Software Engineering CSC 4350

Fall 2016

Patio

Group 4

Noah Aragon, Simon Phillips, Nick Clark, Rett Gerst, Terashia Blake

November 29, 2016

Table of Contents

Introduction:

Nick Clark

Introduction

Background: I am a junior, undergraduate Computer Science student. I started showing interest in Computer
Science freshmen year of high school and have been taking courses since.

Programming Knowledge: I am most familiar with Java and have been using it for many different classes for 5 years. I have
taken most of the 2000-3000 level CSC courses here at GSU.

Software Development: This is my first software engineering course and I do not have any real world experience with software development. My most extensive software project would probably be a small online game I created several years ago. Unfortunately, my free domain has expired so it is no longer accessible. It was a Connect Four remake scripted in JavaScript.

Interests: I am not entirely sure what exact field I am most interested in pursuing, but I have always enjoyed web development so that may be my choice.

Take Away: Unlike most other courses I have taken thus far, I believe the SWE will provide with a tangible project at the end of the course. Other classes have many small assignments, none of which can be used to really show what I am capable of or have been a part of. Also, I have not had any experience with databases prior to this class, so I am excited to learn about those.

Noah Aragon

Introduction

Background: My father sparked my interest in computers at a young age by letting me help complete projects with him. Since, I have decided that it is what I would like to make a career out of. I am currently in my senior year as a Computer Science major.

Programming Knowledge: So far, most of my courses as a Computer Science major have been focused on Java. I have had experience, though very little, with PHP, MySQL, HTML, C, and C++.

Software Development: I currently do not have any software development work or projects. I am hoping at the end of the semester I will be able to revise this and say that I have successfully taken part in a software development project.

Interests: My current academic interest is accomplishing what I need to do in order to graduate. Behind that, on a more serious note, comes web development. I took a Web Programming course here at GSU and I really enjoyed it. I would like to pursue that further.

Top Two Takeaways: The two things I would like to take away from this course are: a good grade and working better with a team. One of my main concerns, probably alongside everyone taking the course, is walking away with a good grade. Secondly, I am looking forward to gaining experience working with a team on a long term project. I have not worked on a team with assigned roles for a major project in the past. I believe that having that experience will benefit me in preparation for the real life workplace.

Gerald Gerst (Rett)

Current address: 450 16th St NW, Atlanta, Georgia 30363, United States

Permanent address: 6810 Center Grove St., Cumming, Georgia 30040, United States

Phone: 678-327-5512

Email: tricagamma@gmail.com

Summary

21-year-old college student majoring in Computer Science. Passionate about and proficient in computing technology.

Professional Experience

Apago, Inc. - Alpharetta, Georgia, United States

Intern (Summer 2015, Winter 2015-2016, Summer 2016)

- MEAN (MongoDB, Express.js, AngularJS, NodeJS) stack, full-stack web development
- Linux server administration (Amazon Linux, a Red Hat-based distribution)
- Created pdfshrink.com over the course of the summer
  Note: I did not do the graphic design, I was provided with a mockup image of the design of the site and then implemented it from scratch myself
- Created a tool for internal company use based on the same technology stack over the course of the winter break
- Created a product currently under NDA, using same technology stack over summer 2016
- Created several HTML/AJAX/node form interfaces for integration with various client workflows and CRM systems

Marco's Pizza - Cumming, Georgia, United States

Driver (December 2014 – August 2015)

- Delivery
- Cleaning, recovery and other miscellaneous activities

Endurance Karting - Alpharetta, Georgia, United States

Sales Associate (January 2014)

- Data entry (Excel)

American Eagle Outfitters - Cumming, Georgia, United States

Sales Associate (December 2013 - January 2014)

- Customer service, Stocking, Recovery

Dollar General - Cumming, Georgia, United States

Sales Associate (June 2013 - August 2013)

- Run register (scan items, handle money including checks, bank cards, and cash, work with a wide variety of customers)
- Recovery (pay attention to any stray items or unorganized shelves and replace or tidy items when needed) • Stocking

Unisource Worldwide – Cumming, Georgia, United States

Temporary Data Entry (October 2012 – March 2013)

- Enter data from various sources into a uniform, pre-defined format in an Excel/Numbers spreadsheet.
- Detect and correct database errors when needed.

ClosetSmith, Inc. – Alpharetta, Georgia, United States

Web Designer and Assistant (June 2011 - August 2011)

- Perform or direct web site updates
- Clean the product assembly area when necessary, including organizing materials and cleaning specialized equipment

Education

West Forsyth High School - Cumming, Georgia, United States

Graduated May 2013

Georgia State University - Atlanta, Georgia, United States

Expected graduation May 2017, majoring in Computer Science, holding a 3.73 GPA at the time of writing.

Additional Skills

High, mostly-self-taught familiarity with Unix-like systems, including GNU/Linux (server and desktop, especially Debian-based distributions) and Mac OS X. Also familiar with Microsoft Windows.

Mix of self-taught and professional experience in full-stack web development, including NodeJS, MongoDB, Firebase, AngularJS, vanilla HTML5/CSS3/JavaScript, Git, GitHub, Amazon AWS, and others.

School-taught experience with Java and C development.

SIMON PHILLIPS

Creative Designer

ABOUT ME

Hi there! My name is Simon Phillips, I'm currently a Senior at Georgia State University pursuing a Bachelor's degree in Computer Science with a focus on human-computer interaction and a minor in Psychology. I'm a total User Interface and User Experience geek, it is one of my passions to make programs as easy as possible for users. Some of my hobbies include making video games and mods using 3D modeling programs, designing art in Photoshop and Illustrator, traveling the world, and going to indie/alternative music concerts! Let's get to know each other!

CONTACT ME

112 Courtland Street NE
Atlants, GA, 30303

simonxphillips.com
simonxphillips@gmail.com

(678) 459-5032

MY SKILLS

- Adobe Photoshop
- Adobe Illustrator
- Cross-Platform Mobile-First Design
- Web Design
- Print Design
- User Interface/Experience
- Blender
- Final Cut Pro

WORK EXPERIENCE

COLLEGE MAC+ TECHNICAL SUPPORT

Apple, Inc. June 2014-December 2015

Technical support advisor for various Apple products and software, which include iPhones, iPods, iPads, Macintosh computers, iTunes for Windows and OS X, iCloud, OS X 10.6 Snow Leopard and onward, and iOS 1.0 and onward. Iso granted Tier 2 account security clearance for Apple ID-related issues.

