DELIVERABLE 3

PART 1 + 2

Team_01(GitHub)
Anson Feng
Jiawei Qiao
Vanessa Pierre
Yolanda Su

Table Of Contents

Project Setup

What tools will we use for our task board?

We will use Trello to set up and manage our task board.

What tools will we use for our burndown chart?

We will use Google Sheets to set up and manage our burndown chart.

Who will maintain the burn-down chart? How?

The scrum master (Yolanda) will maintain the burn-down chart. The scrum master will update the charts data at the end of every working day with the most current number of story points left.

What is every team member's role?

Yolanda is our Scrum Master and will manage the task board and burndown chart. She is one member of the front-end developers of our team. She is responsible for holding meetings as well. Vanessa is responsible for the final editing of deliverable documents so that they look professional, and she is the other member of the front-end developers. Anson is the Product Owner and takes the role of contacting the client/TA, and he works on the back-end for our team. Jiawei is responsible for updating the GitHub repo README and uploading deliverable documents, and he is the other member of our teams' back-end developers. Everyone in our team takes turns recording meeting minutes.

Adjustments for the front-end and back-end developer roles will be made every sprint based on the stories worked on in that sprint and the decision of the team's pairs.

The team will be split into pairs of 2, each pair will have 1 person mainly responsible for the front-end and 1 person mainly responsible for the back-end, and the pair will decide among themselves who fills which role. Pairs will work on the same stories (e.g. one person in the pair works on the front end of a story, the other works on the back end of the same story). Pairs are also responsible for reviewing each other's code.

If there are fewer back-end points on a team than front-end points, the back-end developer will help with front-end points after finishing back-end work, and vice versa if there are less front-end than back-end points. If a card is still incomplete by the end of the sprint, the original assignees to those cards will first complete the leftover work before moving on working with their new partner for the next sprint.

<u>Team member pairing cycle (cycle repeating every 3 sprints):</u>
Yolanda + Anson, Vanessa + Jiawei
Yolanda + Jiawei, Anson + Vanessa
Yolanda + Vanessa, Anson + Jiawei

What tools, if any, will we use for communication?

We will use Slack for general online discussion and Zoom for team meetings.

When do we plan to meet in person?

Daily stand up meetings will take place on Zoom, these meetings will be held from 5:30 - 5:45 pm. During these meetings, members will discuss their accomplished daily tasks or issues that are holding them back from competing for their daily tasks.

Weekly team meetings will take place on Wednesdays from 6:00 - 7:00 pm. This meeting will act as our midsprint team check-in meeting. During this weekly meeting members will go more in-depth with how the progression of the application is proceeding so that we may revise our sprint plan and estimates as needed. We also have retro meetings on Sundays at 1:00 - 3:00 pm, or longer if needed. Additional meeting times will be added when necessary based on team discussion.

TA meetings will take place on Thursday from 6:00 - 7:00 pm.

How will we use our repository on GitHub?

We will develop features on individual branches, and will push branches to git after functionality is complete. We will push bug fix commits on a feature to that feature's branch if the branch has not yet been merged into master, and or we will push to a new bugfix branch if the original feature branch is already merged.

We will merge branches to master after they have met our definition of done (DoD).

Commits must abide the following:
1. Only commit the source code of the file to have regression tests, and never commit unnecessary files.
2. A single commit should only include one completed feature (function) so that it is easy to test and revert if any problems occur.
3. A commit message should have the first character in uppercase, using the past tense, and be less than or equal to 50 characters. It should describe what changes we are committing briefly, and should not describe specific lines of code.

For naming the branches, we will use task ID to name the branch as it is easier to read than the task name.

Which machines will be used for development by each team member?

Yolanda and Anson both use Windows home computers, Vanessa uses a Macbook, Jiawei has and can use both of them.

What is our DoD (Definition of Done)?

For us to consider a story "done", all subtasks of that story must be complete. For us to consider a task "done", completed code must adhere to the Google style guide. All code must also be peer reviewed by at least 1 other team member. The code must

be able to be built successfully. The code must also pass all automated tests relevant to the task's acceptance criteria, as well as pass all previously completed automated tests from related features.

