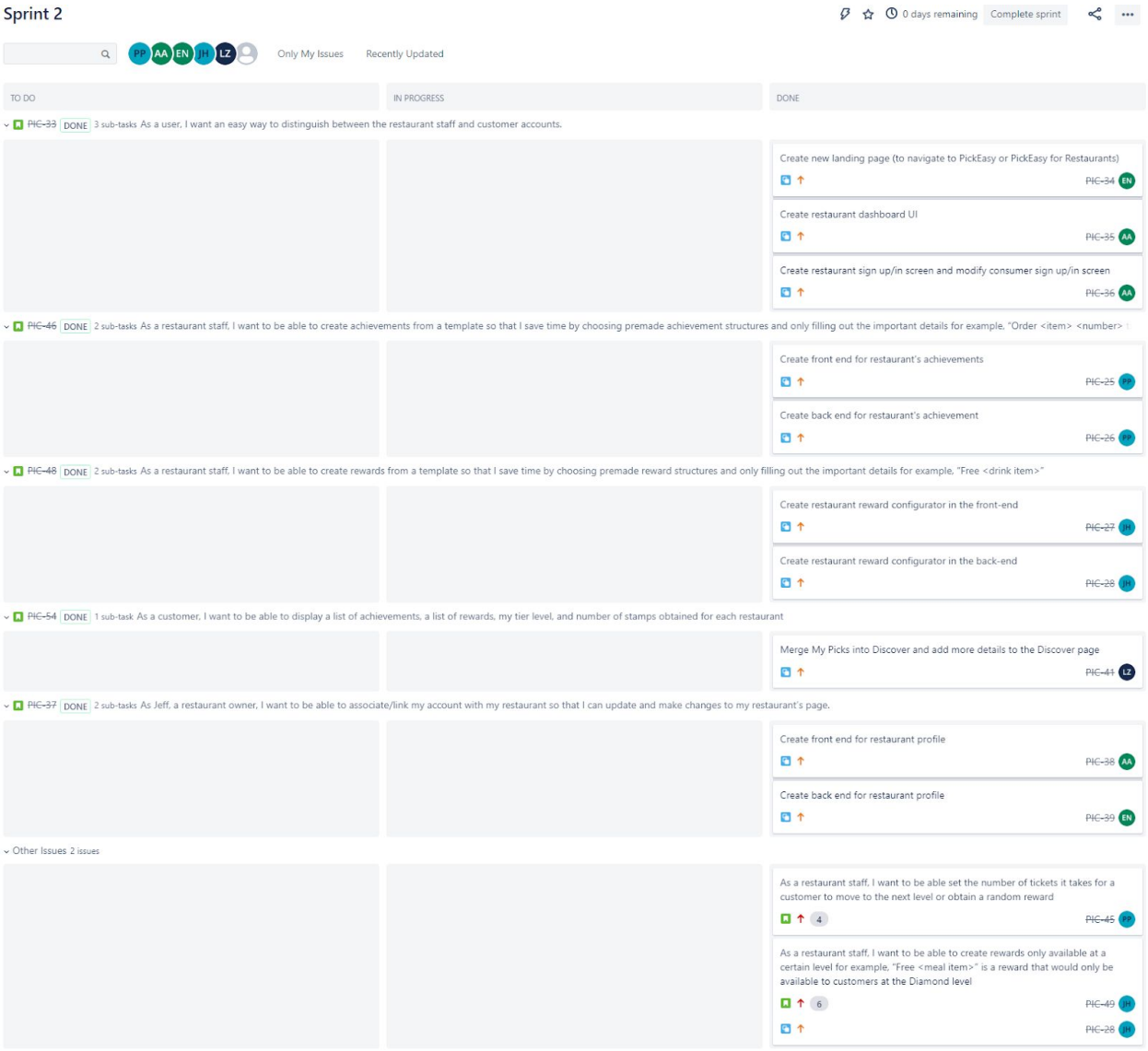
Task Board

Start of Sprint



[Link to Start of Sprint Task Board Snapshot]

