University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 1

Date: 01 - 10 - 2024

Group Number: 60

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Jennifer Tanojo	40026460	b7a5y	jennifer.olive.tan@gmail.com
Jennifer Wang	72344476	f9p0l	jennifer26wang@gmail.com
Christopher Lew	87799201	a8t6b	christopherlew4@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Milestone 1 - Project Proposal, Conceptual Design: ER Diagram

A brief project description:

a) Domain of the application

The domain of the application is food journaling and restaurant interaction, combining elements of personal food documentation, restaurant review and waitlist platforms, and online food services.

b) Aspects of the domain modeled by the database

The application's goal is to serve as a comprehensive food experience platform, where users can not only document their food experiences but also interact with other users and restaurants in various ways. This project is best understood as a personal food journaling and interaction platform, where users can not only document and share their dining experiences but also have restaurant-related activities in ways that enhance their journaling: leaving reviews, creating a food bucket list, ordering food online, and joining a waitlist for dine-in.

From the ER diagram, we represented seven general entities and one weak entity:

1. **Journal**: we chose the primary key to be the journal's ID that we assign uniquely for each journal that is published in the application.
2. **Media**: we chose the primary key to be its file name, such that there are no duplicate files in the database.
3. **Account**: we chose the primary key to be the account ID that we also assign uniquely for each account that is created in the application.
4. **Waitlist**: we chose the primary key to be the waitlist ID that we assign uniquely for each waitlist that is associated with its corresponding restaurant.
5. **Review**: we chose the primary key to the review ID that is also assigned uniquely for each review that is made towards restaurants in the application.
6. **Order**: we chose the primary key to be the order ID, which is assigned uniquely for each order that is made through the application.
7. **Restaurant**: we chose the primary key to be the Restaurant's name and location, to make sure that there are no duplicates (i.e. a restaurant can have the same name but it may be a branch at another location, etc.).
8. **Staff (Weak Entity)**: staff will have a staff ID for identification, we have it as a weak entity so that high-end restaurants may want to feature certain chefs or star staff so they can be highlighted on a Restaurant's page to boost their reputation.

And nine general relationships:

1. Contains – Account contains Journal with a 1-to-many relationship. Journal has a total participation constraint with Account, where if a Journal exists, an Account must be linked to it.
2. Holds – Journal can hold multiple Media and Review. Journal has a 1-to-many with both Media and Review entities.

3. Friends – Account entities can befriend one another to promote interaction in the application, with a many-to-many relationship.
4. Makes – Account entities can make multiple Reviews with a 1-to-many relationship. The Review entity has a total participation constraint with the Account entity, where an Account must be linked to any existing Review entities.
5. Rates – The Review entity mentions the ratings and comments made on a Restaurant entity with a many-to-1 relationship.
6. Joins – Many Accounts can join a Waitlist. Waitlist has a 1-to-many relationship with Account with a participation constraint where many Accounts can join a Waitlist
7. Creates – Account can create an Order. Account has a 1-to-many relationship with Order and a participation constraint; an Account can make multiple Orders and these Orders are linked back to the Account.
8. Handles – A Restaurant handles its Waitlist through a 1-to-1 relationship. Each waitlist is linked to its restaurant and vice versa. There's no participation constraint because a Restaurant doesn't need to have a Waitlist.
9. Fulfills – A Restaurant fulfills Orders with a 1-to-many relationship and a participation constraint. Many Orders can link to the same Restaurant and an Order needs to connect to a Restaurant to be valid.
10. Has (Weak Entity) – Staff can belong to only one restaurant at a time.

Database specifications:

a. Database functionality:

The database will allow users to track their dining experiences through an account. The user can record meals, assign ratings and reviews, and reflect on their food habits–for their own records or serve as recommendations for others with similar palates as a reference. The users can also create and manage their food journals (including reviews and media), view others' journals, place online "order-ahead" orders, join restaurant waitlists, manage their personal account information, view restaurant descriptions, and track their own food-related goals.

However, the developers/managers of the database can access all entities and attributes, instead of restricted access of the users, in which they can only view their personal information in their account but not other users' account. Another one we can point out is the Waitlist entity where the users can view only their information and position in the waitlist, but not other users' information. Similarly, for the Order entity, the developers/managers of the database also have more access compared to the restricted access of the users wherein they can view their orders but not other users'.