Project Backlog

1 point = 1 working hour

Persona 1: David (Restaurant Owner)

Type	Priority	Story	Back-end Points	Front-end Points	Total Story Points
Story	Critical	As David (restaurant owner), I want to be able to register my account.	6.5	4	11.5
Story	High	As David (restaurant owner), I want to create a coupon with my selected requirements.	4	3	7
Story	High	As David (restaurant owner), I want to be able to see all my restaurant's current coupons.	1	6	7
Story	High	As David (restaurant owner), I want to be able to delete an existing coupon.	1	1	2
Story	Medium	As David (restaurant owner), I want a graphical interface allowing me to see how many coupons a customer currently have that can be redeemed.	1	4	5
Story	Low	As David (restaurant owner), I want to be able to use existing coupons as a template for new coupons.	1	2	3
Story	High	As David (restaurant owner), I want customers' progress towards achievements and coupons to update only after they buy an item.	7	0	7
Story	High	As David (restaurant owner), I want to be able to add new achievements to my restaurant and set their experience and/or point values.	4	3	7

Story	High	As David (restaurant owner), I want a default template for customer achievements, so that I can set up achievements more easily.	1	4	5
Story	High	As David (restaurant owner), I want to be able to see all my restaurant's current achievements.	1	4	5
Story	High	As David (restaurant owner), I want to be able to delete achievements from my restaurant.	1	1	2
Story	Low	As David (restaurant owner), I want to be able to modify achievements from my restaurant.	1	1	2
Story	High	As David (restaurant owner), I want an interface for editing restaurant settings where I can manage all aspects of my restaurant on the application easily.	0	7	7
Story	High	As David (restaurant owner), I want to be able to see all employee accounts associated with my restaurant	2	3	5
Story	High	As David (restaurant owner), I want to be able to delete accounts associated with my restaurant.	2	2	4
Story	Medium	As David (restaurant owner), I want the option to give customers experience or points based on how much they spent at my restaurant, so that customers will feel rewarded when they dine at my restaurant.	3	1	4
Epic	Low	As David (restaurant owner) I want to be able to configure the benefits customers can get from their user level at my restaurant.	3	5	8
Epic	Low	As David (restaurant owner), I want a menu system allowing me to upload, preview and modify my menu onto the application.	6	9	15

Persona 2: Daniel (Employee)

Type	Priority	Story	Back-end Points	Front-end Points	Total Story Points
Story	Critical	As Daniel (employee), I want to have an account that is connected to the restaurant I work at.	3	2	5
Story	High	As Daniel (employee) I want to be able to scan customer coupon QR codes and mark the coupon used.	3	7	10

Persona 3: Kevin (Customer)

Type	Priority	Story	Back-end Points	Front-end Points	Total Story Points
Story	Critical	As Kevin (customer), I want to be able to register my account.	0.5	0.5	1
Story	Critical	As Kevin (customer), I want a search bar so that I can easily find and check a restaurant's coupons and achievements.	3	2.5	5.5
Story	High	As Kevin (customer), I want to be able to link my account to Shopify or Square so that after purchases my achievements and points will automatically updated.	7	2	9
Story	High	As Kevin (customer) I want to be able to redeem coupons at the related restaurant.	4	2	6
Story	High	As Kevin (customer) I want to be able to see available coupons and purchase coupons by redeeming my points.	2.5	4	6.5
Story	High	As Kevin (customer) I want to be able to see what achievements are available.	2	2	4

Story	High	As Kevin (customer), I want a progress bar for each achievement that tracks how close I am to completing the achievement.	1	3	4
Story	High	As Kevin (customer), I want a display for each restaurant that tracks how many points I have earned at that restaurant so far.	1	1	2
Story	Medium	As Kevin (customer) I want to be able to see how many points I gain after completing an achievement.	0.5	1	1.5
Story	High	As Kevin (customer), I want to be able to gain experience and levels as I eat at restaurants, so that I can feel a sense of progression as I'm using the application.	3	3	6
Story	Medium	As Kevin (customer), I want to see user leaderboards on the app so that I can compare my level with others and compete to see who has the highest level at a restaurant.	2	4	6
Story	Medium	As Kevin (customer), I want a profile page so that I can easily see which restaurants I've visited before.	3	2	5
Epic	Low	As Kevin (customer), I want to have a customizable character that can represent my progress, so that I will be more engaged in the app and have more fun.	9	10	19

Release Plan

Our team has decided on weekly sprints, with sprints ending on Sundays at 6 pm. Team pairs will pull about 15 points worth of stories from the top of the product backlog every sprint. We chose to have weekly sprints so that we can receive feedback on our feature development and direction more frequently, in terms of both internal feedback between team members and external feedback from the TA and our client. The shorter sprint length also allows us to check our velocity more frequently via our end of sprint meetings, so we have a better idea of how on track we are at any point in time. The frequent sprint summary meetings also allow us to uncover any possible issues in developing planned features earlier on, so that we can review and adjust our product backlog in our learning process as early as needed.

