ScheduleSwift

CS30700

Design Document

Team 13
James Corder, Jenny Ha, Dominic Nale, Micky Santiago-Zayas

Index

Purpose

Making a reservation is often a convoluted process involving various methods of contact depending on the location you are making a reservation for, and often results in the frustration of the person making the reservation because they cannot find an easy way to make the reservation.

The purpose of this project is to develop a reservation application. Existing reservation forms include a website, phone calls, walk-in reservations, and a whole lot of other ways to make a reservation. This project still aims to make reservations, but it seeks to streamline the process into a "one stop shop" for making reservations at all different kinds of businesses through a centralized interface with a unified process for making reservations. On top of just making reservations, our project aims to allow customization beyond just time and date, including things like number of tables at a restaurant, or number of lanes at a bowling alley, as well as the ability to view account statistical information such as number of active reservations, soonest reservation, etc.

Functional Requirements

1. Business Manager,
   - i. I would like to change what employees are able to access and change
   - ii. I would like to create and change event details
   - iii. I would like to change and provide business hours
   - iv. I would like to have control over all features available to business employees
   - v. I would like to customize the items to be reserved (e.g., tables, rooms, etc.)
   - vi. I would like to customize or alter my business' reservation request forms
   - vii. I would like to be able to add a security deposit requirement
   - viii. I would like to change the prices that appear for customers
   - ix. I would like to change what details concerning my business (FAQs, contact information, etc.) appear to customers
   - x. I would like to set up a loyalty rewards system for customers who come often
   - xi. I would like to create employee accounts for my employees to access

2. Business Employee,
   - i. I would like to be able to see all the events at my business
   - ii. I would like to see all the private and public events in the business calendar
   - iii. I would like to access the confirmation number of any timed reservation
   - iv. I would like to cancel an active reservation at my business
   - v. I would like to be able to change active reservations at my business

vi. I would like to be sent a notification when a reservation at my business has been made
vii. I would like to be sent a notification when a reservation at my business has been cancelled
viii. I would like to be sent a reminder of active reservations occurring during my shift at the beginning of my shift
ix. I would like to see contact information for customers who have a reservation at my business
x. I would like to change what is available at my business for customers to choose from (e.g., space availability, time availability, items available, etc.)
xi. I would like to access the information regarding a specific reservation
xii. I would like to forward events for manager review
xiii. I would like to have a work account to access notifications regarding shifts and tasks.

3. Customer,

i. I would like to view pricing options for non-reservable services, reservations, and public events
ii. I would like to see FAQs of a business
iii. I would like to be able to contact the business if I have a question or concern not answered in the FAQ
iv. I would like to be able to register for a ScheduleSwift account
v. I would like to be able to login to my account with my username and password
vi. I would like to be able to manage my account information
vii. I would like to recover my username and password should I forget them
viii. I would like to connect my email to receive notifications or confirm my account through it
ix. I would like to see any events I am able to make a reservation for
x. I would like to see the details and policies of the events I am able to make a reservation for
xi. I would like to fill out an information form regarding my requests for a reservation I am planning
xii. I would like to see my active reservations
xiii. I would like to be able to modify my reservations by changing the date, time, or specific reservation information
xiv. I would like to cancel my active reservations
xv. I would like to be sent a reminder 24 hours before my reservation
xvi. I would like to be sent a notification when my reservation has been cancelled
xvii. I would like to be sent a confirmation receipt when I make a reservation
xviii. I would like to be sent confirmation when I modify a reservation

xix. I would like to be sent a notification when a business modifies my reservation
xx. I would like to access my confirmation number as proof of reservation
xxi. I would like to see an itemized receipt and my total before I make a purchase
xxii. I would like to have a payment method available to pay
xxiii. I would like to access all information regarding my reservation (e.g., payment status, reservation details, etc.)
xxiv. I would like to view statistics about my account such as number of reservations, last login, next upcoming reservation, etc.

