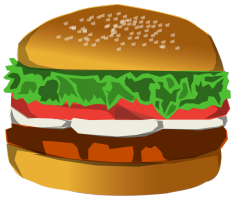


U Eat:

A Dining Concierge App

-User Stories

Group 28
Luke Carter
Ibrahim Mahmoud
Miles McCoy
Mathew McDade
Timothy Tseng

User Stories

1. **General Login:**
    a. **AS A** busy software developer **I WANT** to be able to login to U Eat app easily **BECAUSE** I would like the app to remember me.
2. **Social Login:**
    a. **AS A** busy student **I WANT** to be able to use my social media logins **BECAUSE** I'm in love with instagram and don't want to make another login.
3. **App Personalization:**
    a. **AS A** famous photographer **I WANT** to be able to add pictures of restaurants and dishes to my app **BECAUSE** I love taking pictures and use color to bring objects to life.
4. **Profile Editing:**
    a. **AS A** lazy typist **I WANT** to be able to edit my profile **BECAUSE** I made a typo.
5. **Order History:**
    a. **AS A** history teacher **I WANT** to be able to view my past orders **BECAUSE** history teaches us many lessons such as what I have ordered before.
6. **Favorites:**
    a. **AS A** single girl in a new city **I WANT** to be able to keep track of my favorite restaurants **BECAUSE** I would like to visit them again.
7. **Reward Points $$$:**
    a. **AS A** coupon clipper **I WANT** to be able to accrue reward points every time I order **BECAUSE** I like getting things for free.
8. **Dashboard Search:**
    a. **AS A** mom of 9 kids **I WANT** to be able to search for restaurants quickly **BECAUSE** I would like to get my screamy hungry kids off my back as quickly as possible.
9. **Search by Name:**
    a. **AS A** pilot **I WANT** to be able to search for restaurants by name **BECAUSE** I would like to find my destination on the fly.
10. **Map Search:**
    a. **AS A** tourist **I WANT** to be able to view the local restaurants on a map view **BECAUSE** I am unfamiliar with the area.

11. **Post Restaurant Menu:**
    a. **AS A** restaurant owner **I WANT** to be able to post my menu **BECAUSE** I would like customers to order from my restaurant.
12. **View Restaurant Menu:**
    a. **AS A** hungry person **I WANT** to be able to view the menu **BECAUSE** I would like to see what I can order from the restaurant.
13. **Restaurant Reviews:**
    a. **AS A** grumpy granny **I WANT** to be able to leave restaurant reviews **BECAUSE** I think my cooking is better than these Michelin restaurants and I got to share my opinions.
14. **Order Ahead:**
    a. **AS A** busy bee **I WANT** to be able to order ahead **BECAUSE** I don't want to wait in the lobby.
15. **Estimated Time:**
    a. **AS A** wreck-it Ralph **I WANT** to be able to know the estimated time for my order to be ready **BECAUSE** I would like to break the internet and still get there on time.
16. **Confirm and Complete Order:**
    a. **AS A** type A personality **I WANT** to be able to confirm and then complete my order **BECAUSE** I would like to centuple-check my order.
17. **Order Status:**
    a. **AS A** serious business man **I WANT** to be able to track my order status **BECAUSE** I would like to know when my food is going to reach my tummy.
18. **Order Cancellation:**
    a. **AS A** forgetful fish (Dory from Finding Nemo) **I WANT** to be able to cancel my order **BECAUSE** I forget everything anyways.
19. **Tableside Order:**
    a. **AS A** 100 hotdog eating contest winner **I WANT** to be able to add more items to my orders **BECAUSE** the more I eat, the more hungry I get and I would be in need of more food.
20. **Tableside Payment:**
    a. **AS A** CPA **I WANT** to be able to know how much I owe for my table **BECAUSE** I would like to pay with my apple pay and keep the receipts for tax return season.

