

Landon Bedell

Deep Desai

Branden Romero

Justin Tang

CSCI 4448

Massimo: Final Report

1. What features were implemented?

- Requirements:
 - i. Rate specific meals
 - ii. Create a receipt of items a user ordered in a sitting and Display it in payment screen
 - iii. Highlight items that the user clicks on and pop a pay for button
 - iv. Move user to rating screen after payment
 - v. Calculate tip and display 10%, 15%, and other then add selected to selected bill
 - vi. Have a button on home screen to bring up camera to scan QR code
 - vii. Easily spit check between other users
 - viii. As restaurant places order, create and update the table's receipt
 - ix. Use QR Codes to determine which check is associated with the user
 - x. Tag entrée based on the type of food they have
 - xi. Once user scans QR code for table associate them with receipt
 - xii. If more than two users scan the QR code for one sitting, enable them to select which items to pay for.
 - xiii. Selecting specific items to pay for, instantly updates on all user's phones
 - xiv. Have app suggest meals for me based on what I have previously had
- Use Cases:
 - i. UC-003 (partial)
 - 1. Rate Experience
 - ii. UC-005
 - 1. Search by Item
 - iii. UC-006
 - 1. Check in to Table
 - iv. UC-007
 - 1. Update Order
 - v. UC-008
 - 1. Checkout of Restaurant

2. Which features were not implemented from Part 2?

- Requirements
 - i. Sign in through Facebook
 - ii. Sign in through Google+
 - iii. Use preferred payment method
 - iv. Search through nearby entrees based on meal type
 - v. Rate the restaurant
 - vi. See what dishes are trending near me
 - vii. Keep me logged in
 - viii. Automatically charges users that forget to pay
 - ix. Able to upload new entrée to app
 - x. View statistical data about restaurant's entrees
 - xi. Show total tips earned for the day so splitting between servers can be calculated
 - xii. Know if an entrée is available at restaurant.
 - xiii. Auto detect accounts that user is already signed into
 - xiv. USE GPS to determine if a user is at a restaurant
 - xv. Auto pay for forgotten tabs at midnight
 - xvi. Store the frequency at which specific entrée is ordered
 - xvii. Display which entrees are trending based on frequency of it ordered
 - xxviii. Store the average user rating for each entrée
 - xix. Enable search that can be organized by distance, name or rating
 - xx. Store pictures for entrees
 - xxi. Display entrees' photos on search list
 - xxii. Create a homepage with a list of entrees based on location and rating
 - xxiii. Send notification to user when they enter a restaurant that support our app
 - xxiv. Update user that payment is complete after transaction goes through. Or tell them that it failed
 - xxv. Restaurant can view a list of entries available at their restaurant and select and update which items are out of stock.
 - xxvi. Signing in with other accounts takes no longer than 10 seconds
 - xxvii. Protecting sensitive user data including username, password and location
 - xxviii. Search functionality should be quicker than 30 seconds
- Use Cases
 - i. UC-001
 - 1. Log in to app (customer)
 - ii. UC-002
 - 1. Log in to app (Employee)
 - iii. UC- 004
 - 1. Search by Restaurant
 - iv. UC-009
 - 1. Update Restaurant information