Sprint Plan

Priority	Story	Back-end Points	Front-end Points	Tasks	Back-end Points	Front-end Points
Critical	As David (restaurant owner), I want to be able to register my account as a restaurant owner.	6.5	4	T001: Design the user database with restaurant owner and customer type accounts	1	0
				T002: Design the restaurant database	1	0
				T003: Code the user database with user account creation functionality	2	0
				T004: Code the restaurant database and link restaurant creation to the creation of a restaurant owner user account	2	0
				T005: Design registration and login UI	0	0.5
				T006: Implement UI using database	0	3

				T007 + T008: Automate tests	0.5	0.5
Critical	As Kevin (customer), I want to be able to register my account as a customer.	0.5	0.5	T026: Add customer user type to the user database implementation	0.5	0
				T027: Add customer as an account type option in the registration UI	0	0.5
Critical	As Kevin (customer), I want a search bar so that I can easily find and check a restaurant's coupons and achievements.	3	2.5	T009: Design search algorithm for restaurants	1	0
				T010: Code search function with respect to restaurant database	1.5	0
				T011: Design search bar UI	0	0.25
				T012: Implement UI using back-end functions	0	1.75
				T013 + T014: Automate tests	0.5	0.5
High	As David (restaurant owner), I want to create a coupon with my selected requirements.	4	3	T015: Design coupon database for restaurant-specific coupons	2	0
				T016: Code the user database with user coupon creation functionality	1.5	0
				T017: Design coupon creation UI	0	0.5
				T018: Implement UI using restaurant-coupon database	0	2
				T019 + T020: Automate tests	0.5	0.5
High	As David (restaurant owner), I want to be able to see all my restaurant's current coupons.	1	6	T021: Code database query for fetching coupons specific to a given	0.8	0

				restaurant owner account		
				T022: Design UI for displaying a list of coupons	0	1
				T023: Implement UI using back-end functions	0	4
				T024 + T025: Automate tests	0.2	1

Acceptance Criteria

Task	Acceptance Criteria
Design the user database with restaurant owner and customer type accounts	**Scenario:** Multiple users want to use the app. Given that I am not the first user to use the app, when I register my account, the database is able to store multiple user account entries. **Scenario:** The user only wants to see features relevant to their account type. Given that I have created my account as a certain user type, when I log in, I only see the features related to my user type. **Scenario:** There are multiple users with the same name. Given that there is already a user on the app with the same name as me, when I log in, the correct account data is returned. **Scenario:** The user wants to access their user data securely. Given that I have already created an account, when I want to access my data again, the data is securely locked behind my account's login email and password.
Design the restaurant database	**Scenario:** Multiple restaurant owners want to use the app. Given that I am not the first restaurant owner to use the app, when I register my account and restaurant, the database is able to store multiple restaurants at once. **Scenario:** An owner wants customers to be able to find a restaurant on the app. Given that I want customers to be able to find my restaurant, when I add my restaurant to the app, I can set the restaurant's name.

	Scenario: Multiple restaurants have the same name. Given that my restaurant has the same name as another restaurant on the app, when I register my account and restaurant, the database is able to distinguish between the restaurants by a unique restaurant ID. **Scenario:** A restaurant owner wants to manage their restaurant on the app. Given that I want to be able to edit my restaurant after setting it up the first time, when I log in to my restaurant owner account, I can access the restaurant settings as my restaurant is associated with my owner's account.
Code the user database with user account creation functionality	The implemented database fulfills all the acceptance criteria for the user database design task. **Scenario:** A user creates an account for the first time. Given that I want to create an account, when I register my information, the database saves the data successfully. **Scenario:** The user does not provide sufficient information for registration. Given that I have not provided login email, password, or account type information, when I try to register my account, the registration is unsuccessful. **Scenario:** The user tries to register separate accounts with the same email. Given that I have already made an account with a given login email, when I try to register a new account with that same email, the registration is unsuccessful. **Scenario:** The restaurant owner does not provide sufficient information for registering a restaurant. Given that I want to register as a restaurant owner and have not provided my restaurant's name and other basic restaurant information, when I try to register my account, the registration is unsuccessful.

Code the restaurant database and link restaurant creation to the creation of a restaurant owner user account	The implemented database fulfills all the acceptance criteria for the restaurant database design task. **Scenario:** The user wants to register a new restaurant as a restaurant owner. Given that I want to create a restaurant as a restaurant owner, when I register my restaurant's information, the data is saved successfully. **Scenario:** The user wants to register a new restaurant but does not have a restaurant owner account. Given I am not a restaurant owner, when I try to create a restaurant, it is not possible without first creating a new restaurant owner account.
Design registration and login UI	The implemented UI fulfills all the acceptance criteria as the related back-end tasks above. **Scenario:** The user wants to register as a customer. Given that I am registering an account, when I select a customer account type, the UI redirects to a form with customer specific fields. **Scenario:** The user wants to register as a restaurant owner. Given that I am registering an account, when I select a restaurant owner account type, the UI redirects to a form with restaurant owner specific fields.
Implement registration and login UI using the database	The implemented UI fulfills all the acceptance criteria for the associated UI design task **Scenario:** The user registers an account via the UI after providing valid and complete user information. Given that the user has filled all required fields with valid data, when the user clicks the register button, the UI displays a confirmation screen for a successful registration. **Scenario:** The user registers an account via the UI after providing invalid or incomplete user information. Given that the user has not filled all required fields with valid data, when the user clicks the register button, the registration form prompts the user to refill the form fields and try again. **Scenario:** The user tries to login with valid login information. Given that I have entered valid login information, when I press the login button, the database returns the correct account and I stay logged in on the app. **Scenario:** The user tries to login with invalid login information.

