CSCC01 Project Deliverable #3 - Part 1 & 2

Group: Leap C

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

Table of Contents

NOTE: NEW CHANGES IN PART 1 ARE HIGHLIGHTED IN YELLOW

Overview of Part 1

- What tools will you use for your task board?
    - Jira
- What tools will you use for your burn-down chart?
    - Jira does this automatically for us
- Who will maintain the burn-down chart? How?
    - Pravinthan through Jira
- What is every team member's role?
    - Scrum Master: Evan
    - Lead Developer: Prav
    - Product Owner: Luke
    - Developers: Anthony and Jeremy

  Lead Developer (Prav): We chose Pravinthan for this role as our Lead Developer for his past experience with developing angular based projects. In addition, as lead developer, Pravinthan will be in charge of code reviews, pull requests, and helping anyone who is having difficulty during development.

  Scrum Master (Evan): We chose Evan for the role of Scrum Master based on his past experience as a Scrum Master in prior projects. As Scrum Master, Evan will be in charge of monitoring and time managing the team's progress as well as helping anyone who is having any blocks in development.

  Product Owner (Luke): We chose Luke for the role of Product Owner based on his leadership experience. As a Product Owner, Luke will be in charge of attending client meetings and addressing any issues he has found to the team. In addition, he will also be in charge of assigning tasks and moving tasks around in order to keep the team on track. He will also be in charge of writing meeting minutes.

  Developers (Anthony and Jeremy): As developers, Anthony and Jeremy will be working solely on the code of our projects under the supervision of our Lead Developer, Pravinthan.

- What tools, if any, will you use for communication?
    - Discord, FB Messenger
- When do you plan to meet in person?
    - Twice a week Wed and Sat
- How will you use your repository on GitHub?
    - One folder for each deliverable
    - One folder for development

- Which machines will be used for development by each team member? E.g., a CMS virtual machine, a Linux laptop, a Windows home computer, etc.
    - Windows: Evan, Jeremy, Prav
    - Linux: Prav
    - Mac: Luke, Anthony
- What is your DoD (definition of done)?
    - Unit testing
    - Acceptance testing, we will do a demo with PickEasy every Thursday to ensure the validation of our requirements.
    - Code reviews, Pravinthan, our Lead Developer will review our code during our two weekly meetings on Saturday and Wednesday.
    - Code compiles

NOTE: We will start holding daily scrum meetings from June 28, 2020, onwards.

Product Backlog

1. (Priority: 1) (Cost: 5) As Ben, a non-tech-savvy restaurant owner, I want to be able to manually create coupons without needing a strong understanding of modern technology, so I can have better control of the business.
    a. Acceptance Criteria: Able to create coupons manually with ease and simplicity.
2. (Priority: 2) (Cost: 4) As Steven, a student, I want to be able to search up any restaurants that are offering discounts and deals, so that I can easily find cheap and affordable food at reasonable prices.
    a. Acceptance Criteria: Able to search for any restaurants that are offering deals
3. (Priority: 3) (Cost: 6) As Jeff, a business owner, I want to be able to see the customer's attendance by just scanning their apps.
    a. Acceptance Criteria: Able to scan customers' phones and record their visit for the purposes of loyalty tracking.
4. (Priority: 4) (Cost: 4) As Jenny, a busy user, I want to be able to view all the coupons that are currently available to me, so that I can make quick decisions on which restaurant to choose.
    a. Acceptance Criteria: Able to view all coupons for the available restaurants that I am near.
5. (Priority: 5) (Cost: 8) As Steven, a student, I want to be able to look at my account and track my loyalties so that I can see what benefits I can receive.
    a. Acceptance Criteria: Able to check account options and track loyalties and benefits towards restaurants.
6. (Priority: 6) (Cost: 4) As Tim, a restaurant owner, I want to be able to delete existing coupons at any point and not cause any confusion among customers who wanted to redeem them beforehand so that I will not get a bad reputation among any future customers.
    a. Acceptance Criteria: Able to delete any coupons that are available for customers.
7. (Priority: 7) (Cost: 7) As Steven, a student, I want to be able to view ratings of any restaurant I want and rate them myself, so I can learn about the restaurant's quality before having to visit.
    a. Acceptance Criteria: Able to rate and view ratings for different restaurants.
8. (Priority: 8) (Cost: 2) As Sammy, an avid consumer, I want to have my coupons sorted by most percentage saved so that I can save money when I eat out seven times a week.
    a. Acceptance Criteria: Able to sort restaurants by most savings in the restaurant list.
9. (Priority: 9) (Cost: 3) As Brock, an active user, I want an option to view the coupons that a restaurant offers before I choose if I like it or not, so I can see if there are items that fit my budget.
    a. Acceptance Criteria: Able to view coupons during the “Discovery” screen in order to judge whether I like the restaurant or not.

