

Database Project

Concept Paper

Group (23)

|  |  |
| --- | --- |
| Name | Student Number |
| Mohammad KANDAKJI | 3141578 |
| Omar Kotb | 3144713 |
| Moamen Hatem | 3168571 |
| Mohamed Abdelbar | 3144718 |

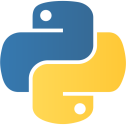
* Participants:



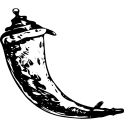
Software

**Architecture**

* Backend
* Frontend
* Database
* Stack Diagram
* **Backend**
* **Language/Framework:** Python with Flask



Python is a versatile language with a clean syntax, making it ideal for rapid development. It has a strong ecosystem of libraries for a variety of tasks.



Flask is a lightweight WSGI web application framework. It's designed to make getting started quick and easy, with the ability to scale up to complex applications.

* **Tasks Covered:**
* Request Handling: Flask will serve as the server that handles HTTP requests from the frontend.
* Business Logic: Python code will process data, implement the business rules of the application, handle authentication, and manage user sessions.
* API Development: Flask will be used to create RESTful APIs that the frontend can use to send or retrieve data.
* Database Interaction: Flask will interact with SQLite using an ORM like SQLAlchemy. This layer will handle all CRUD (Create, Read, Update, Delete) operations with the database.
* **Frontend**
* **Technologies:** HTML, CSS, JavaScript

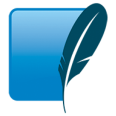


HTML provides the structure for the web pages.

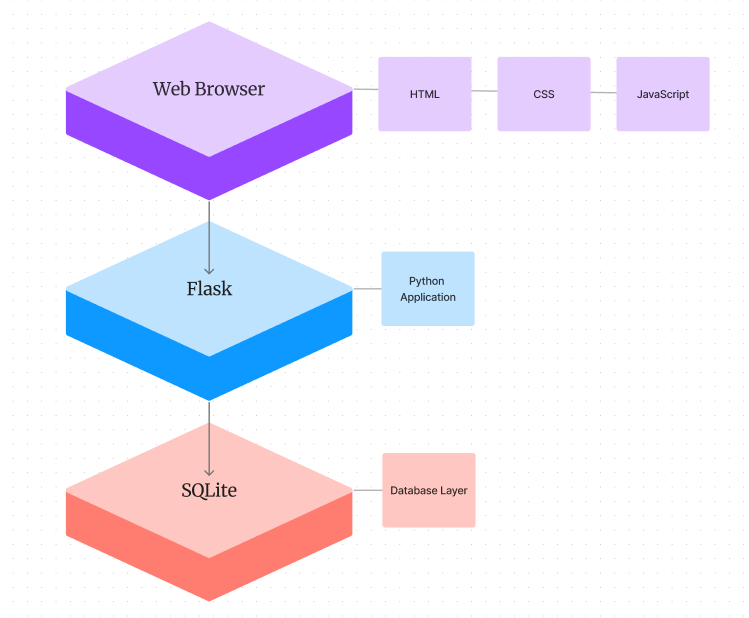
أوراق الأنماط المتتالية - ويكيبيدياCSS is used for styling and ensuring the application is visually appealing and provides a good user experience.

JavaScript allows for dynamic interactions on the frontend. It will handle asynchronous requests to the backend (AJAX), form validations, and manipulation of the DOM to update the user interface without needing to refresh the page.

* **Tasks Covered:**
* User Interface Rendering: Displaying all the web pages and UI components that the user interacts with.
* Client-Side Logic: Handling user input and interactions, data validation before sending it to the backend, and controlling navigation between different parts of the application.
* **Database**
* **Database Management System:** SQLite



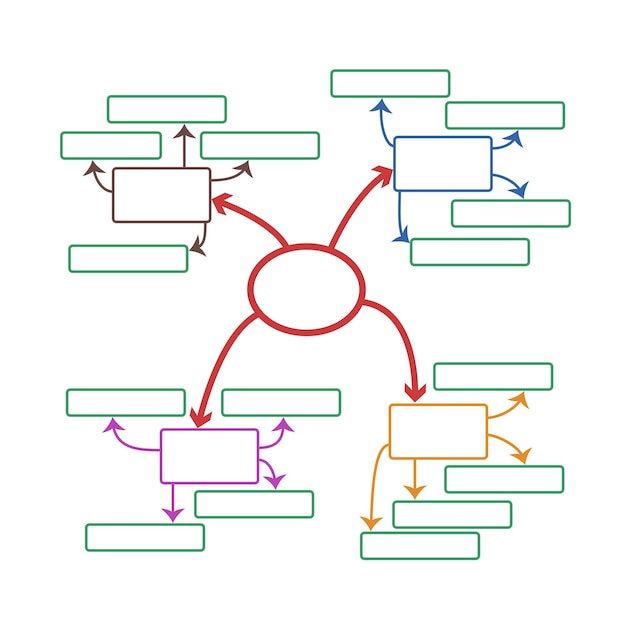
**SQL**ite is a C library that provides a lightweight disk-based database and is capable of handling lower to medium traffic HTTP requests.

