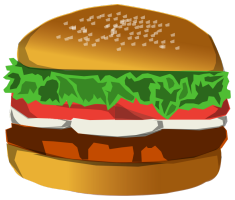


U Eat:

A Dining Concierge App

-Implementation Stage 2

Group 28

Luke Carter
Ibrahim Mahmoud
Miles McCoy
Mathew McDade
Timothy Tseng

Project Access

The test site of our mobile web app is being hosted on:
http://flip3.engr.oregonstate.edu:3716
Note: To access the site you must connect to the OSU VPN. Our site is designed as a mobile web application, so for best viewing open the site on a mobile phone or use a Chrome browser, inspect the page, and press Ctrl + Shift + M to enter mobile view mode.

Access credentials:

1. **email:** team@osu.edu **password**: ueat
2. **email:** test@test.com **password**: test

Registered Restaurants:

1. Salad House **zipcode**: 95811
2. pizzapizza **zipcode**: 11111
3. Old Ironsides **zipcode**: 95811
4. The Refuge **zipcode**: 94025
5. Kabab & Curry **zipcode**: 95050

*Additional Note: Search the zipcode '95811' to see the full effect of the multiple markers being added to the google maps search results from the zipcode search.

User Stories Completed

- User story tasks were broken down into work groups wherein one member of the work group would pursue testing, refactoring, and ancillary tasks while the other two members of a work group engaged in pair programming on the designated tasks before switching off to reflect development task assignments from the previous week's implementation planning. This allowed all five team members to be working productively and simultaneously.

User Story 10: Map Search

- **Work Group:**
  - Mahmoud, McCoy, McDade
- **Unit Tests:**
  - Render marker for restaurant on new page after sersach for known zip code: passed.
  - Render multiple markers for multiple restaurants in a single zip code: passed.
  - Return a "No restaurants found" page when no restaurants in zip code: passed.
  - Each marker populates with a message flag upon click: passed.

- The link in the message flag renders the restaurant page: passed.

➢ **Problems/Issues:**

- The main issue was determining how to provide the latitude and longitude data to the Google Map API. Mahmoud and McCoy researched the Google api and discussed to alternative solutions. One, use the stored address data and make a call to the api to return the latitude and longitude for the restaurants. That method required making a call for each restaurant before sending the data back to google a second time to generate the marker object. The second option was to make the location call to the google api upon creating the restaurant data in the database. Then our internal database would store and provide the latitude and longitude as part of the SQL call. The team chose the second method.
- The second issue was parsing the text data provided by the SQL call as the required float data type. McCoy provided an initial draft code. McDade refactored the original javascript and solved the parsing issue. McCoy reviewed McDade's final code.
- Finally, the team needed to have multiple markers on a single map. McCoy used the rendering features of handlebars to add a loop in the script for making the map markers. McDade and Mahmoud reviewed that code. McDade suggested the flags that pop up upon clicking a marker not merely contain the location data but also link to the Restaurant pages.

➢ **Time Tracking:**

- Research Google API (2 Units)/ (~4 hours paired programing)
- Implement initial map search on the server side (2 units) / (~4 hours paired programing)
- Implement initial map search view and script (2 units) / (~4 hours paired programing)
- Refactor script (1 unit) / (~2 Hours paired programing)
- Add multiple marker functionality (2 units) / (~4 hours paired programing)
- Add link to page functionality (.5 units) / (~1 hour paired programing)

➢ **Status:**

- User story successfully implemented

➢ **Remaining Tasks & Enhancements:**

- In future implementations we will get the current location from the user and return the correct restaurants based on a radius from that location. The computation would be done server side and then the view rendered with the same handlebar.

User Story 13: Restaurant Review

➢ **Work Group:**
- McDade, McCoy, & Mahmoud

➢ **Unit Tests:**
- When clicking on “Reviews” review page is correctly rendered and displayed: passed.
- Query results are correctly displayed and rendered on page: passed.
- Alert pops up if user tries to submit review with no rating: passed.
- Submitting a review correctly displays on once “Submitted:” passed.
  - Number of “Stars” is correctly displayed after submitting: passed.
  - Most recent review is displayed on bottom: passed.
  - User can not submit another review once submitted: passed.