FREELANCE GRAPHIC DESIGNER

Self-Employed August 2012-Current

Contracted to work with multiple organizations for graphic design-related projects. Organizations include All Saints Catholic Church, North Springs Charter High School, and campaign posters and graphics-related projects for the Athens chapter of N.O.R.M.L.

SENIOR GRAPHIC DESIGNER

Young Americans for Liberty August 2013-May 2015

Head graphic designer, helped raise awareness of political issues through outreach via graphics-related tasks, such as advertising and design projects meant for forwarding the Georgis State University chapter.

EDUCATION

BACHELOR OF COMPUTER SCIENCE

Georgia State University August 2013-Current

Currently pursuing a Bachelor's degree in Computer Science, specifically a concentration in human-computer interaction, user interface, and user experience.

REFERENCES

TED LEE

Manager at Apple, Inc. May 2015-December 2015

Office phone: (916) 390-1745
Email: ted.lee@apple.com

JENNY BYRD

Manager at Apple, Inc. June 2014-May 2015

Office phone: (209) 304-2217
Email: jennifer_byrd@apple.com

ADAM LINDENAU

Founder, Lindy Interactive

Phone: (404) 769-6963
Email: adamlindenau@gmail.com

TERASHIA BLAKE

DEGREE: Senior here at Georgia State University. I am thriving to graduate this upcoming December with hopes of receiving my Bachelor's Degree in Computer Science and a concentration in Networks & Parallel Distribution.

PROGRAMMING KNOWLEDGE: Since my entrance, I have taken a great deal of CSC classes teaching me many different languages such as: Java, Assembly, C, Python, SQL, & Cassandra. On my own time, I dedicated a good amount of time on 'Codacademy' learning HTML & CSS. I even made a mock website while learning. In all of my classes I have wrote several Java programs. Programs ranging from creating two arrays with random numbers, multiplying those arrays, printing each one and the result (Data Structures) to creating a mail user client where the user can send email (Computer Networks). Alongside a group of team members, we created an implementation of a Hangman game using the Python language. In the Big Data course I took, we chose a dataset, used SQL to query the data, created a data visualization using Tableau and presented the data to the class. Although I have gained knowledge and programming experience of these different languages, I do not have any work experience, i.e. an internship.

ACADEMIC INTERESTS: I always knew that programming was not one of my strong points. At one point in my collegiate career, I wanted to change my major from Computer Science, CSC to Computer Information Systems, CIS. My interest leans more towards Databases & Networks. The Big Data course I took was my first introduction to data. I never knew the complexities of data and how much work is required when dealing with it. Data is everywhere! It can be expressed, presented and broken down in several different ways. The opportunities and career possibilities involving data is also more than what I thought. As a programmer, my skills aren't as proficient as I would like for them to be but I do have a good understanding of the syntax. I can map out a program, including what needs to happen, suggested methods or classes. I can explain a program and understand why. Actually writing the code, is one of my weaknesses. The beauty and what I love about being a Computer Science major is that I am not limited to just programming. I was able to choose a concentration focusing more on my interest and I get to take two more classes outside of that concentration, to explore other options.

TAKEAWAYS: Due to the fact that I do not have work experience, I want to use this class as a foundation of learning how to work with a team to create something from start to finish. Being able to witness and experience the work it takes to develop a software/project that can potentially be used in reality. I look at this class as a final project that is applying all my skills and everything I have learned. I hope to create something that I can use on my resume. I also want to take away more knowledge about Software Engineering and Databases in hopes that I understand both aspects more and have more concrete idea of what I would like my first job to be.

Introduction to Our Project:

Our project will be a restaurant reservation service that will provide the user with the ability to place and cancel reservations at local restaurants online as well as provide helpful information such as the weather forecast for the time of the reservation and easy access to GPS directions. We are looking to cut out the need to use multiple applications to plan a trip to restaurant and condense all of the steps into one, easy to use application.

Requirements Elicitation:

Problem Statement:

Our project will help combine the function of several applications into one, easy to use application. The application will allow users to place reservations at restaurants, and then be provide with additional information pertinent to the date and time, such as notifying the user of the weather conditions. The user shall be able to log in to an account, view available restaurants, the estimated wait times at said restaurants, and then place a reservation if desired. We shall provide information such as restaurant contact information and the address with a link to directions. The user shall be sent a confirmation confirming the successfully placed reservation.

Requirement Traceability Matrix:

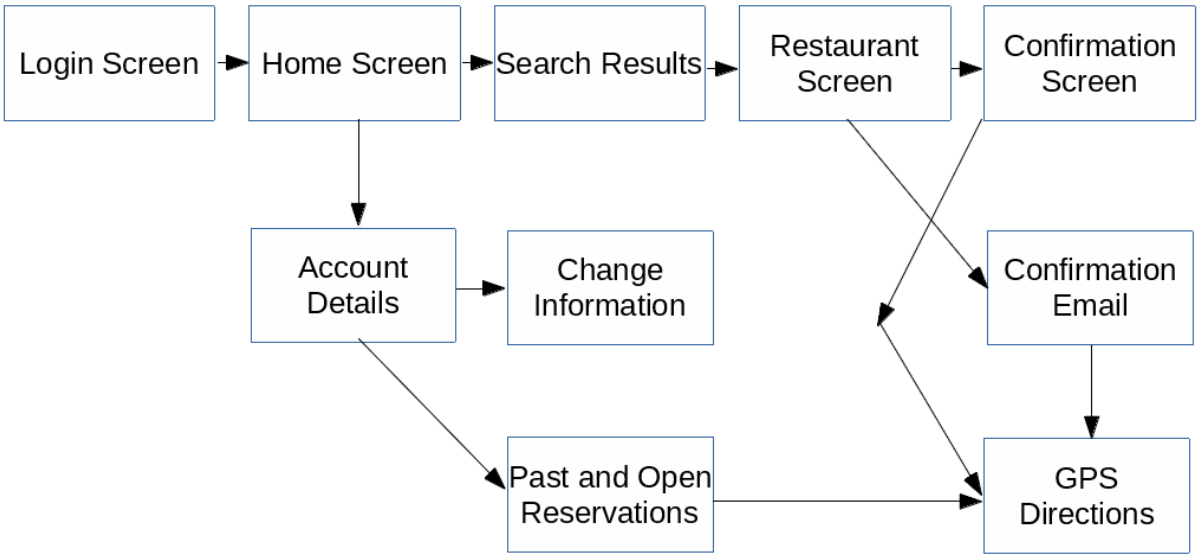
Entry Number	Parameters	Functionality	Type	
1	Verify Login	Checks server and verifies login details	Login (Authentication)	Y
2	Check Resturant Data	Calls to restaurant to see open tables	Restaurant API	Y
3	Weather Forecast	Calls to weather API to get weather conditions	Weather service API (ex. Weather Underground)	Y
4	GPS Directions	Uses Maps API to get directions	Google Maps API	Y
5	Menu	Uses resturant data to grab menu items	Yelp API (possibly)	N
6	SQL server	Server to back-up data	Server	Y
7	User	Profile for user	Login profile/account	Y

