lightwait

Final Report

May 4, 2014

Adrian Hernandez

Patrick Leopard II

Luke Oglesbee

Alec Siems

Ryan Sligh

Joe St. Angelo

lightwait

Product

lightwait is an ordering solution for SMU Dining at Mac's Place. It allows the customers of Mac's Place to order through a mobile app, on their own computers, or through a local kiosk in Mac's Place. However, mobile users are notified when their order is complete. Mac's Place chefs can view a digital queue of orders, bumping orders from the queue as they are finished. Administrators of Mac's Place have access to all the information from the digital orders, giving them access to valuable analytics.

Team

We are Jimmy and the Dragons, and we believe that we can make the Mac's Place experience more enjoyable for everyone. We want to make a product that is comfortable, consistent, and highly usable for all. Here is our team:

Adrian Hernandez - Database Management

Patrick Leopard II - Primary iOS Developer

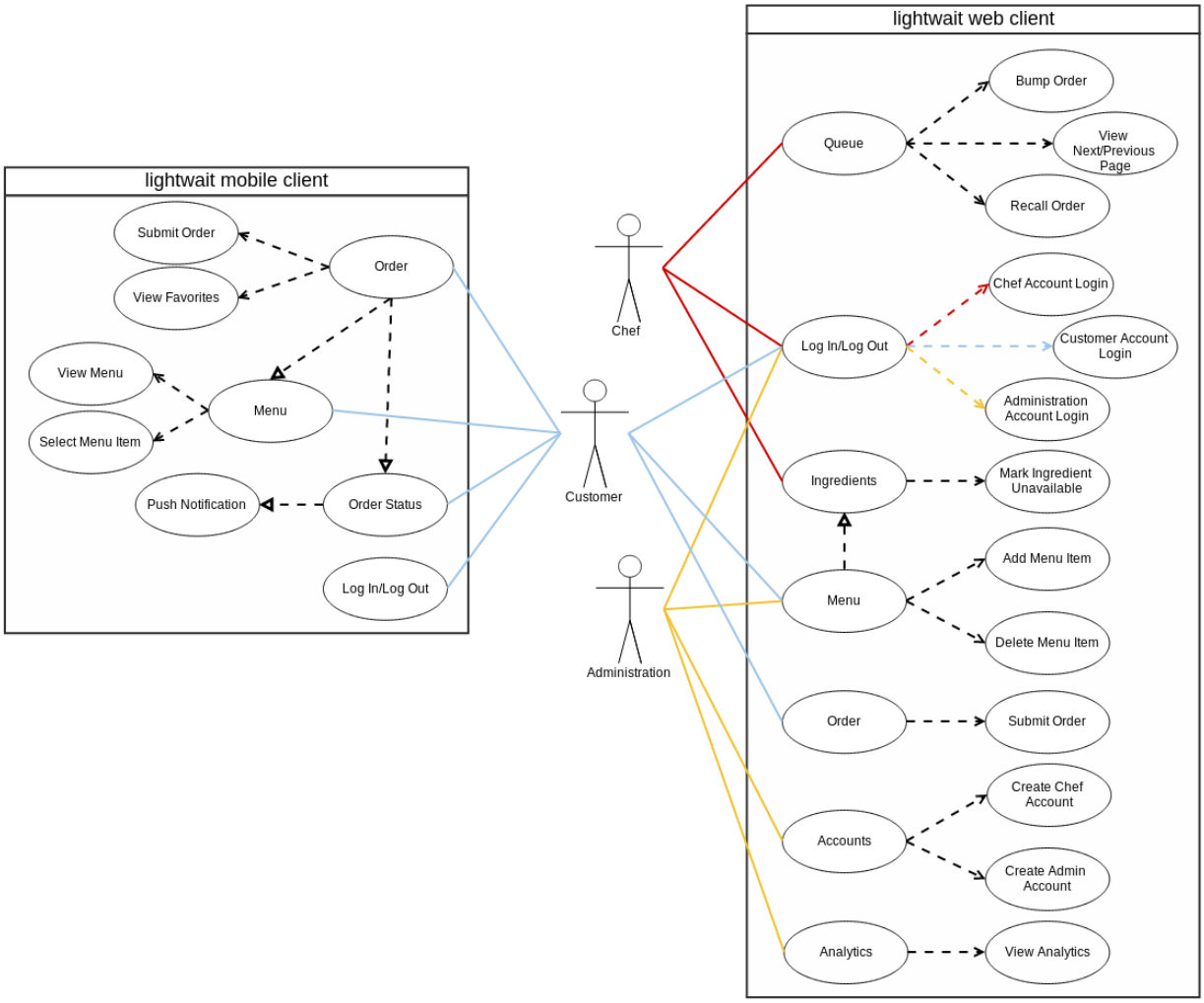
Luke Oglesbee - Primary Web Stylist

Ryan Sligh - Layout and Javascript Developer

Joe St. Angelo - Javascript and JQuery Manager

Alec Siems - Middleware Development

Use Case Diagram



Features

Customer

lightwait customers can place an order through either our web or mobile app. Our mobile app will alert you when your order is bumped, and you can save your favorite orders on the mobile device.