* **Tasks Covered:**
* Data Storage: Storing all application data, including user accounts, restaurant details, menu items, and orders.
* Data Retrieval: Facilitating queries to fetch data based on user requests.
* Transaction Management: Ensuring data integrity through ACID (Atomicity, Consistency, Isolation, Durability) transactions.
* **Stack Diagram**
* In this diagram:

Web Browser: This is the layer that users interact with. It includes all the front-end technologies that run in the user's browser – HTML for markup, CSS for styling, and JavaScript for interactivity and making asynchronous calls to the server.

Flask: Representing the backend or server-side layer where the Flask application runs. It processes incoming HTTP requests, interacts with the database, and sends back responses to the client.

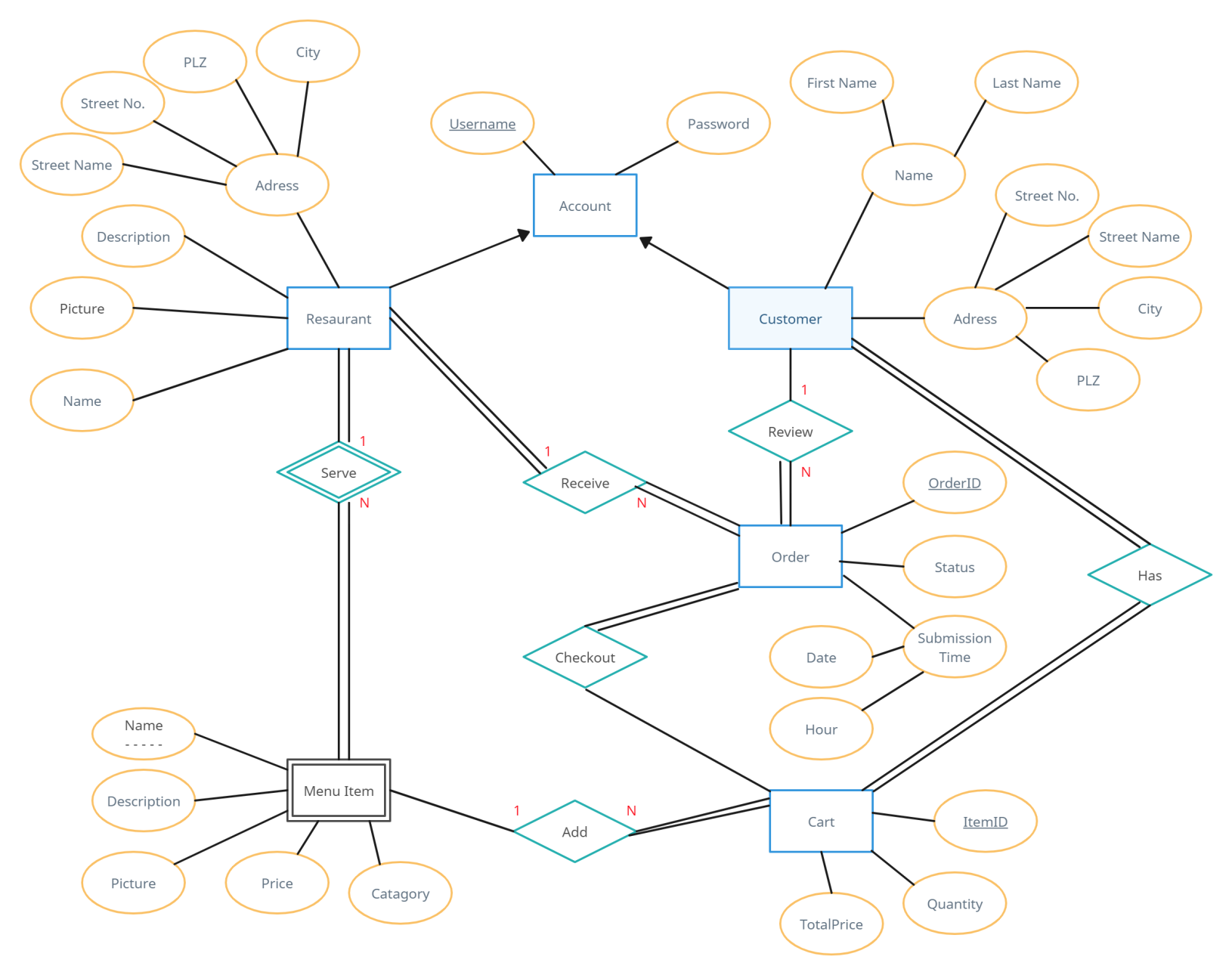
SQLite: The database layer where data is stored and managed. Flask communicates with this layer to perform CRUD operations.



**ER-**

Diagram

* **Entity-Relationship Diagram**
* for **Lieferspatz**

Our Entity-Relationship (ER) diagram represents the structured data model of the Lieferspatz. The diagram encapsulates the essential entities, their relationships, and attributes that define the core functionality of our system.



**UI**

Mockups

* Website Design

* UI Mockups (Wireframes)
* **Technologies:** Balsamiq

The user interface (UI) mockups for the Lieferspatz project have been meticulously designed to provide an effective user experience. Developed with Balsamiq. Our mockups serve as the visual blueprint for the application's interface, detailing the layout and interaction flow of the platform.