Reasoning:

The topic was chosen in lieu of the frustration that comes with waiting for countless minutes at popular restaurants. We wanted to make it easier for users to find restaurant information such as hours, reservations, wait times, directions, etc. in one place. The requirements will be implementing multiple existing applications into one easy to use web application.

System Analysis & Design:

Horizontal Prototype:



Use Cases:

Login	UC_001_Login
Actors/Participants	User
Flow of Events	1. Click login button 2. Navigate to login page (if not already there) 3. Type in username and password 4. Verify username and password a. If verified, user is authenticated and system redirects user to home screen of app b. If not, return to login screen or ask if password needs to be reset
Entry Condition	Click login button and enter authentication details

Exit Condition	Successful authentication
Quality Constraints	Browser version, supported OS (for browser)

Navigation	UC_002_Navigation
Actors/Participants	User
Flow of Events	1. User clicks UI element corresponding to page they wish to navigate to 2. System redirects them to corresponding page
Entry Condition	Wishing to navigate to a different page
Exit Condition	Successfully navigating to page
Quality Constraints	None (?)

Searching	UC_003_Searching
Actors/Participants	User
Flow of Events	1. User clicks search bar at top of page (or wherever it is) and enters search term 2. System returns results (substring match based on restaurant name or cuisine) 3. User clicks/selects restaurant 4. System navigates to restaurant page
Entry Condition	Click on search bar
Exit Condition	When search button is pressed, searched results will appear.
Quality Constraints	Realtime search results (?), relevant matches

Account Management	UC_004_Account Management
Actors/Participants	User
Flow of events	1.Update account (Basic Information: Name, Address, Billing Information, Email Address) 2.User is able to view past/upcoming reservations
Entry Condition	Able to click account details page
Exit Condition	Successfully locate exit button or end-of-session
Quality constraints	Browser support

Reservation	UC_005_Reservation
Actors/Participants	User
Flow of events	1. User navigates to restaurant page 2. System shows list of availability and weather forecast in a table 3. User selects an availability result from the table (if desired) 4. System reservation confirmation email to user's email address
Entry Condition	Navigate to restaurant page
Exit Condition	User creates reservation
Quality constraints	?

Weather Forecast	UC_006_WeatherForecast
Actors/Participants	Service
Flow of events	1. System detects address of restaurant

	2. System makes call to weather service API for weather conditions 3. Weather service responds with conditions at requested address 4. System displays weather info on restaurant page when resturant page is open
Entry Condition	Navigate to restaurant page
Exit Condition	Service displays weather conditions of the desired restaurant
Quality Constraints	Internet connection is required, API must be online, browser support

GPS Directions	UC_007_GPSDirections
Actors/Participants	System
Flow of events	1. User sees/receives link/button to directions, clicks/taps it 2. User's browser/os opens relevant information in google maps/mapquest/openstreetmap etc.
Entry Condition	API must be online
Exit Condition	System displays address with appropriate directions based off of API
Quality Constraints	Browser support for API, valid address

Account Creation	UC_008_CreateAccount
Actors/Participants	User
Flow of events	1. User is asked for desired username. a. If available, user is then asked for a password.

	b. If not avaliable, user is asked to try a new username. 2. User is asked to input information such as name, email, phone, address. 3. Information is submitted into database.
Entry Condition	Not having an account. Click create account.
Exit Condition	User information is sent to database.
Quality Constraints	Valid information with available username.

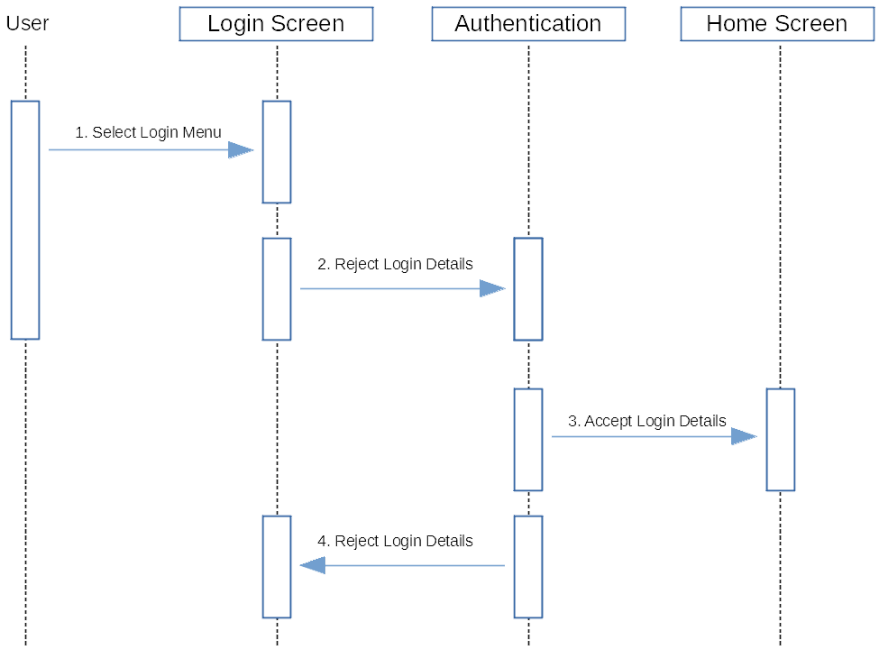
Email Confirmation	UC_009_EmailConfirmation
Actors/Participants	User
Flow of events	1. User receives confirmation email, opens and clicks confirmation link 2. System sends reservation information to restaurant for processing 3. User is shown confirmation page and sent email with directions
Entry Condition	User receives and opens confirmation email
Exit Condition	User is shown confirmation page and sent post-confirmation email
Quality Constraints	Valid email address, online servers for both email service and site