10. (Priority: 10) (Cost: 10) As Bo Ling, a power user and a restaurant owner, I want to randomly generate coupons from my menu given a max percentage discount so I don't have to manually enter each individual coupon myself.
    a. Acceptance Criteria: Coupons should have the option to be randomly generated given a percentage discount range.
11. (Priority: 11) (Cost: 9) As Jeff, a business owner, I want to be able to automatically assign flat-rate discounts, so that I can increase the loyalty and consistency of my customers.
    a. Acceptance Criteria: Coupons should be able to be automatically generated using a flat-rate discount.
12. (Priority: 12) (Cost: 15) As Joe, a new restaurant owner, I want to be able to automatically customize my restaurant's achievements based on customer feedback, so those achievements are focused on the customer's wants/needs.
    a. Acceptance Criteria: Achievements should be automatically generated based on customer feedback.
13. (Priority: 13) (Cost: 12) As Jimmy, a frugal consumer, I want to be able to collaboratively collect points and share rewards with another user so that I am able to maximize savings.
    a. Acceptance Criteria: Points and rewards should be able to be shared across multiple users.
14. (Priority: 14) (Cost: 7) As Brock, an active user, I want a feature that will introduce me to new restaurants weekly, so I can explore options I may not have known about before.
    a. Acceptance Criteria: Restaurants should be able to be offered to the user based on user restaurant preferences.

Release Plan

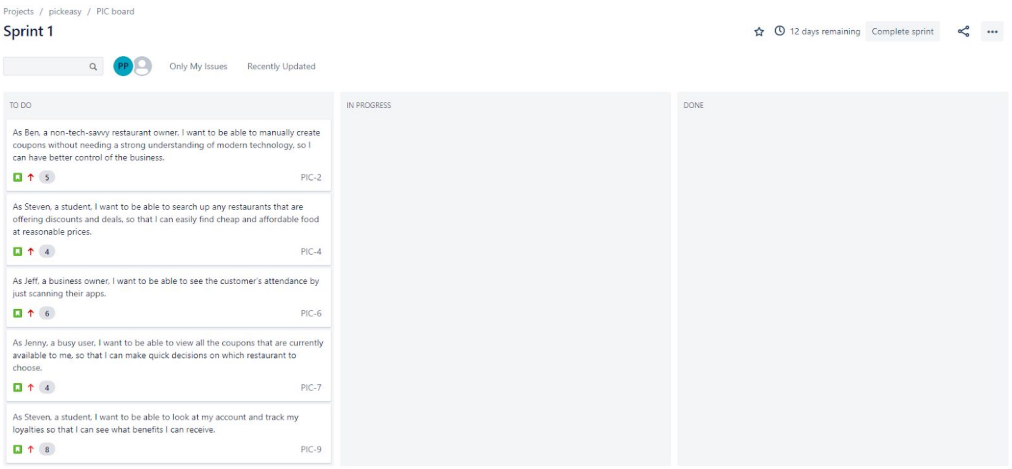
- Choose the length of your sprints and explain your choice.
    - Two weeks because we want to have multiple sprints before the end of the class and any longer would not let us have multiple sprints
- Notice that, whatever duration you choose for your sprints, you must have a release on each deliverable's due date.
    - 1st Sprint: Sat, June 20th
        - Release by: June 29th
    - 2nd Sprint: Sat, July 4th
- What are your rules for what's being committed (eg. autogenerated files?)
    - We have a .gitignore file that does not allow secret files, ide files, node dependencies, and compiled output files to be committed.
- What are your rules for what's being included in commit messages (what was done, progress, format, etc)
    - Each commit should contain a logical increment of progress with a description of what was done
- What are your rules for branch names (eg. user story/task id)?
    - We will use the Task ID for our branch names and master for our production branch (e.g. PIC-19 would be the task ID automatically generated by Jira and we would just use this as the name of our branch. PIC is the first three letters of PICkeasy and then 19 is the unique number Jira allocates)
- What methodology will you be following for code style and documentation?
    - Code Style: Prettier. We chose this code style to follow because it is simple for us to use since all of us have agreed on adopting Visual Studio Code as our IDE
    - Documentation: https://www.writethedocs.org/guide/. We chose this documentation to follow because it will be helpful for us to reference whenever we feel lost on how to write documentation
- Where are standalone tasks? (eg. set up server, database, etc)
    - We separated between frontend and backend development. Jeremy and Pravinthan are focusing on the backend, setting up the server and database. Luke, Evan, and Anthony are working on the frontend.
- What tasks specifically accomplish these user stories?
    - We now have subtasks for each task that, when completed, accomplish the user story
- Make sure you can explain how you calculated the story points /priorities for each task.
    - We calculated story points based on developer-hours and priorities of each task based on our discussion with the Pickeasy team