	Given that I have entered invalid login information, when I press the login button, the login form prompts the user to refill the form fields and try again
Design a search algorithm for restaurants	**Scenario:** The user searches for restaurants that match a given name or keyword. Given that I want to find all restaurants related to my chosen keyword, when I input the keyword, all restaurants whose names have the given name/keyword as a substring appear. **Scenario:** The user searches for a restaurant name that is not on the app. Given that I want to find an unavailable restaurant, when I input the restaurant name into the search bar, no restaurants appear.
Code search function with respect to the restaurant database	The implemented function fulfills all the acceptance criteria for the search algorithm design task.
Design a search bar UI	The implemented UI fulfills all the acceptance criteria as the related back-end tasks above
Implement UI using back-end functions	The implemented UI fulfills all the acceptance criteria for the associated UI design task **Scenario:** The user searches restaurants by partial text Given that I am trying to search for a restaurant quickly and have begun to input my search term, when I add an additional character to the search term input, the list display refreshes with most relevant matches displayed at the top of the returned restaurant list.
Design coupon database for restaurant-specific coupons	**Scenario:** The restaurant owner adds a new coupon. Given that I have entered all required and valid information to create a coupon, when I click create, a new coupon is successfully added to the database and the coupon is associated with my restaurant. **Scenario:** The restaurant owner adds multiple coupons to their restaurant. Given that I have already added coupons to my restaurant before, when I add a new coupon, the database is able to store all my coupons. **Scenario:** The restaurant owner adds multiple coupons with the same description. Given that I already have coupons added to my restaurant, when I add a new coupon with the same name as an existing

	coupon, the database saves a new coupon separate from the one of the same name. **Scenario:** The customer browses the selection of offered coupons. Given that I want to select a coupon to exchange for, when I browse a restaurant's coupon, each coupon displays a description describing what the coupon can be used for. **Scenario:** The customer wants to exchange their points for coupons. Given that I have chosen a coupon I want to exchange for, when I go to spend my points, each coupon displays its cost in points.
Code the user database with user coupon creation functionality	The implemented database fulfills all the acceptance criteria for the coupon database design task
Design coupon creation UI	The implemented UI fulfills all the acceptance criteria as the related back-end tasks above **Scenario:** The restaurant owner tries to add a coupon after providing invalid or incomplete coupon data. Given that I have not entered all required and valid information to create a coupon, when I click create, the coupon is not successfully added to the database and the form prompts me to try entering data again. **Scenario:** The restaurant owner has begun to create a new coupon but changes their mind. Given that I have already opened the coupon creation interface, when I click the cancel button, I exit the creation interface and return to the original page.
Implement UI using restaurant-coupon database	The implemented UI fulfills all the acceptance criteria for the associated UI design task
Code database query for fetching coupons specific to a given restaurant owner account	**Scenario:** The user checks the existing coupons offered at a restaurant. Given that I want to browse coupons relevant to a specific restaurant when I go to that restaurant's page, the display only contains coupons associated with that restaurant. **Scenario:** The user checks the existing coupons offered at a restaurant with multiple coupons. Given that I want to browse all coupons relevant to a specific restaurant when I go to that restaurant's page, the display containsall coupons associated with that restaurant.