➢ **Problems/Issues:**
- Posting and rendering a restaurant review was implemented as a client side operation. In order to accomplish this, the jQuery module was utilized. There were issues at first getting jQuery functions to work in the client application. This was traced back to the specific jQuery package that was being included, which only included core functions and not the DOM manipulation functions. This was easily fixed by updated what jQuery package was included in the client application.

➢ **Time Tracking:**
- Create “Reviews” table in Database (1 unit) / (2 hours paired programming).
- Implement “Reviews” tab within Restaurant search (2 units) (4 hours paired programming).
- Implement “star” rating option as well as Submit a review field (1.5 units) (3 hours paired programming).
- Create SQL script to get data from “Reviews” table in database (1.5 units) / (3 hours paired programming).
- Res / Render query onto webpage (1 unit) (2 hours paired programming).

➢ **Status:**
- The User Story has been successfully implemented.

➢ **Remaining Tasks & Enhancements:**
- There are no remaining implementation tasks for this user story.
- Enhancement: We will add the ability to sort restaurant reviews by multiple parameters to be decided by the customer.
- The Restaurant Review user story is otherwise complete.

User Story 17: Order Status

➢ **Work Group:**

- Carter / Tseng

➢ **Unit Tests:**

- Food items are correctly displayed and rendered when viewing current order: passed.
- Food amount is correctly displayed and rendered after when viewing current order: passed.
- Quantity is correctly displayed and rendered after viewing current order: passed.
- Order total is correctly displayed when viewing current order: passed.

➢ **Problems/Issues:**

- Issue with type and bugs caused view to not render the query results correctly.
- Had database issue which caused problems when trying to see if query results were correctly rendered.
- CSS / HTML problems when trying to render correctly on webpage.

➢ **Time Tracking:**

- Create table and attributes for Order Status (1 unit) / (1 hour paired programming)
- Create SQL code to get query (2 units) / (~4 hours paired programming)
- Create route for order status (1.5 units) / ~2 hours paired programming)
- Create view to render query results (1 unit) / (~1 hour paired programming)
- Integration / Testing (1 unit) / (~ 1 hour paired programming)

➢ **Status:**

- The User Story has been successfully implemented.

➢ **Remaining Tasks & Enhancements:**

- Add an “estimated time of arrival” display to let users know when their food will be ready.
- The Order Status user story is otherwise complete.

Spikes and Diagrams

Google Maps API / Map Markers:

- As a Team, we decided that we committed to a roughly 2 hour time-boxed research on the best way to implement and connect to the Google Maps API. Google's documentation on getting a key and connecting to the Google Maps API was very clear. The bigger challenge was researching how we would implement having multiple markers on our displayed map-- we looked at several sources, such as stack overflow for a rough idea of how we would implement multiple markers on the map. Even though we had prior experience with APIs, this Spike was very useful as none of us had too much experience with the Google Maps API.

Restaurant Review implementation:

- As a team we committed to another two hour spike to research on the best way to implement a restaurant review implementation. Our research showed that while many websites and applications such as Yelp have much more sophisticated methods for implementing reviews using front end frameworks, our spike showed that given our current time constraints and available team skills, we should combine server-side rendering for existing reviews and a client side script using the jQuery module to add new reviews. This approach minimized the complexity of the review functionality while giving a good user experience when leaving a review.