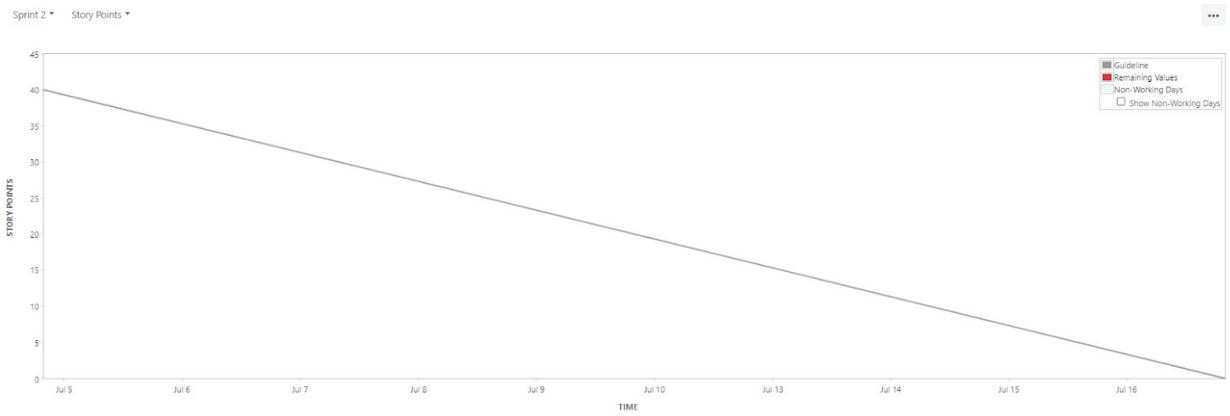
End of Sprint



[Link to End of Sprint Task Board Snapshot]

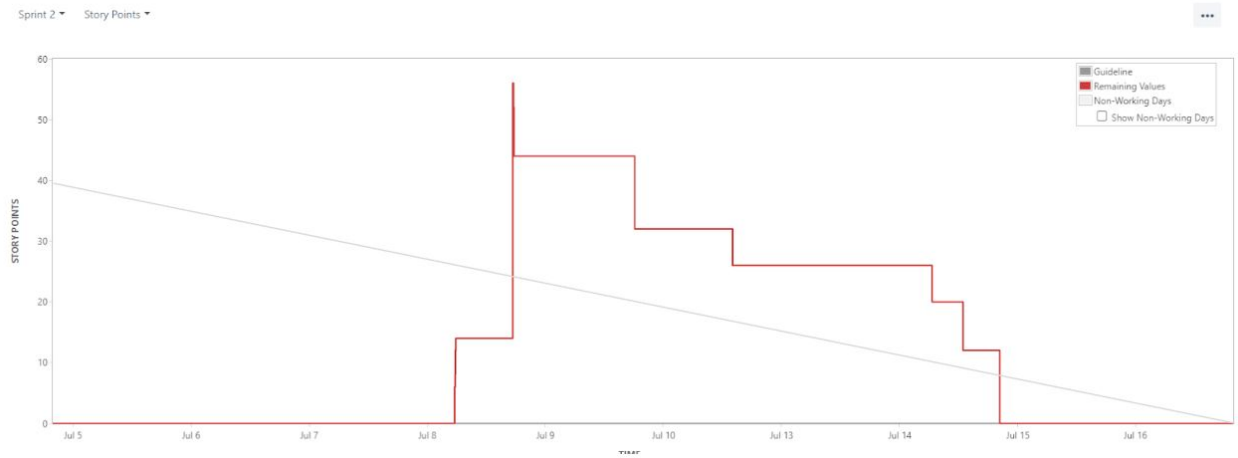
Burn Down Chart

Start of Sprint



[Link to Start of Sprint Burn Down Chart Snapshot]