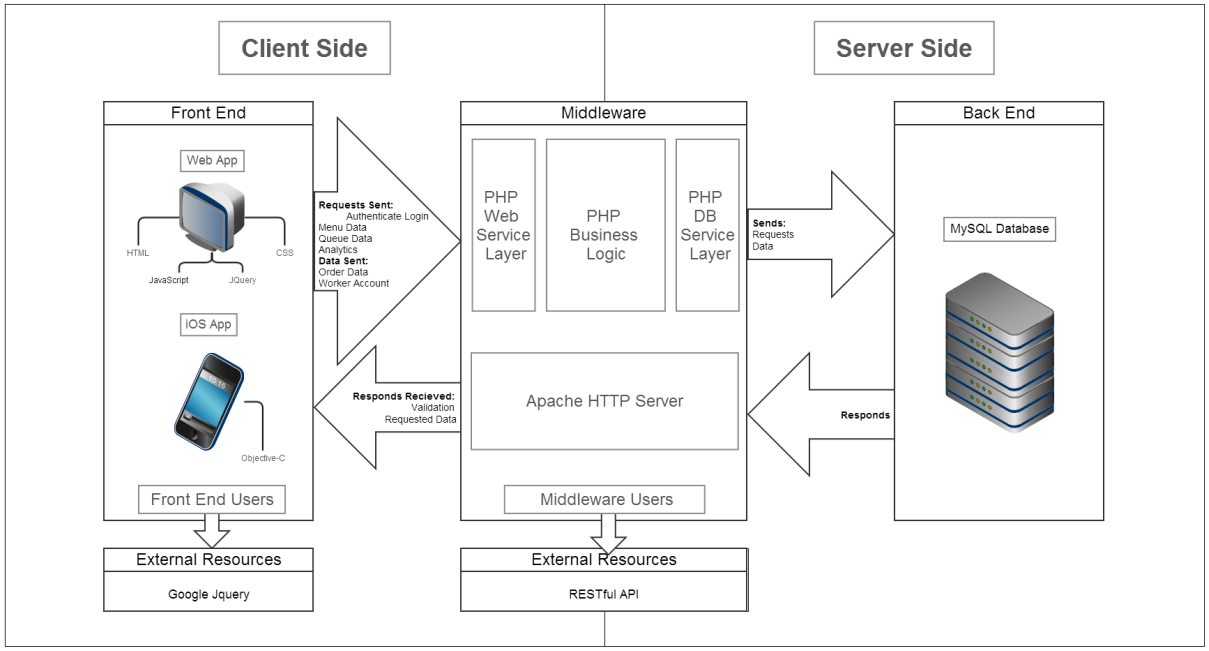
Chef

Orders sent by the customers are displayed neatly in a queue with a sidebar showing the total number of bases ordered to allow the chefs to quickly see what needs to be placed on the grill. When the chefs finish an order, they can the bump order off the queue, and *lightwait* will automatically notify the customer that their order is finished. If the kitchen runs out of, or restocks, an item, the chefs can easily make that item unavailable or available on the menu. The chef's interface is designed to minimize the input needed to perform any action to the bare minimum to allow for more efficiency in busy work environments.

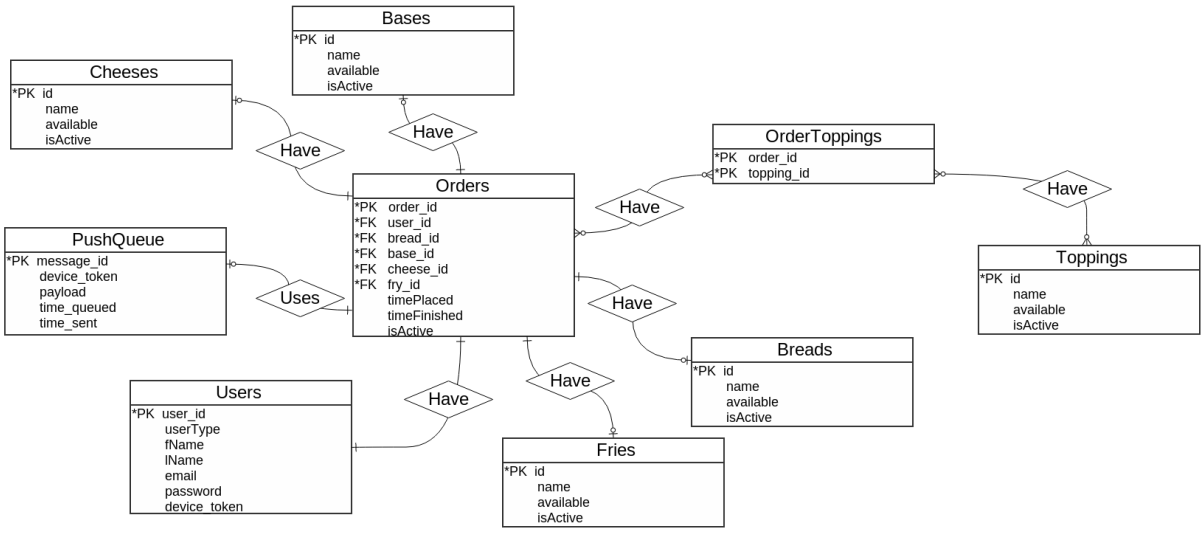
Administrator

Administrators have the capability to manage both the menu and personnel in *lightwait.* They can add or delete items as well as make an item temporarily unavailable for customers to order. They can also make new chef or administrator accounts, and view analytics for orders.