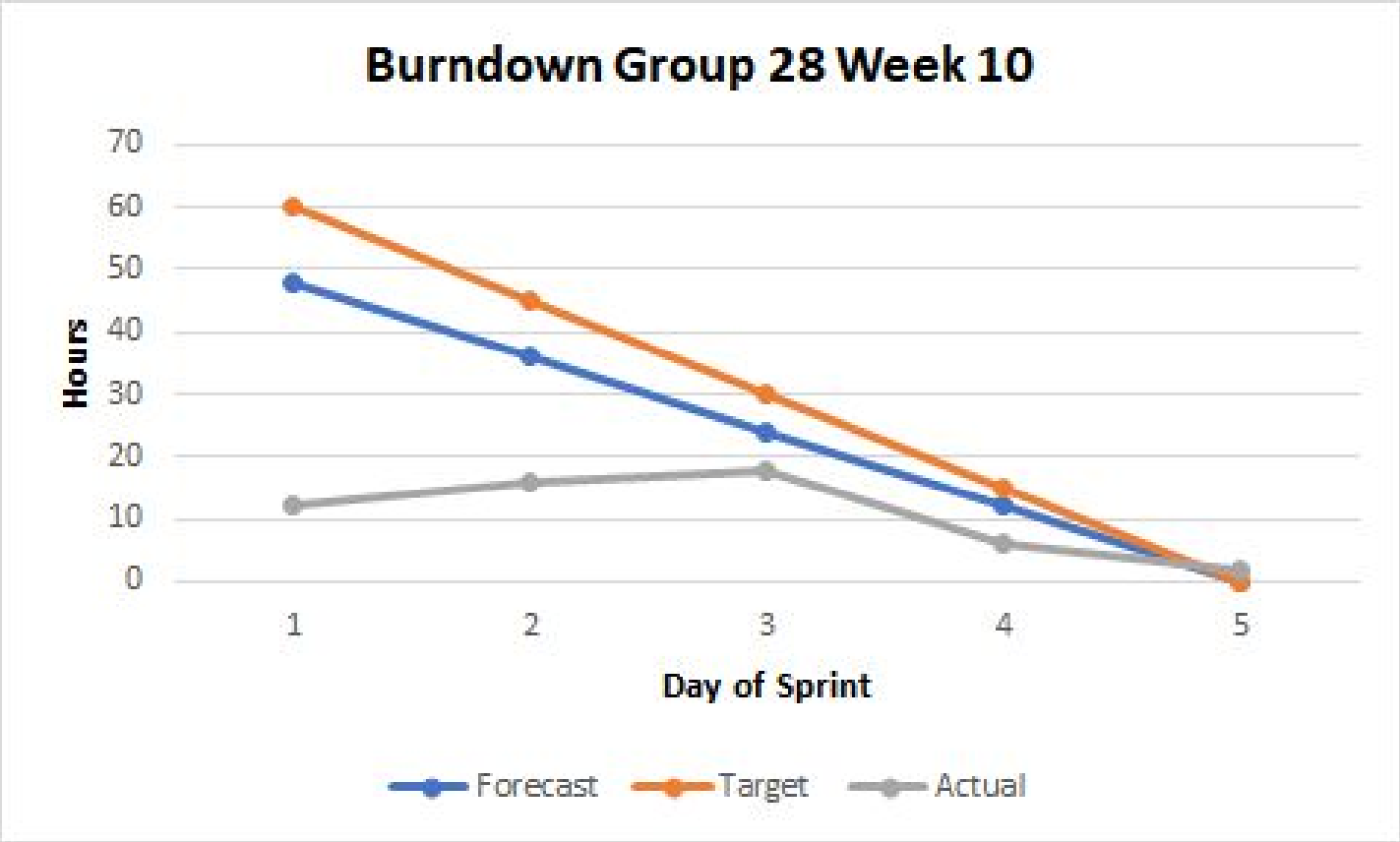
Order Status / Current Order Implementation:

- We as as team wanted to keep the layout similar to make keep a cohesive product despite having a diverse group of developers. The big piece of developing this particular functionality was creating a mysql query that would get the required information from the database. This particular page was pulling from three of the more interconnected databases which meant we had to be very precise with the information we queried. The groundwork we have laid leaves plenty of opportunities to create more of the functionality that the customer wants just by adding small pieces to the already developed larger chunks. Which with additional time will result in more well rounded software solution.
- There were no additional diagrams we desired in addition to the ones described in the previous week's implementation document.

Burndown Diagram

Notes on burndown

- Overall this week we spent more time that we did in the previous week, we spent about 44% more units of time than what we planned for. Our person per day hours was about 2.3 hours, and our average hours per person was about 11 hours with a Standard deviation of about 3.3 hours as compared to last weeks of 4.8 hours. We had planned for 38 hours and spent 55 this week so we accomplished more in less percentage of time, and in a smaller sprint. Overall we are still behind the 80% effectiveness but making progress.



Refactoring

- The team undertook various refactoring efforts throughout this week's development process, following the lesson from the refactoring lecture that it is much easier to refactor early and refactor often than to build up a refactoring debt and attempt to address all issues at once. Much of the refactoring had to be done in conjunction with integration testing between our three main components: the database, the server, and the page views.

jQuery Package:

- We performed a very simple refactoring by changing which version of the jQuery module we were using for the client application. The original version didn't have adequate functions for DOM manipulation. While this was a very simple refactor, we ensured that proper regression testing was performed. This seems especially important when changing module packages or versions as the contents of the module are largely not visible to the developers.

