



# Boston Foodies


By Madie Simmons, Katherine Kelly,  
Anthony Yang, Shameer Ahmad, and Joseph Kim

# Real-World Problem

- People new to the area or recent grads lack an easy way to find fun, local restaurants
- No way to meet other foodies!
- Small businesses struggle to compete against the trendy chains and get their name out there





The background is a light beige color with various food-related illustrations. At the top left, there is a brown paper bag with a white drawstring and several brown dots. To its right is a brown swirl. On the right side, there is a brown paper cup with a white drawstring. At the bottom left, there is a brown swirl. At the bottom center, there is a brown paper bag with a white drawstring and several brown dots. At the bottom right, there is a blue plate with a variety of food items, including green leafy vegetables, orange slices, a brown fried item, and white rice.

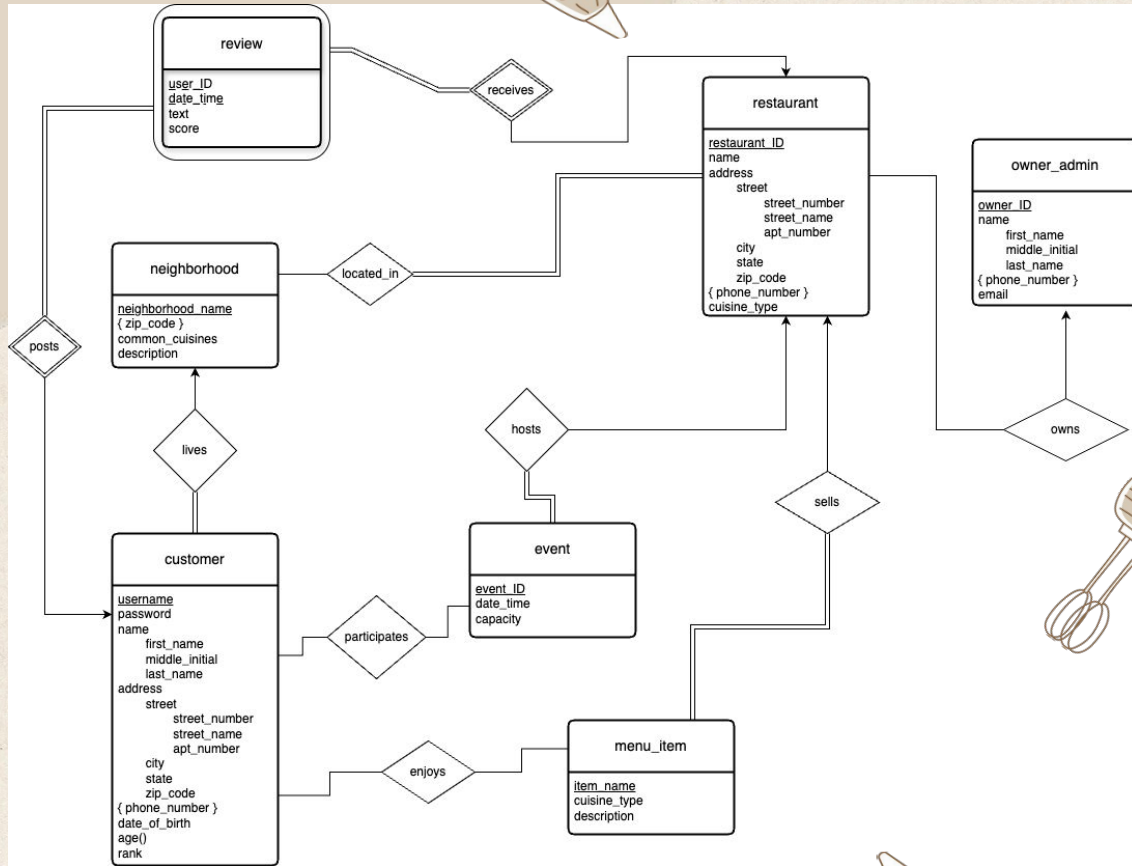
# Our Solution!

## Aviram's Diners

An application to connect avid foodies with a like-minded community to explore local Boston cuisine!

- Users can sign up to join, meeting fellow foodies to explore the Boston restaurant scene
- Restaurants can host events with special discounts

# ER Diagram





# Key Features

```
5 CREATE TABLE IF NOT EXISTS user (  
6     user_id INTEGER PRIMARY KEY AUTOINCREMENT,  
7     username TEXT NOT NULL UNIQUE,  
8     password TEXT NOT NULL,  
9     role TEXT CHECK(role IN ('customer', 'owner')) NOT NULL,  
10    is_logged BOOLEAN NOT NULL DEFAULT 0  
11 );  
12
```

- Changes to the schema as we built out our product
- One user table
  - Users can be customer or owners
  - Different privileges
  - Owners manage restaurants
  - Customers leave reviews