Software Architecture

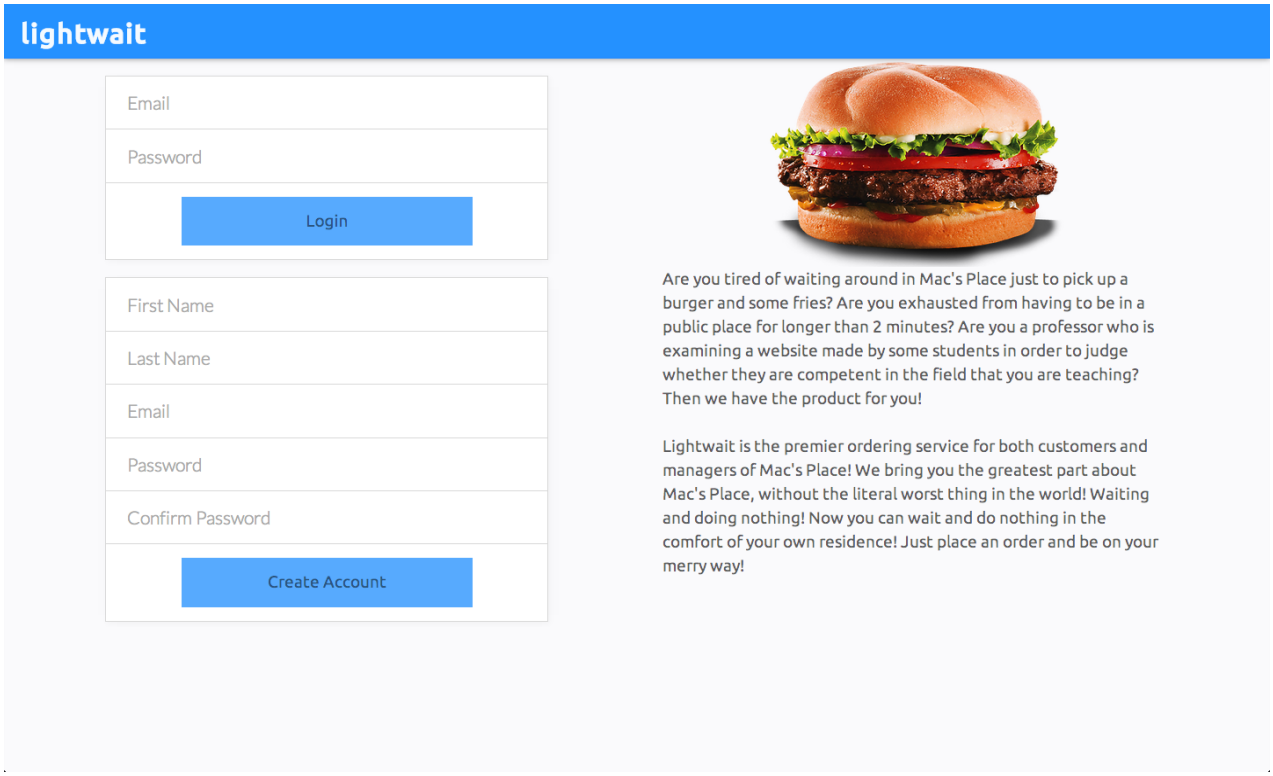


Database

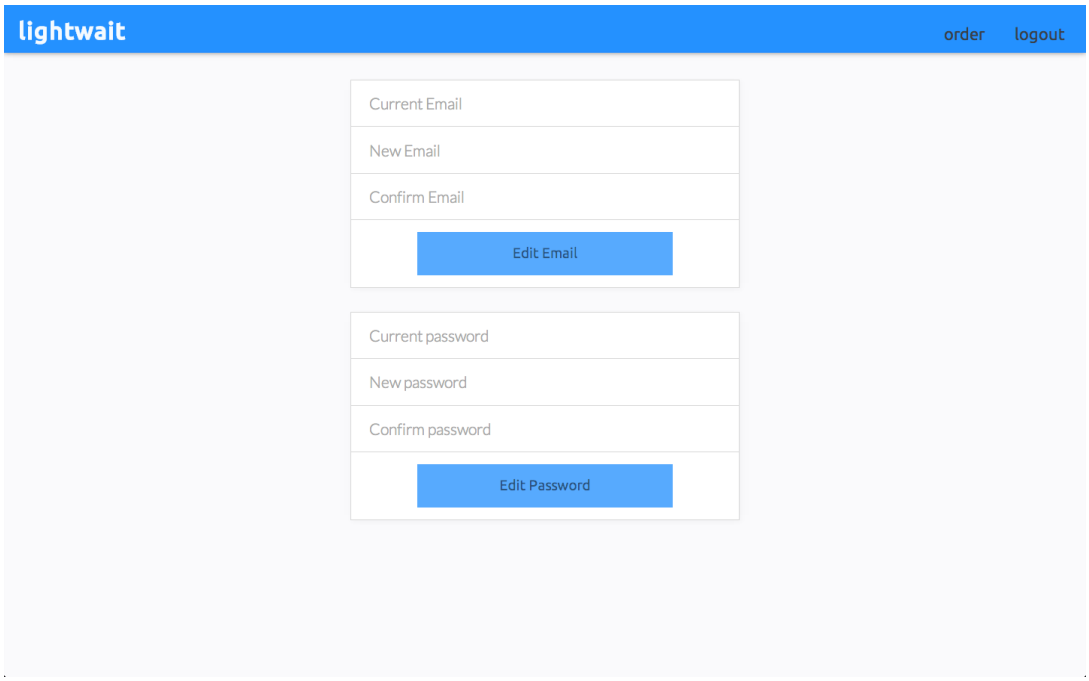


Database Model (for Data Dictionary see Appendix)