User Story Planning

1. ## General Login:
    a. **Due:** Story due December 1st.
    b. **Tasks:**
        i. Set up database to store user login credentials.
            - 2 units (~4hrs pair programming).
        ii. Create web application page with appropriate layout for login functionality.
            - 1 unit (~2hrs pair programming).
        iii. Build server 'login' routes that POST user info and return login verification or error.
            - 2 units (~4hrs pair programming).
2. ## Social Login:
    a. **Due:** Story not due.
    b. **Tasks:**
        i. Look up social media/AuthO API requirements and implementation details.
            - 2 units (~4hrs pair programming).
        ii. Use the social media api as the authentication.
            - 2 units (~4hrs pair programming).
        iii. Add relationship in DB.
            - 1 unit (~2hrs pair programming).
3. ## App Personalization:
    a. **Due:** Story not due.
    b. **Tasks:**
        i. Create DB entity to hold picture object references.
            - 1 unit (~2hrs pair programming).
        ii. Create relationship between entity and user.
            - 1 unit (~2hrs pair programming).
        iii. Create relationship between entity and restaurant name entity.
            - .5 units (~1hr pair programming).
        iv. Build backend API to make the CRUD SQL calls for photo entity.
            - 1 unit (~2hrs pair programming).
        v. Build frontend HTML/CSS, JavaScript to implement CRUD frontend.
            - 2 units (~4hrs pair programming).

4. **Profile Editing:**
    a. **Due:** Story not due.
    b. **Tasks:**
        i. Build backend API to make the Update SQL calls for user relationships.
            ➢ 2 units (~4hrs pair programming).
        ii. Build frontend HTML/CSS, JavaScript to implement Read and update frontend for the user data.
            ➢ 2 units (~4hrs pair programming).
5. **Order History:**
    a. **Due:** Story not due.
    b. **Tasks:**
        i. Create order user relationship in DB.
            ➢ 1 unit (~2hrs pair programming).
        ii. Build backend API to make SQL calls to display the past orders.
            ➢ 2 units (~4hrs pair programming).
        iii. Build frontend HTML/CSS JavaScript to display past orders.
            ➢ 2 units (~4hrs pair programming).
        iv. May want to add filter functionality (need to talk to customer).
6. **Favorites:**
    a. **Due:** Story not due.
    b. **Tasks:**
        i. Add favorites table to DB.
            ➢ 1 unit (~2hrs pair programming).
        ii. Table is one to many User to restaurants.
        iii. Build backend SQL call to add relationship.
            ➢ 2 units (~4hrs pair programming).
        iv. Build front end HTML/CSS, javascript to mark the relationship. Ie radio button.
            ➢ 2 units (~4hrs pair programming).
7. **Reward Points $$$:**
    a. **Due:** Story not due.
    b. **Tasks:**
        i. Add Reward attribute to User entity.
            ➢ 1 unit (~2hrs pair programming).
        ii. Build backend SQL to update Reward attribute after customer orders.
            ➢ 2 units (~4hrs pair programming).
        iii. Build front end HTML/CSS, and javascript to show reward points in the user profile.
            ➢ 1 unit (~2hrs pair programming).

8. ## Dashboard Search:
    a. **Due:** Story not due.
    b. **Tasks:**
        i. Add front end HTML / CSS for Search bar.
            ➢ 1 unit (~2hrs pair programming).
        ii. Add javascript to to enable Search function
            ➢ 2 units (~4hrs pair programming).
        iii. Add backend API to make SQL call based on user search and display results.
            ➢ 2 units (~4hrs pair programming).
9. ## Search by Name:
    a. **Due:** Story due December 1st.
    b. **Tasks:**
        i. Add front end HTML / CSS for Search by name (user)
            ➢ 1 unit (~2hrs pair programming).
        ii. Add javascript to enable Search function and buttons
            ➢ 2 units (~4hrs pair programming).
        iii. Add backend API to make SQL call search on users and display query results
            ➢ 2 units (~4hrs pair programming).
10. ## Map Search:
    a. **Due:** Story due December 6th.
    b. **Tasks:**
        i. Add front end HTML / CSS to display Map
            ➢ 1 unit (~2hrs pair programming).
        ii. Add script to get user location (or simply an input for address)
            ➢ 2 units (~4hrs pair programming).
        iii. Create GET request to google maps API with provided user information
            ➢ 4 units (~8hrs pair programming).
        iv. Res / Render map on webpage
            ➢ 1 unit (~2hrs pair programming).
11. ## Post Restaurant Menu:
    a. **Due:** Story not due.
    b. **Tasks:**
        i. CreateHTML / CSS to display Menu
            ➢ 1 unit (~2hrs pair programming).
        ii. Add script for backend SQL call to search query for specified restaurant
            ➢ 2 units (~4hrs pair programming).
        iii. Res / Render query results to webpage
            ➢ 1 unit (~2hrs pair programming).