End of Sprint



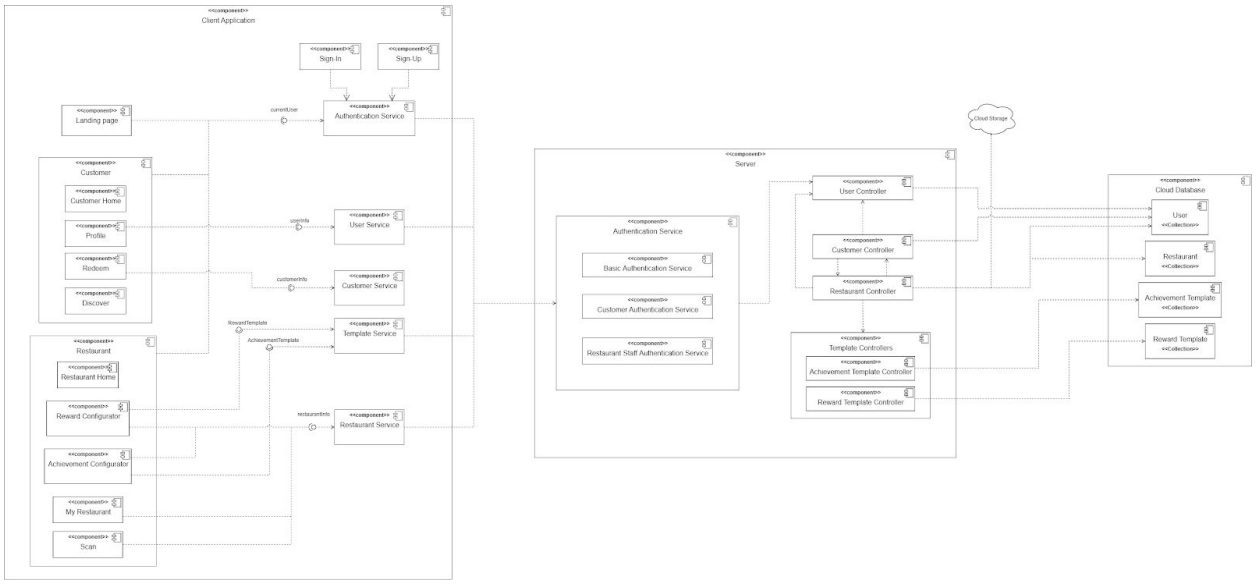
[Link to End of Sprint Burn Down Chart Snapshot]

Architecture

Changes

We completely overhauled our system architecture diagram for this deliverable. We have more fine-grained components that exemplify minimal coupling and maximal cohesion, and we have many services that we use for dependency injection. We split the main components into Client Application, Server and Cloud Database so that it is easier for us to manage dependencies. Finally, our diagram now focuses on UML dependencies rather than the flow of component interaction.

Diagram



[Link to Architecture Diagram]

Explanations

Client Application

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 four subcomponents (Customer Home, Profile, Redeem, 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.

Customer Home Component

The Customer Home Component is responsible for allowing users to sign in or sign up as a customer. Once signed in, this component provides access to the three other main components (Profile, Redeem, and Discover) through tabs and address routing.

Profile Component

The Profile Component is responsible for showing the customer's information such as username, first name, and last name. It depends on the User Service Component to receive the necessary information.

Redeem Component

The Redeem Component is responsible for allowing customers to present their QR code for restaurant staff as well as showing customers the rewards they have accumulated from different restaurants. It depends on the User Service Component to receive the necessary information.

Discover Component

The Discover Component is responsible for displaying the list of restaurants to the user. There will be some extra features too such as preference-based suggestions of restaurants, filtering and search functionality, and sort functionality.

Restaurant Component

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

Restaurant Home Component

The Restaurant Home Component is responsible for allowing users to sign in or sign up as a restaurant staffer. Once signed in, the Restaurant Home Component provides access to the four other main components (My Restaurant, Achievements, Rewards, and Scan) through tabs and address routing.

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. 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. 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, logo, name, description, cuisine, and cost. This component depends on the Restaurant Service Component to retrieve the aforementioned information.

Scan Component

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

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.