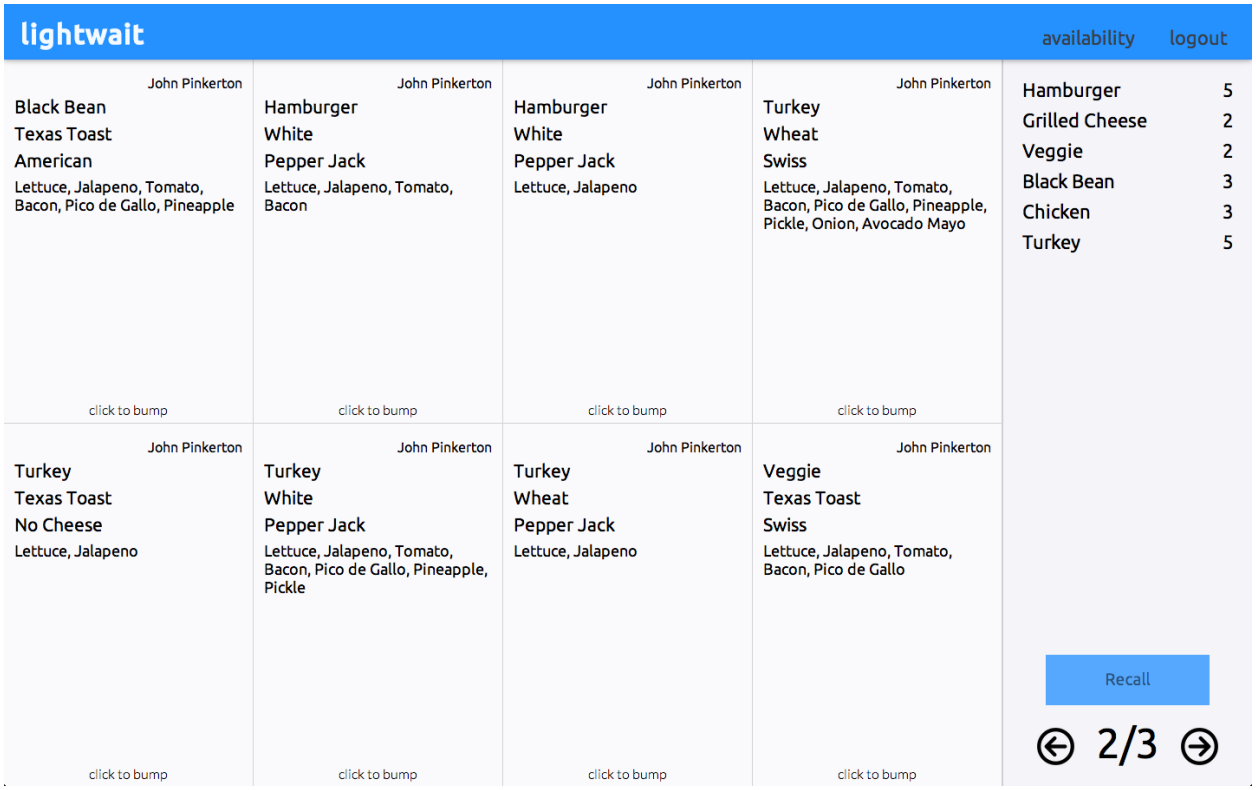
User Interface



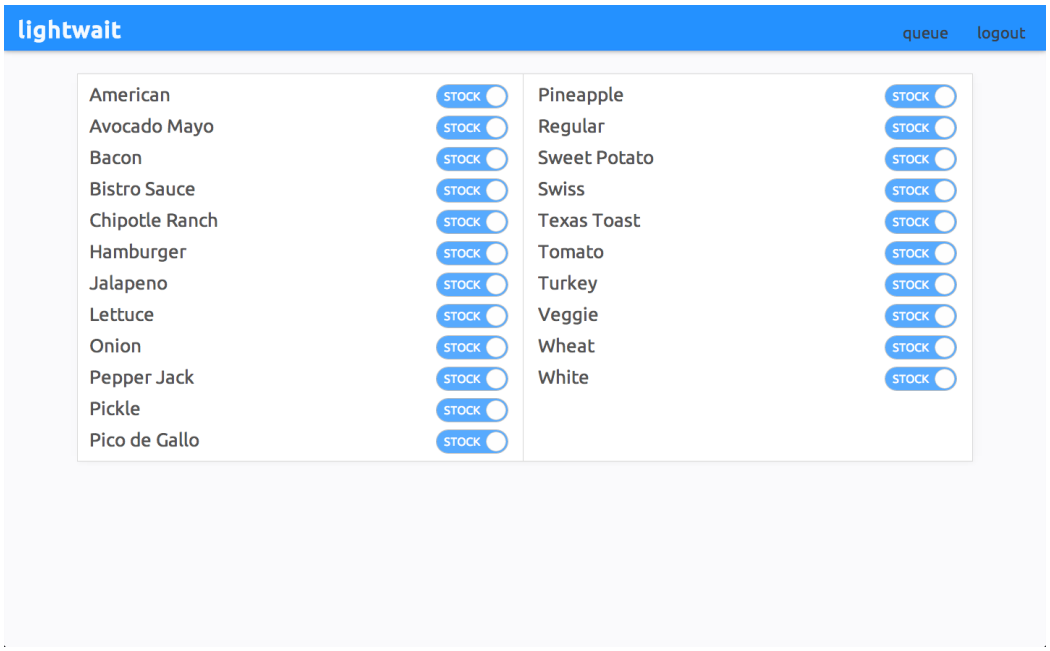
Home - Users can login or create an account.