Reservation Cancellation	UC_010_ReservationCancellation
Actors/Participants	User
Flow of events	1. User goes to Account management

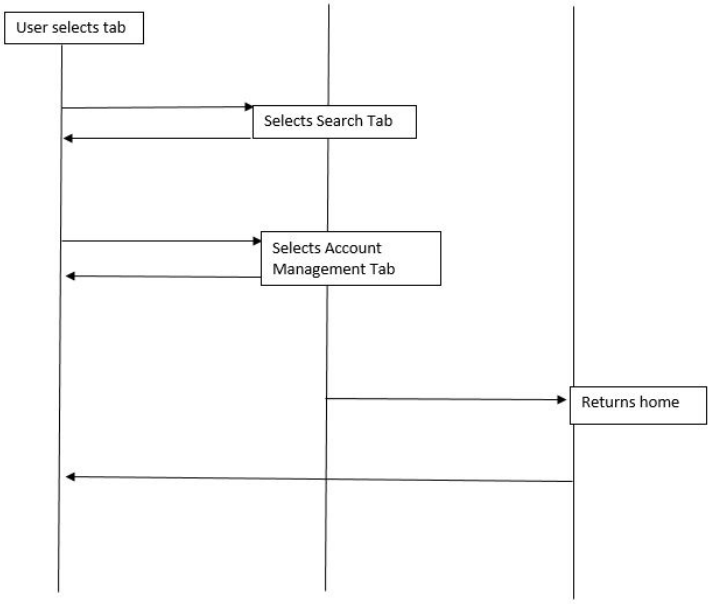
	page, clicks active reservations. 2. System would display current reservations 3. User would select which reservation they would like to cancel 4. System removes reservation, and notifies restaurant
Entry Condition	User navigates to account management page
Exit Condition	Reservation has been successfully canceled, and email confirming cancellation has been sent
Quality Constraints	User has an active reservation

Interaction Diagrams:

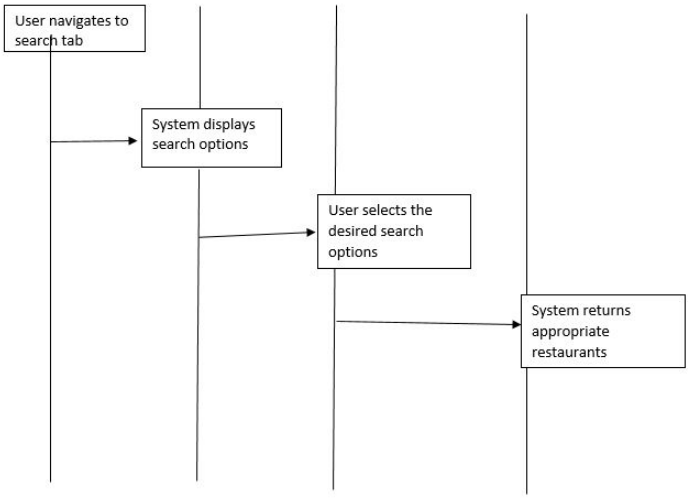
UC_001_Login:



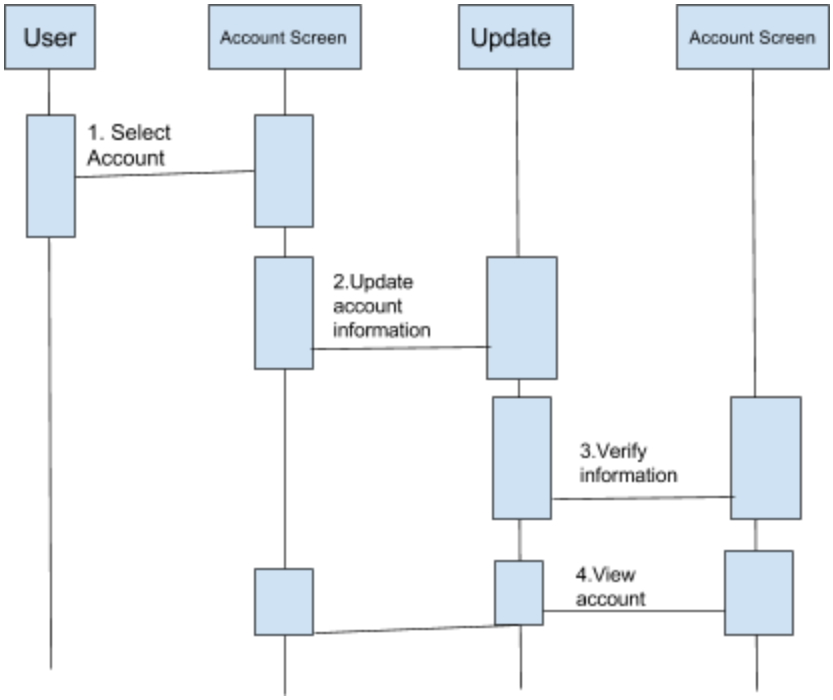
UC_002_Navigation:



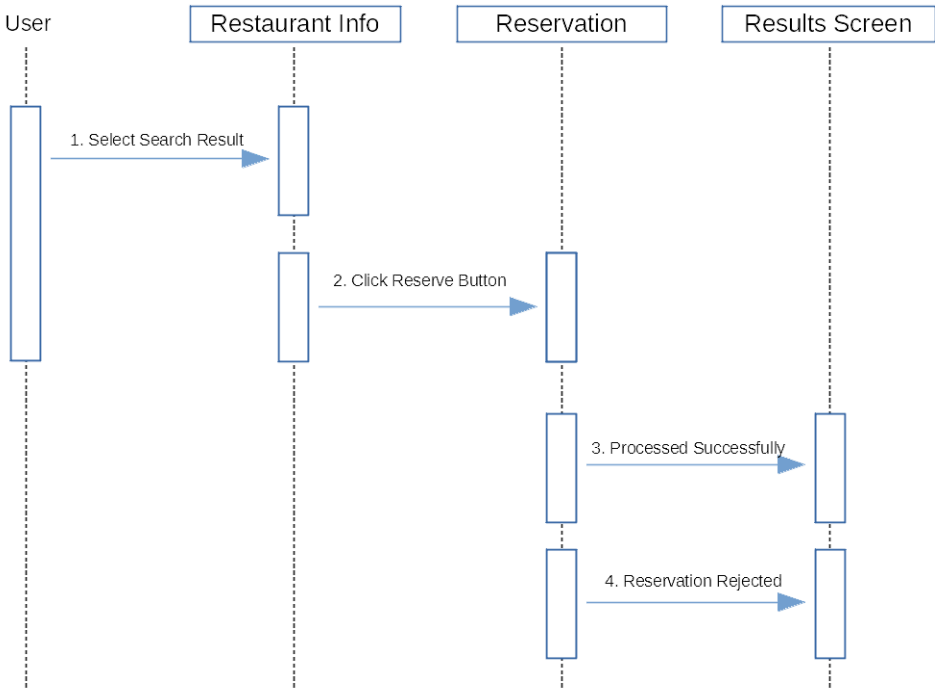
UC_003_Searching:



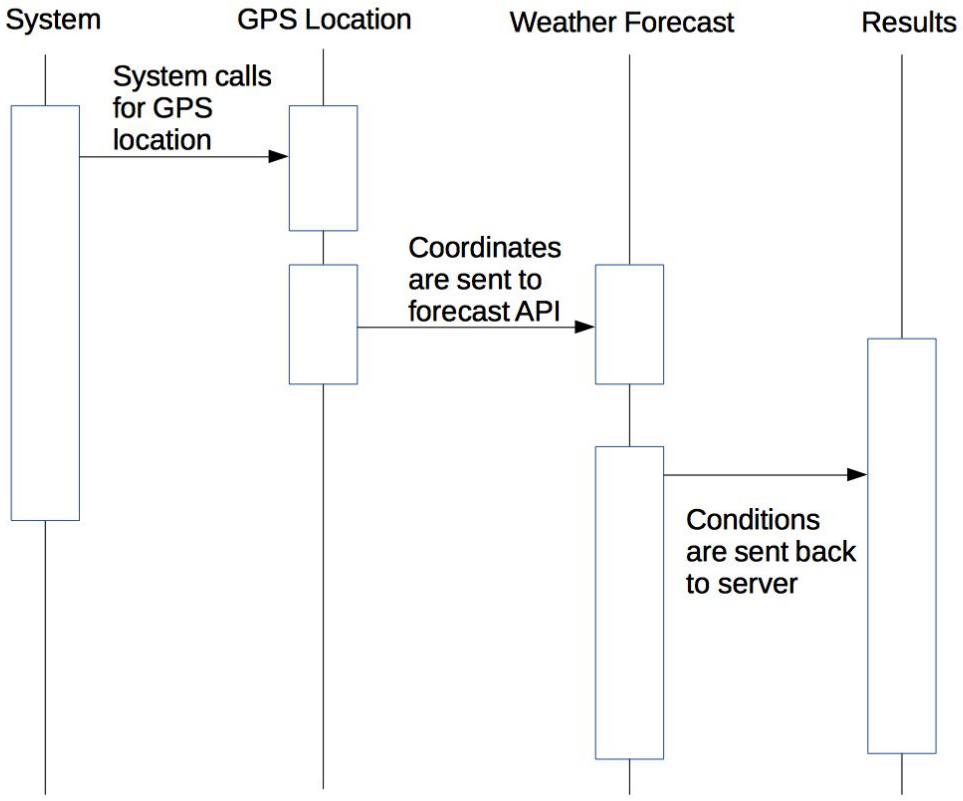
UC_004_Account Management:



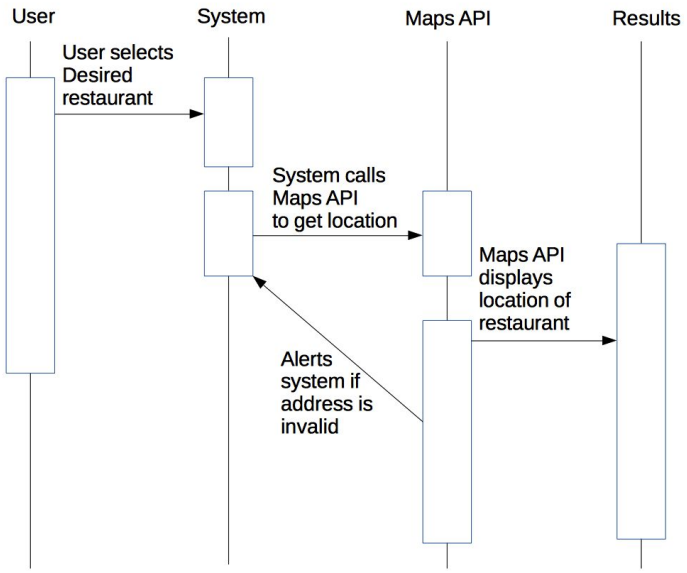
UC_005_Reservation:



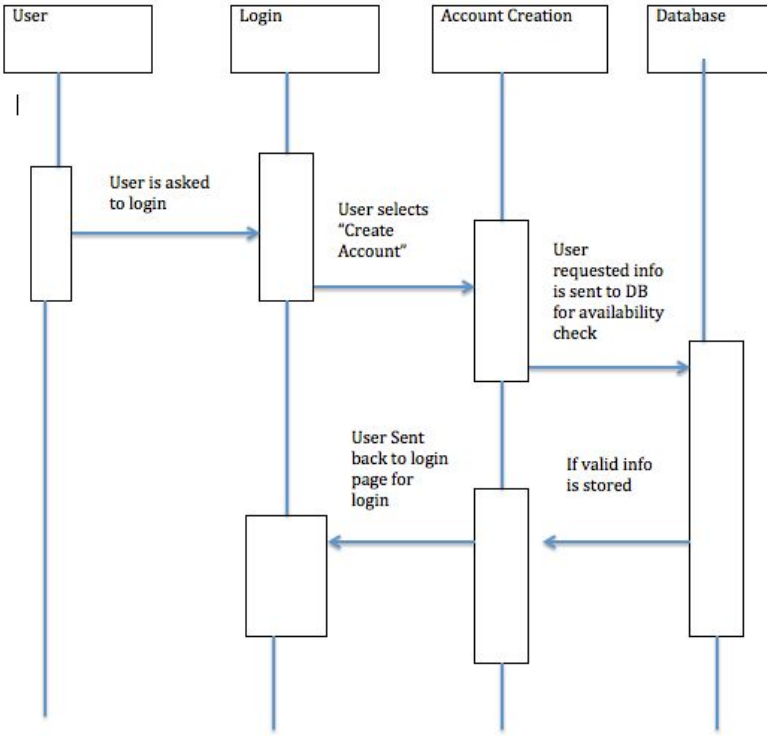
UC_006_WeatherForecast:


System
GPS Location
Weather Forecast
Results
System calls for GPS location
Coordinates are sent to forecast API
Conditions are sent back to server

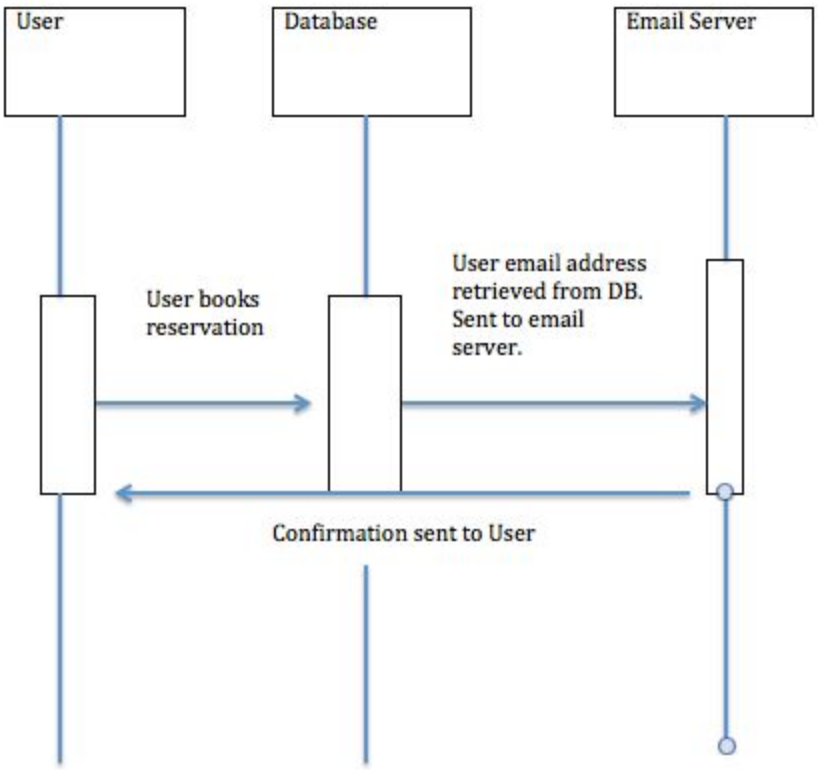
UC_007_GPSDirections:



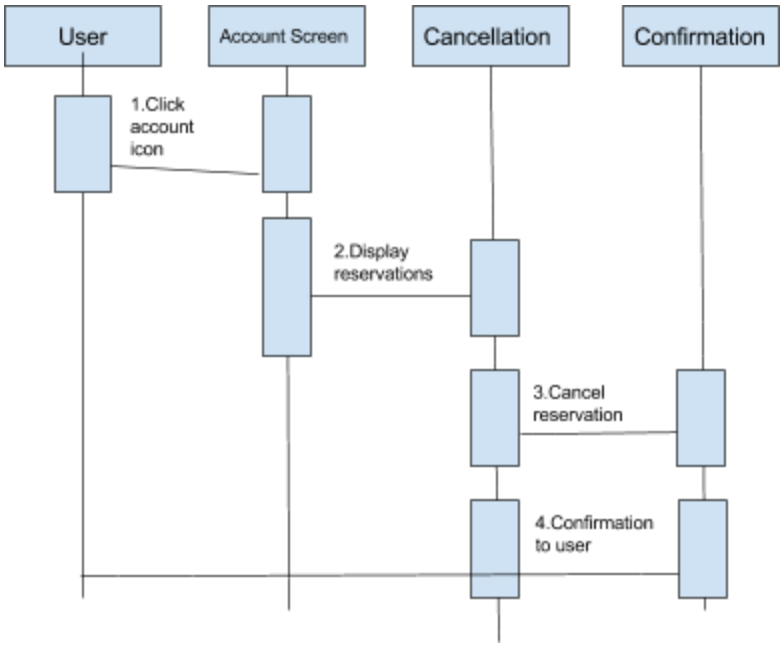
UC_008_CreateAccount:



UC_009_EmailConfirmation:



UC_010_ReservationCancellation:



Object Design:

Objects

Name	Authentication
Type	Control
Function	Verifies login information (username/password), communicates with database to retrieve user information
Attributes	● Database login information
Methods	● Authenticate user (returns user object)

Name	User
Type	Entity
Function	Holds deserialized user information and communicates with the database to update information
Attributes	● Name ● City ● Cuisine preferences (?)
Methods	● Get user information ● Set user information

Name	Restaurant
Type	Entity
Function	Holds information about a restaurant and provides availability information
Attributes	● Name ● Location ● Cuisine ● Seating (indoor and/or outdoor)