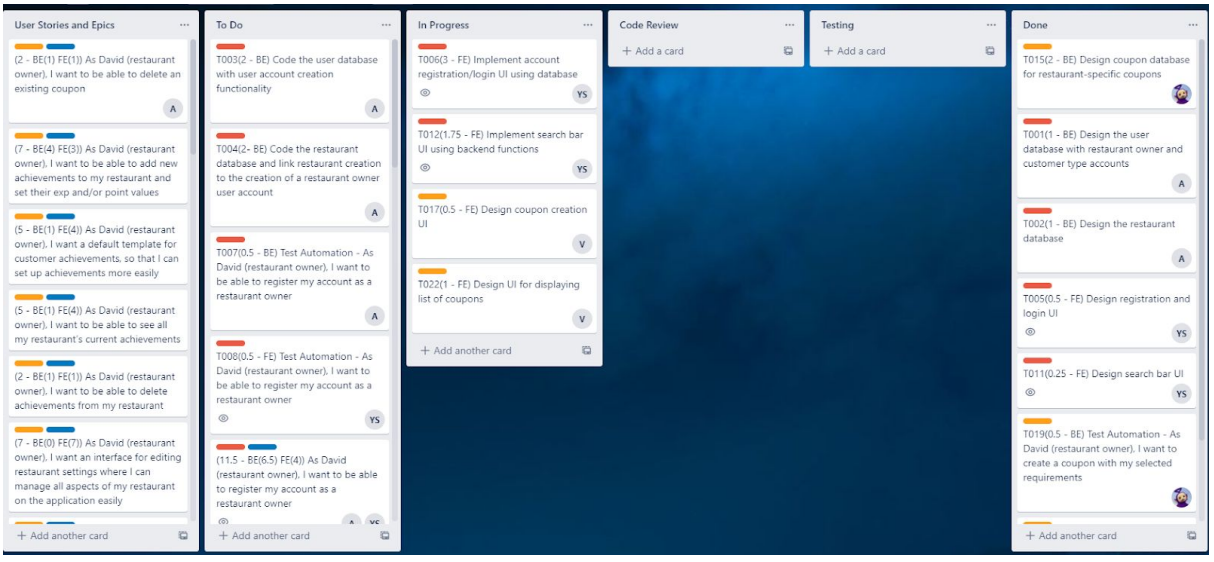
Design UI for displaying a list of coupons	The implemented UI fulfills all the acceptance criteria as the related backend tasks above **Scenario:** A restaurant has multiple coupons with the same description and/or point cost. Given that the restaurant has already added several coupons with overlapping coupon descriptions and/or point cost, when I browse the restaurant's coupon list, each coupon has its coupon ID displayed alongside its description and cost.
Implement UI using back-end functions	The implemented UI fulfills all the acceptance criteria for the associated UI design task **Scenario:** The restaurant owner checks the existing coupons offered at their own restaurant while logged in. Given that I login as a restaurant owner, when I go to my restaurant's coupon management page, the display only contains coupons associated with my restaurant.

Team Task Assignments

Task ID	Cost	Priority	Day 0	Day1	Day 2	Day 3	Day 4	Day 5	Day 6	Assignment
T001	1	Critical	0	0	0	1	0	0	0	Anson
T002	1	Critical	0	0	0	1	0	0	0	Anson
T003	2	Critical	0	0	0	0	2	0	0	Anson
T004	2	Critical	0	0	0	0	0	2	0	Anson
T005	0.5	Critical	0	0	0.5	0	0	0	0	Yolanda
T006	3	Critical	0	0	0	1.5	1.5	0	0	Yolanda
T007	0.5	Critical	0	0	0	0	0	0	0.5	Anson
T008	0.5	Critical	0	0	0	0	0	0.5	0	Yolanda
T026	0.5	Critical	0	0	0	0	0.5	0	0	Anson
T027	0.5	Critical	0	0	0	0	0.5	0	0	Yolanda
T009	1	Critical	0	0	0	1	0	0	0	Anson
T010	1.5	Critical	0	0	0	1	0.5	0	0	Anson
T011	0.25	Critical	0	0	0.25	0	0	0	0	Yolanda
T012	1.75	Critical	0	0	0	0	0	1	0.75	Yolanda

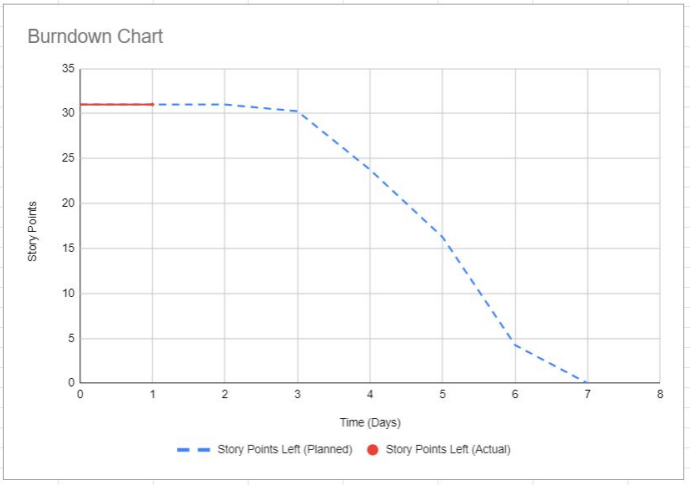
T013	0.5	Critical	0	0	0	0	0	0.5	0	Anson
T014	0.5	Critical	0	0	0	0	0	0	0.5	Yolanda
T015	2	High	0	0	0	0	1	1	0	Jiawei
T016	1.5	High	0	0	0	0	1	0.5	0	Jiawei
T017	0.5	High	0	0	0	0	0.5	0	0	Vanessa
T018	2	High	0	0	0	0	0	2	0	Vanessa
T019	0.5	High	0	0	0	0	0	0	0.5	Jiawei
T020	0.5	High	0	0	0	0	0	0	0.5	Vanessa
T021	0.8	High	0	0	0	0	0	0.5	0.3	Jiawei
T022	1	High	0	0	0	1	0	0	0	Vanessa
T023	4	High	0	0	0	0	0	4	0	Vanessa
T024	0.2	High	0	0	0	0	0	0	0.2	Jiawei
T025	1	High	0	0	0	0	0	0	1	Vanessa
Total Points Completed:			0	0	0.75	6.5	7.5	12	4.25	
Total Points Left:			31	31	30.25	23.75	16.25	4.25	0	