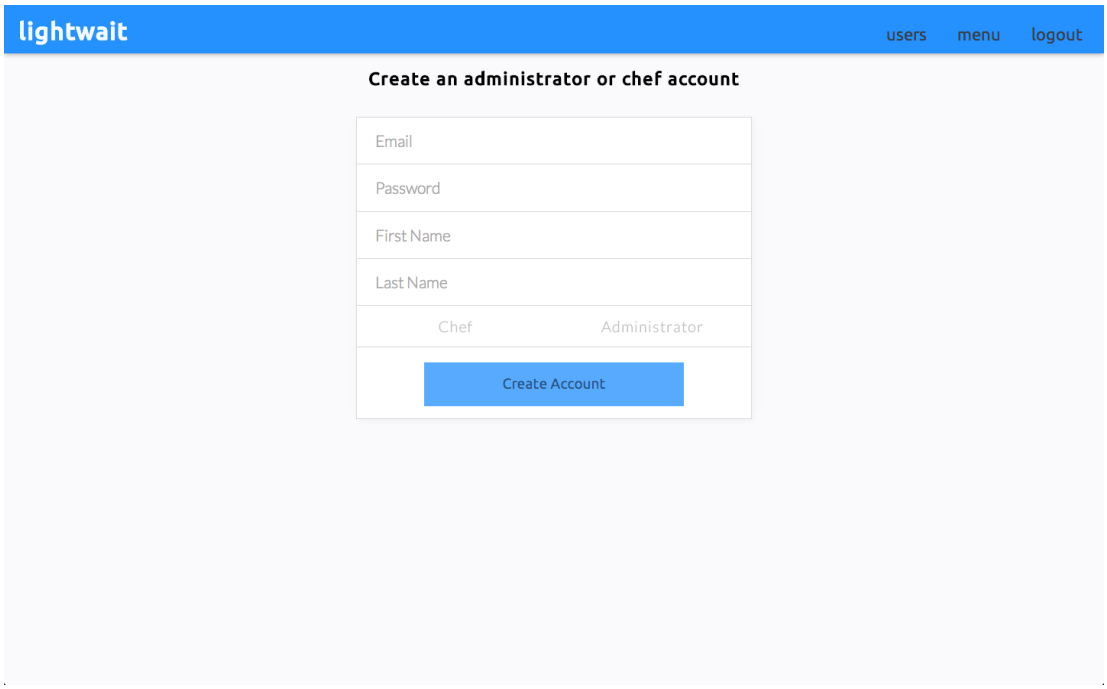
Edit Account - Users can change email or password.



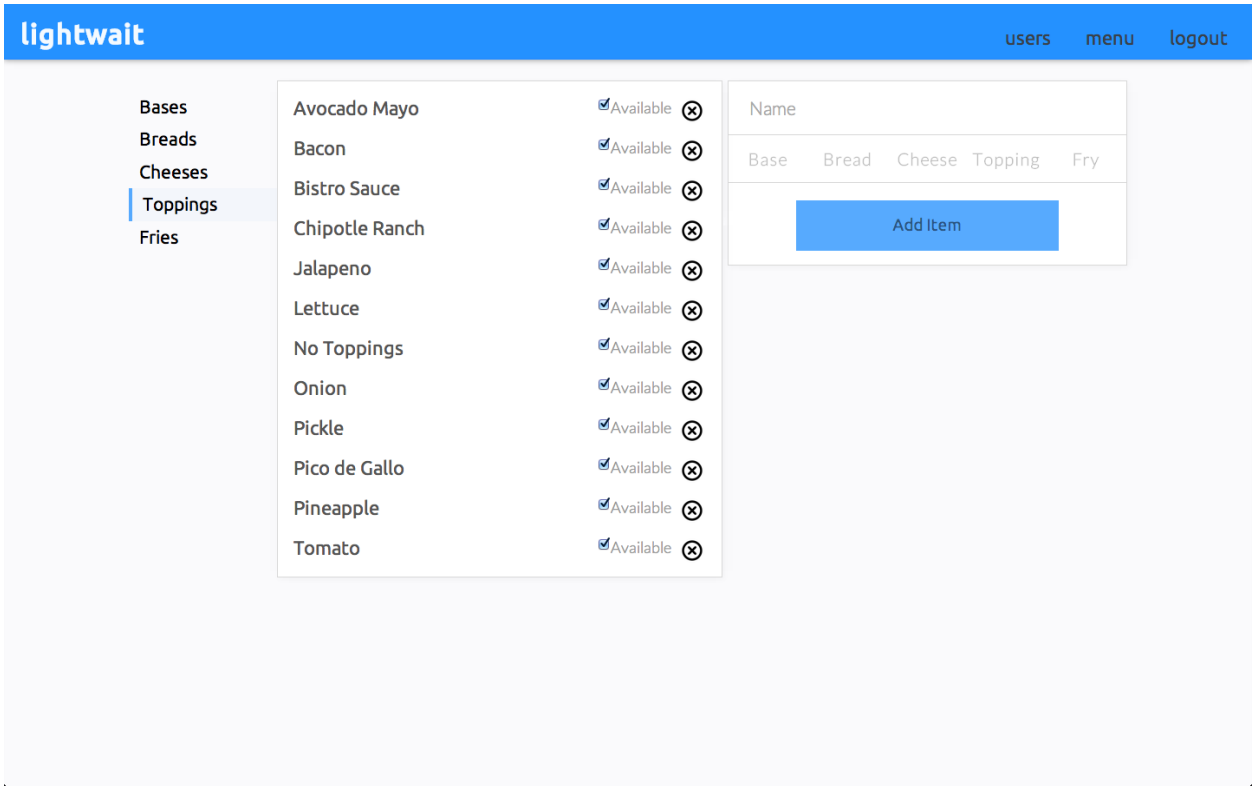
Chef Queue - Chefs can view, bump and recall orders. The sidebar contains a quick view, showing the bases of all active orders.



