Restaurant App

Work Statement:

We are planning on doing a restaurant app that allows for users to view a map or seating chart of a restaurant and determine how busy the restaurant is and where they would like to sit. Customers can view the map and if the restaurant is busy or if the tables that are available are occupied, the tables will be grayed out to indicate that those tables are in use at that specific time. If a table is open, the customer will be able to click on that specific table and a code will be given to that customer. The customer will need to then use that code and present it to the host at the restaurant in a timely manner to ensure that they get the table they have selected.

Team:

Doug Mellon - Database Architect, Network Architect
Jacob Schroeder - Business Analyst, Software Developer
Sean Strawmatt - Project Manager, Security Engineer
Jessica Torres - Web Developer, UI/UX Designer

Project / System Request:

Sponsor:

For this project, we have decided that our project sponsor will be Grace Hector. Grace is currently a server at Chili's in Greeley and she has a lot of experience with how difficult restaurant seating can be. Grace has mentioned some of the difficulties with seating in a restaurant, especially when it gets busy and also during COVID. Customers call and try to be placed on the call ahead list or try to put their name in for a reservation and have so many issues with that, as well as the frustration that comes from sections being closed due to the pandemic. With the use of this app, this would allow for customers to look at the seating chart of the restaurant and will be able to determine if they want to eat at this particular restaurant or not. This app would allow customers to view a map of a restaurant at the time they want to go and see which seats are available.

Need:

This restaurant app would eliminate the use of a customer calling the restaurant and asking "are you guys busy" or "how many tables are available". Instead, a customer would be able to open up the app, and view the seating chart of the restaurant. The seating chart of the restaurant will also indicate how busy the restaurant is by having the tables "filled in" or grayed out to show that a specific table is being occupied at that current moment. If the restaurant looks busy the customer can choose to go to another restaurant or wait it out until a table opens up. If a customer happens to view the seating chart and a few tables are open, that customer can select the table and that table will be reserved for them until they arrive. While the table is reserved, the customer will be sent a verification code and must arrive at the restaurant within a timely manner otherwise the table will be up on the app and shown as "available" on the seating chart.

Requirements:

- Need to design a restaurant layout
- Need a hosting service
- Need a DB
- Admin (us) would need to have login credentials and edit the seating chart as needed (edits would include if we need to close down portions of the restaurant)
- Customers must be able to view a seating chart of the restaurant at all times
- If a customer selects a table, they must arrive at the restaurant and be ready to be seated within a certain time (10 minutes?)
- If two customers happen to click on a table at the same time only 1 of them would be able to get it.
- App will show how much of the restaurant is open (we could gray out some tables or an entire section or something like that)
- Customers will be sent a verification code for the table they reserved so that when they arrive they can just tell the host their code. The code can be sent through email, through text, or can be viewed directly within the app.
- The restaurant should also have access to the admin page to make adjustments as needed.
    - Restaurant owners can go in and edit the seating chart of the restaurant if issues or problems occur.
    - Example: if they need to close down a handful of tables, the tables would appear grayed out on the customers end of viewing the app.
- Seating chart must be updated at all times to ensure that customers are viewing the "current" seating chart.
- People with "reservations" get first pick - would not cause issues with people using app selecting a table and those walking in at the time the user selected the table
    - People that walk in "now" get a table based on the unreserved tables
    - We should have several tables that cannot be reserved online. These tables would only be for guests who walk in the restaurant and these can be randomized or toggled

Value:

This project would end up being valuable for both employees working at the restaurant and customers going to the restaurant. Grace mentioned that "the value would be huge in the customer service aspect because they know what they're walking into". Simply meaning that when the customer views the app, they can see which parts of the restaurant are open and closed and how busy the restaurant is at that time. Another valuable item this app would bring to the table would be helping out the actual staff within the restaurant. COVID impacted just about every type of business but especially the restaurant business. Grace explained that "Because of COVID, restaurants are understaffed and don't have full capacity every night based on staffing. This allows people to see the capacity of the restaurant before they get there, and so if they have a section closed down, people will know in advance and not get angry when they see an empty

section, but are put on a wait". This could potentially end up bringing in more revenue and value to restaurants if another pandemic hits so that customers know how long their wait will be and will know what to expect, rather than walking in and getting disappointed with their visit. Grace also mentioned that some people, when going into a restaurant, insist on a booth, and will wait for one. This allows people that want a booth to wait for one, and others that don't care, can wait a separate time for the first available table.

Constraints:

- We will only be able to design this app for one restaurant within the given time frame.
- Possible budget constraint from hosting service
- Possible design of the restaurant seating chart
- Identifying different users and generating different codes for them
- Real time DB management to ensure that everything is up to date with tables customers have selected

Level Of Effort:

Task	**Estimate Time**
Need a hosting service (DigitalOcean / AWS?)	**1 week**
Need a Database (design, host (managed?), build, etc.)	**2 days**
Setup version control and CI/CD pipeline ahead of development.	**2 days**
Design, build, document, and implement API layer	**3 weeks**
Implementing user stories correctly	**2-3 weeks**
Admin (us) would need to have login credentials and edit the seating chart as needed (edits would include if we need to close down portions of the restaurant)	**1.5 weeks**
Need a graphical display of the restaurant layout	**1 week**
End user signup / form to	**1 week**

submit reservation	

Total Time: 12 weeks, 2 days

Feasibility Analysis:

Technical:

- Can we build it (yes)
- What risks are associated with this project?
    - Risks of not being able to successfully implement everything that we need to within the timeframe and on budget.
    - The possibility of having to learn new tooling, frameworks, etc.
    - Limited budget for infrastructure which could impact server reliability in terms of scaling.
    - Small team size will require individual members to wear multiple hats — creating the potential for redundant tasks without proper management or strategy.
    - Small team size could create issues if one or more team members have to take time away from the project.

Economic:

The economic feasibility aspect for this project does make sense. This is an app that could improve how the restaurant industry works by possibly bringing in more value. If customers have the ability to view the restaurant they want to go to at that particular time, they would be able to make the decision then rather than taking the risk of going to the restaurant and having to wait or deal with some other issue. Initially, the cost to build this app would not be very high because we do not need access to many items. Essentially all we need is access to the restaurant's layout, and some sort of time calculator that indicates how busy the restaurant is. From how I am understanding this, our return on investment would be huge because we are developing a simple app that could potentially be useful to everyone who has any experience going to or working at a restaurant.

Organizational:

Ultimately this project would fit in with any restaurant. For a restaurant to have this feature it would impact them in a positive way just because it would allow for restaurants and customers to be on the same page in terms of understanding how busy the restaurant is at that time. If we were to ask any other server about this idea they would essentially say the same thing. Since we have been going through a pandemic for a while I think that this app would have much more value now rather than before the pandemic because restaurants have lost so much revenue and employees.

Project Plan:

Approach:

We will be using the waterfall method.