Description of the application platform:

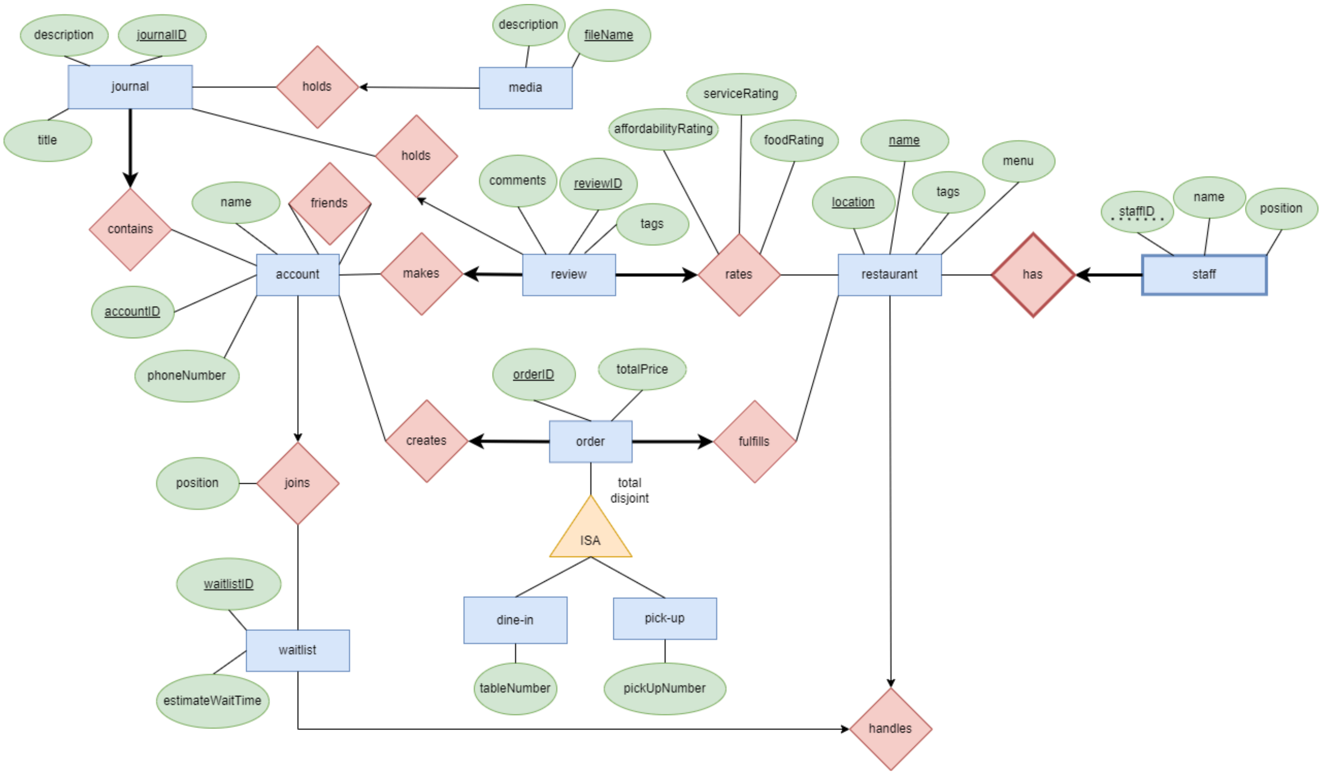
a. Database for our project:

Our project will use the department-provided Oracle server, with the provided details here (taken from the milestone 1 specifications' pdf file). The database will store the application's data (all of them, some that can serve as examples are the users' accounts/profiles, journals, restaurants, waitlists, and order details).

b. Expected application technology stack:

At the moment, we are expecting to use SQL, JavaScript, and/or PHP as our technology stack for our application. For version control, we can use Git to track changes in code and collaborate with team members. GitHub will be used for repository hosting and collaboration.

ER diagram for the database that our application will use:



Other comments:

To reaffirm that project differs from the blacklisted topics, here are the differences that we have compiled in a comprehensive list with explanations:

- Focus on Food Experiences/Journaling

The core functionality of our project is centered around personal food experiences and journaling, creating a space for users to reflect on, track, and share their dining experiences. Retail systems do not focus on such personal reflection and interaction.

- Personalized Interaction
  Our project enables users to maintain and view personal food journals, leave reviews, and track their dining experiences. This is a lifestyle-driven approach that focuses on reflection and documentation not just transactional sales.
- No Transaction
  Unlike retail ordering databases/apps, our project does not include transactions within the app. All transactions are to be done at the restaurant and the customer's discretion, and our app provides a platform for them to connect.
- Social and Lifestyle Elements
  The project encourages users to interact with each other by sharing journal entries, following friends' food activities, and engaging in a shared food journey. The social and collaborative aspect is a significant departure from traditional user management systems, which are more transactional and activity-focused.
- Restaurant Waitlist and Dining Engagement
  Unlike simple retail ordering, our project introduces real-world restaurant interaction through waitlists for dine-in and reviewing restaurants, creating a more dynamic interaction with real-time restaurant services.