12. View Restaurant Menu:

a. **Due:** Story due December 1st.
b. **Tasks:**
   i. Add menu entity to DB.
      ➢ 1 unit (~2hrs pair programming).
   ii. Add item entity to DB.
      ➢ 1 unit (~2hrs pair programming).
   iii. Add relationship for item to menu.
      ➢ .5 units (~1hr pair programming).
   iv. Add relationship for menu to restaurant.
      ➢ .5 units (~1hr pair programming).
   v. Build backend API making SQL call to display items by restaurant menu.
      ➢ 2 units (~4hrs pair programming).
   vi. Build front end HTML/CSS to display JSON data returned by API call.
      ➢ 2 units (~4hrs pair programming).

13. Restaurant Review:

a. **Due:** Story due December 6th.
b. **Tasks:**
   i. Add Restaurant Review table to DB
      ➢ 1 unit (~2hrs pair programming).
   ii. Create relationship (A restaurant can have one to many reviews)
      ➢ 1 unit (~2hrs pair programming).
   iii. Add HTML / CSS for to allow user to create a review and submit
      ➢ 1 unit (~2hrs pair programming).
   iv. Add script to make SQL to update review table.
      ➢ 2 units (~4hrs pair programming).

14. Order Ahead:

a. **Due:** Story not due.
b. **Tasks:**
   i. Create HTML form to allow user to order ahead
      ➢ 1 unit (~2hrs pair programming).
   ii. Create script to send POST request to restaurant via API.
      ➢ 3 units (~6hrs pair programming).

15. Estimated Time:

a. **Due:** Story not due.
b. **Tasks:**
   i. Add HTML / CSS for estimated wait time
      ➢ 1 unit (~2hrs pair programming).
   ii. Create script to send GET request for wait time via API
      ➢ 4 units (~8hrs pair programming).
   iii. Res / Render
      ➢ 1 unit (~2hrs pair programming).

16. Confirm and Complete Order:

a. **Due:** Story not due.
b. **Tasks:**
   i. Add HTML / CSS
      ➢ 1 unit (~2hrs pair programming).
   ii. Add script to create “Alert” to confirm order
      ➢ 1 unit (~2hrs pair programming).
   iii. Add script to send POST request to restaurant when order is confirmed.
      ➢ 4 units (~8hrs pair programming).

17. Order Status:

a. **Due:** Story due December 6th.
b. **Tasks:**
   i. Add HTML / CSS
      ➢ 1 unit (~2hrs pair programming).
   ii. Create script to send GET request for order status via API from restaurant.
      ➢ 4 units (~8hrs pair programming).
   iii. Res / Render order status
      ➢ 1 unit (~2hrs pair programming).

18. Order Cancellation:

a. **Due:** Story not due.
b. **Tasks:**
   i. Create HTML / CSS buttons for Order Cancellation
      ➢ 1 unit (~2hrs pair programming).
   ii. Create script to send some type of request to have restaurant cancel order
      ➢ 4 units (~8hrs pair programming).

19. Tableside Order:

a. **Due:** Story not due.
b. **Tasks:**
   i. Create HTML/CSS button for Add Additional Items to Order.
      - 1 Units (~2hrs pair programming)
   ii. Create a script that sends the additional items to the tables bill.
      - 2 Units (~4hrs pair programming)
   iii. Create a script that creates a new order and calls User story #16
      - 2 Units(~4hrs pair programming)
   iv. Create script that adds additional entry to list of orders for table.
      - 0.5 Unit(~1hrs pair Programming)