Tasks:

Interview Grace, designing the infrastructure, design database, setting up hosting service, designing an API, designing both front and back end, graphic design, authentication.

Timeline:

Fall 2021 through Spring 2022.

Analysis Phase

Requirements Definition Statement:

- **Business Requirements:**
    - The goal with this restaurant app is to make life easier for both restaurant workers and customers. Customers can view at any time how busy the restaurant is by a seating chart and view how many tables are taken or not taken. This would bring value by allowing for less stressed staff and customers and hopefully allow for the restaurant to run much more smoothly.
- **User Requirements: (what does the user need to do in order to meet all business requirements)**
    - An admin user must be able to login and edit the way the restaurant is layed out
        - This helps when customers go to the app and can view the "current" seating chart if any major changes have been made
        - This is mainly for when large parties of people come in or when a section of the restaurant needs to be closed down.
    - A customer needs to be able to go to the app and view the seating chart to determine whether or not they want to eat at the restaurant
        - Customers can use the app to view how busy or not busy the restaurant is
        - By allowing for customers to see a current time of the restaurant they can choose if they want to reserve a table or not.
    - If customers do want to eat at the restaurant, they need to be able to select a table that is open
        - If a table is open it will not be grayed out and if the customer selects the table then a confirmation code will be displayed and they must arrive at the restaurant within a certain number of minutes
        - This is the main functionality of the restaurant where customers can view the seating chart and select their table

    - If customers choose a table that is already taken, then they need to select another table
        - Customers cannot choose tables that are currently being used by other customers in the restaurant.
        - If a customer tries to select a grayed out table, then they will receive an error message on the screen.
- **Functional Requirements: (what does the system need to be able to do inorder for the user to do what they need to do.**
    - System must have a specific time slot for when customers "open" the app till the time they select their table
        - This is important because we don't want multiple people choosing the same tables at the same time. Customers should be given a time slot of 2-5 minutes from when they open up the app to when they choose their table
        - This is a similar concept for choosing seats at a movie theater.
    - System must recognize that an admin has logged in
        - The system needs to be able to allow for only admin users to login and when they log in they need to be able to have admin abilities such as changing the seating chart, opening/ closing tables, possibly posting any important messages out to the restaurant that can be viewed on the customers side.
        - System must also require a specific password length for the admin when they log in.
        - System must be able to differentiate between which admin login if there are multiple accounts.
    - System must have a "current" time showing the layout of the restaurant
        - This is one of the features that makes this app so important is that it allows for customers to view the current time and how busy the restaurant is whenever they open up the app.
    - System must have hours of operation implemented into it
        - Hours of operation can be edited when the admin login.
        - The system should have a timer running in the background for normal hours of operation
            - If a user logs in at 2 AM the restaurant should be greyed out
            - Messages should be displayed to users when the restaurant is going to close in an hour or less.
    - System needs to be able to allow for customers to select which tables they want
        - Another main functionality of this app is allowing the users to click on the table they want
        - Once a user clicks on a table they want, the system needs to generate a random code (length 5-8) for the customer to use
    - System needs to be able to select 1 customer if multiple customers select the same table at the same time
        - System needs to be able to choose which one
        - This could be done randomly or first come first serve (whichever would be easier to implement)

- **Non-functional Requirements:(what characteristics does the system need to have in order for all other requirements to happen)**
    - Operational:
        - The system needs to connect to a database to store admin and other information
        - The system should run on all major browsers
    - Performance
        - The system should support up to 15 customers at a time
        - The system should always be updated to show the current view of the restaurant
        - The system should have the five nines uptime.
    - Security
        - No customers can access the admin view of the restaurant
        - Only the admin may change the layout of the restaurant
        - No external services can connect to the database
        - No plaintext passwords (auth system)
    - Cultural/political
        - Admin information is protected
        - The web app will be available in english

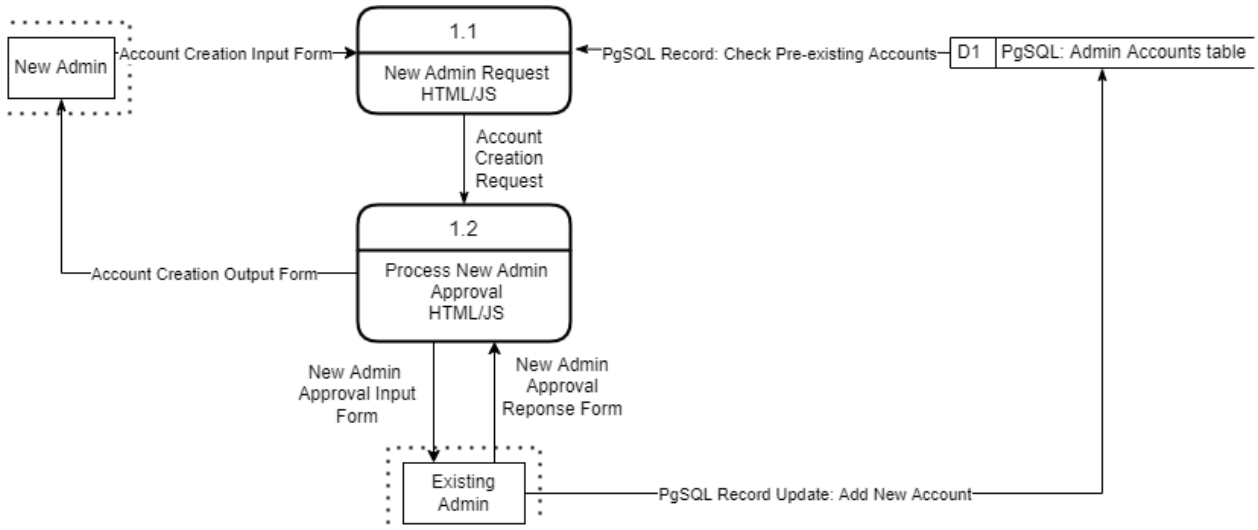
User Stories:

- Admin:
    - 1) As an admin, I want to be able to create an account so that I can access the seating chart.
    - 2) As an admin, I want to be able to edit the seating chart so that I can close off portions of the restaurant as needed.
    - 3) As an admin, I want to be able to post announcements to the app so that customers can be informed of anything important (closing times, deals,...)
    - 4) As an admin, I want to be able to remove my restaurant if I don't want it on the app anymore.
- Customers:
    - 5) As a customer, I want to be able to search for restaurants in my area so I can create a reservation.
    - 6) As a customer, I want to be able to view which restaurants have the most available tables
        - Can view a couple of different restaurants in the area
    - 7) As a customer, I want to be able to view the tables that are available so I can reserve one.
    - 8) As a customer, I want to be able to see a message about how busy the restaurant is at the time I'm looking at the app
        - Could display how busy the restaurant is ("not too busy", "busy", "expect a 20 minute wait")