4. If time allows,

i. As a business manager I would like to have a work clock for employees to clock in and out with
ii. As a business manager I would like to select what type of business I own (recreational, restaurant, etc.)
iii. As a business manager, if I own a recreational business, I would like to add default features such as a leaderboard or scoreboard for my customers
iv. As a business manager, I would like to send notifications about sales and discount events to users
v. As a customer, I would like to add my reservations to my personal calendar, such as Outlook or Google Calendar
vi. As an employee, I would like to choose my shifts and block unavailable times
vii. As a business manager, have a shifts calendar generator according to employee availability

Non-Functional Requirements

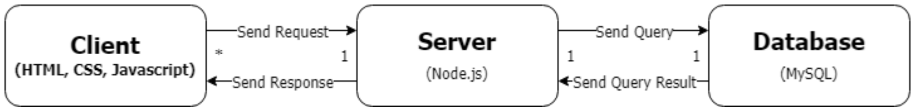
1. Performance, Availability, and Scalability
    1. We would like the application to run smoothly
    2. We would like the application to avoid all crashes
    3. We would like the application to launch in 10 seconds or less
    4. We would like the application to be able to support at least 10,000 concurrent users
    5. We would like the application to be available 24/7
    6. We would like to be able to increase the concurrent user count by adding more server capacity in the future
2. Server
    1. We would like the server to process server-client communication requests in real time without noticeable delays
    2. We would like the server to be able to store important data in a database
    3. We would like the server to be able to retrieve important data from a database
3. Appearance
    1. If time allows, we would like to create a smooth, seamless, interactable user interface
    2. If time allows, we would like to add animations and more sophisticated appearance effects to attributes on our web application
    3. If time allows, we would like to add the ability for our web application to scale mobile devices accordingly
4. Security
    1. We would like to store passwords in an encrypted format and decrypt them as needed to lessen the chances of passwords being stolen
    2. We would like to prevent access to a user's personal information via a key that only business employees or managers would have access to
    3. We would like give users the option to save their payment method, and encrypt payment method information should they choose to save it
    4. If time allows, we would like to implement account restrictions should a user attempt to misuse or abuse the services made available to them
    5. If time allows, we would like to limit the number of accounts a user can create to 1 account to avoid users getting around account restrictions
5. Usability
    1. We would like the application to be straightforward enough that a tutorial or explanation of features is not required
    2. We would like the application to run on all browsers

iii. If time allows, we would like to allow the scaling of components to fit a mobile device, as mentioned above
iv. If time allows, we would like to add customizability to the layout to assist people with disabilities

Design Outline

High Level Overview

This project will be a web application that allows customers to view and make reservations for events that are created and published by business employees and managers. This application will follow the traditional client-server model in which one server will handle all the requests from any number of clients using Node.js. The server will be used to accept client requests, access, or store information in a database, and send responses back to the client(s) as need be.