v. UC-010

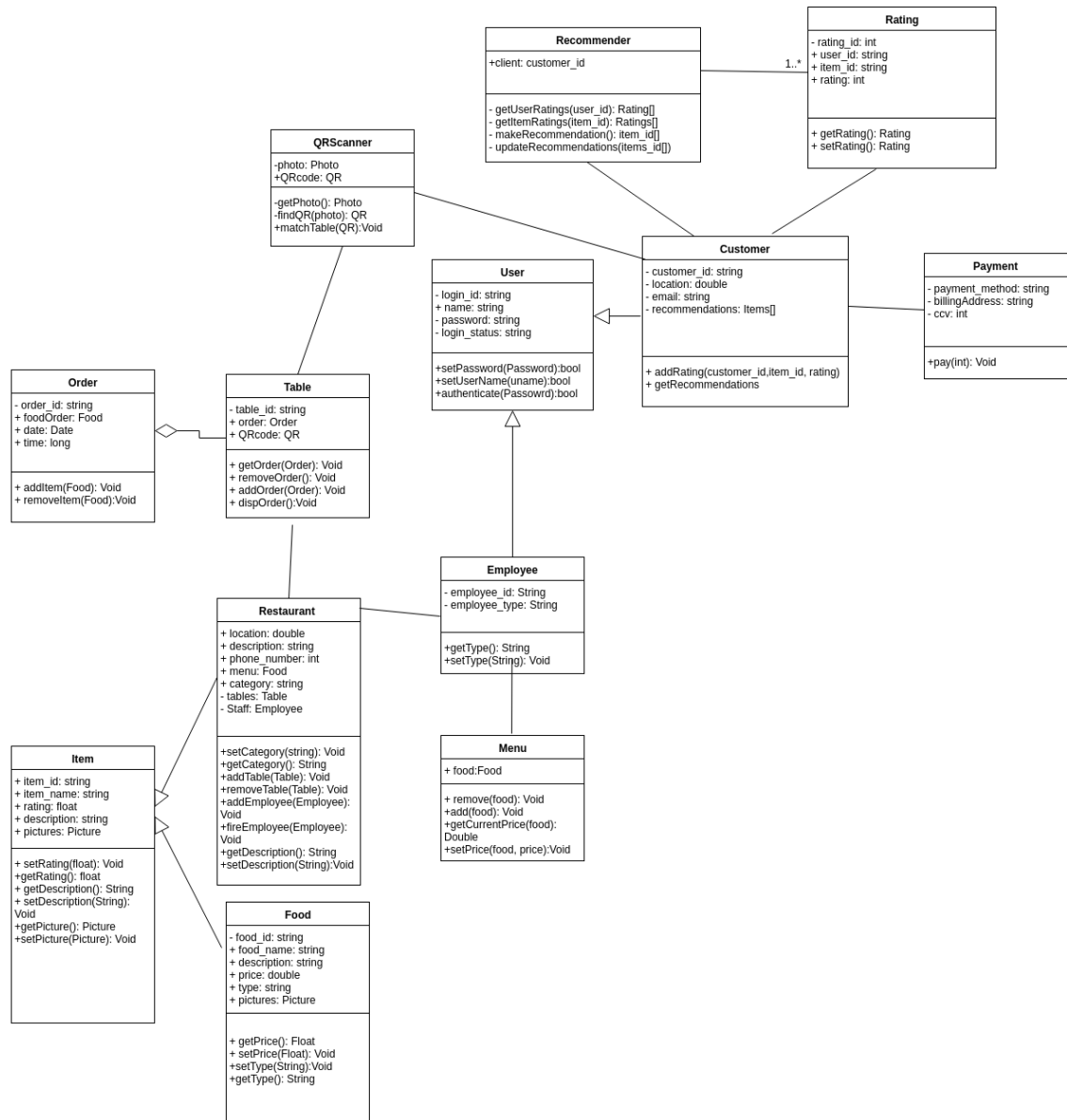
1. Update Restaurant Menu

vi. UC-011

1. Modify Account Info

3. Show your Part 2 class diagram and your final class diagram. What changed? Why? If it did not change much, then discuss how doing the design up front helped in the development.

Class Diagram from Part 2:



Final Class Diagram:

Rating class. Delegating these methods to the Rating class make the system more cohesive since all rating related variables and methods are together, while reducing coupling by removing the transient class.

4. Did you make use of any design patterns in the implementation of your final prototype? If so, how? If not, where could you make use of design patterns in your system?
 - MVC - OrderController acts as the controller for the views for the QR scanner (CameraView), the order screen (OrderView) and pay screen (PayView) and the models for orders (OrderModel) and food (FoodModel) classes. It controls the data flow into models and updates the views whenever updates come in. It keeps view and model separate.
 - Observer- created an observable class (TableModel) and a observer class (OrderController). This enables us to have the TableModel get updates which then passes the updates to OrderController, which then updates the view accordingly.
 - Singleton- A variation of Singleton was used implemented in customer class.
 - Adapter- We used a listview adapter for creating RateActivity and HomeFragement views, by linking to Android Classes which aid in the generations of views . An adapter was also used to link to the Android Fragment class.
 - State- Was used to make multiple contexts in the MainActivity, which the user can switch between.
5. What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

Planning through UML diagrams makes implementation a lot easier especially for complicated systems especially class diagrams. We realized that pre designing really helps speed up the implementation process because it eliminates time usually spent on debugging poorly designed code. Also there are many ways to code smartly and efficiently.

Another thing that we learned is the importance of design patterns. Without the knowledge of some of the design patterns like the observer design pattern, we would not have been able to accomplish some of the objectives that we put forth. Therefore, it is worth learning the design patterns, and asking yourself how to incorporate them into your code to improve the design of your system.