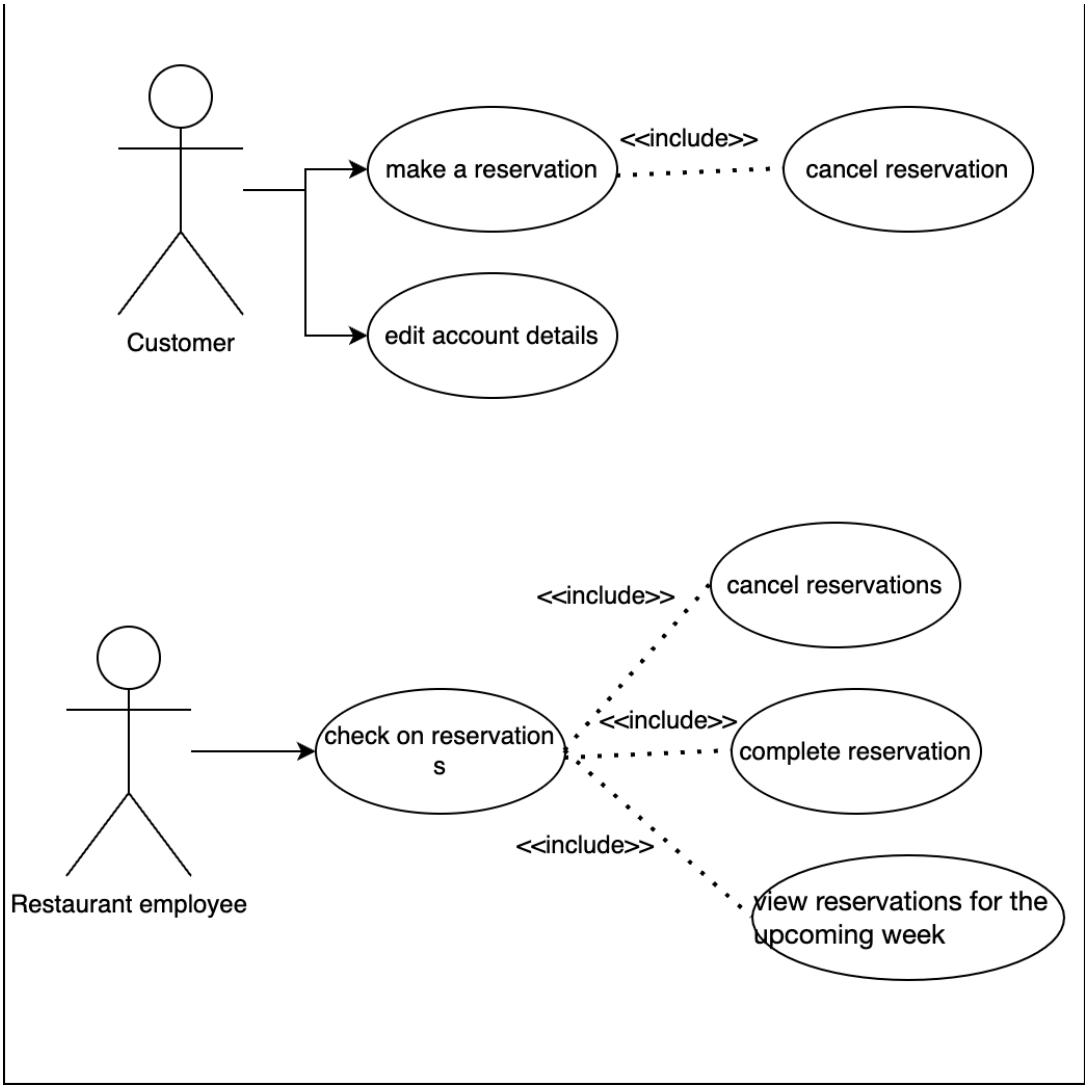
Book-A-Bite: D2 Design Document

1. Project Overview

   We are creating a restaurant reservation web application. We are trying to reduce the workload on restaurant hosts and servers by reducing the number of calls and paper writing that the hosts and servers would have to perform when customers want to schedule a reservation. This would also help customers as they can log on themselves to check available times and then set a reservation, all without having to make a single phone call. This provides a better system for customers and it saves restaurant employees time and thus, money.



