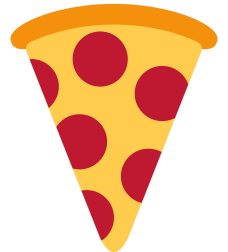


U Eat:

A Dining Concierge App

-Implementation Stage 1



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
2. pizzapizza

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 1: General Login

- **Work Group:**
 - McDade, McCoy, & Carter
- **Unit Tests:**
 - The application redirects non logged in users to the login page: **passed.**
 - The application accepts valid credentials and creates a new session: **passed.**
 - The application displays an error message if the user enters invalid credentials: **passed.**
 - Logging out ends the current session and redirects the user to the login page: **passed.**
- **Problems/Issues:**
 - We had some environmental setup issues, primarily related to the database. There were initially issues with using the ONID database. The CS361 database resolved some of the issues, but it still created difficulties coordinating with local development databases. It was difficult to make the SQL calls match across several databases. We resolved these issues by



creating shared sql scripts to build and populate a database based on the independent databases we had been working on.

- Login redirect blocking other page views: The user login route was structured to set a user session boolean value that if set to false, meaning the user was not logged in, the user would be redirected from any page back to the login page. While this is desired functionality for the end product, it slowed the development of other features as each time the server was restarted, the developers were redirected to the login page. This was easily overcome by temporarily, manually setting the session flag to true, thus saving a great deal of development time.

➤ **Time Tracking:**

- Create SQL table for login credentials:
 - 2.5 units (5hrs pair programming).
- Build web application view with appropriate layout for login:
 - 1.5 units (3hrs pair programming).
- Write server login route with verification and login redirection:
 - 3 units (6hrs pair programming).

➤ **Status:**

- This user story has been implemented and successfully tested.

➤ **Remaining Tasks & Enhancements:**

- There are no remaining implementation tasks for this user story.
- Enhancement: We will implement 'remember password' functionality.
- The General Login user story is otherwise complete.

User Story 9: Search by Name


➤ **Work Group:**


- Mahmoud, McCoy, & Tseng

➤ **Unit Tests:**

- Searching for a registered restaurant by exact name returns the specified restaurant on the search results page: **passed**.
- Searching for a restaurant by partial name returns all restaurants with a partial match to the search string: **passed**.
- Searching for a restaurant with a string not contained in any registered restaurant name will display an informational message to the user: **passed**.
- Searching for a restaurant with an empty string will return all restaurants to the user: **passed**.

➤ **Problems/Issues:**

- Single page vs. server-rendering: we moved from executing this user story as a single page application as was proposed in our initial agile planning
- 



document to having it rendered on the server side. From a data point of view, we had it working with either method, but because the extensive view changes between the initial search screen and the search results screen we opted for the server-side rendering and save single page functions for a later user story task requiring less view manipulation.

➤ **Time Tracking:**

- Create SQL table for restaurant details:
 - 1.5 units (3 hours).
- Build web application view for search page:
 - 2 units (4 hours).
- Build web application view for search results page:
 - 1 unit (2 hours).
- Write server route for querying the database and rendering search results:
 - 2 units (4 hours).

➤ **Status:**

- This user story has been implemented and successfully tested.

➤ **Remaining Tasks & Enhancements:**

- There are no remaining implementation tasks for this user story.
- Enhancement: Add single page result sorting functionality based on customer preferred customer sort parameters.
- Enhancement: After the next iteration of user stories implements Map Search and restaurant location is localized in relation to the user, we will need to decide how restaurant search return values are prioritized/limited based on proximity to the user.
- The Search by Name user story is otherwise complete.

User Story 12: View Restaurant Menu


➤ **Work Group:**

- Tseng, McDade, & Carter

➤ **Unit Tests:**

- When the user selects a restaurant, all menu items from that restaurant are displayed: **passed**.
- The application will display any images associated with menu items: **passed**.
- The application will display a default U Eat logo image beside any menu items that do not have an associated image: **passed**.