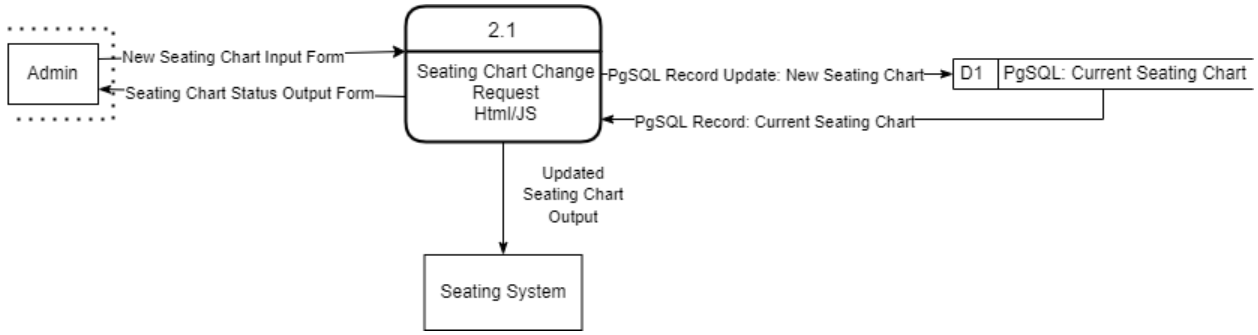
- 9) As a customer, I want to be able to view the contact information of a restaurant so I can reach them if I have to.

Process Models: flow chart of the tasks that happen (visualization of a business process)

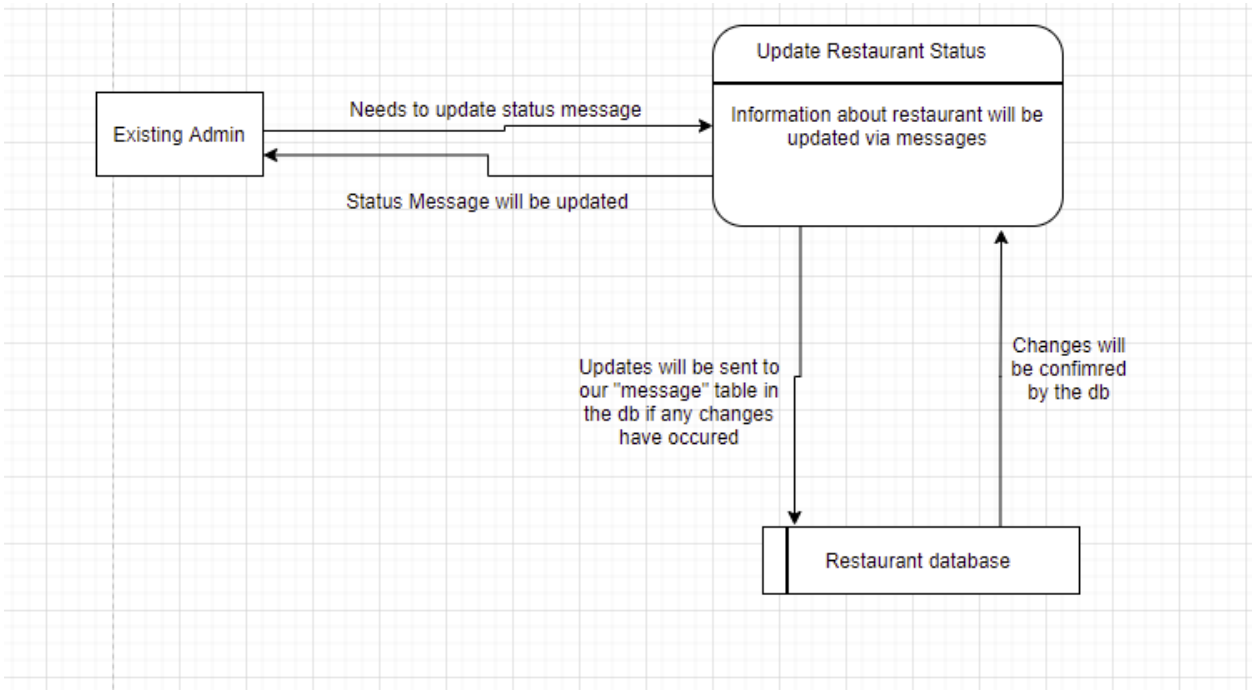
Process Model #1 (user story #1):



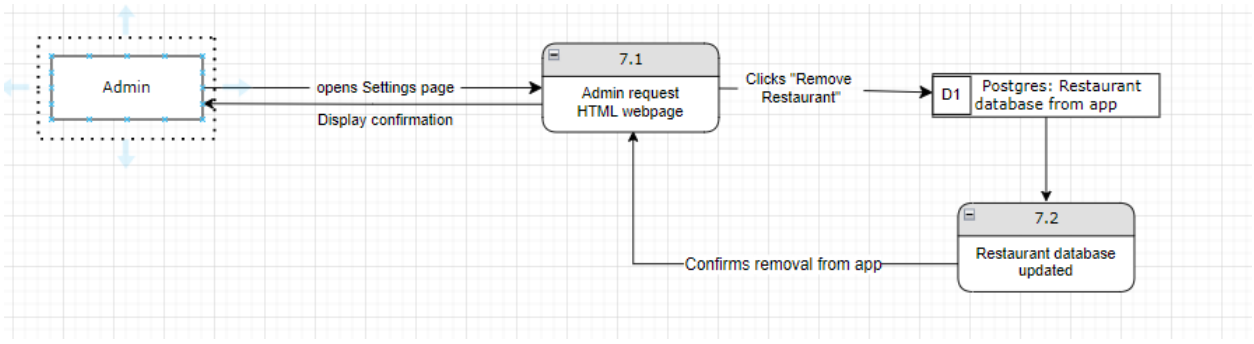
Process Model #2 (user story #2):



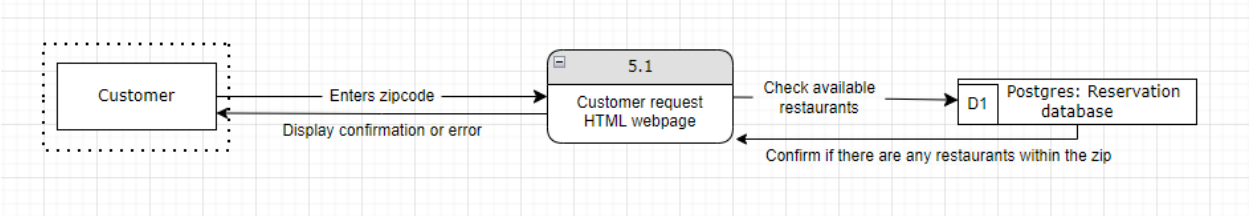
Process Model #3 (user story #3):