Methods	● Get availability for a time range

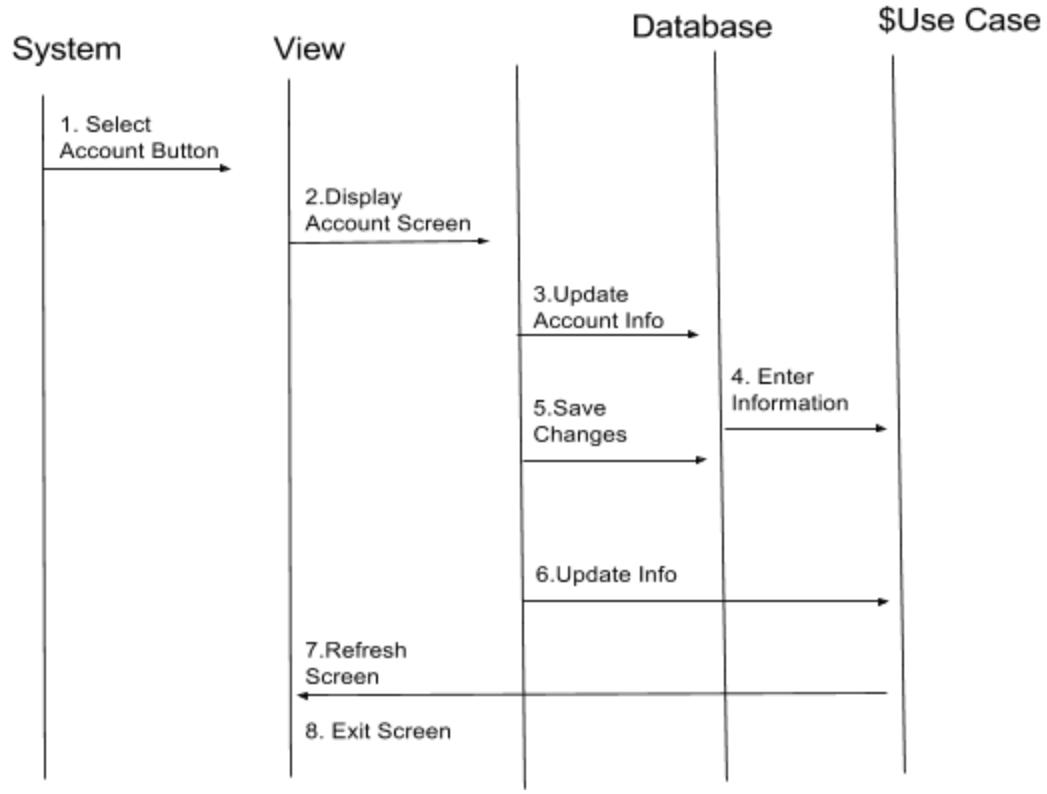
Name	Weather
Type	Entity
Function	Represents “the weather” as a general concept, communicates with third-party weather API(s) to retrieve forecast for a location and time range
Attributes	● Weather API information
Methods	● Retrieve forecast for location and time range

Name	User Interface
Type	Boundary
Function	Allows user to navigate and retrieve information, submit a reservation, and set user account information
Attributes	None
Methods	● Login and Logout ● Search ● Browse

Category Interaction Diagram:

UC_4_AccountManagement_CID

Steps	**Categories**
1. User selects “Account” icon/button	System
2. Account screen displays	GUI (View)
3. User changes account information	
4. User selects save changes	
5. Information is updated to database	Database
6. Account screen is reloaded with updated information	
7. Exit Account Screen	$Use_Case



Test cases:

Tests will be written in Javascript using the Mocha testing framework, and will be fully automated. All tests will be located in the test/test.js file (relative to the project root) and, if the mocha-cli utility is installed from npm and all of the dependencies were installed using 'npm install', running 'mocha' from the project root directory should run all the tests. User intervention is not necessary and the framework will handle logging which tests passed or failed, and how they failed. For these reasons, the "location" and "log" rows have been omitted for brevity. The data in the "input" rows will be either embedded in the file or created at runtime, so once again, user input or intervention is not necessary. Tests will also be executed in the order in which they are listed below.

In the below tables, an *error* describes a response given by the application, which sometimes happens in expected/desired behavior of the application, if for example the user inputs unexpected or invalid data. This can be expressed by either an instance of the Javascript Error class, or a non-200-or-300 HTTP status code. A *fault* describes a difference between the expected/desired behavior and its actual behavior in the test suite. If the "oracle" row just says "Success" or "Error", this means that this is a void method that does not return data, and is only expected to either return/throw an Error or not depending on its input.

Attributes	**Description**
name	Register (invalid characters)
input	Randomly-generated username and password which contains at least one invalid character
oracle	Error

Attributes	**Description**
name	Register (valid)
input	Randomly-generated username and password which are valid
oracle	Success

Attributes	**Description**
name	Register (taken)
input	Username from the “Register (valid)” test and any username
oracle	Error (if executed in the correct order, this should return an error, since the desired username had been taken)

Attributes	**Description**
name	Change account information
input	Randomly generated valid data for user display name, location, other relevant fields
oracle	Success

Attributes	**Description**
name	Query restaurants (by search string)
input	Run multiple times with a mixture of randomly-generated strings, an empty string, as well as some predefined strings that expect to return at least one result
oracle	No error thrown, should be a list, and at least one of the lists should have at least one result

Attributes	**Description**
name	Query restaurants (by category)
input	Run multiple times with a mixture of randomly-generated strings, an empty string, as well as some predefined strings that expect to return at least one result
oracle	No error thrown, should be a list, and at

	least one of the lists should have at least one result

Attributes	**Description**
name	Query restaurant openings (valid)
input	Run multiple times with different restaurants obtained from the previous step
oracle	No error thrown, should be a list/matrix/table of openings with corresponding weather forecast data. Although it is unlikely, *it should not be a fault if any or all lists are zero-length*, because this is technically possible in a real-world application.

Attributes	**Description**
name	Query restaurant openings (invalid)
input	Run multiple times with randomly-generated restaurant ids for restaurants that are not in the database
oracle	Should return an error. If the module returns a zero-length array instead of an error, this is a fault.

Attributes	**Description**
name	Restaurant reservation (valid)
input	A valid restaurant id and opening time
oracle	Success

Attributes	Description
name	Restaurant reservation (invalid)
input	Run once with an invalid restaurant id and any opening time, and once with a valid restaurant id, but an invalid opening time
oracle	Error

Rationales:

Object Rationale:

Authentication: We need a central authentication control object to handle authentication and communication with the database.

User: We need a User class to instantiate several user objects to contain and handle the updating of user information.

Weather: We should have one central object to provide weather information, as well as handle communication with the chosen weather API. It should provide one method which will return the current weather forecast for a location and time range.

Restaurant: We should have a restaurant class to hold information about restaurants, retrieve information about them and facilitate submitting reservations.

User Interface: We should have a user interface boundary object to simplify the structure of the user interface.

Use Case Rationale:

UC_001_Login: Our service must have the ability to log in users in order to provide the user with proper information and tie reservations to specific users.

UC_002_Navigation: The web application will rely heavily on the user being able to easily navigate throughout the different pages or tabs to access the information they desire (ie. specific restaurants or their account profile).

UC_003_Searching: Searching is a necessity in order to provide the user with an accurate list of available restaurants according to their entered specifications. Instead of making the user search through every restaurant by hand, they will be able to enter specific cuisines to narrow the options.

UC_004_Account Management: The user will be provided the ability to update their account's information, such as providing a new email address or phone number. From here, the user is also able to view current and past reservations.