➤ **Problems/Issues:**

- Similar to the previous user story, we weighed two approaches of rendering restaurant menu data, with viable functionality for both approaches, and opted for a server-rendered page for this story.
- 

- In implementing the restaurant menu database entity, we realized that there isn't actually a need for a 'menu' entity as the relationship between restaurants and menu items is one to many, and so not requiring an independent relationship table, and thus were able to remove the task of building that specific table and combine two of the story's subsequent, related tasks.
- While testing the 'View Restaurant' function, not having everyone connect to the same database also led to some confusion that was easily fixed by everyone connecting to the same database build, as discussed further under the Refactoring section of this document.

➤ **Time Tracking:**

- Create SQL table for menu items:
 - 2 units (4 hours, including refactoring).
- Build web application view for restaurant menus:
 - 2 units (4 hours).
- Write server route for querying the database and rendering menus:
 - 2 units (4 hours).

➤ **Status:**

- This user story has been implemented and tested successfully.

➤ **Remaining Tasks & Enhancements:**

- There are no remaining implementation tasks for this user story.
- Enhancement: Menu items will eventually be grouped by food category. This will require additional database type tables as well as front end display modifications, but will enhance the user experience.
- Enhancement: We may add additional descriptions and pairing recommendations to menu items that would reflect the services traditionally provided by a professional waiter.
- Enhancement: We may add a way to display a food item's "rating" or popularity next to menu items to help the user decide on what to order.
- Enhancement: We will modify the menu items display to fix aspect ratio distortion of item images and menu item div length.
- The View Restaurant Menu user story is otherwise complete.



Spikes


Mobile Native vs. Web App:

- We conducted an initial spike to determine the feasibility of a mobile application. Based on concern with database connectivity and learning a new front end technology that none of the team mates could act as a resource on, the team decided to proceed with a web based mobile application. This spike was helpful in that it allowed us to proceed on a development path where we would be successful.

Server/DB Login Functionality

- Since there are many resources available on Server/DB Login functionality we time-boxed our spike to only 2 hours, but to be honest it was not necessary as team members have already had experienced with creating login functionality in previous classes. The spike did introduce the use of sessions to the login functionality, which improved application functionality as a whole.

Multi-Item Display

- For story 12, we also had a roughly two hour time-boxed spiked, where we researched the most effective way to display the restaurants' menus. Similar to the spike for story 1, it was not really needed as team members also had previous experience with writing a server script to pull and render a SQL query search. On the other hand the system flow diagram we created for story 12, made it easy to implement the code via code programming because each person could pick a “step” to code which made implementation much easier.
- 