20. Tableside Payment:

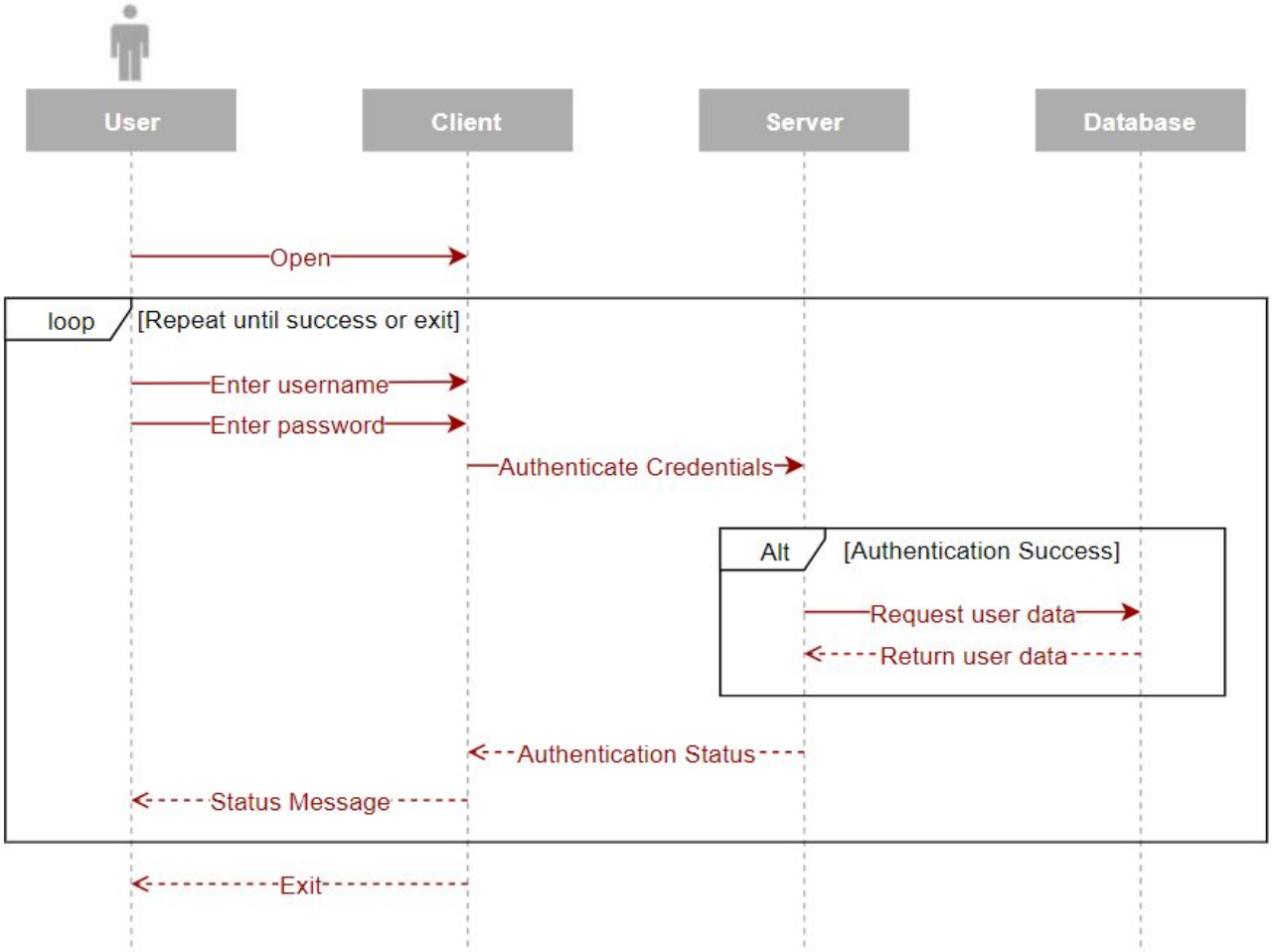
a. **Due:** Story not due.
b. **Tasks:**
   i. Create HTML/ CSS display element for current table total.
      - 1 Unit (~2hrs pair programming).
   ii. Create Script that Gets the total of all orders for the Table.
      - 1 Unit (~2hrs pair programming)..
   iii. Create HTML/CSS pay bill button and corresponding payment selection interface.
      - 2 Units (~pair programming).
   iv. Research Apple Pay API implementation and requirements.
      - 2 Units (~4hrs pair programming).
   v. Create script to pass total, restaurant, and table to Apple pay using API.
      - 2 Units (~8hrs pair programming).
   vi. Create Script to post the paid bill to restaurant with Apple API information.
      - 2 Units (~8hrs pair programming).

