Objective

Create a structured wireframe and define the data models for the backend of your full stack web application. The backend will be built using Python FastAPI, and this assignment will serve as a template for the implementation phase. The final design should be open api format. https://editor.swagger.io/Links to an external site. You should use this to create the endpoints. This will look like what is your final fast api application.

Part 1: Project Overview

Project Title: *Caffeine Compass*
Project Description: A social media website focusing on Cafe Restaurants in San Diego. Featuring anonymous review exchanges, profile customization and recommendations based on the user's personal taste. User's are able to post their own photos of their items and view other user's photos.

Part 2: Define the API Endpoints

Create a list of the primary API endpoints your backend will need. For each endpoint, include the following:

Endpoint URL: (e.g., /api/users, /api/products/{id})
HTTP Method: (GET, POST, PUT, DELETE)
Description: A brief description of the purpose of the endpoint.
Request Parameters: List any required and optional parameters.
Response Structure: Describe the structure of the response (e.g., JSON format) including success and error responses.

User:

Endpoint	HTTP Method	Description	Request Parameters	Response Structure
/api/users	GET	Retrieve a list of users	None	[{ id, name, email, ... }]
/api/users	POST	Create a new user	{ name, email, password }	{ success: true, user: {...} }
/api/users/{id}	PUT	Update user information	{ name, email }	{ success: true }
/api/users/{id}	DELETE	Delete a user	None	{ success: true }

Restaurant Page:

Endpoint	HTTP Method	Description	Request Parameters	Response Structure
/api/restaurant page	GET	Retrieve a list of restaurants	{restaurant name, address}	[{ address, restaurant name, owner's email, ... }]
/api/restaurant page	GET	Retrieve reviews of restaurant	{restaurant, address}	[{response, review number}]
/api/restaurant page	POST	Create a new restaurant page	None	{ success: true,restaurant : {...} }

/api/restaurant page/{id}	PUT	Update restaurant information	{ restaurant name, address }	{ success: true }
/api/restaurant page/{id}	DELETE	Delete a restaurant	None	{ success: true }

Comment Threads/Forums:

Endpoint	HTTP Method	Description	Request Parameters	Response Structure
/api/threads	GET	Retrieve a list of threads	None	[{ Thread ID, Thread Authors name}]
/api/threads	POST	Create a new thread	None	{ success: true, thread: {...} }
/api/threads /{id}	PUT	Update information	{ Thread Name, Thread id}	{ success: true }
/api/threads pageid}	DELETE	Delete thread	None	{ success: true }

Filtered Tags:

Endpoint	HTTP Method	Description	Request Parameters	Response Structure

/api/TagsPage	GET	Retrieve a list of tags	None	[{tags }]
/api/TagsPage/{id}	PUT	Update user tag preference	{tags }	{ success: true }
/api/TagsPage/{id}	DELETE	Delete a tag	None	{ success: true }

Cosmetic Store:

Endpoint	HTTP Method	Description	Request Parameters	Response Structure
/api/cosmetics	GET	Retrieve a list of cosmetics	None	[{ Cosmetic id, name }]
/api/cosmetics/{id}	PUT	Update profile cosmetics	{ Cosmetic id}	{ success: true }
/api/cosmetics/{id}	DELETE	Delete a cosmetic	{ Cosmetic id}	{ success: true }

Bookmarks:

Endpoint	HTTP Method	Description	Request Parameters	Response Structure
/api/bookmarks	GET	Retrieve a list of bookmark	None	[{ Restaurant id, name }]
/api/bookmarks/{id}	PUT	Update bookmark page	{ bookmark id}	{ success: true }

/api/bookmarks /{id}	DELETE	Delete a bookmark	{ Cosmetic id}	{ success: true }

Part 3: Data Models

Define the data models that your application will use. For each model, include:

Model Name: (e.g., User, Product)
Attributes/Fields: List the attributes of the model along with their data types and any constraints (e.g., required, unique).
Relationships: Describe any relationships between models (e.g., one-to-many, many-to-many).

User Model

Attributes:

- id: int, primary key, auto-increment
- name: str, required
- email: str, required, unique
- password: str, required
- created_at: datetime, default to current time
- bookmarks: string

Relationships:

- A user can have many cosmetics and comments

Restaurant Model

Attributes:

- id: int
- address: string, required
- name: string, required
- review/rating: int

- phoneNumber: string
- filtered_tags_id: int
- comment_id

Relationships: A restaurant can have many tags.

Address Model

Attributes:

- id: int, required
- street: string, required
- city: string, required
- state: string, required
- zip: string, required

Tags Model

Attributes:

- tags: str, unique
- id: int

Relationships:

- Posts/Restaurants/Comments can have many tags

Comments/Reviews Model

Attributes:

- Title: str, required
- comment_id: int, unique, auto-increment
- Created_at: datetime, default to time

Relationships: Reviews can have many comments

Cosmetic Model

Attributes:

- cosmetic_ID: png, unique
- name: string, unique
- Date_acquired: datetime, default to current time
-

Relationships: Many cosmetics can be applied to a user.

Bookmark Model

Attributes:

- ID:, unique
- name: string, unique
- Date_bookmarked: datetime, default to current time
- Restaurant_name: Restaurant Model{...}

Relationships: A user can have many bookmarks

Part 4: Database Schema

Based on the defined models, outline a basic schema for your database. Describe the tables and their relationships.

