The Big Result
System Design Document

Team members
Amy Yao
Ananya Poddar
Ava Oveisi
Carlos Fei Huang
Fariha Fyrooz
Noor Nasri
Vishal Sahoo

Table of Contents

CRC Cards

Class Name: User	
Parent Class: *None* **Subclasses:** Customer, Professional, Admin	
Responsibilities: Knows their username and password. Knows their name, email, and status. Knows the image-link for their picture.	**Collaborators:** *None*

Class Name: Customer	
Parent Class: User **Subclasses:** *None*	
Responsibilities: Knows their location and price range.	**Collaborators:** *None*

Class Name: Professional	
Parent Class: User **Subclasses:** *None*	
Responsibilities: Knows their average rating. Knows their description. Knows their offered services and associated costs.	**Collaborators:** Service

Class Name: Admin	
Parent Class: User **Subclasses:** *None*	
Responsibilities: Knows all customers, professionals, bookings, and reviews.	**Collaborators:** Customer Professional Bookings Reviews

Class Name: Settings	
Parent Class: *None* **Subclasses:** *None*	
Responsibilities: Knows the app preferences of the user. Knows the billing data of the user.	**Collaborators:** User

Class Name: Calendar	
Parent Class: *None* **Subclasses:** *None*	
Responsibilities: Knows the schedule of the user.	**Collaborators:** User

Class Name: Service	
Parent Class: *None* **Subclasses:** *None*	
Responsibilities: Knows its name and description.	**Collaborators:** *None*

Class Name: Booking	
Parent Class: *None* **Subclasses:** *None*	
Responsibilities: Has an identifier. Knows booking time, date, location, price, and status. Knows the Service of the booking. Knows the Professional of the booking. Knows the Customer of the booking.	**Collaborators:** *Service* *Professional* *Customer*

Class Name: Review
Parent Class: *None*

Subclasses: *None*	
Responsibilities: Has an identifier. Knows rating and description of review. Knows image-links attached to the review. Knows the associated booking.	**Collaborators:** *Booking*

DAO CRC Cards

Note that we have a DAO for every class that requires database interactions, rather than a single DAO interface.

Class Name: UserDAO	
Parent Class: *None* **Subclasses:** CustomerDAO, ProfessionalDAO, AdminDAO	
Responsibilities: Get a user, given correct username and password.	**Collaborators:** User

Class Name: CustomersDAO	
Parent Class: UserDAO **Subclasses:** *None*	
Responsibilities: Get a customer, given correct username and password Create a customer, given username and password	**Collaborators:** Customer

Class Name: ProfessionalsDAO	
Parent Class: UserDAO **Subclasses:** *None*	
Responsibilities: Get a professional, given correct username	**Collaborators:** Professional

and password Create a professional, given username and password.	

Class Name: AdminDAO	
Parent Class: UserDAO **Subclasses:** *None*	
Responsibilities: Get an admin, given correct username and password. Remove any customer, professional, booking, or review, given that object.	**Collaborators:** Admin Customer Professional Bookings Reviews

Class Name: SettingsDAO	
Parent Class: *None* **Subclasses:** *None*	
Responsibilities: Get a user's settings, given that user.	**Collaborators:** Settings User

Class Name: CalendarDAO	
Parent Class: *None* **Subclasses:** *None*	
Responsibilities: Get a user's calendar, given that user. Modify a calendar, given a booking.	**Collaborators:** Calendar User Booking

Class Name: BookingsDAO	
Parent Class: *None* **Subclasses:** *None*	
Responsibilities:	**Collaborators:**

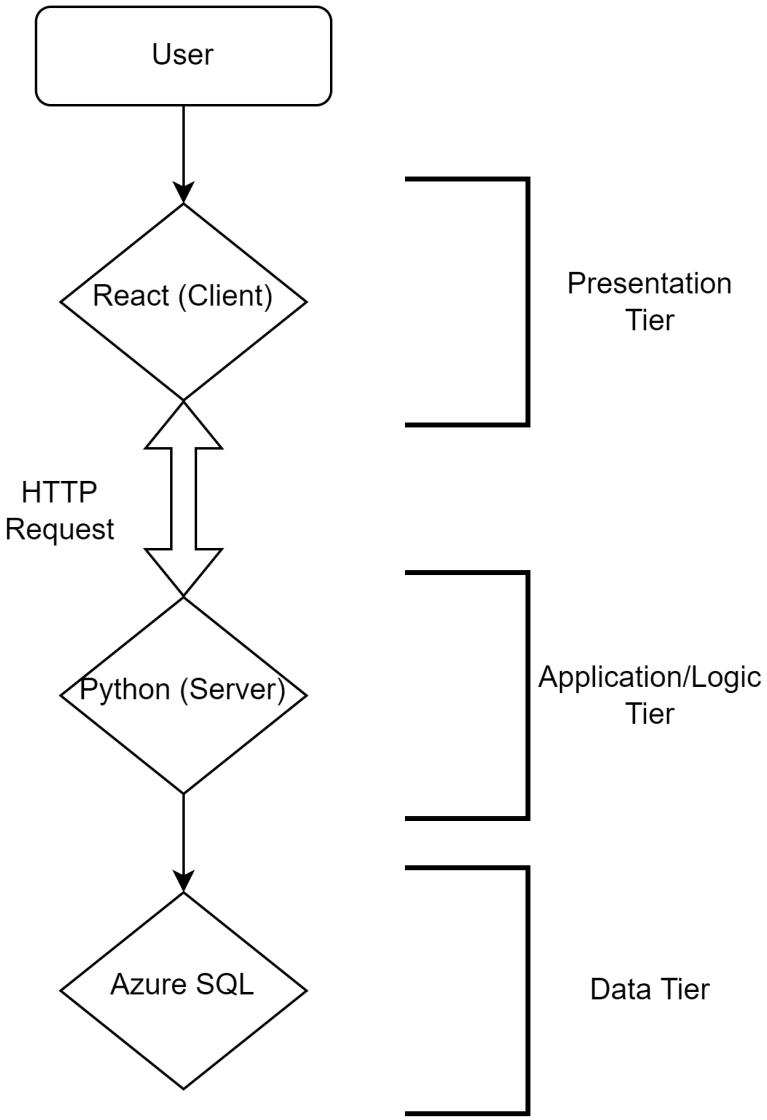
Get all bookings associated with a professional. Get all bookings associated with a customer. Get a booking, given its identifier. Modify status of a booking. Create a booking.	Booking Professional Customer

Class Name: ReviewDAO	
Parent Class: *None* **Subclasses:** *None*	
Responsibilities: Get all reviews associated with a professional. Get all reviews associated with a customer. Get a review, given its identifier. Post a review. Delete a review.	**Collaborators:** Review Professional Customer

Software Architecture

This project uses the **Three-Tier Architecture** - https://www.ibm.com/cloud/learn/three-tier-architecture. The user accesses the presentation tier, which is made with React, by opening the web app on a browser. The app will make HTTP requests to a flask server, which is the Application tier. This handles all logic, and acts as an internal firewall before accessing the Data tier, which uses Azure SQL. Since billing is involved in our application, the security of that data is very important. The project uses this design due to the disconnect between the client tier and the data tier, as opposed to the MVC structure.

Diagram



Three-Tier Architecture - https://www.ibm.com/cloud/learn/three-tier-architecture