Desired Diagrams

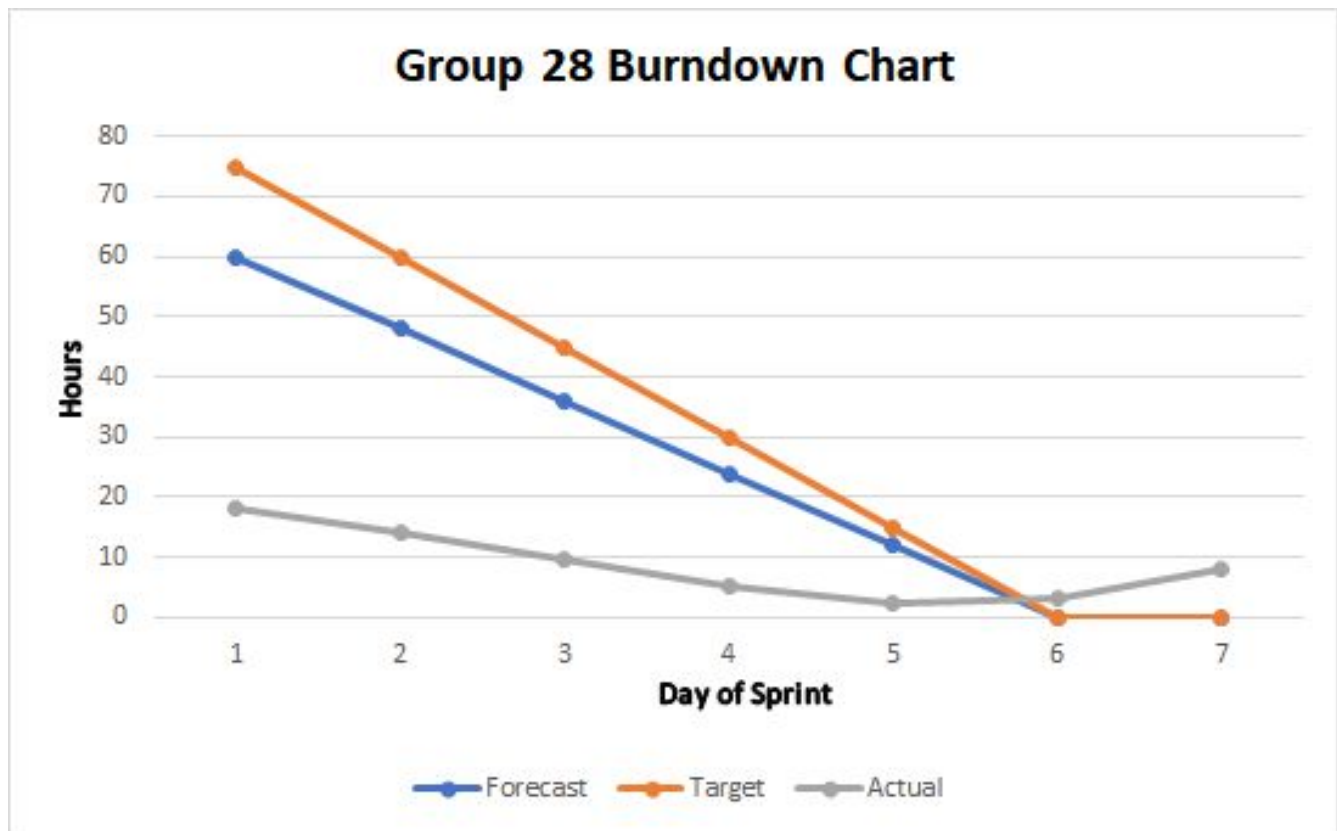
Program Implementation Diagram

- It would have been helpful to have a diagram of the overall system or design. We started with a shell of the web app, but without any indication regarding the intended functionality of designated routes. That led to building functionality in nonsensical locations and duplicate coding, even with cooperation inherent to pair programming.

Entity-Relationship Diagram

- Creating an Entity Relation Diagram specific to our implementation format would have been useful as earlier in the week we had problems with working on an unstandardized database. This would have also made implementing the restaurant menu display function smoother as mentioned before, it was until we were actually implementing the function that we realized that there wasn't a need for a menu entity.
- 

Burndown Diagram



Note: this chart includes our work on the document for this week as well.

Notes on burndown

- We had estimated that it should take us 34 hours of developing for this week's stories and it took us about 45 hours of developing. We were able to get about 70% of our development done in the first three days. On this week's assignment we as a group spent about 60 hours total. Which is about 12 hours per person.
- For this next week we will be implementing a better hours tracking system to get more precise data for our next sprint.

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.

Database:

- The database was refactored several times in order to reconcile our various development environments. The database refactoring included refactoring both table and attribute names and structures. Several of the database relationships were also refactored as we found what different relationships would require during the process of development.

Server Routes:

- Our server routes were also refactored several times to optimize when queries were being performed and which routes were rendering which views. This included deciding on what method each handler should take--we tried both GET and POST methods for several routes--and whether to package our SQL queries within functions or directly within a route. Each of these refactorings was carried out singularly in order to allow for regression testing at each step. The use of a version control system was also very helpful during this process as it allowed us the option to roll back any application breaking changes.

Web Page Views:

- Web page views were refactored around the data being passed into *handlebars* views by the routers, which were in turn based primarily on the current state of the database. The most interesting refactoring for the page views was moving from passing every individual attribute to the page view to handling JSON level logical operations using handlebars functions that allowed rendering more dynamic page structures based on what information was retrieved from SQL queries.
-

Customer Feedback

- We met with the customer to solicit additional feedback after we had the mobile web application functional and running on a flip server. We asked the customer several questions we had about specific aspects of aesthetics and functionality as well as for more generalized feedback and thoughts on our progress in meeting the customer's vision.

Initial Impression and General Feedback

- **Initial thoughts?** The customer gave very positive feedback on aesthetics and flow of the application. For aesthetics, the customer appreciated the “hunger inducing” color scheme and attention to detail in the view components, which stayed on brand with the product we had been proposing throughout our documentation. The customer was also satisfied that our application had delivered on the functionality of the first priority user stories. Overall, the customer was impressed with polished product delivered after only one week of agile development.