Task Board

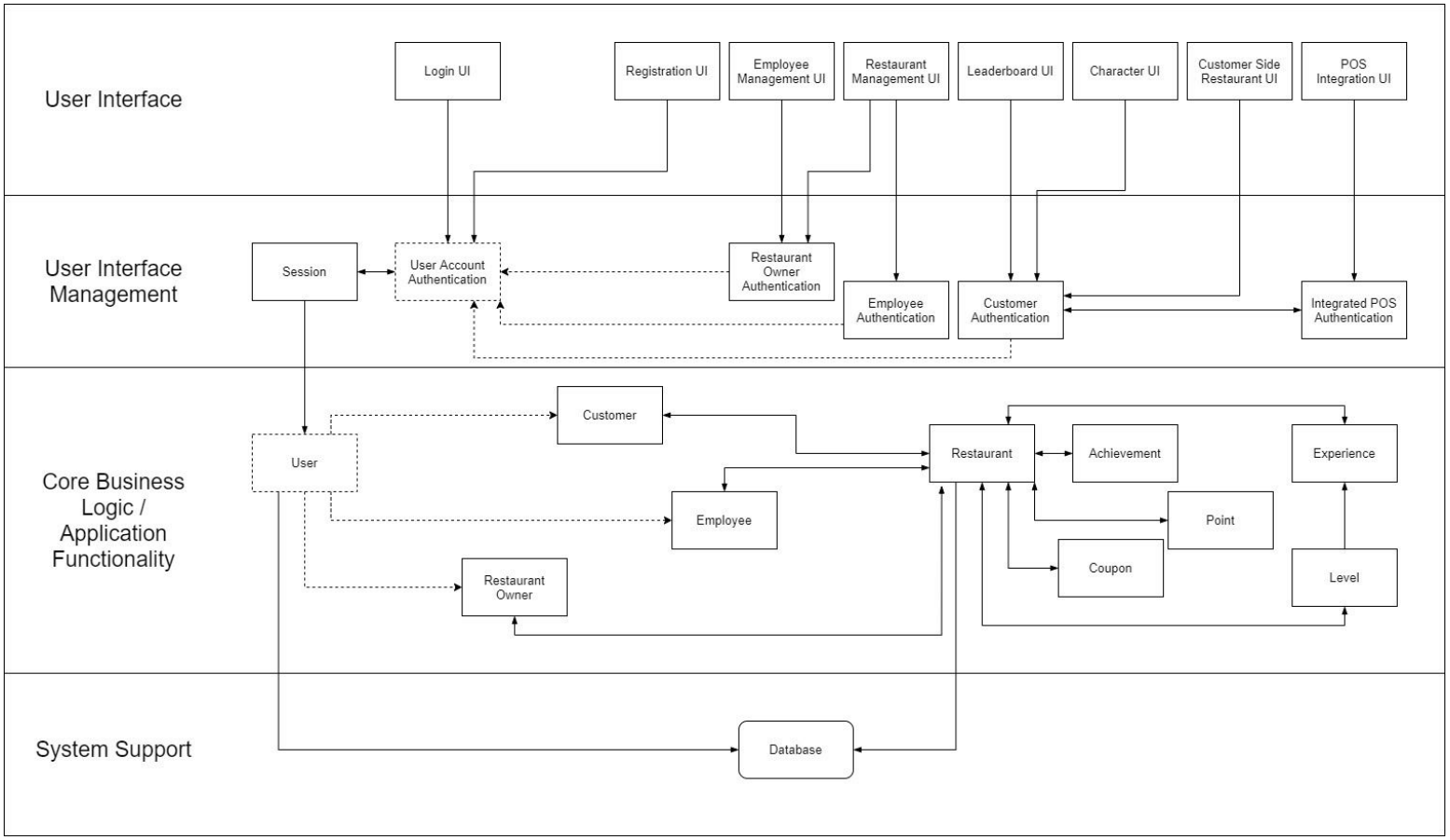


Burndown Chart

Time (Days)	Story Points Left (Planned)	Story Points Left (Actual)
0	31.00	31
1	31.00	31
2	31.00	
3	30.25	
4	23.75	
5	16.25	
6	4.25	
7	0.00	