<u>Users Table</u>

Columns: user_id, name, email, password, created_at

Foreign Key: user_id references comment_id and cosmetic_id

<u>Restaurant Table</u>

Columns: restaurant_id, address, name, review/rating, phoneNumber

Foreign Key: restaurant_id references tag_id

<u>Address Table</u>

Columns: street, city, state, zip

<u>Comment Table</u>

Columns: Title, comment_id, created_at

Foreign Key: user_id references comment_id

<u>Filtered Tags Table</u>

Columns: tags_id, name

Foreign Key: posts_id references tags_id

<u>Cosmetic Table</u>

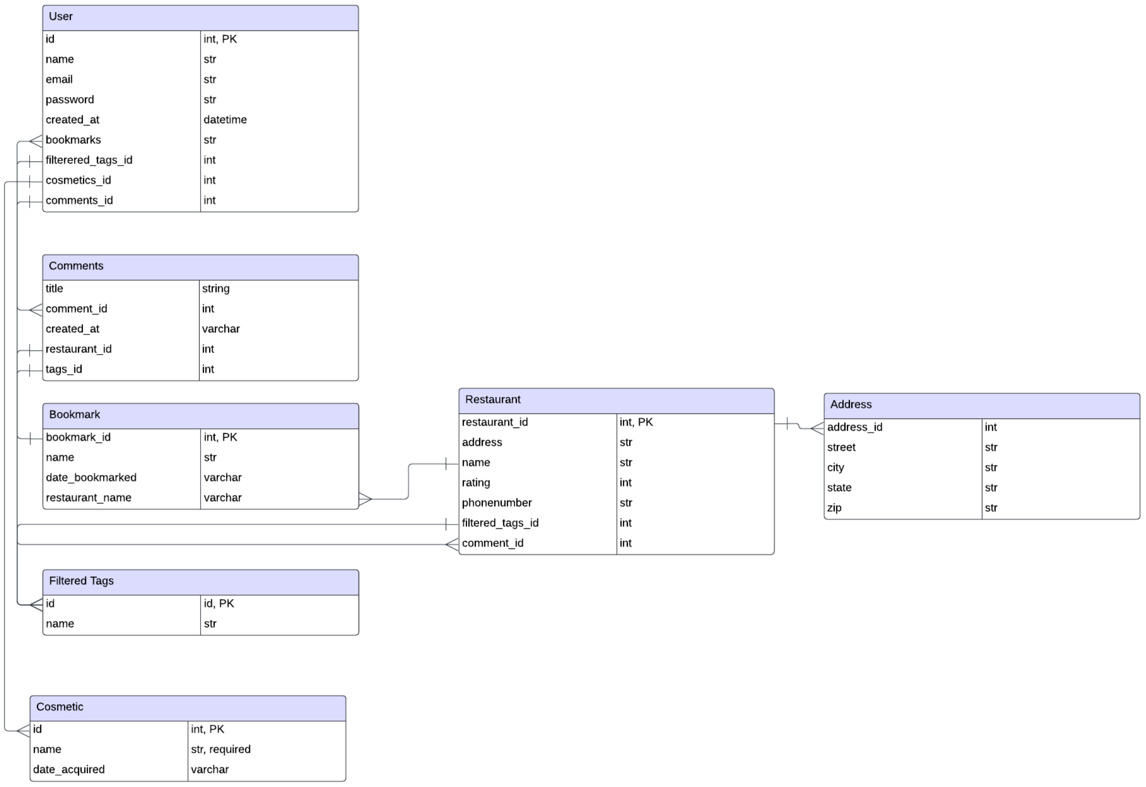
Columns: cosmetic_ID, name, Date_acquired

Foreign key: User_id references cosmetic_id

<u>Bookmarks Table</u>

Columns: bookmark_id, name, date_bookmarked, restaurant_name

Foreign Key: user_id references bookmark_id



User	
id	int, PK
name	str
email	str
password	str
created_at	datetime
bookmarks	str
filterered_tags_id	int
cosmetics_id	int
comments_id	int

Comments	
title	string
comment_id	int
created_at	varchar
restaurant_id	int
tags_id	int

Bookmark	
bookmark_id	int, PK
name	str
date_bookmarked	varchar
restaurant_name	varchar

Restaurant	
restaurant_id	int, PK
address	str
name	str
rating	int
phonenumber	str
filtered_tags_id	int
comment_id	int

Address	
address_id	int
street	str
city	str
state	str
zip	str

Filtered Tags	
id	id, PK
name	str

Cosmetic	
id	int, PK
name	str, required
date_acquired	varchar

Part 5: Additional Considerations

Authentication: Describe the authentication method you will use (e.g., JWT, OAuth).

Middleware: Outline any middleware you plan to implement (e.g., CORS, logging).

Error Handling: Provide a brief overview of how you will handle errors and what standard responses will look like.
Testing: Mention how you plan to test the API (e.g., using Postman, automated tests).

Submission Guidelines

Compile your wireframe and model definitions into a single document. Use clear headings and bullet points for easy readability. Submit your assignment in PDF or Markdown format by the specified deadline.

Attach the yaml file for swagger editor.

Grading Criteria

- Completeness of API endpoint definitions
- Clarity and accuracy of data models
- Logical structure of the database schema
- Consideration of authentication, middleware, and error handling
- Overall organization and presentation of the document

Good luck! This assignment will set the foundation for your backend development using FastAPI.

Features:

- Profile Creation
- Restaurant Pages
- Comment Threads
- San Diego based cafe
- Reviews

CaffineCompass

Social Media	**Cafe Aspects**
- Profile Creation	- Restaurant Pages
● Comment Threads	● San Diego based cafe
- Reviews/ Likes system	- Filters/ Tags of interest
- Posts/Image Sharing	- Anonymity?
- Cosmetics? Points based by post	● Map/ Map radius
- Themes	