UC_005_Reservation: This use case is arguably the most vital, as it provides the entire concept of our product. It will allow for the user to place a reservation at one of the restaurants they found via the search.

UC_006_WeatherForecast: The additional features that our service provides, such as showing the weather forecast for the time of the reservation, are what separates it from the existing reservation services. This use case is different from the previous ones in the way that the user is not the actor, the use case will be called by the system after the reservation has been set.

UC_007_GPSDirections: This use case is much like the Weather Forecast, the system will automatically call the GPS Directions use case to provide the user with directions after they place a reservation.

UC_008_CreateAccount: A user, in the case that it is their first time using our service, must have the ability to create an account. This will consist of the user entering their desired user name, password, and other basic account information such as their email address.

UC_009_EmailConfirmation: In order to ensure the reservation was successfully placed. If there was an error placing the reservation or an issue with the restaurant, the user will be notified via email.

UC_010_ReservationCancellation: If for some reason the user decides they would like to cancel a reservation they previously placed, they will be able to navigate to their account management page. They will be able to view any current reservations they have placed and the ability to cancel it.

Topic Chosen Rationale:

The topic was chosen in lieu of the frustration that comes with waiting for countless minutes at popular restaurants. We wanted to make it easier for users to find restaurant information such as hours, reservations, wait times, directions, etc. in one place. The requirements will be implementing multiple existing applications into one easy to use web application.

Software Architecture Used Rationale:

Variation on MEAN (MongoDB, Express.js, Angular, Node.js) software stack using MySQL in place of MongoDB.

We will be using a client-server architecture. The server will be written in JavaScript and run on a Node instance, and communicate with a MySQL database instance. The server, when accessed with a web browser, will return an HTML page with embedded CSS and JavaScript which will act as the client.

Within the client we will use a model-view-controller architecture, with the assistance of the Angular framework.

Function Point Cost Analysis & COCOMO:

Weighting Factor Estimation						
Measurement Parameter	**Count**		**Simple**	**Average**	**Complex**	**Total**
Number of user inputs	3	**X**	1	2	3	15
Number of user outputs	3	**X**	2	4	6	36
Number of user inquires	4	**X**	4	5	7	64
Number of files	0	**X**	0	0	0	
Number of external interfaces	3	**X**	3	5	7	45
	Count				=	160

Prototype:

Project Legacy:

Our goal with this project was to create a web application that simplified the reservation process. In addition to simply being able to place reservations, we wanted to provide the user with additional information, such as GPS directions and weather forecast. After working on this project for several months, we have been able to achieve most of our goals. We were able to implement the majority of the features and requirements that were found during the requirement elicitation stage. Overall, our project was a success and accomplished what we set out for it to do. All team members fulfilled their designated roles successfully.

Work Share Document:

Group 4

Members

Noah Aragon- Team leader/Group Coordinator

Gerald "Rett" Gerst- Programmer and Database

Simon Phillips- Database Programmer

Nick Clark- Technical Writer

Terashia Blake- Technical Writer

Gantt Chart:

Task	Task Description	Duration	Start Date	End Date
Task 1	Problem Identification. Figure out what we will be working on.	5 days	8/22/16	8/27/16
Task 2	Role selection. Group members identify their skills and choose roles accordingly.	1 day	8/30/16	8/30/16
Task 3	Organize team. Decide how our team will be communicating , when we will meet, and how we will accomplish all assignments and meet deadlines.	3 Days	9/1/16	9/3/16
Task 4	Project Presentation. Formalize our project and create slideshow.	3 Days	9/3/16	9/6/16
Task 5	Requirement Elicitation. Put together a formal problem statement, work share document, Gantt chart, and technical term dictionary.	8 Days	9/10/16	9/18/16

Task 6	Use Cases. Identify and describe the use cases for our project. Update Gantt chart, RTM, and glossary of terms.	8 Days	9/20/2016	9/28/2016
Task 7	Presentation #2. Create Prototype and presentation to show proof of concept and explain in more detail how our product will work.	4 Days	10/2/2016	10/6/2016
Task 8	Object design. Isolate the objects our product will use and design their attributes and how they will function.	7 Days	10/9/2016	10/16/2016
Task 9	Project Rationale and CID. Combine the rationals for all aspects, and create the Category Interaction Diagrams for our product.	7 Days	10/16/2016	10/23/2016
Task 10	Test cases and coding. Began coding certain aspects of our project and preparing and executing test cases to ensure	13 Days	10/24/2016	11/6/2016

	proper functionality.			
Task 11	Finish coding and assemble the final report. The programmers put the finishing touches onto the project and the technical writers and project manager put together the final report.	14 Days	11/13/2016	11/17/2016

Dictionary:

Angular.js

Angular.js (or "Angular") is a model-view-controller (MVC) framework for the web platform, designed to assist in creating front-end user interfaces for web applications. Angular as well as applications which leverage it are written in javascript.

API

API, which stands for Application Program Interface, is the interaction between applications and software components. The API specifies the systems and libraries the application will be interacting with. APIs are what make it possible to move information between programs.

Express.js

Popular software library for Node.js assisting with the development of web applications.

Node.js

Node.js (often referred to as just "node") is a runtime environment that allows javascript to be used outside of the browser. It is most popularly used with the Express framework to create web servers. The Node runtime environment is distributed as a binary and Node applications are distributed as javascript source files. Node uses the same V8 javascript JIT compiler used in Google Chrome.

MongoDB

Popular *non-relational* database management system which is very popular among the Node.js ecosystem, but does not fit our requirements for a relational database system.

MySQL

MySQL is a Relational Database Management System (RDBMS) created by Oracle Corporation.

User Guide:

Returning users will firstly log into the application with their existing credentials; first time users will have the ability to create a new account and provide basic account information to the application. Once logged in successfully, the user will be brought to the home screen that will have the ability to navigate through all of the application's functionalities. The main function will be to search for a desired restaurant by entering a keyword, such as type of cuisine, into the search bar. A list of restaurants meeting the user's criteria will be returned and the user is free to parse through it. Once the user has decided on a restaurant of their liking, they will be able to place a reservation at a variety of timeslots. Once a reservation is placed, the user will receive a confirmation email and the application will display the weather forecast for the relevant time period and access to GPS directions. Once returned to the home screen, the user can either search for more restaurants for additional reservations, edit account information by visiting account management, or log out and put an end to their session. The application is very user-friendly and there will be prompts to help the user successfully use our product.