High-level Architecture



Component	Description	Dependencies
Login UI	The login UI is responsible for taking login email and password input.	The login UI depends on the user account authentication component in order to validate provided login information and retrieve the corresponding account data.
Registration UI	The registration UI is responsible for taking account type, login email, password, and other initial data for creating a new user account.	The registration UI depends on the user account authentication component in order to validate provided registration information and save the new account data.
Employee Management UI	The employee management UI is responsible for listing all employee accounts under a given restaurant and displaying the options to add or delete employee accounts.	The employee management UI depends on the restaurant owner authentication component in order to check the user's management permissions and retrieve restaurant data.
Restaurant Management UI	The restaurant management UI is responsible for listing coupons and achievements for restaurant owners and employees to manage.	The restaurant management UI depends on the restaurant owner authentication component in order to check the user's management permissions (for coupons and achievements) and retrieve restaurant data. The restaurant management UI depends on the employee owner authentication component in order to check the user's management permissions (for coupons) and retrieve restaurant data.
Leaderboard UI	The leaderboard UI is responsible for displaying a list of the highest level customers at a given restaurant.	The leaderboard UI depends on the customer authentication UI to check the customer data retrieval permissions and retrieve relevant customer data.
Character UI	The character UI is responsible for displaying a user's custom character.	The leaderboard UI depends on the customer authentication UI to check that the character feature has been unlocked and retrieve the logged in customer's account data.

Customer Side Restaurant UI	The customer side restaurant UI is responsible for displaying a user's level, points, and achievement progress at a given restaurant	The customer side restaurant UI depends on the customer authentication UI to check and retrieve the logged in customer's account data.
POS Integration UI	The POS integration UI is responsible for linking and redirecting to external POS systems in order.	The POS integration UI depends on the Integrated POS authentication to verify the linked POS account details and process transaction details.
Session	The session component is responsible for recording the data of the current logged in user account, as well as recording and executing user actions in the current session.	The session relies on the user component to retrieve user-relevant data upon request. The session relies on the account authentication component to determine which user actions and requests are permitted.
Account Authentication (Interface)	The account authentication component is responsible for verifying user login credentials and user permissions.	The account authentication UI relies on the login UI to provide login credential input. The account authentication UI relies on the login UI to provide account registration data input. The account authentication UI relies on the session to return data requested by the user.
Restaurant Owner Authentication	The restaurant owner authentication component is responsible for verifying user login credentials and user permissions for a restaurant owner.	The restaurant owner authentication component implements the account authentication component.
Employee Authentication	The employee authentication component is responsible for verifying user login credentials and user permissions for an employee.	The employee authentication component implements the account authentication component.
Customer Authentication	The customer authentication component is responsible for verifying user login credentials and user permissions for a customer.	The customer authentication component implements the account authentication component.

		The customer authentication relies on the integrated POS authentication component to verify linked POS system accounts match the given user account.
Integrated POS Authentication	The integrated POS authentication is responsible for verifying the linked POS system accounts and processing transaction data securely.	The integrated POS authentication relies on the customer authentication component to verify the linked user account matches the given POS system account.
User (Interface)	The user component is responsible for knowing the user data of a given user.	The user component relies on the database to store user data.
Customer	The customer component is responsible for knowing all the user data of a given customer user, such as all restaurants the customer has visited.	The customer component implements the user component. The customer component relies on the restaurant component to know the customer's achievement progress, coupons redeemed, points, and level at a given restaurant.
Restaurant Owner	The restaurant owner component is responsible for knowing the user data of a given restaurant owner user.	The restaurant owner component implements the user component. The restaurant owner component relies on the restaurant component to know the achievements and coupons offered at the owner's restaurant.
Employee	The employee component is responsible for knowing the user data of a given employee user.	The employee component implements the user component. The restaurant owner component relies on the restaurant component to know the coupons that can be redeemed at the employee's restaurant.