Login Questions and Feedback

- **Does the login functionality as reflected by the navigation icons meet the customer's expectations?** The customer was very satisfied with the flow of the login functionality and like the dynamic dropdown components.

Search Questions and Feedback

- **Would you prefer that an empty search string return all possible results--current state--or no results?** The user prefers the current state of our search functionality on an empty search string. Returning all available restaurants on an empty search allows the user to discover new restaurants when they don't know the name of a restaurant. **Customer Note:** Once location based searching is enabled, name search will ideally return a limited number of results based on proximity to the user. The customer also noted an aesthetic problem with the aspect ratio of the search page carousel images which has subsequently been corrected.

Menu Questions and Feedback

- **Are there any additional enhancements you would like to see for the application's menu display?** The customer would eventually like to see detailed item descriptions, pairing recommendations, and other enhancements related to menu items. The long term goal of these enhancements would be to provide U Eat users with services akin to a virtual server or, as our tagline goes, a virtual dining concierge.
-

Integration Testing

We conducted integration testing throughout the process. The first test was to check login compatibility with development database. Login worked with the local database, but had difficulty connecting to the original development database. This led to moving away from the ONID database and switching to the CS361 database. To facilitate the change, several tables in the development database were renamed.

The second issue concerned the .env file required for launching a working copy of the application. The primary issue was login credentials for the database and port binding. That issue became more of a development issue than a final prototype issue after refactoring. The issue resulted in additional time discussing how to install and launch the application. We solved the problem by making the development database credentials public during the initial pair programming and switching to a database with hidden credentials after refactoring. This allowed access to the development database by all team members with a simple git clone or pull.

After the first round of pair programming pieces of the system operated, but did not have a consistent style. They also did not follow a logical organization. The menu item page did not have any of the style of the main page or search page. The search result page had some of the same style but did not match the overall mobile design. The route handlers both needed to be refactored. The views both needed to be refactored.

After initial refactoring, the menu item page no longer operated. It returned a 404 error. The issue resulted from an incorrect route included in the route handler .get call. After the call was changed to the correct route the page loaded correctly.

The current state of the application provides full functionality. The login screen accepts a username and password. After logging in the search screen populates with search by name and the search by zip code map. The search by zip code map was not an implemented story. If the user searches no name, all available restaurants populate. The two individual restaurants in the database both return when individually searched. When the name is selected a new page populates with the menu data. All three of the user stories operate as intended.

User Story Implementation Stage 2 Plan

Selected User Stories

- User Story 10: Map Search
- User Story 13: Leave Restaurant Review
- User Story 17: View Order Status

Schedule	Team Tasks
Monday	
<i>McCoy / Ibrahim</i>	Create routes for map search. 2.5 units (~5 hrs).
<i>Tseng / Carter</i>	Menu item aesthetics and detail enhancements. 2 units (~4 hrs).
Tuesday-Wednesday	
<i>McDade / Ibrahim</i>	Create routes for leaving a restaurant review. 2.5 units (~5 hrs).
<i>Carter / Tseng</i>	Create handlebars views for map search. 2 units (~4 hrs).
Wednesday-Thursday	
<i>McDade / Carter</i>	Create routes for viewing order status. 2.5 units (~5 hrs).
<i>McCoy / Ibrahim</i>	Create handlebars view for restaurant review. 2 units (~4 hrs).
Thursday-Friday	
<i>Ibrahim / Tseng</i>	Unit and integration testing. 3 units. (~6 hrs).
<i>McDade / McCoy</i>	Create handlebars view for viewing order status. 2 units (~4 hrs).

Customer Meeting:

- **11/30/2019:** The customer was able to meet with us Saturday evening 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 - Pair programming for restaurant menu display function as well as testing all other website functions / code as well as contributing to various documentations.
 - Miles McCoy - Created development database and schema, populated with dummy data for testing, all SQL calls to that database, pair program on search by name and menu display, integration testing and unit testing.
 - Ibrahim Mahmoud - Created express routes for searching by name for restaurants and displaying menu items for each one, as well as devising the schedule for the following week.
 - Luke Carter - Burndown diagram creation, and time spent analysis.
 - Mathew McDade - Document Setup, User Stories Completed, Spikes, Refactoring, Customer Meeting & Feedback.
-