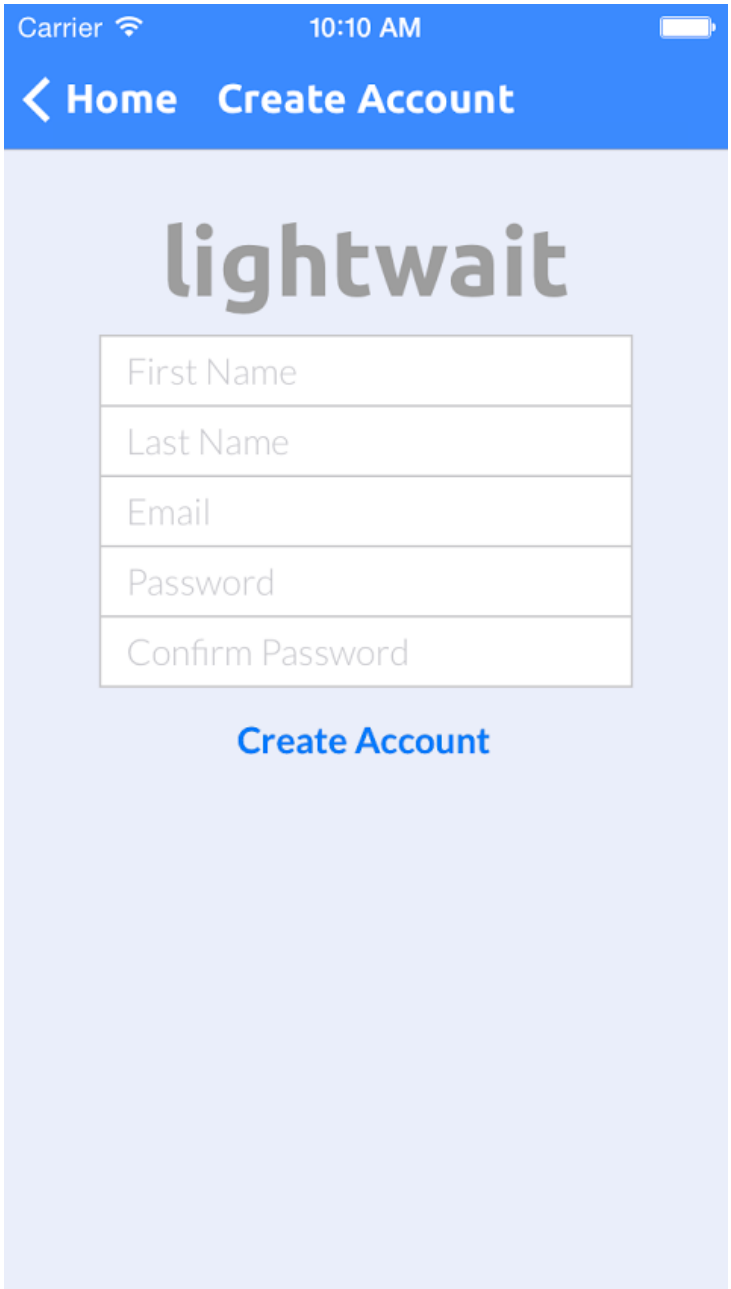
Chef Availability - Chefs can make ingredients unavailable for customers to order.



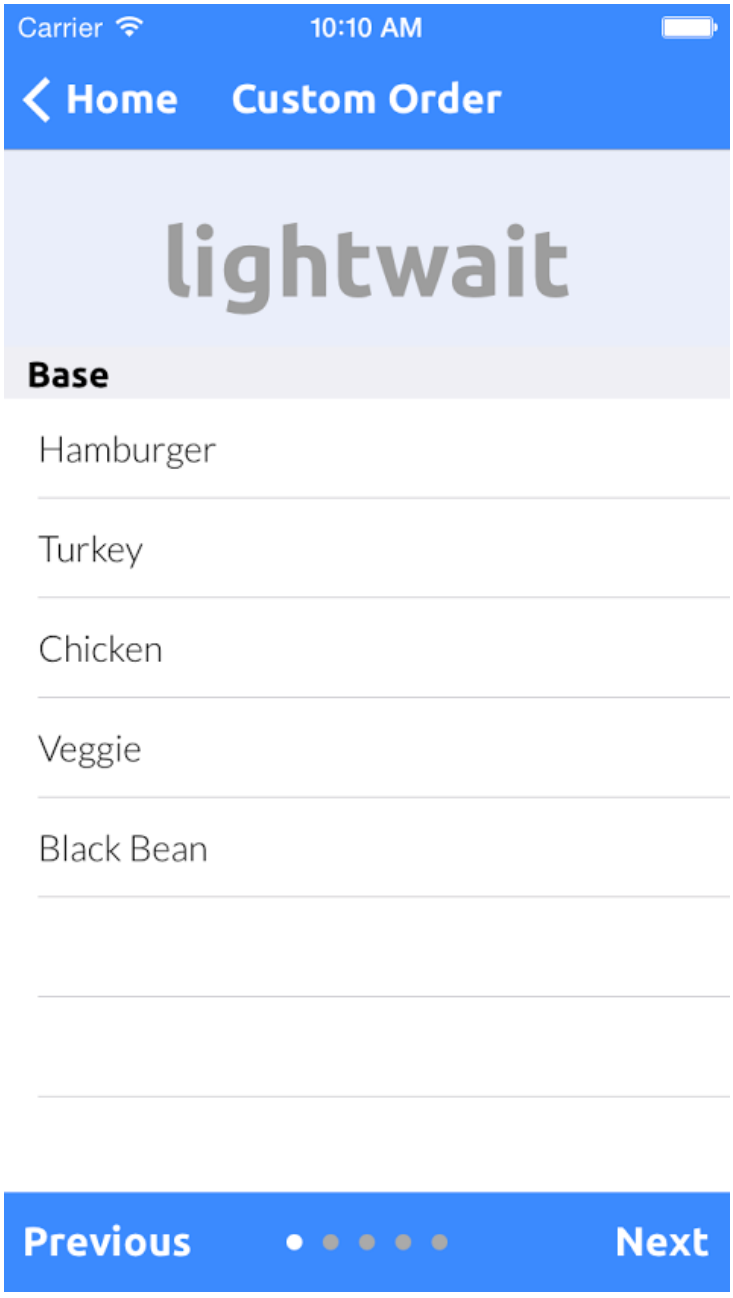
Administrator Create Account - Administrators can create chef or administrator accounts.