Restaurant	The restaurant component is responsible for knowing which achievements/coupons/experience/level correspond to a given restaurant.	The restaurant component relies on the database to store all restaurant data, including data of coupons/achievements/etc associated with the restaurant. The restaurant component relies on the customer component to know which customers have been to the given restaurant. The restaurant component relies on the restaurant owner component to know who owns the given restaurant. The restaurant component relies on the employee component to know which employees work at the given restaurant. The restaurant component relies on the achievement component to store a list of the given restaurant's achievements. The restaurant component relies on the coupon component to store a list of the given restaurant's coupon. The restaurant component relies on the point component to store a list of points earned by each individual customer at the given restaurant. The restaurant component relies on the experience component to store a list of experience earned by each individual customer at the given restaurant.
Achievement	The achievement component is responsible for knowing the data of a specific achievement at a given restaurant.	The achievement component relies on the restaurant component to know which restaurant a given achievement is associated with.

Coupon	The coupon component is responsible for knowing the data of a specific coupon at a given restaurant.	The coupon component relies on the restaurant component to know which restaurant a given coupon is associated with.
Point	The point component is responsible for knowing a user's points at a given restaurant.	The point component relies on the restaurant component to know which restaurant a given user's points are associated with.
Experience	The experience component is responsible for knowing a user's experience at a given restaurant.	The experience component relies on the restaurant component to know which restaurant a given user's experience is associated with.
Level	The level component is responsible for knowing a user's level at a given restaurant.	The level component relies on the restaurant component to know which restaurant a given user's level is associated with. The level component relies on the experience component to calculate a given user's level.
Database	The database is responsible for storing data.	The database does not have dependencies.

Retrospection

Since it is the first time we worked together as a team and is the first time doing an actual Agile process, our user story grouping in the product backlog was only sorted by priority at the beginning. After receiving some suggestions from the TA, we decide to make the grouping more readable by grouping them by persona; followed by feature subgrouping within the persona group so that it will make more sense to the reader, and finally sorting stories from the same feature by priority.

As mentioned above, as we were new to Agile, we made some user stories that were too vague and thus not testable. Therefore, after some discussions in one of our weekly meetings, we decided to split some user stories into smaller and more specific stories so that they could be more testable and precise. We also updated and re-evaluated the story points for some stories based on the TA's feedback and our first sprint experience, by re-voting the points as a team.

During development, we realized that we might need to add a new persona for representing the employee user type, who will handle some of the work of the restaurant owner, thus we added two extra user stories for the employee persona, in addition to reassigning some existing restaurant owner user stories. We also added two extra owner user stories that represented the owner's need for being able to add or delete employee accounts under their own restaurant.

Changes From Deliverable 2

As mentioned at the former section, we rewrote user stories which were too vague and split some of the user stories into several smaller stories that are easier to implement and test. We changed our prioritization system to one with only 4 different levels of priority, rather than rating from 1-20, in order for our priority system to be easier to understand. We re-evaluated and updated the priorities of our stories so that the priorities would correspond to the client's expectations of the app more closely. Based on these changes, we also deleted redundant stories. We revised acceptance criteria and project setup based on the feedback TA provided and based on examples from lecture, in order to be more professional.

Furthemore, we came up with one additional persona that covers employee type users based on TA feedback. We are still waiting for the response of the client to confirm the persona. This persona is as follows:

Daniel is a 24 near-graduate university student living in the GTA area who is currently working in a bubble tea restaurant as a cashier. This restaurant is planning to promote user retention through a loyalty system, and is aiming to use an online app for these planned promotions. Since Daniel knows he will be responsible in part for handling the technological

side of running the loyalty system, he hopes that he will not need to deal with a complicated app. He is good at using technology, but he is not interested in dealing with features beyond the scope of his coupon processing responsibilities as a cashier.

Release

We have fully implemented and tested the user registration, restaurant creation and coupon code creation, as well as corresponding database setups, both front-end and back-end. These will be in our release.

Project Velocity

Our estimated project velocity was 31 points in a sprint.
Our actual project velocity was 23.25 points in a sprint.

We believe this gap was caused by our underestimating how many points our tasks and stories were worth, as well as our decision mid-sprint to refactor a lot of our completed work.

Planning Retrospection

Our plan was a bit rushed and was revised repeatedly during the sprint, thus everyone in our team began work late and had to work harder on the latter half of the sprint to get work done. Additionally, most of us haven't had too much experience doing actual app and database related development, both front-end and back-end, which was one of the major difficulties we encountered so far as we needed quite a few work hours to learn new frameworks. Due to this additional time cost, we ended up behind on our plan, and had a limited time period to complete the tasks while trying to learn new technologies simultaneously, which made the development process less smooth.

During our retro meeting, we discussed these issues, and thus we changed to a new plan for our next sprint. We decided on reducing amounts of points that are in each sprint so that we have more flexibility when a task turns out to be underestimated, and adding story pointed tasks allocated specifically for learning new frameworks. We also chose to refactor our completed work to and do our future code in Python and Flask, as it is easier and faster to learn, and at least one member is already familiar with this language and framework.