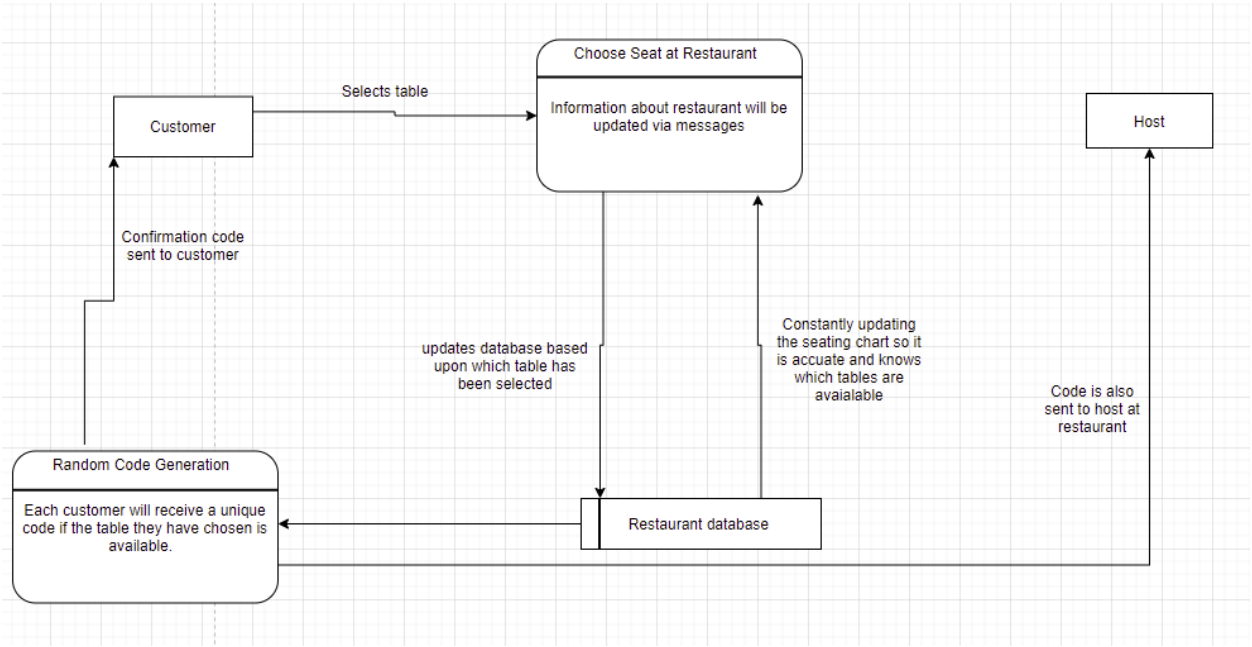
Process Model #4 (User Story #4)



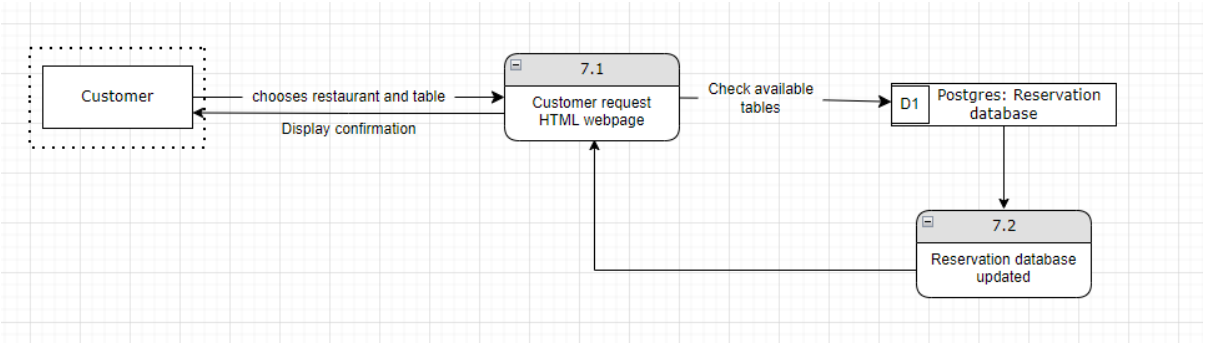
Process Model #5 (User story #5)



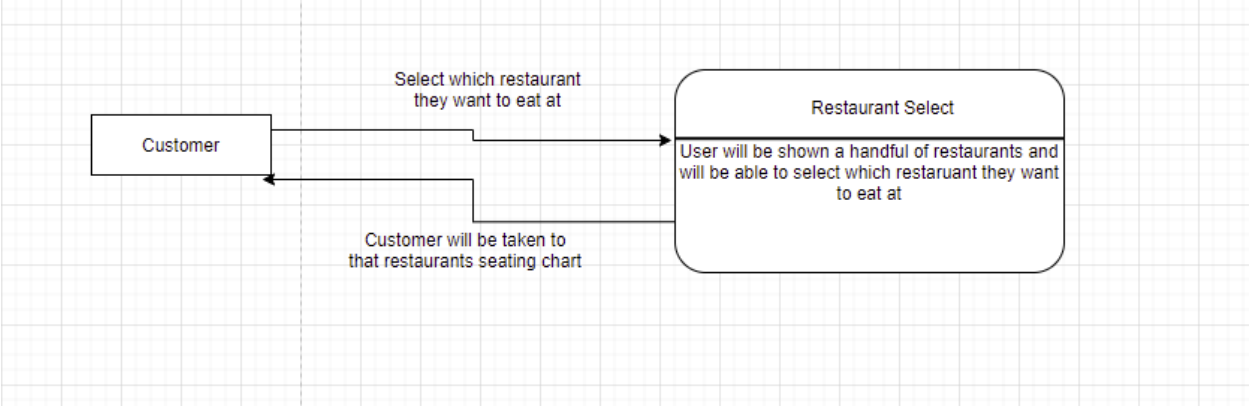
Process Model #6 (User story #6)



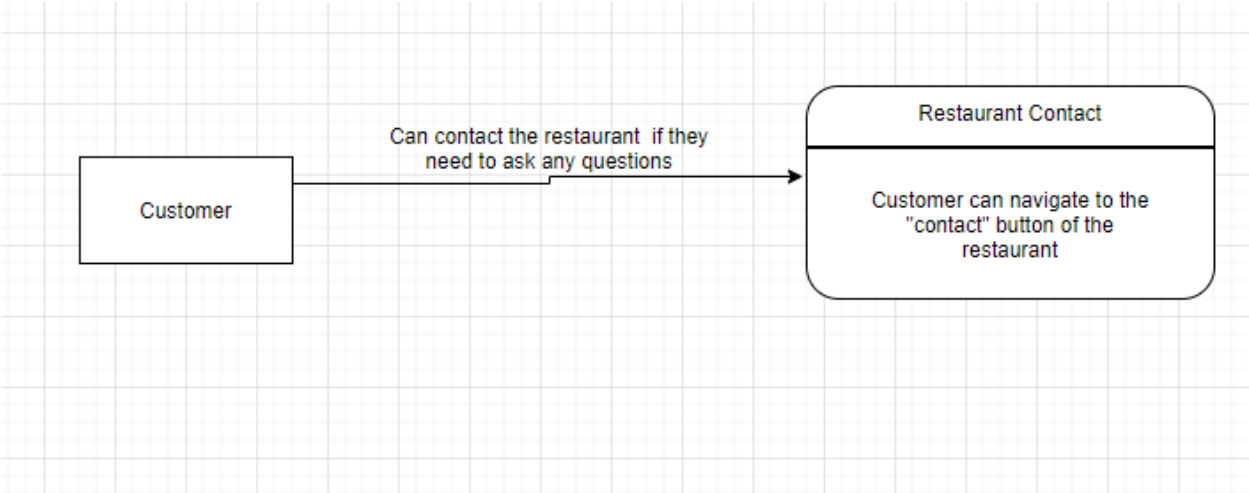
Process Model #7 (User Story #7)