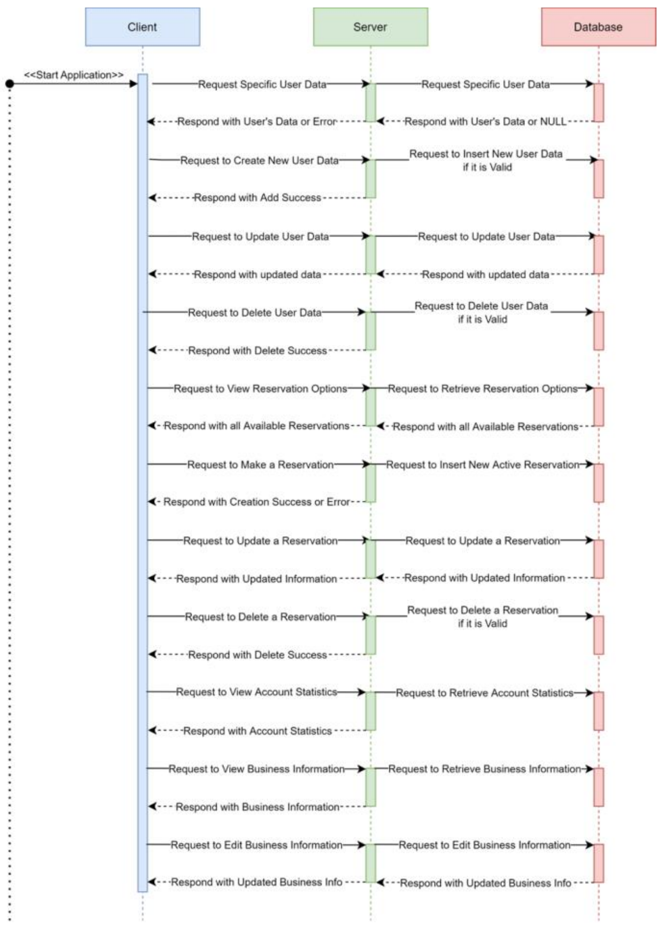


1. Client
    a. The client will provide the user interface for our web application according to account type (i.e., customer, business employee, business manager)
    b. The client will send React requests to the server
    c. The client will receive a response from the server and will render necessary changes according to the results
2. Server
    a. Upon connection the server will provide the client with information (e.g., account information, reservation information, etc.)
    b. The server will receive the React requests from the client
    c. The server will validate the request and formulate the appropriate response to the request
    d. The server will send requests to create, read, update, or delete entries from the database as needed
    e. The server will send the formulated responses back to the client
3. Database
    a. The database will store all the data for our application, such as user information, reservation information, etc.
    b. The database will receive queries from the server and perform the queries on the datasets within the database
    c. The database will send the result of the queries back to the server for processing
    d. The database will respond with an error if it is unable to perform the requested query on the given dataset
    e. The database to allow reservation requests to go as far as needed per business. However, per business there is a constant MaxNumReservation. To compensate for storage, we will implement a garbage collector to remove past reservations/events from the database and make more space.

i. For user convenience, there would be an “open for reservations” status each business will have. They can also set to be notified whenever it is available for new reservations.

Sequence of Events Overview

The sequence diagram below illustrates the typical set of interactions between the client, the server, and the database. The sequence will always be initiated by a user starting our web application and it will run indefinitely until the user closes the web application. Upon login, the client will send a request to the server. Once the server receives the request, it will query the database to validate the login attempt. Once the login has been validated, the client will then be able to send the server other actions, such as viewing available reservations, making a reservation, viewing account details, etc. In response to these requests, the server will formulate an appropriate response via querying the database to retrieve the necessary information to generate the response. The database will respond with the data generated by the queries sent to it via the server. The database will also respond to any of the standard CRUD operations, those being create, read, update, and delete.


Client
Server
Database
<<Start Application>>
Request Specific User Data
Request Specific User Data
Respond with User's Data or Error
Respond with User's Data or NULL
Request to Create New User Data
Request to Insert New User Data if it is Valid
Respond with Add Success
Request to Update User Data
Request to Update User Data
Respond with updated data
Respond with updated data
Request to Delete User Data
Request to Delete User Data if it is Valid
Respond with Delete Success
Request to View Reservation Options
Request to Retrieve Reservation Options
Respond with all Available Reservations
Respond with all Available Reservations
Request to Make a Reservation
Request to Insert New Active Reservation
Respond with Creation Success or Error
Request to Update a Reservation
Request to Update a Reservation
Respond with Updated Information
Respond with Updated Information
Request to Delete a Reservation
Request to Delete a Reservation if it is Valid
Respond with Delete Success
Request to View Account Statistics
Request to Retrieve Account Statistics
Respond with Account Statistics
Request to View Business Information
Request to Retrieve Business Information
Respond with Business Information
Request to Edit Business Information
Request to Edit Business Information
Respond with Updated Business Info
Respond with Updated Business Info

Design Issues

Functional Issues

1. What information should be used to create an account?
    a. Username and Password
    b. Username, Password, and Email Address
    c. Username, Password, Email Address, and Phone Number

Choice: Option C

Justification: A username and a password are the bare minimum requirements for setting up an account, and they will be all that is needed for a user to login to an already created account. By including Email Address and Phone Number, it provides us with a lot of options from the development standpoint. First, having those fields allows us to display that information to business workers if they need to contact a customer regarding a particular reservation. Second, having that information allows us to configure notifications, reminders, or updates to reach the customers through those mediums, rather than just through the web application. Lastly, should a user forget their username or password, we have the option to use those contact methods to recover that information for them.