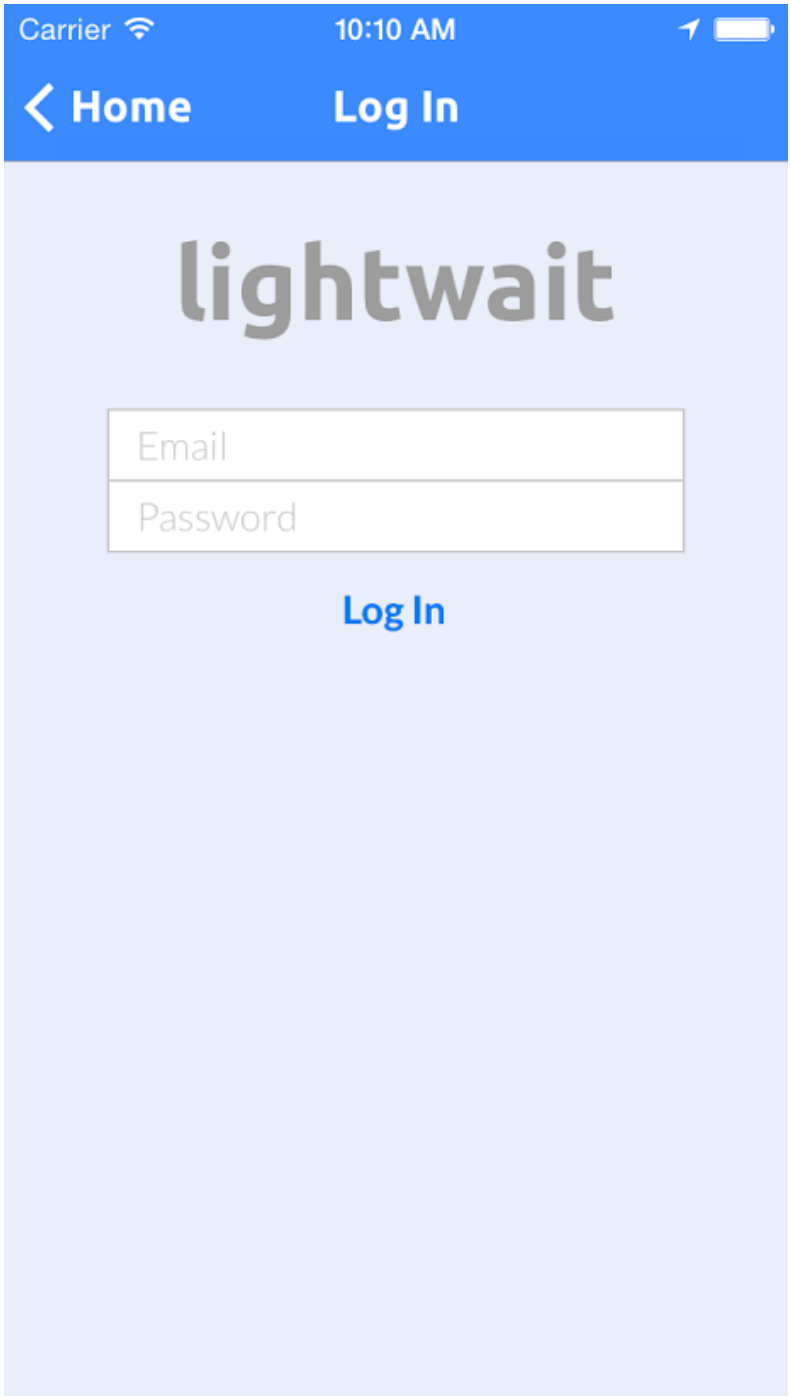
Administrator Menu - Here administrators make changes to the menu, including adding items, deleting items, and making items unavailable for customers to order.



Create Account - Users can create an account.



Custom Order - Users can create a custom order and then save the order should they choose to.



Log In - Users can log in with their username and password.

Testing

Testing Methods

We had the pleasure of having Team 5 test two iterations of *lightwait*. We gave them access to our source code via github so they could stress test our system and find problems. For each iteration, we provided them with a list a functional features so they knew what to test. When a member of Team 5 found a problem they would file an issue in github's integrated issue tracking application. While Team 5 was our largest group of testers, we also had several other students test our product.

Members of Team 5 are Taylor Bishop, Bruce George, Kenneth Politz, Hector Curi, Matthew Morris.

Testing Results

- Recall bumped more than one order
- Our input validation was not complete enough
- Long orders were overflowing off the visible order content area
- Session problem when using the back button
- Multiple accounts with the same email were allowed

Our Testing

While testing Team 5's applications, we reported 35 issues out of their total of 78. We notified them that you could hack into accounts using a SQL injection, which is a major security flaw. We also flagged issues relating to the functionality of loading and displaying the presentations. Lastly, we gave design and usability advice and condoned their industrial design, which they have since modified.

Team Reflection

Technical Challenges

This greatest challenge of creating *lightwait* was figuring out how to use Slim, which none of us had used before. There were many small roadblocks, but the rest of production was relatively straightforward after we understood Slim.

Outside Knowledge

Several sources of outside knowledge assisted in the development of our product. First, prior

knowledge of iOS development helped lead to the decision to develop an iPhone app. Also, several members had previous experience working with databases that helped lead to the development of our API. Additionally, we acquired a common sense of design through studying outside websites such as *Facebook* and *The Noun Project* in order to provide a consistent web and mobile experience. We also used *proto.io* to make flip switches.

Advice to Future Semesters

The Taco Truck assignment is a solid testing ground for design and team dynamic ideas, so put full effort into it. We met very often and got a lot of work done during our meetings, but we often didn't do work outside of our meetings. Future students of this class should find a good balance between working in a group and working individually. If team members do their individual work, group meetings can be concise and productive instead of long and laborious. Team meetings can be time to touch base with team members and tackle technical roadblocks. We believe this will allow meetings to be shorter if they arrive ready with a new set a problems to get help with.

Retrospection

Looking back, we may have picked a different PHP framework to create our RESTful API. After figuring out the finer details of Slim, we had few problems implementing it. Though finding other, more feature-rich frameworks, we could have added additional functionality to our product. We also would have been more open to changing the database design instead of working with what was already created. Finally, as stated above, we would have worked more individually in addition to our group meetings.