User Service Component

The User Service Component is responsible for retrieving advanced user information such as first name and last name from the back-end. This component depends on the Server's Authentication Service Component to ensure authorization when accessing relevant 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, logo, 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. This component and all its subcomponents depend on the User Controller Component to retrieve the current user's session 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 uses a given user ID and retrieves a corresponding user from the Cloud Database

Customer Controller Component

The Customer Controller Component is responsible for retrieving and updating customer-specific information such as achievement progress and obtained reward. This component depends on the User Controller Component and 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, logo, name, description, cuisine, cost, achievements, and rewards. This component depends on the User Controller Component and 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 retrieve restaurant logos and 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, and type of user (Customer or Restaurant Staff).

Restaurant Collection

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

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' logos and 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.

Dependencies

Any dependencies are listed along with the explanations above.

System Validation

The following are the three validation meetings Leap C attended with PickEasy to ensure that our implementation met the client's needs. Each of the meeting minutes contains everything that was talked about, so for an overview of the minutes scroll to the bottom under the "Main Takeaways" heading to see the benefits of the meeting summarized.

7/16/20 Minutes: the following are meeting notes taken on 7/16/20 with Daniel with the project demoed on a July 16th commit, 18c67e8

- Needs Work
    - Restaurant Staff
        - Drag picture in when setting up the restaurant would be nice, but a would like to have
        - Maybe duplicate tier rewards in rewards page for restaurant staff
            - Maybe categorize tier rewards for visibility
        - The onboarding could flow better, after clicking save it should be quicker. The click economy is not that good, and should automatically bring you to achievements. Seeing the number of tickets, it is not intuitive that it is the same.
        - Add a subtitle so that what you click is exactly what you chose, because the input fields make it confusing
        - Might be more intuitive to group similar achievements together. Should be able to group it visually, so it's clear
        - Number of tickets a customer must collect to receive a reward or progress to the next tier". Try to explain this a bit better
        - Rewards seem detached from achievements. Hard to see that achievements are related to rewards.
        - Number of tickets to get a reward (this would be a coupon). Instead of "purchase" to how. Visually should be lower to higher. Maybe the number of tickets should be on the rewards slide?
        - More intuitive that better rewards need more tickets to claim. Maybe can let restaurant users claim a certain reward, but still keep the tier system for the rewards. Could spend number tickets for each level of the rewards be different (so it feels like the user is getting a better reward). In Starbucks you spend more points for a better reward. Should still be some difference to claim better or worse rewards
        - In general explain how the system works better
    - Customer
        - No way to see available rewards for customers. Have what you're eligible all in one place, so they don't have to remember. It's crucial to have all the rewards in one place. It would be a plus to be able to see the rewards

in the past, but not is important. From both user side and customer side and do a better job explaining how the program works, it is very open-ended. There should be a bit more guideline on how restaurants should do tiers, that it should be progressing on how good the rewards are

- Questions Asked
    - Does our system meet your requirements?
- Main Takeaways
    - The Air Miles inspired system could be more intuitive with spending more tickets leading to better rewards.
    - Additional features should be implemented such as: seeing available rewards for customers, and categories for achievements and rewards

7/9/20 Minutes: the following are meeting notes taken on 7/9/20 with Choyin with the project demoed on a July 9th commit, a376ce8

- Needs Work
    - Branding the word stamp in “number of stamps” on the achievements page for restaurant staff should be better, so it’s clearer that obtaining stamps leads to a reward. Maybe tickets?
    - Use the word customer instead of consumer. Use the word restaurant staff instead of restaurant owner. If you have time, separate owner and staff responsibilities.
- Good
    - Achievement and rewards look good!
    - Air Mile inspired system is good
    - Redeeming QR codes for redeeming coupons okay. For implementation, manual checks are totally fine, as long as it can be used by the restaurant. Doesn’t have to be automatic
- Additional Notes
    - None, no need to think about scalability projects for C01 project
    - Scalability: number of users is 1,000-10,000. How would this change our design, we don’t know. Out of scope. Ask our TA again why we have to address scalability. Apply design patterns instead, like SOLID. Highly important to follow SOLID, follow Agile methodologies, Angular singleton (lots of services)
    - Security: out of scope. Ask our TA again why we have to address security. Don’t spend over 2 hours on authentication. Username and password should be super easy. Doesn’t have to be super secure, not the worries of this course. User has to be identified
- Questions Asked
    - What is some advice you can give us about the potential technical challenges with scalability?
    - What is some advice you can give us about the potential technical challenges with security?