Process Model #8 (User Story #8)

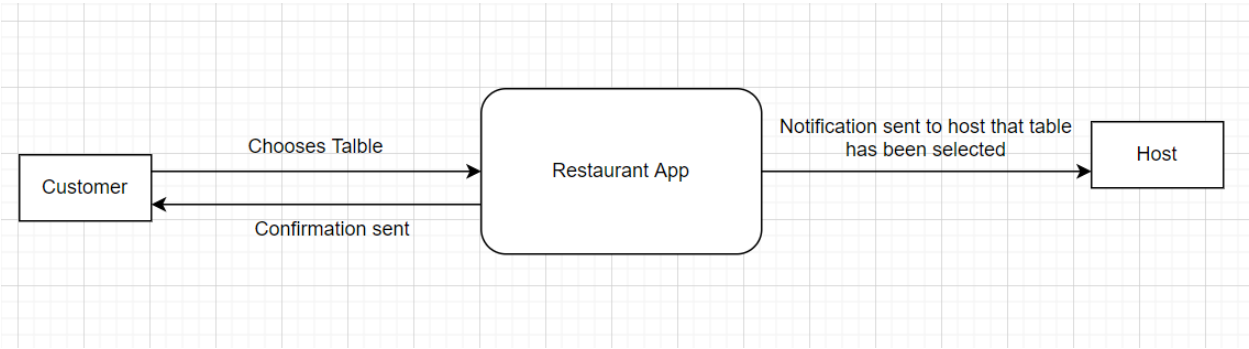


Process Model #9 (User Story #9)

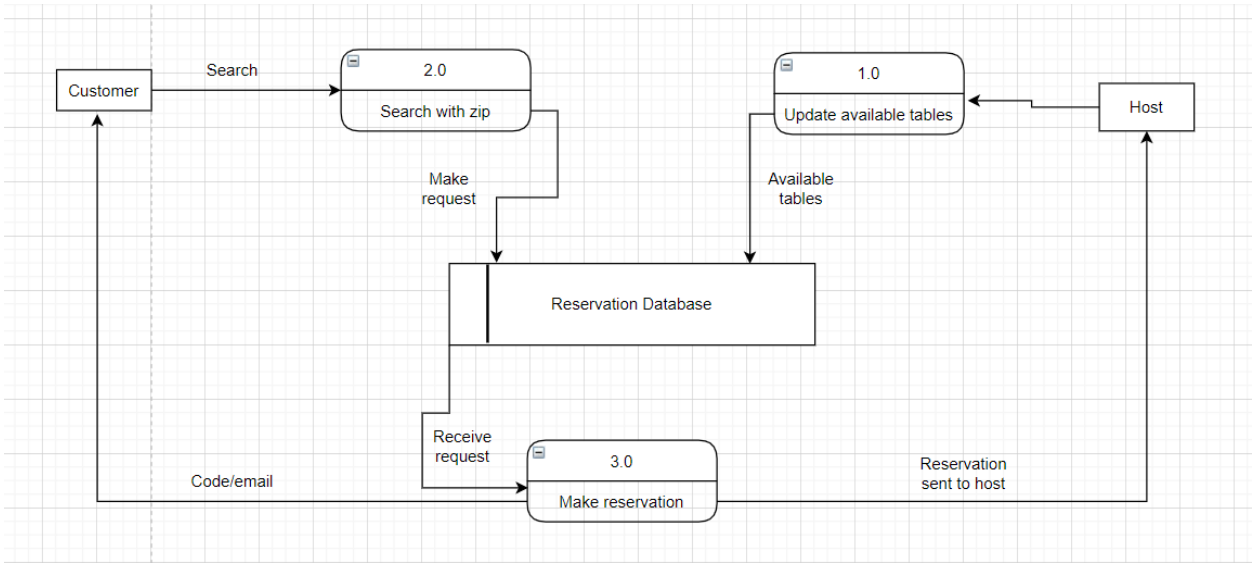


Data Models:

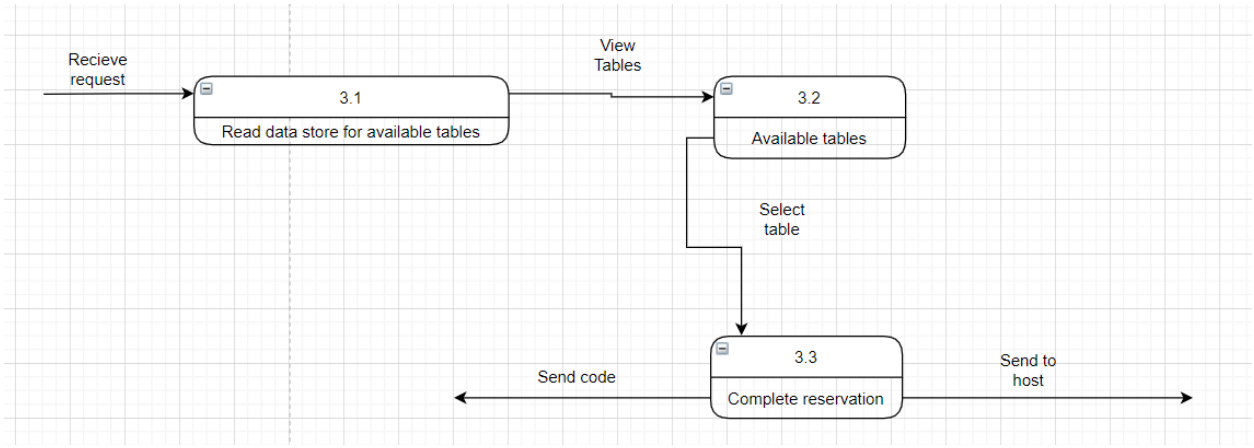
Context data flow diagram



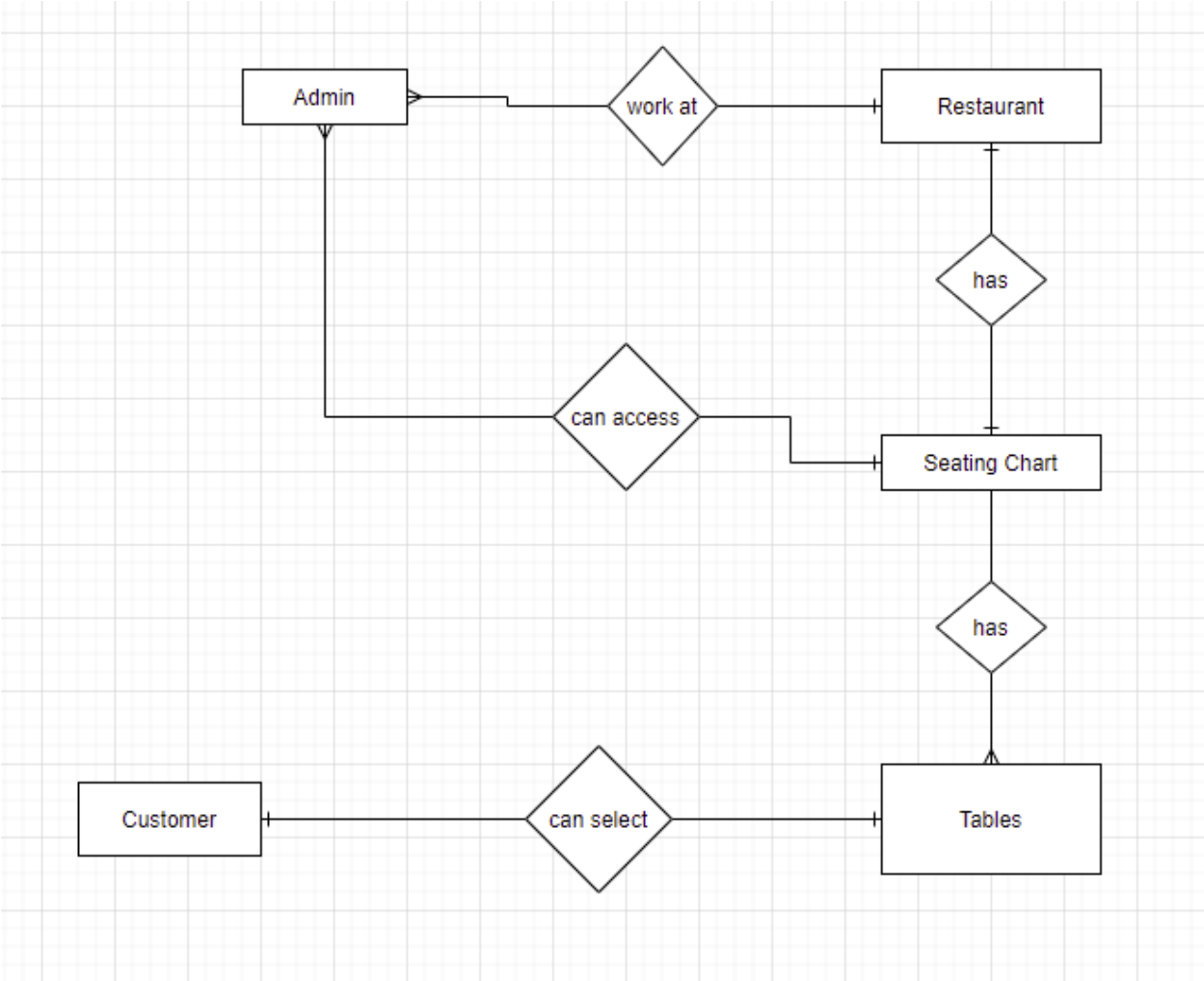
Data flow diagram level 0



Data flow diagram level 1



High level ERD



Design

Design Narrative:

- Design db from user stories
    - Why we chose Postgres (performance, libraries, etc.):
        - We will be utilizing PostgreSQL for our database. PostgreSQL is a free and open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance. However, because it is technically an object-relational database, it is capable of higher performance, more advanced data types (ie. JSON support), while also being SQL compliant. We have chosen PostgreSQL due to the above advantages, the available object-relational mapping (ORM) libraries, and

no need to target an RDMS with faster read capabilities.

    - The database design process:
        - The first step in designing our database will be to analyze the user environment. We will work closely with the client to gain a strong understanding of their use case, and deliver what the client has requested in the project request phase. We will also consider future use cases to ensure proper transition with new applications. A conceptual data model will be created to identify business concepts and the relationships between them to build a better understanding of their requirements from a data perspective.

        - A logical model will be developed to map the conceptual model to PostgreSQL and a physical model to plan the data layout while considering our hardware and software decisions. Finally, we will evaluate the physical model to estimate performance and implement it to an operational level. We will also be documenting the structure of our database in an easy-to-read format.

- Build out the db, write tables + relationships
    - What tools will we be using
        - For the creation of our database we will be using psql which is a terminal-based front-end to PostgreSQL as well as datagrip for generating diagrams and management. The database itself will either be hosted on a PostgreSQL managed server at the DigitalOcean SFO location or on a DigitalOcean droplet in the same region. Backups and system monitoring will be completely automated through the DigitalOcean platform.

- Design what endpoints we want to make available for our API
    - Determine what data needs to be accessed and by who.
        - To identify which data needs to be accessed, we will review what information is needed from the restaurants as well as the end user. All access will be available through CRUD operations in a REST API that is designed according to the initial requirements in Roy Fielding dissertation. We will be developing the API in Django using the Python language as it fits the skill sets of the team. The API itself will be used as an access layer to separate our system into microservices — ultimately allowing for future growth and scaling without having to make significant alterations to the backend. Upon completion of the API, we will use both the Postman application and curl in our terminal to test the responses to our application.

- Design UI/UX

    - We will research, design, and test different layouts to ensure an inclusive design that is easy to use for both the customer and end user. All frontend code will be completed using CSS, HTML, and Javascript with React using modern hooks. The layout will be completed using the Bootstrap framework.

- Use github actions to automate deployment
    - Github will be used for version control of all source code as well as GitHub Actions to automate builds, testing, and deployment from Github to our Ubuntu Server. Code reviews will be conducted before pull requests are approved from each member of the team.

- Containerization
    - Because we want to develop the application with future growth in mind, we will deploy with scalability in mind. Both the frontend and backend will be deployed to our server in Docker containers. This will allow us to isolate each deployment and keep all requirements, configuration files, and libraries synced. We will open up communication between containers using Docker Compose and YAML files for configuration which will enable us to scale the containers as necessary. Using Docker containers will also allow us to easily transition to Kubernetes should we desire.