Wrap Up

A Next Set of Features

- Data - provide an advanced query functionality
- Provide estimated order completion time
- Color coded queue windows corresponding to certain base types
- Promotions and sales
- Mobile application for administrators

Appendix

Design Sources

Proto.io - http://proto.io/freebies/onoff/

Facebook - www.facebook.com

The Noun Project - www.thenounproject.com

Users - Customer, chef, and administrator accounts

- **user_id** is an INT(30) that is a NOT NULL, a PRIMARY KEY and AUTO_INCREMENTED. This number uniquely identifies each user.
- **userType** is an INT(1) that is NOT NULL and DEFAULT 1. This number identifies the user as either customer, chef, or administrator.
- **fName** is a VARCHAR(255). This is the user's first name.
- **lName** is a VARCHAR(255). This is the user's last name.
- **email** is a VARCHAR(255). This is what user logs in with.
- **password** is a VARCHAR(255) that is hashed. This is what the user logs in with.
- **device_token** is a VARCHAR(64). It is a hex number that is a unique identification number for each iPhone and the *lightwait* app used to send out messages from the Apple Push Notification Service.

Fries - The types of fries on the menu

- **id** is an INT(30) that is NOT NULL, a PRIMARY KEY, and AUTO_INCREMENTED. It is used to uniquely identify the type of fry.
- **name** is a VARCHAR(30). It is the name of the type of fry.
- **available** is a BOOLEAN that is DEFAULT TRUE. This determines whether customers are able to order that particular type of fry.
- **isActive** is a BOOLEAN that is DEFAULT TRUE. This determines whether or not an order has been deleted.

Breads - The types of breads on the menu

- **id** is an INT(30) that is NOT NULL, a PRIMARY KEY, and AUTO_INCREMENTED. It is used to uniquely identify the type of bread.
- **name** is a VARCHAR(30). It is the name of the type of bread.
- **available** is a BOOLEAN that is DEFAULT TRUE. This determines whether customers are able to order that particular type of bread.
- **isActive** is a BOOLEAN that is DEFAULT TRUE. This determines whether or not an order has been deleted.

Bases - The types of breads on the menu

- **id** is an INT(30) that is NOT NULL, a PRIMARY KEY, and AUTO_INCREMENTED. It is used to uniquely identify the type of base.
- **name** is a VARCHAR(30). It is the name of the type of base.

- **available** is a BOOLEAN that is DEFAULT TRUE. This determines whether customers are able to order that particular type of base.
- **isActive** is a BOOLEAN that is DEFAULT TRUE.This determines whether or not an order has been deleted.

Cheeses - The types of cheeses on the menu

- **id** is an INT(30) that is NOT NULL, a PRIMARY KEY, and AUTO_INCREMENTED. It is used to uniquely identify the type of cheese.
- **name** is a VARCHAR(30). It is the name of the type of cheese.
- **available** is a BOOLEAN that is DEFAULT TRUE. This determines whether customers are able to order that particular type of cheese.
- **isActive** is a BOOLEAN that is DEFAULT TRUE.This determines whether or not an order has been deleted.

Toppings - The types of toppings on the menu

- **id** is an INT(30) that is NOT NULL, a PRIMARY KEY, and AUTO_INCREMENTED. It is used to uniquely identify the type of topping.
- **name** is a VARCHAR(30). It is the name of the type of topping.
- **available** is a BOOLEAN that is DEFAULT TRUE. This determines whether customers are able to order that particular type of topping.
- **isActive** is a BOOLEAN that is DEFAULT TRUE.This determines whether or not an order has been deleted.

OrderToppings - Matches the toppings to orders

- **order_id** is an INT(30) and PRIMARY KEY. It represents the order that a particular topping relates to.
- **topping_id** is an INT(30) and PRIMARY KEY. It represents the toppings that a particular order relates to.

Orders - Meals that customers order

- **order_id** is an INT(30) that is NOT NULL, a PRIMARY KEY, and AUTO_INCREMENTED. It uniquely identifies every order placed.
- **user_id** is an INT(30) that is NOT NULL and a FOREIGN KEY that references user_id in the Users table.
- **bread_id** is an INT(30) that is NOT NULL and a FOREIGN KEY that references id in the Breads table.
- **base_id** is an INT(30) that is NOT NULL and a FOREIGN KEY that references id in the Bases table.

- **cheese_id** is an INT(30) that is NOT NULL, a FOREIGN KEY, and DEFAULT 0 that references id in the Cheeses table.
- **fry_id** is an INT(30) that is NOT NULL, a FOREIGN KEY, and DEFAULT 0 that references id in the Fries table.
- **timePlaced** is a TIMESTAMP that is NOT NULL and DEFAULT 0. It represents when the order was placed.
- **timeFinished** is a TIMESTAMP that is ON UPDATE CURRENT_TIMESTAMP. It is the time that isActive was set to FALSE.
- **isActive** is a BOOLEAN that is DEFAULT TRUE. It represents whether an order is in the order queue.

PushQueue - Information for push notifications

- **message_id** is an INT(11), NOT NULL, a PRIMARY KEY, and AUTO_INCREMENTED. It uniquely identifies that table's row.
- **device_token** is a VARCHAR(64) and NOT NULL. It is a hex number that is a unique identification number for each iPhone and the lightwait app used to send out messages from the Apple Push Notification Service.
- **payload** is a VARCHAR(256) that is NOT NULL. It is the JSON that contains the notification information.
- **time_queued** is DATETIME and NOT NULL. It represents the time when it was added to the queue
- **time_sent** is DATETIME and DEFAULT NULL. It represents when the notification was sent from the Apple Push Notification Service.