Group Number 12
Project Name: Book-A-Bite
Description: Make Reservations for Resturants Seemless

PRODUCT BACKLOG (TO-D

Team Members Diego Lopez, Cameron Boydd, Marcel Newman, Cayden Renegar, Abrar Mian

Story #	Card Front	Card Back	Sprint Number	Priority	Assigned To
ST-0	As a user, I want to be able to see	Add a view all avaiable times button	Sprint 1		
ST - 1	As a parent, I need to reserve a table	Fill out user information showing how	Sprint 1	1st	
ST - 2	As a waiter I need a way to view	User can be shown a list of booked	Sprint 1	8th	
ST - 3	As a manager, I need to see how	The manager should be able to see	Sprint 1	9th	
ST - 4	As a restuarant host, I need to be	Add a cancellation button, that	Sprint 1	10th	
ST - 5	As a student, I want to be able to	Add a filter option for the time of day	Sprint 2	6th	
ST - 6	As a bussiness person, I need to be	Add an option that shows the peak	Sprint 2	11th	
ST - 7	As a Moderator, I want our website	Site encryption, SSO, is enabled.	Sprint 2	4rd	
ST - 8	As a UNCC student I would like to	Implement a system for giving a	Sprint 2	12th	
ST - 9	As a UNCC student, I would like the	Web Standards and Usability	Sprint 2	3rd	
ST - 10	As a UNCC student, I would like to be	Profile page implemented	Sprint 2	5th	
ST - 11	As a Moderator, I would like the page	Implement a table in the database to	Sprint 3	7th	
ST - 12	As a Moderator, I want first time	Tutorial splashscreen after user	Sprint 3	2nd	

2. Architectural Overview

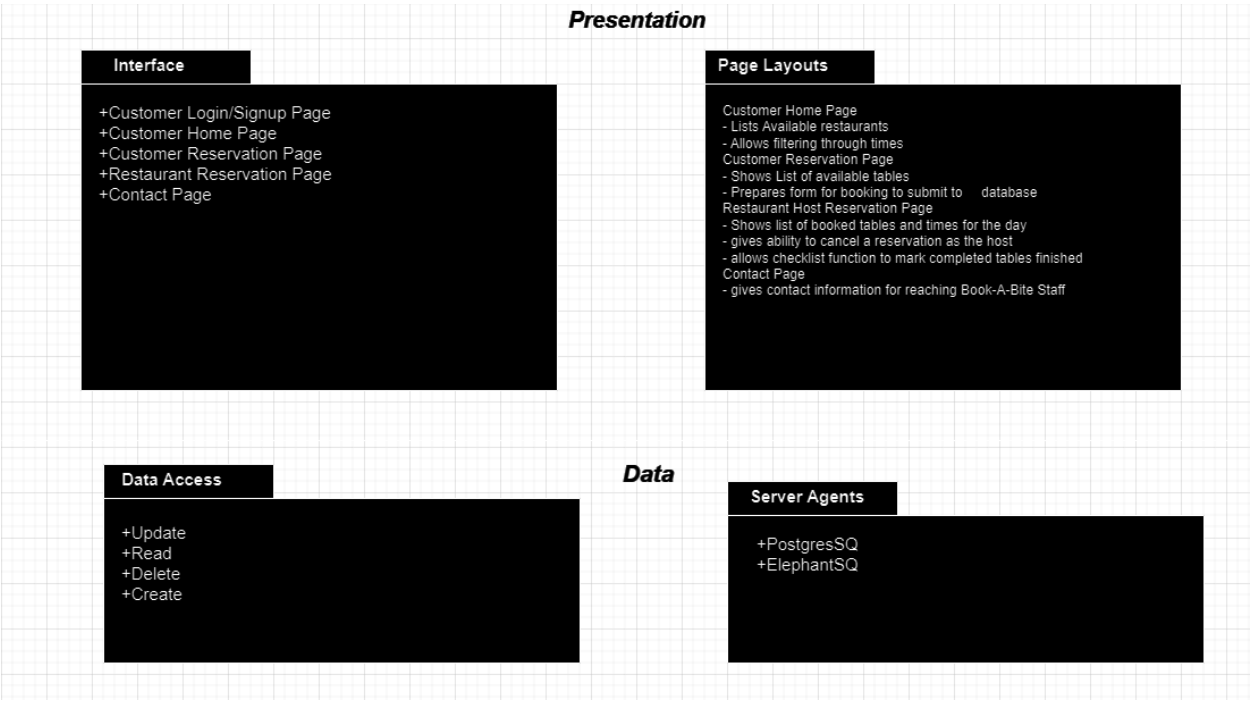
Our application will run off of a custom API back-end, running on node.js and express.js. Our front-end will consist of vanilla javascript, HTML and CSS. The frontend will make calls to the backend to call functions based on the user's input. We originally wanted to try to create a React.js front-end, but based on the team's varying experience in this technology we decided it was no longer feasible. So we scrapped React.js and decided to use vanilla languages, since this is what more team members were comfortable with.