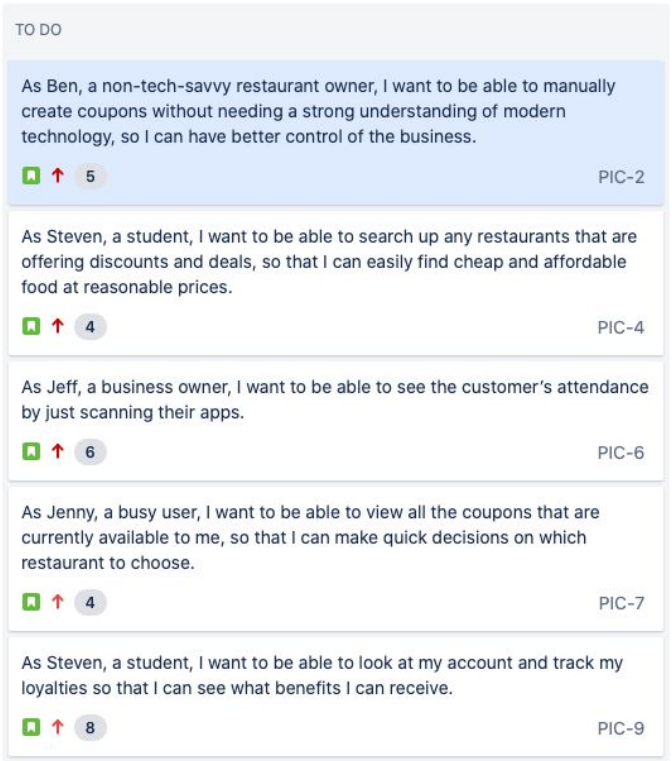
Sprint Plan

- Pravinthan:
  - **User Story:** As Steven, a student, I want to be able to search up any restaurants that are offering discounts and deals, so that I can easily find cheap and affordable food at reasonable prices.
  - **Sub Task:** Create backend systems to be able to search for restaurants
  - **User Story:** As Steven, a student, I want to be able to look at my account and track my loyalties so that I can see what benefits I can receive.
  - **Sub Task:** Create an account system in the backend (and sign in/up UI)
  - When: June 20th
- Anthony:
  - **User Story:** As Steven, a student, I want to be able to look at my account and track my loyalties so that I can see what benefits I can receive.
  - **Sub Task:** Create user interfaces for account system
  - When: June 20th
- Luke:
  - **User Story:** As Steven, a student, I want to be able to search up any restaurants that are offering discounts and deals, so that I can easily find cheap and affordable food at reasonable prices.
  - **Sub Task:** Create a user interface that allows users to search for restaurants.
  - When: June 20th
- Jeremy:
  - **User Story:** As Steven, a student, I want to be able to look at my account and track my loyalties so that I can see what benefits I can receive.
  - **Sub Task:** Create a loyalty system in the backend
  - When: June 20th
- Evan:
  - **User Story:** As Steven, a student, I want to be able to look at my account and track my loyalties so that I can see what benefits I can receive.
  - **Sub Task:** Create user interfaces for a loyalty system
  - When: June 20th

Task Board and Burndown Chart

After starting our first stories, we realized that we bit off more than we can chew. So we decided to only pick the PIC-9 and PIC-4 stories shown below. Additionally, below our original two pictures of our task board, we have our updated task board with the stories broken down into sub-tasks. Our initial burndown chart did not change and the new burndown chart can be seen in our deliverable 3 folder on Github.