2. How should we confirm a user owns the account when they request to change their username or password?
    a. SMS Code
    b. Email Code
    c. Security Questions
    d. All of the above

Choice: Option D

Justification: Account confirmation is very important because we do not want to give someone access to someone else's account, so we must have at least one of these methods available to customers. We chose to have all three because we chose to make users provide an email and phone when signing up. This means that we can internally confirm the phone and email belong to the user who created the account and allow them to pick which method they are most comfortable with. The Security Question acts as the simplest form of verification, as it does not require accessing anything outside the web application to confirm ownership of the account. Including all 3 is primarily for usability, providing the users with a choice of which alternative they feel most comfortable with; however, it also allows users multiple way to recover their password should they forget the answer to their security question, forget the password to an email, or change phone numbers.

3. What kinds of specific information do we allow to be a part of the reservation forms?
    a. Predetermined Templates of Information
    b. Fully Customizable by Business Employees or Managers
    c. Partially Predetermined with the Addition of Customizable Fields

Commented [SZMD1]: Conflict with design idea

Commented [CG2R1]: Changing to Option B, include justification

Choice: Option C

Justification: Allowing only predetermined templates of information completely defeats one of the goals of our product, which was to allow more customization within reservations, so that is completely out of the question. Fully customizable poses the problem of having a unique entity for every single reservation that is created, which makes storing the data for these reservations harder, as we would not have a unified number or type of fields to put into a database. This leaves us with option C. Under this option, we have a pseudo-unified template which makes storing the data in the database much easier, but the addition of a few customizable fields allows business workers to customize their reservations in a way that is convenient for them, without inconveniencing the data storage and retrieval.

4. What kind of business information do we allow customers to see?
    a. All Business Information
    b. Business Information Relevant to a Specific Reservation
    c. Business Information Chosen by Business Managers

Choice: Option C

Justification: Allowing the user to see all business information without the agreement of the business could lead to some privacy issues that we do not want to have to handle with our web application, so we must choose a more restrictive approach. Allowing users to see information relevant to a specific reservation is nice, but the user might wish to see information that is not relevant to that reservation and is more of a universal piece of information. By allowing the business managers to choose what information is available, we avoid privacy issues because the managers agree to make that information public, and we also avoid having unique instances of available information depending on which reservation you are looking at. We are aiming to achieve a very user-friendly design and digging through multiple reservations to find a specific piece of information that is not present in the other reservations goes against that policy, this is why option C is the best for us.

5. What kinds of permissions do we allow business employees to have in the system?
    a. Unrestricted Permission to Create, Update, and Delete Reservations
    b. Global Permissions Assigned and Modifiable by Business Managers
    c. Permissions Based on Individual Reservations

Choice: Option B

Justification: Allowing unrestricted permissions for any standard employee opens up a variety of risks we are not willing to take with our web application, which means we need some sort of restriction on the permissions given to employees. Permissions based on individual reservations becomes annoying for many reasons. Having unique instances of permissions means we must store the permissions related to each reservation, which means we use more storage on less reservations. Furthermore, this means that training employees will be difficult because they do not have a standard set of permissions to get accompanied with. This leads us to option B. Option B allows the managers to set up permissions for their employees, which solves the 3 problems the other options pose. First, they do not have unrestricted permission, as it is controlled by the manager. Second, it doesn't require vastly more storage as we have universal permissions across all reservations. Lastly, training and reusability is more consistent because we have a global permission set.

6. How should we allow customers to contact businesses if they wish to do so?
    a. Fill out a Request Form
    b. Provide Phone Number
    c. Provide Email Address
    d. Provide Phone Number and Email Address

Choice: Option D

Justification: Filling out a request form is not a bad option, but we have chosen to discard it for several reasons. First, a form of that manner must be handled by an employee, which means users may see a long delay between submission and a response, which we would like to avoid. Second, the information present in the form can be hard for employees to decipher, especially if there are large text boxes where customers can write almost anything. Providing either a phone number or an email address would be okay, although it limits the customers to only one stream of contact. Therefore, we have chosen option D. It provides the users with two forms of contact, with a different purpose behind each one. Phone numbers give users the ability to chat with an employee in real time and ask whatever questions they have. Email provides users the option to get in contact with the business if they are okay with waiting for a response or if they do not have time to sit on a phone call with an employee. Lastly, using these options over a request form means we can globalize it. With request forms, the fields would be different for all different businesses, and although each business will have its own unique phone and email, it is more organized to store and display the same two types of fields for each business.