Database:

- The database was refactored to add attributes appropriate to this week's user stories. The changes to the database were performed incrementally and unit tested using direct queries on a development copy of the database. After the changes, regression testing was performed to ensure that attribute changes didn't adversely affect existing queries.

Customer Feedback

- We met with the customer to solicit additional feedback Thursday afternoon. We provided the customer with their own login credentials and began by allowing free exploration of the new features before moving on to specific questions.

General Feedback

- Initial feedback on the look and feel of the application were once again very positive, noting that the aesthetics were on point with the app's intended use. The customer was particularly pleased with the functionality of the Restaurant Review user story, saying that it "works perfectly" and "It's awesome." The customer was also very pleased with the look and functionality of the Map Search and View Order user stories. The customer also noted improvements to image aspect ratio displays throughout the app and was very pleased to see that we had enacted an enhancement based on the previous iteration's feedback.

Map Search Feedback

- The customer was interested in the potential implementation of automatically getting user location and providing estimated distances between the user and various restaurants. This functionality would likely require its own independent spike and be a high priority in future iterations.

Order Status Feedback

- The customer liked that the order status is being stored on the database so that an order will be saved even if the client closes or is disconnected from the application. This was one of our major goals when we initially designed the system--to use statefulness to make the application resilient against error states. The customer did ask that we add additional interactivity to the order status page, such as edit and delete functions as well as automated tax and tip calculations, as future enhancements to this user story.

Integration Testing

- **User Story 1: General Login**
  - Login with all accounts are functional
  - Login successfully redirects to main page
- **User Story 9: Search by Name**
  - Inputting a string in the textbox and pressing the “search” button will return all restaurant names with that string included.
  - Pressing the “search” button without any included string will return all restaurants.
- **User Story 10: Map Search**
  - Inputting an integer in the textbox and pressing the “world” button will display all restaurants with the inputted zipcode as markers on the Google Map.
  - Pressing on a marker should redirect to its respective restaurant page.
- **User Story 12: View Restaurant Menu**
  - After choosing a restaurant, either by search or map, the website will display all appropriate menu items and their respective prices and icons.
- **User Story 13: Leave Restaurant Review**
  - After choosing a restaurant, either by search or map, and then pressing the “Reviews” header, the website will display all reviews to the respective restaurant, as well as the option to submit a new review.
  - After submitting a review, the user will not be able to submit another.
- **User Story 17: View Order Status**
  - After pressing the “Home” icon, then “Current Order” button, the website will redirect to allow the user to view the current orders on the cart, as well as the total cost.

User Story Implementation Stage 3 Plan

Selected User Stories

- 5: Order History
- 6: Favorites
- 8: Dashboard Search

Schedule	Team Tasks
Monday	
McCoy / Ibrahim	Update database to include order history. 1 unit (~2 hrs).
Tseng / Carter	Create routes for order history. 1 unit (~2 hrs).
Tuesday-Wednesday	
McDade / McCoy	Update database to include user favorite restaurants/orders. 1 unit (~2 hrs).
Carter / Tseng	Create routes for accessing favorites restaurants/orders. 1 unit (~2 hrs).
Wednesday-Thursday	
McDade / Carter	Create views to display order history. 2 units (~4 hrs).
McCoy / Ibrahim	Create views to display favorite restaurants/orders. 2 units (~4 hrs).
Thursday-Friday	
Ibrahim / Tseng	Create routes for dashboard search. 2 units (~4 hrs).
McDade / McCoy	Create views for dashboard search. 2 units (~4 hrs).

Customer Meeting:

- **12/5/2019:** The customer was able to meet with us Thursday afternoon for a very productive meeting providing feedback on development progress and answering our questions about implementation details.

Team Contributions:

- All: Communication via private Slack channel and collaborative Google Doc.
- Timothy Tseng - Helped with integration and testing, contributed to document.
- Miles McCoy - Updated restaurant table in DB, updated DB test data to allow for map search functionality, researched Google Map API, server side script for map search, handlebar for map search, researched how Handlebars compiles at runtime, multiple marker functionality, User story 10: Map Search.
- Ibrahim Mahmoud - Created schedule for the following week, helped where it was needed during the coding, and outlined and performed the required testing for what was implemented this week.
- Luke Carter - Burndown analysis, Order status implementation, additional commenting for maintainability.
- Mathew McDade - Document Setup, Customer Meeting and Feedback, Restaurant Review User Story, Refactoring.