User Story Diagrams and Spikes

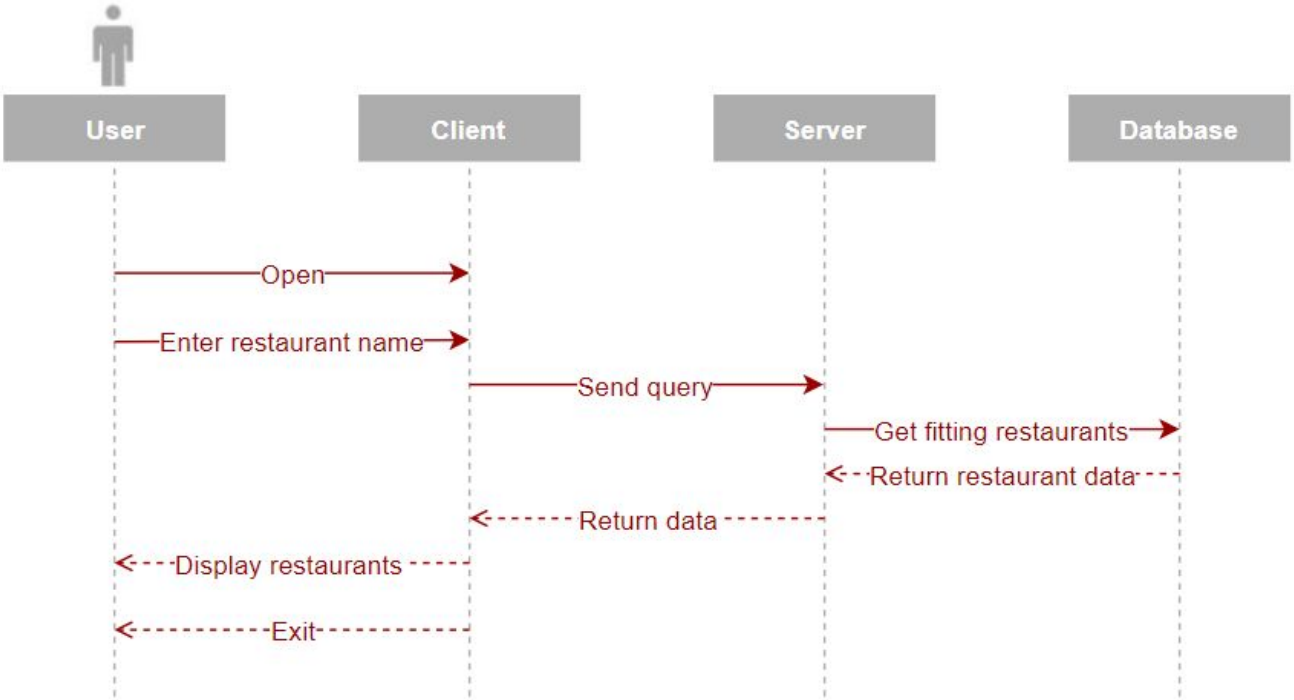
Spike. Mobile Native Application:

We performed a spike in reference to the potential to deploy a mobile native app, which is the end goal of our project, and found that the current team composition and time constraints weren't sufficient to allow deployment of a functional mobile native app as it would require the introduction of completely new development languages and environments, which would likely require a spin up time exceeding our project's schedule. The spike was performed using the Android Studio application and Flutter/Dart development environment and language. We decided instead to deploy the service as a web application at this time, with future plans to develop a mobile app depending upon customer feedback, team skill mix, and time constraints.