PIC-4 TO DO 2 sub-tasks As Steven, a student, I want to be able to search up

Create backend systems to be able to search restaurants

PIC-20 PP

Create a user interface that allows users to search restaurants

PIC-21 LZ

PIC-9 TO DO 4 sub-tasks As Steven, a student, I want to be able to look at my

Create an account system in the backend (and sign in/up UI)

PIC-16 PP

Create a loyalty system in the backend

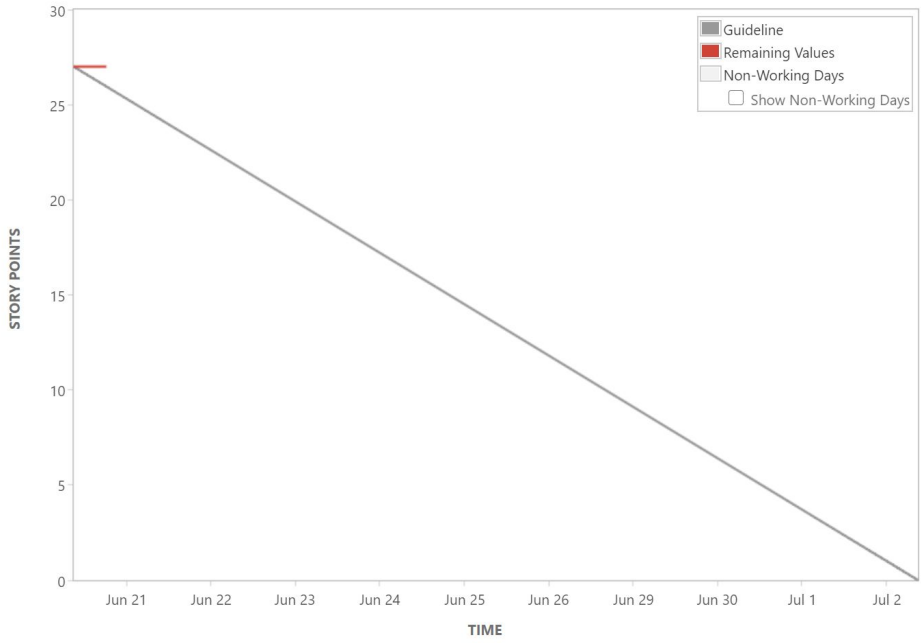
PIC-17 JH

Create user interfaces for account system

PIC-18 AA

Create user interfaces for loyalty system

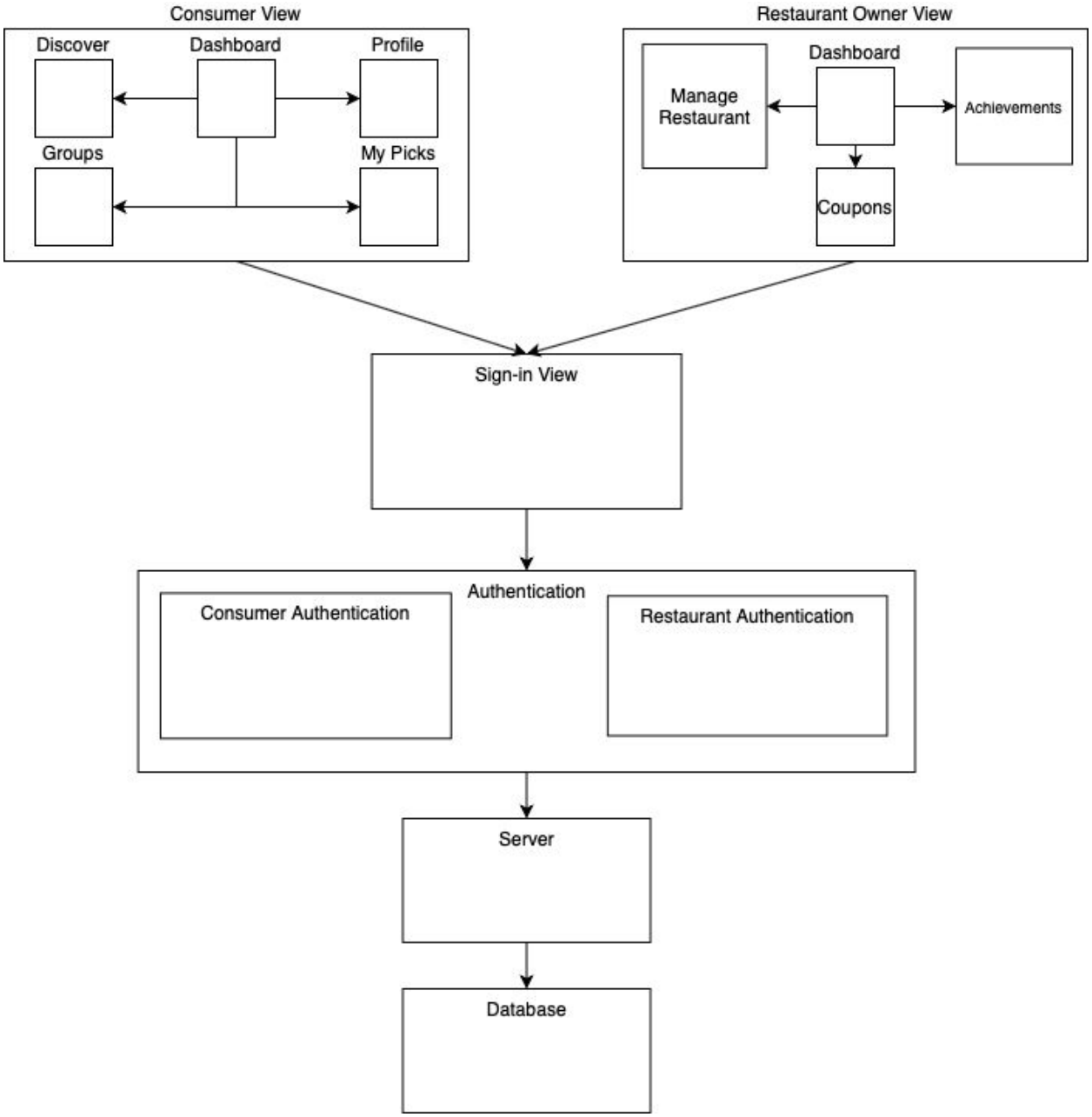
PIC-19 EN



Overview of Part 2

- Submit a brief overview of your project as it progressed from deliverable 2 to deliverable 3.
    - One of the first challenges we faced is taking on too many stories at once, but we quickly resolved this issue on the first day of our sprint. We decided on developing a web app using the MEAN stack to build a loyalty system for the Pickeasy team because we have more experience as a team in web development than mobile development. Under the guidance of our lead developer, Pravinthan, we were able to get a fully functioning web app started that is connected to a local database. In our deliverable 3 release, we have implemented login functionality, a home page, a basic restaurant loyalty system, and a profile interface in our web app. So to summarize, from deliverable 2 to deliverable 3 we picked a few user stories from the personas we created from deliverable 2 and implemented those features in a web app.
- What was your estimated project velocity?
    - Estimated Project Velocity: 27 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 Deliverable3 folder in our Github repository.
- What was your actual project velocity?
    - Actual project velocity: 15 story points for 1 week
    - Since our sprint spans two weeks and our deliverable 3 is due in the middle of the week, we were only able to complete the first week of our sprint. In one week, we have completed 15 story points. This leaves us with 12 points left to complete in the next week. Considering we have completed 15 in a week, we think we can achieve our estimated project velocity of 27 story points in two weeks.
- Did you follow your plan(s) exactly? What difficulties have you encountered? Was your contingency plan useful at that point or did you have to come up with a new solution?
    - We had trouble, in the beginning, implementing the features of our user stories because we needed an existing basic loyalty system on our app to build upon. The main difficulty we encountered was taking upon more than we could actually do, so we had to tone down the number of user stories to complete in our sprint, and instead, break down two user stories into many tasks. Our contingency plan was useful because it gave us a better idea of what mistakes we had in our solution. For example, we found that often we had to change the design of our architecture; at the beginning, we started with three components in our consumer view, but as we started implementing we ended with five components. This has been a great learning experience for all of us because we are learning to realize what it means for change to be inevitable while working in an Agile team.