    - Does our system meet your requirements?
- Main Takeaways
    - Continue with Air Mile inspired loyalty system
    - Clarify terminology and refactor the names of some components, such as stamps, consumer, and restaurant owners

7/2/20 Minutes: the following are 7/2/20 meeting notes with Choyin from PickEasy, where we demoed our project which was deployed on pick-easy.herokuapp.com on a June 30th commit, 3a89468e47b49cf0d2b9bc422bccb12cf6cffe71

- Needs Work
    - Choyin said they don't want a recreation of their PickEasy app, instead, they want more of a focus on gamified achievements. Additionally, they don't want the loyalty system to just be a "per visit" system like every other restaurant is using nowadays. Instead, they want a gamified achievement system with levels, that is unique, something we would enjoy using as university students
    - The Discover page is not what they want since it has nothing to do with achievements. This can be refactored to be used for achievement searching instead
    - The restaurant owner and customer module should be completely separate. They don't want a checkbox to signify you're a restaurant owner, instead, they want two completely separate sign-in modules. They prefer if the two are two separate web apps, but combined is okay but not preferred. We will probably implement two separate sign-in screens for restaurant owners and consumers
- Good
    - The video that plays in the background looks cool
    - QR code scanning for achievements is a good idea
- Additional Notes
    - Style is not important. Functionality is the highest priority. Don't spend more than an hour on the CSS
- Questions Asked
    - Thoughts on QR code scanning for loyalty system implementation?
    - Does our system meet your requirements?
- Main Takeaways
    - Merge Discover and My Picks component on the Consumer view to be a singular Achievements component
    - Rethink our loyalty system of 5 visits for a reward to be more unique, such as using stamps to either get a random reward or level up
    - Create separate sign-in interfaces for restaurant owners and consumers

Overview

Project Progression

At the beginning of the deliverable, our team had to redesign our product backlog such that our user stories better fit the client's requirements. We also redesigned the architecture, by including more subcomponents, removing unnecessary components, and correctly connecting dependencies between them. We created the interface for the restaurant staff components and connected it to the customer side of the application. Next, we created a landing page to differentiate between restaurant staff and customers, directing the user to their correct page. We also implemented our reward and achievement pages on the restaurant staff side. The restaurant staff is now able to create and add rewards and achievements using templates. Additionally, we implemented the my-restaurant component, where restaurant owners can update their restaurant's picture, description, cuisine, and cost. Lastly, we merged the discover and my-picks pages on the customer side, removing any unnecessary functionality between the two.

The difference in work done in D4 in comparison to D3 is quite substantial. In D3 we focused on front-end features with the focus on how certain pages looked. In D4 we decided to focus on new features and the backend development. In addition, for this deliverable, we decided to move from mockup data to an actual database. For D5 since we have mastered creating user stories and we have overhauled our architecture, now our main focus can be solely on developing the application and meeting the client's requirements. We aim to implement more features in D5 than what was completed during D4.

Project Velocity

- What was your estimated project velocity?
    - Estimated Project Velocity before sprint backlog changes: 40 story points for 2 weeks
    - 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 Deliverable4 folder in our Github repository.
- What was your actual project velocity?
    - Actual project velocity after sprint backlog changes: 56 story points for 2 weeks
    - The reason why our actual project velocity is greater than our estimated project velocity is due to changes in our sprint backlog that we detailed earlier in this report. The actual project velocity is based on the number of story points after our changes to our sprint backlog, which is calculated by the sum of story points for each team member's task. Our burndown chart can be found in the Deliverable4

folder in our Github repository. Since our sprint spans two weeks and our deliverable 4 is due after our last sprint day, we were able to complete all of our tasks a bit ahead of the projected velocity.