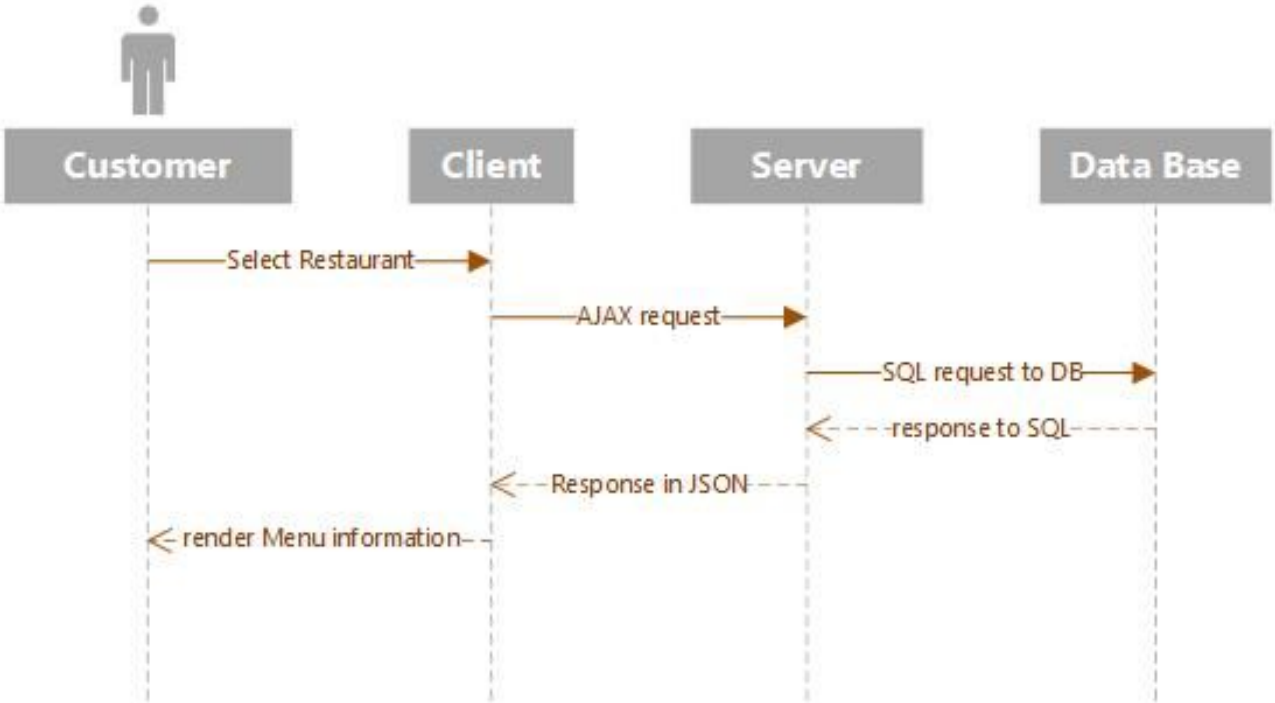
User Story 1: General Login



User Story 9: Search by Name



Story 12: View Restaurant Menu



User Story Implementation Plan

Selected User Stories

- User Story 1. General Login
- User Story 9. Search by Name
- User Story 12. View Restaurant Menu

Schedule	Team Tasks
Monday	
McCoy / Ibrahim	Configure SQL database on OSU server and communicate access credentials to the team.
Tseng / Carter	Create SQL database tables based on anticipated entities and attributes required for the first round of user story implementations. Populate tables with sample data for unit and integration testing.
Tuesday-Wednesday	
McDade / Ibrahim	Configure node.js + Express server and create appropriate application routes for the first round of user story implementations.
McCoy / Tseng	Implement and test SQL database, nodejs server integration with queries to provide the needed data elements for the first round of user stories.
Tuesday-Wednesday	
McDade / Carter	Create web application view for login functionality.
McCoy / Ibrahim	Create web application view for search by name functionality.
Thursday-Saturday	
Ibrahim / Tseng	Create web application view for displaying a restaurant's menu.
McDade / McCoy	Refine web application views for consistency and perform integration testing before customer presentation.

Customer Meeting:

- **11/14/2019:** The customer was able to meet with us Thursday afternoon for a productive meeting elaborating potential user stories. The customer provided excellent user stories with a creative flair and was very reasonable in prioritizing story implementation around providing core functionality first.

Team Contributions:

- All: Communication via private Slack channel and collaborative Google Doc.
- Timothy Tseng - User Story Tasks #7-11 and #13-20.
- Miles McCoy - User Story Tasks #2-6 and #12, User Story #12 UML Sequence Diagram.
- Ibrahim Mahmoud - User Story #1 UML Sequence Diagram, User Story #9 UML Sequence Diagram.
- Luke Carter - User Story Tasks #19 & #20.
- Mathew McDade - Document Setup, Customer Meeting, User Stories, User Story Planning, Mobile App Implementation Spike, User Story Implementation Plan & Considerations, Setup Github Repositories, Document Submission.