Architecture Diagram



Architecture Explanations

Sign-in View

The sign-in view is responsible for taking the inputs for new and existing users. These inputs include username, password, and type of user (consumer or restaurant). The screen then passes these inputs to be verified by the authentication component. All the front-end

components are built using Angular. The entire project uses the MEAN stack which stands for MongoDB, Express.js, Angular, and Node.js.

Consumer View

After a consumer user logs in, the sign-in view will pass responsibility to the Consumer View which consists of five main components: Dashboard, Discover, My Picks, Groups, and Profile.

Dashboard

The Dashboard component provides access to the four other main components (Discover, My Picks, Groups, and Profile) through tabs and address routing.

Discover

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.

My Picks

The My Picks component is responsible for displaying detailed information about restaurants such as its name, description, picture, cost, rating, menu, loyalty, and achievements.

Groups

The Groups component is responsible for the creation and management of groups. This component is subject to change because it was not a requirement of the PickEasy team.

Profile

The Profile component is responsible for showing the current user's information such as name and username.

Restaurant Owner View

After a restaurant owner logs in the sign-in view will pass responsibility to the Restaurant Owner View which consists of four components: Dashboard, Manage Restaurant, Coupons, and Achievements. These components are different from the Consumer View's components because the restaurant owner will be the one customizing their loyalty, coupons, and achievements.

Dashboard

The Dashboard component provides access to the three other main components (Manage Restaurant, Coupons, and Achievements) through tabs and address routing.

Manage Restaurant

The Manage Restaurant component is responsible for displaying and providing editable information about the restaurant such as its name, description, picture, cost, menu, loyalty, and achievements.

Coupons

The Coupons component is responsible for displaying active/inactive coupons and allowing creation, modification, and deletion of coupons. Restaurant owners will also be able to automatically generate coupons based on common user preferences.

Achievements

The Achievements component is responsible for displaying active/inactive achievements and allowing creation, modification, and deletion of achievements. Restaurant owners will be able to customize how consumers interact with the achievement tailored to their restaurant.

Authentication

The Authentication component encapsulates the consumer and restaurant authentication components. It ensures that users are logged in at the very basic level and passes individual requests through the consumer and restaurant authentication components to ensure correct authorization.

Consumer Authentication

The Consumer Authentication component is a server component that ensures that consumers are authenticated when trying to access their own information such as name, username, loyalty, and achievements as well as information about restaurants.

Restaurant Authentication

The Restaurant Authentication component is a server component that ensures that consumers are authenticated when trying to access and edit their own information such as restaurant name, description, picture, cost, loyalty, and achievements as well as the management of loyalty for individual users through bar code scanning.

Server

The Server component uses Node.js as our main back-end runtime that will serve the Angular Application. The Server component receives HTTP/S requests and, assuming the user is authorized, responds using data from the database.

Database

The Database component is built using MongoDB and will store all information about the consumers and restaurants.

Architecture Dependencies

- **The Consumer View and Restaurant Owner View depend on Sign-in View:** These views require the user to be signed in as either a consumer or a restaurant owner.
- **The Sign-in View depends on Authentication**: The input requests from the front-end are passed into the Authentication component to ensure the correct information is being accessed.
- **The Authentication depends on Server**: The Authentication component sends HTTP/S requests to the server in order to verify the user is signed in.
- **The server depends on Database**: The Server component acts as the API for the Database component so that the Authentication, Consumer, and Restaurant Owner components' business logic can access data that is in the database. It obtains HTTP/S requests and authorization information to ensure the database is accessed safely.