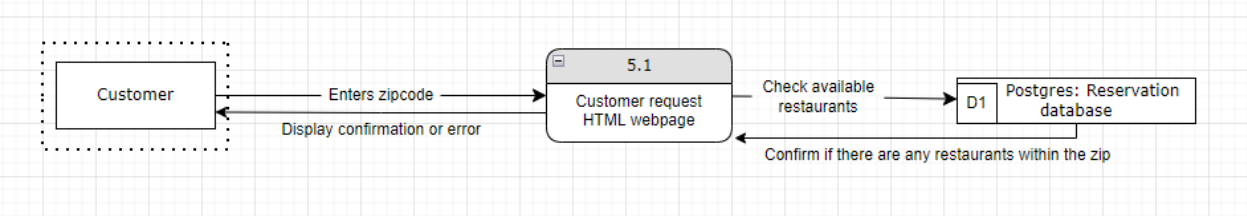
List of Technologies:

- PostgreSQL / pgAdmin - Database
- Docker / Docker Compose - containerization
- Django (Python) - REST API
- Github - Version control and CICD with Github Actions
- React (Javascript / HTML) - front end
- DigitalOcean - Ubuntu server
- Nginx - Reverse proxy

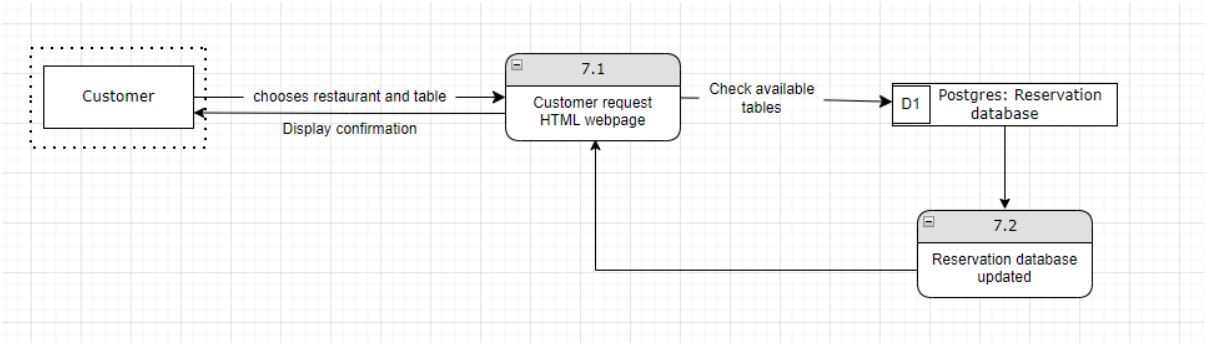
Physical Process Models:

We decided to choose 4 of our most important process models to convert to physical process models.

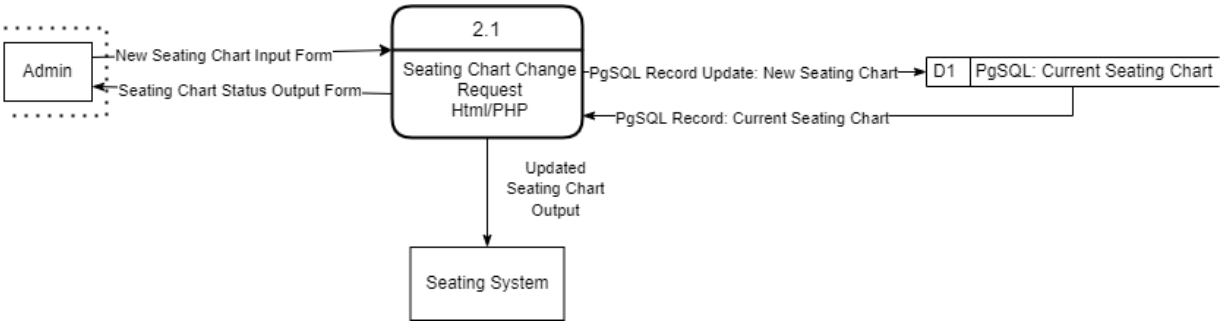
    - 5) As a customer, I want to be able to search for restaurants in my area so I can create a reservation.



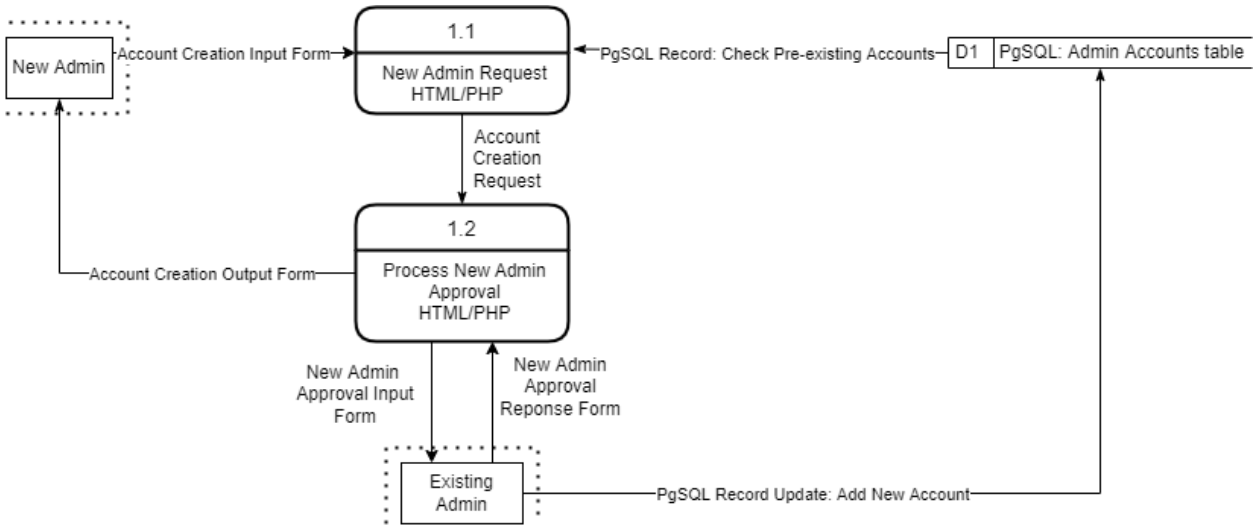
- 7) As a customer, I want to be able to view the tables that are available so I can reserve one.



- 2) As an admin, I want to be able to edit the seating chart so that I can close off portions of the restaurant as needed.