2.1 Subsystem Architecture

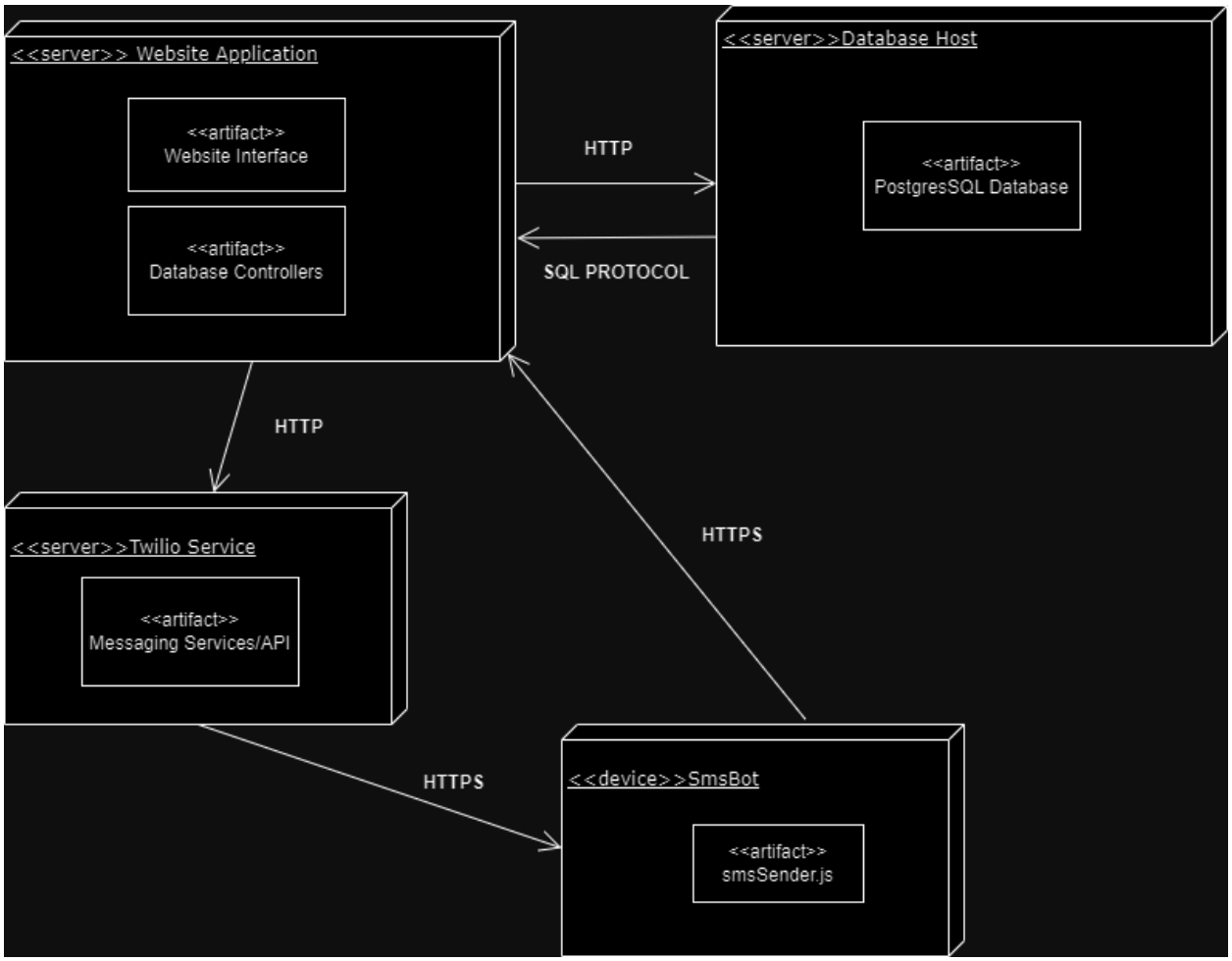
Book-A-Bite Structure Layers:

***Presentation**: This layer shows all functionalities to the user in a clean layout with transparent navigation to each task. Works with the Database to showcase existing/non-existent users as well as availability of restaurants extracted from our continuously updated tables.*

***Data**: Keeps information on the user, all relevant information, and keeps track of table availability for each restaurant and the time slots for their tables. For our presentation, we allow customers and restaurant staff to insert their information and have it displayed and edited based on use case.*



2.2 ***Deployment Architecture***



The client connects to the Book-A-Bite Website Application via HTTP. When data needs to be transferred or retrieved, the application is connected to our SQL Database so information can be quickly delivered.

2.3 Data Model

For our data storage approach, we have opted for a relational database using SQL. This choice enables us to efficiently manage and query the required pieces of information throughout different reservations and users. Within the database, we have designed a schema comprising various tables that tend to organize Users (consumers AND restaurants host accounts) and the reservations that the aforementioned have made. Each table is structured to accommodate identifying user and scheduling information, ensuring clarity and adding security to said database. We focused on scalability due to our potential amount of usage, aiding to seamless integration with our application's functionality.

Our tables (table name - column names):

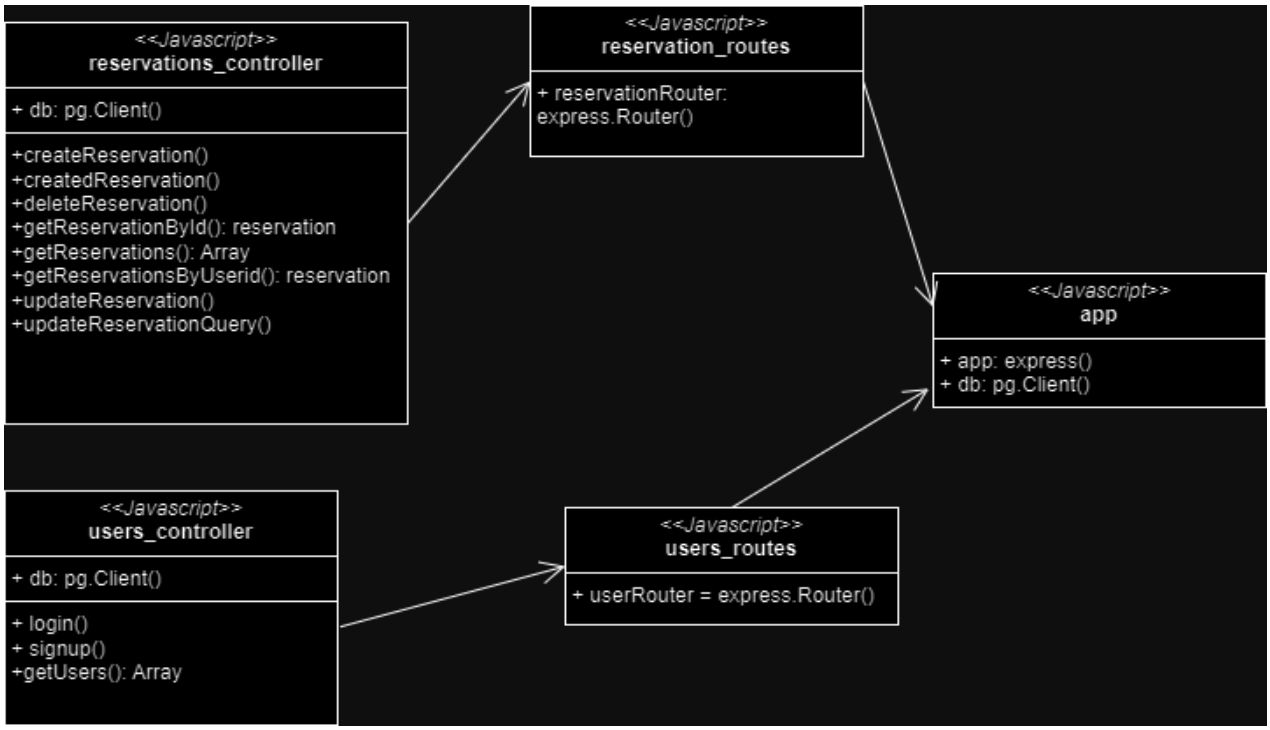
Users - id, name, email, password, phone_number
Reservations - id, user_id, table_number, num_guests, datetime, name, email, phone_number, restaurant
Restaurant - id, name, email, password