Non-Functional Issues

1. What backend language should we use?
    a. Django
    b. Spring
    c. Spring Boot
    d. Node.js
    e. PHP

Choice: Option D

Justification: We wanted to choose a language that has good compatibility with the backend and the database we are choosing, as well as a language that we all have experience with or is fairly straightforward to learn. Spring and Springboot offer a lot of power, but they do not work as seamlessly with the other pieces of our web application. Furthermore, creating a maven or a Gradle project is a very involved process that we do not have a lot of experience with. Django is also a good alternative, although it is based in python, and we wanted to refrain from building our web application from python as we want to use JavaScript primarily. We chose not to use PHP because some of us have worked with it in the past and have found it not very beginner friendly to work with, and we know there are better options with more learning materials out there. This leads us to Node.js. We chose Node for a variety of reasons.

First, it is JavaScript, which means it will go nicely with our frontend. Second, Node has a library specifically for communicating with MySQL, which is the database we want to use, which makes communication very easy. Furthermore, Node is incredibly popular all across the internet, so there is already a lot of material from us to learn from or use to implement the features that we want in our own code. Lastly, we find Node to be more intuitive, especially in the way it connects to the frontend and communicates with the database.

2. What backend extensions/frameworks should we use?
    a. Express
    b. CORS
    c. Nodemon
    d. MySQL 2
    e. All of the Above

Choice: Option E

Justification: All of these extensions offer a lot of features that are very useful for backend development. The most obvious of them all is MySQL 2, which allows us to use Node to communicate with the database language we have chosen, which is MySQL. Express is an all-around useful framework for Node as it provides broad features for building web or mobile applications, most commonly it can be used to build single or multipage web applications, which is exactly what we want. It is also used to help manage servers, which makes it useful since it will only be used server side. CORS is more technical and will require us to delve more deeply into it to get the most use out of it, but in the general sense it will allow us to relax some of the security applied to a specific API, most importantly it allows us to specify which origin points are allowed to access a specific API. Nodemon is the simplest of all of these, it just makes it so that the node application will automatically restart when changes are made to a file in the present working directory of our project. This is useful for live viewing of changes we make in code and is mostly a quality-of-life feature for development. As you can see, all of these are very useful to have in our development, so we have chosen to include them all.

3. What frontend language should we use?
    a. React
    b. Default HTML and JavaScript
    c. Angular
    d. Vue

Choice: Option A

Justification: We have chosen option A in part because of our choices for backend and database. React has a lot of resources which can be used specifically with Node, which we have chosen for our backend. Furthermore, react offers a lot of frameworks which include premade features like buttons and navigation bars and other similar features, which will allow us to improve the appearance of our application easier and more effectively. Furthermore, similar to Node, there are a lot of materials online for us to learn from and incorporate into our project and React is very popular. We also chose React in part because no one in our group is very familiar with it, but we all have an interest in learning more about it, so this project will give us the opportunity to do so. The last big reason we chose React is because React actually uses virtual DOM (Document Object Model) which means that we will have the minimum update time for the real DOM, which will give us more performance and a more polished user experience, which is important to us.

4. What frontend extensions/frameworks should we use?
    a. React Testing Library
    b. Styled Components
    c. Prettier
    d. All of the Above

Choice: Option D
Justification: Similarly to the backend extensions/frameworks, these all provide useful features that will assist us in our development. First, React Testing Library allows us a very simple way to test the React components we use in our product, as it comes equipped with several utility functions. Syled Components will be used to give us a bit more freedom in designing and using React Components, as it allows us to write actual CSS code to style our React components even more. This is purely a cosmetic decision, as the functionality of the components will still be entirely determined by React, but we want to make our interfaces as user-friendly and appealing as possible. Lastly, we want to include Prettier for a developmental benefit. Prettier is used purely to format our code in a nice, consistent manner. Since this is a group project and all of our coding styles are unique, using Prettier will enforce a unified style that will automatically alter the format of the code we type so that we can easily see what code written by other members of our group is doing. As you can see, all of the optional extensions/frameworks provide meaningful benefits to our project, so we chose to incorporate all of them.