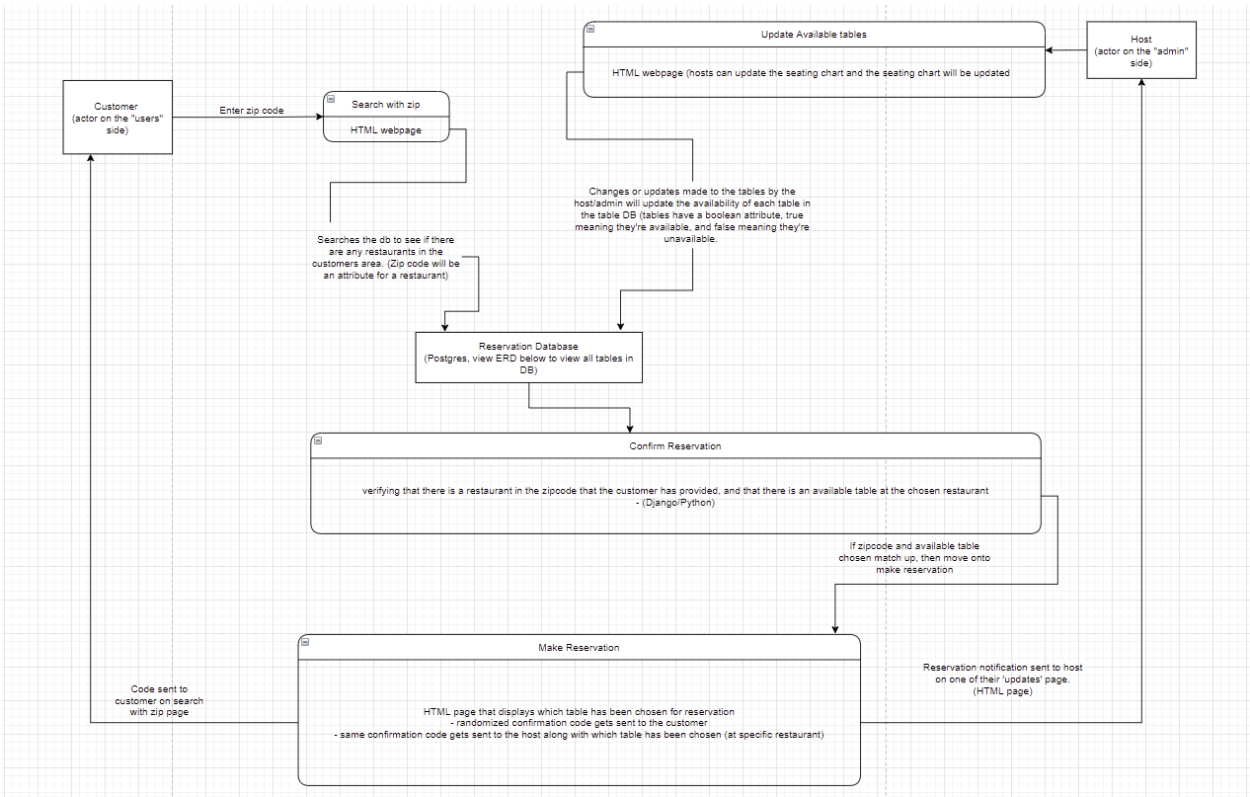


- 1) As an admin, I want to be able to create an account so that I can access the seating chart.

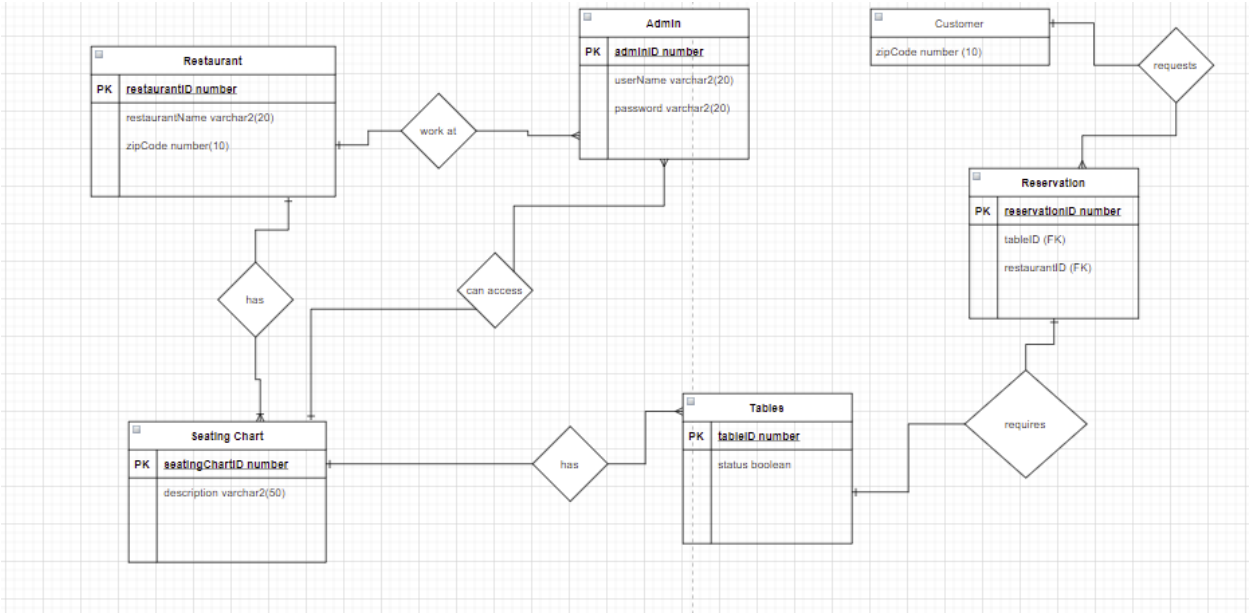


Physical Data Models:

Overall physical data model of how most of our system will work. Requests will be sent to our database, the database will resolve the requests, and send back the data to our frontend.



ERD



Implementation

Source control: for our project we will be using our Github repo which can be found at https://github.com/bacs487-restaurant-reservation/restaurant-reservation

Work Assignments:

As far as work assignments go, we will be using Trello or Jira for our task tracker. This will allow us to understand what stage each user story or other task is at. We don't have specific tasks as of yet of what we will be doing however a general idea would be that we will all take part in the initial phases of the project. Such as setting up accounts, understanding how all of the technologies will fit together and then carrying out the initial steps of the project. Sean will be assigning tasks, and helping in every aspect possible, Jessica will be responsible for understanding the technologies and building out the UI/UX experience of the app. Doug will be responsible for building and setting up the database, and Jacob will be responsible for a large portion of the coding. Since we are

a smaller team, we will all be wearing each other's "hats" because we will all need help and we all want to learn and understand this project as a whole. We will be able to also monitor and track each other and see where we are and how we're doing. Additionally, we will be incorporating Github issues for issue reporting and tracking.

Testing:

We think the route for us to go next semester would be to do integration testing or unit testing. It's hard to say at the moment but thinking about how our project is set up and built, we might need to switch between integration and testing. However, I think that most of our testing will be integration testing because we can split up and all work on different parts and then come back and piece them together.

Documentation:

For this project, we will all take part in documentation — including reviews before updating it. I think that we should come up with a method for documenting so most of our documentation looks similar. We will throw in a documentation folder in our Github repo, and put our documentation there in markdown files. We will have detailed documentation on every process that we do so that if we need to review something or do another project in the future, we have it.