2.4 Global Control Flow

When creating the execution control of our system, we decided on a time-dependent model, particularly crucial when managing restaurant reservations within our application. We implement algorithms and logic that consider factors such as reservation duration, peak hours, and restaurant capacity to optimize the booking process. Through time dependency, our system dynamically adjusts availability of tables/restaurants and allocates resources accordingly.

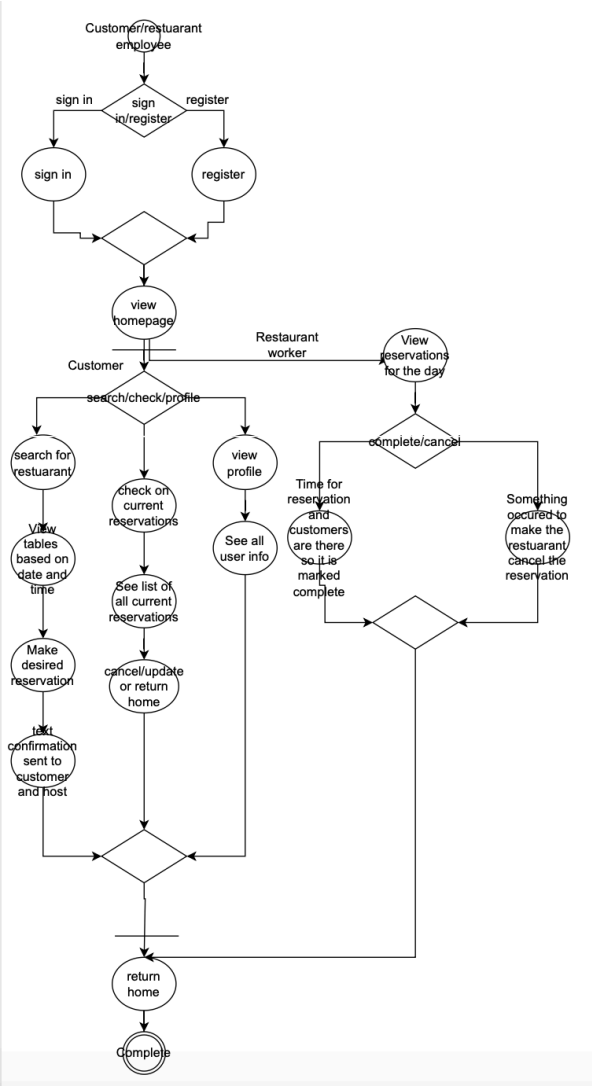
3 Detailed System Design

3.1 Static view



The reservation/users controllers set up the functionality of creating a new account or booking into the database. The request is transferred through to the appropriate route, goes through validation, and then posted as a request.

3.2 Dynamic view



If the user is a customer, they will be able to log in, access their home feed, and select a restaurant of their choice to reserve a table. Once selected, they will be shown available time slots for the table, select their slot, and book a reservation. The reservation is posted and updated in our database.

If the user is a restaurant host or staff member, their home page will immediately show them current/upcoming reservations so they can prepare their areas accordingly for the customer.