# Database Schema

```
1
2 CREATE TABLE restaurant (
3     restaurant_ID INTEGER PRIMARY KEY,
4     name TEXT NOT NULL,
5     street_number TEXT NOT NULL,
6     street_name TEXT NOT NULL,
7     apt_number TEXT,
8     city TEXT NOT NULL,
9     state TEXT NOT NULL,
10    zip_code TEXT NOT NULL,
11    cuisine_type TEXT,
12
13    FOREIGN KEY (zip_code) REFERENCES neighborhood_zip(zip_code)
14 );
15
16 CREATE TABLE restaurant_phone (
17     restaurant_ID INTEGER,
18     phone_number TEXT,
19     PRIMARY KEY (restaurant_ID, phone_number),
20     FOREIGN KEY (restaurant_ID) REFERENCES restaurant(restaurant_ID),
21     CHECK (phone_number LIKE '____-____-____')
22 );
```

```
24 CREATE TABLE owner_admin (
25     owner_ID INTEGER PRIMARY KEY,
26     first_name TEXT NOT NULL,
27     middle_initial TEXT,
28     last_name TEXT NOT NULL,
29     email TEXT NOT NULL UNIQUE
30 );
31
32 CREATE TABLE owner_phone (
33     owner_ID INTEGER,
34     phone_number TEXT,
35     PRIMARY KEY (owner_ID, phone_number),
36     FOREIGN KEY (owner_ID) REFERENCES owner_admin(owner_ID),
37     CHECK (phone_number LIKE '____-____-____')
38 );
39
40 CREATE TABLE event (
41     event_ID INTEGER PRIMARY KEY,
42     date_time DATETIME,
43     capacity INTEGER CHECK (capacity > 0)
44 );
45
```





# Database Schema (cont.)

```
46 CREATE TABLE menu_item (  
47     restaurant_ID INTEGER,  
48     item_name TEXT,  
49     cuisine_type TEXT,  
50     description TEXT,  
51     PRIMARY KEY (restaurant_ID, item_name),  
52     FOREIGN KEY (restaurant_ID) REFERENCES restaurant(restaurant_ID)  
53 );  
54  
55 CREATE TABLE review (  
56     user_ID TEXT,  
57     date_time DATETIME,  
58     text TEXT,  
59     score REAL CHECK (score BETWEEN 0.0 AND 5.0),  
60     PRIMARY KEY (user_ID, date_time),  
61     FOREIGN KEY (user_ID) REFERENCES customer(username)  
62 );  
63  
64 CREATE TABLE neighborhood (  
65     neighborhood_name TEXT PRIMARY KEY,  
66     common_cuisines TEXT,  
67     description TEXT  
68 );  
69  
70 CREATE TABLE neighborhood_zip (  
71     neighborhood_name TEXT,  
72     zip_code TEXT NOT NULL,  
73     PRIMARY KEY (neighborhood_name, zip_code),  
74     FOREIGN KEY (neighborhood_name) REFERENCES neighborhood(neighborhood_name)  
75 );
```

```
77 CREATE TABLE customer (  
78     username TEXT PRIMARY KEY,  
79     password TEXT NOT NULL CHECK (LENGTH(password) >= 8),  
80     first_name TEXT NOT NULL,  
81     middle_initial TEXT,  
82     last_name TEXT NOT NULL,  
83     street_number TEXT NOT NULL,  
84     street_name TEXT NOT NULL,  
85     apt_number TEXT,  
86     city TEXT NOT NULL,  
87     state TEXT NOT NULL,  
88     zip_code TEXT NOT NULL,  
89     date_of_birth DATE,  
90     FOREIGN KEY (zip_code) REFERENCES neighborhood_zip(zip_code),  
91     CHECK (zip_code >= '02108' AND zip_code <= '02467')  
92 );  
93  
94 CREATE TABLE customer_phone (  
95     username TEXT,  
96     phone_number TEXT,  
97     PRIMARY KEY (username, phone_number),  
98     FOREIGN KEY (username) REFERENCES customer(username),  
99     CHECK (phone_number LIKE '____-____') -- Replace this with regex in application code  
100 );
```