5. What database language should we use?
    a. MySQL
    b. Oracle
    c. Postgres
    d. MongoDB
    e. Neo4j

Choice: Option A

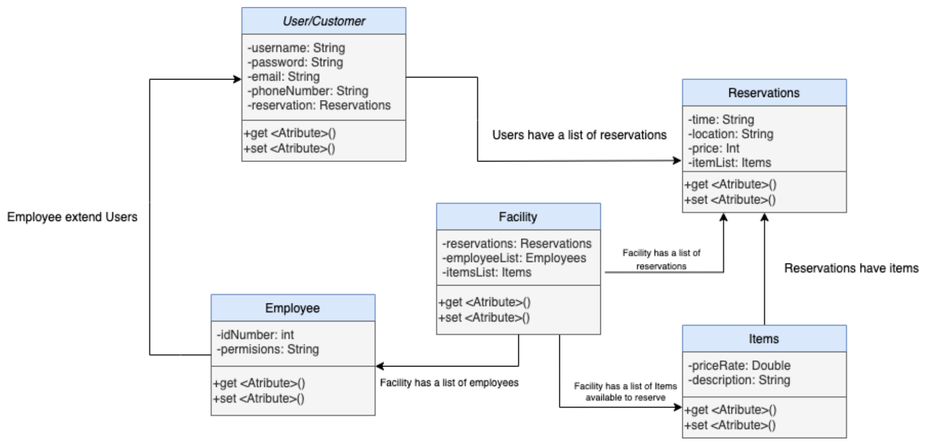
Justification: The most basic reason we chose MySQL is because many of the members of our group have little-to-no database experience at all, and MySQL is very beginner friendly, the most beginner friendly of our options. Furthermore, the MySQL 2 package offered in Node allows us to practically just write raw queries to access the database, which helps with the user-friendliness. MongoDB and Neo4j are out of the picture because they are not traditional relational database schema, they are document and graph schema, respectively, which makes them not only more complicated to learn, but also more complicated to implement effectively. Oracle and Postgres are not bad options, but MySQL is one of the most popular database languages in existence, so there is a lot more information available to help teach those of us who are less familiar with it.

6. What web service should we use?
    a. Microsoft Azure
    b. Amazon AWS
    c. Heroku Hosting
    d. Google Cloud

Choice: Option C

Justification: Originally, we had planned to use Google Cloud hosting, but upon investigation, we found that Heroku more closely aligns with the kind of hosting we are looking for. As none of us have experience building and hosting a web application, we were searching for something that is easy to use and has a lot of information about how to use it. We chose Heroku because we want to use PaaS instead of IaaS, and there are already plenty of tutorials on how to host a React/Node based web application using Heroku. Furthermore, we didn't want to have to spend unnecessary time configuring infrastructure or settings within our hosting options, and since Heroku already has a setup for hosting React/Node applications, we decided it makes more sense to use that as our hosting service.

Descriptions of Classes and Interaction between Classes



- User/Customer
    - User object is created when someone signs up for the application

    - Each user will have a username and password for login purposes
    - Each user will have an email and phone number for contact purposes
    - Each user will have a list of reservations they've made
- Employee
    - Employees extend User object.
    - Each employee will have a Permissions String
        - The String will consist of 1s and 0s. Actions that require permissions will take the String and either allow or disallow the employee from completing that action.
    - A "manager" has all permissions
- Reservation
    - A reservation object is made when a user makes a reservation
    - Every reservation will have a time of when the reservation takes place
    - Every reservation will have a location of where the reservation takes place
    - Every reservation will have a price
    - Every reservation will have a list of Item objects reserved

- Item
    - An Item object is created when an employee creates one for their business
    - An Item object will have a price rate for being reserved
    - An Item object will have a description, of what it is (ie Table, Lane, Room)

- Facility
    - The facility object exists from the beginning
    - The facility contains a list of reservations
    - The facility contains a list of all items in the facility
    - The facility contains a list of all employees

Sequence Diagram

The following diagrams depict the sequence of important events in the application. These include, user login, creating reservable items, making a reservation, cancelling a reservation, and changing a reservation. The sequence displays how data is exchanged in a client-server model. When the user performs an action in the frontend UI, the client sends a request to the server, which then goes to the database to retrieve the requested information. The server then gives the information to the frontend, and the UI is updated as needed.

1. Sequence of user login

2. Sequence of manager, creating a reservable item

3. Sequence of reserving an item

4. Sequence of changing a reservable item

UI Mockups

Navigation Flow Diagram