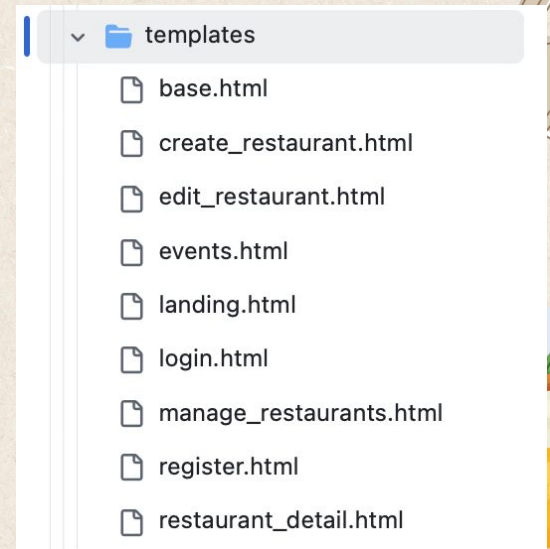
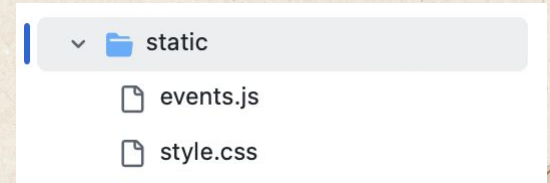
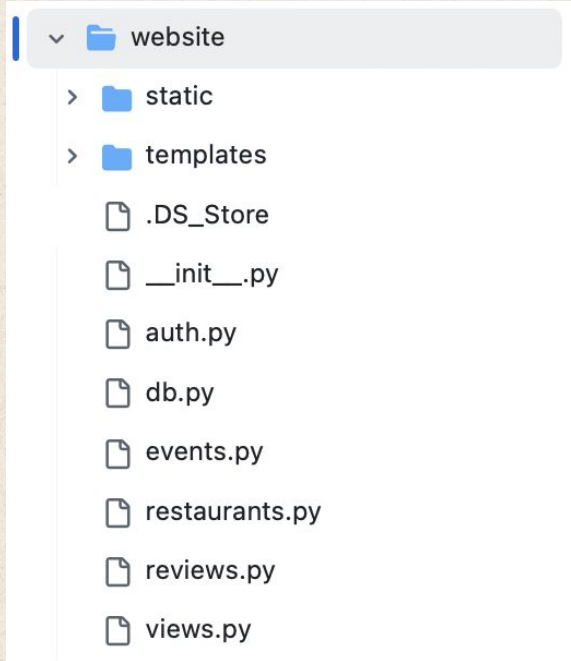
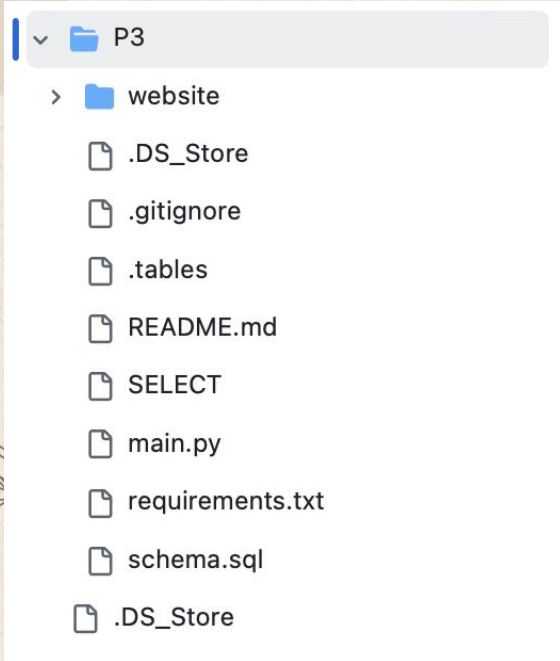
# **Aviram's Diners™**

**(not actually trademarked – yet!)**

**+  
demo**



# File Organization



# auth.py

```
56 @auth.route("/register", methods=["GET", "POST"])
57 ✓ def register():
58     if request.method == "POST":
59         username = request.form.get("username")
60         password = request.form.get("password")
61         role = request.form.get("role")
62         verification = request.form.get("verification")
63
64         if role == "owner" and verification != current_app.config['VERIFICATION_CODE']:
65             return "Wrong verification code", 400
66
67         conn = get_db_connection()
68         cursor = conn.cursor()
69         cursor.execute(
70             "INSERT INTO user (username, password, role, is_logged) VALUES (?, ?, ?, ?)",
71             (username, password, role, 1)
72         )
73         # Get the user_id of the inserted user
74         user_id = cursor.lastrowid
75
76         conn.commit()
77         conn.close()
78
79         # Set user ID in session
80         session['user_id'] = user_id
81
82         return redirect(url_for('auth.landing_page'))
83     return render_template("register.html")
84
```

# \_\_init\_\_.py

```
3 ✓ def create_app():
4     app = Flask(__name__)
5     app.secret_key = "mysecretkey"
6     app.config['VERIFICATION_CODE'] = "foodies"
7     -
```



The background is a light beige, textured surface. It features several hand-drawn food items: a plate of food with a fried item, egg, and vegetables in the top left; a bowl of ramen with a hard-boiled egg, meat, and vegetables in the bottom right; and various kitchen tools like a whisk, a knife, and a rolling pin scattered around. There are also some decorative swirls.

# Further Implementations

- Event suggestions based on past activity
- Social networking features - users can create groups, check activity feeds
- Loyalty programs
- Include details about different Boston neighborhoods and their unique cuisines

**THANK  
YOU**

