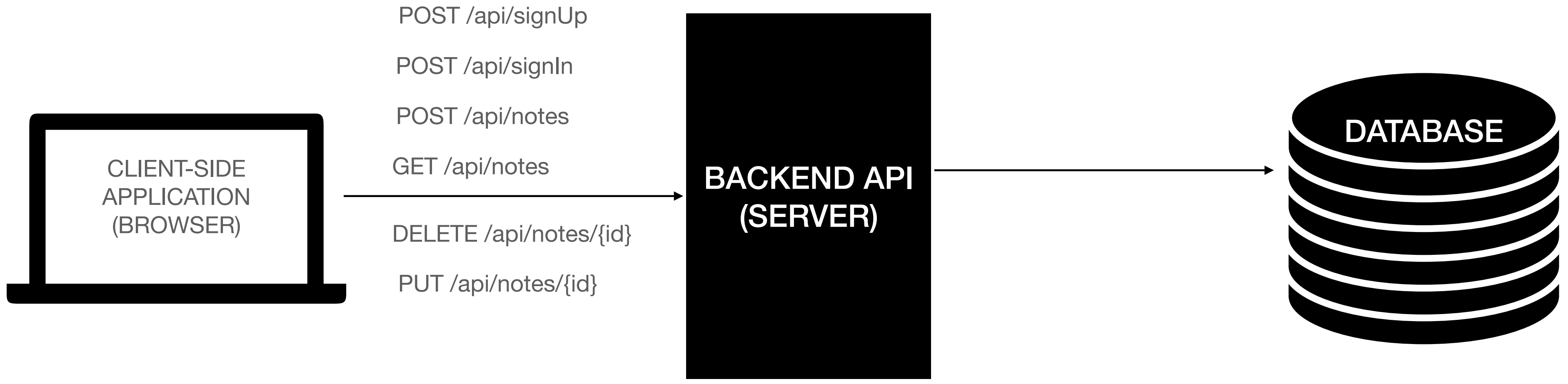
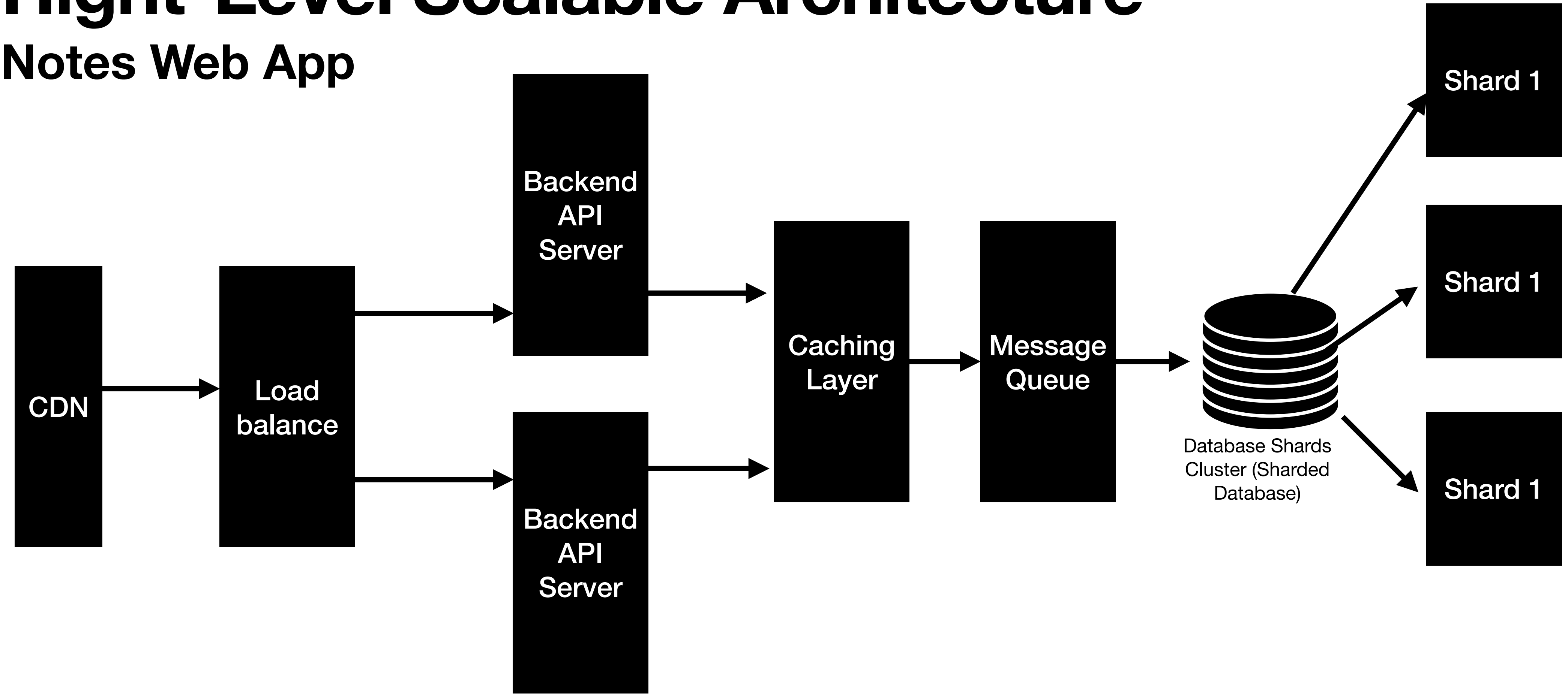
1. Hight-Level Design

Note Web App



Hight-Level Scalable Architecture

Notes Web App



2. Web App UI

Notes Web App

3. Data Model

User Model

Before diving into the note model, we need a user model since the application requires login and signup functionalities. Each user will have unique notes associated with them.

- **id**: Unique identifier for the user (UUID).
- **username**: The user's chosen username.
- **email**: The user's email address.
- **passwordHash**: Hashed password for security.
- **createdAt**: Timestamp when the user was created.
- **updatedAt**: Timestamp when the user information was last updated.

3. Data Model

Database Schema

Column	Type	Constraints
id	INT	Primary Key/Auto Increment
username	Varchar(100)	Unique, Not Null
email	Varchar(80)	Unique, Not Null
passwordHash	Varchar(100)	Not Null
createdAt	Timestamp	Not Null
updatedAt	Timestamp	Not Null

3. Data Model

User Model

JSON example:

```
{
  "id": "user-1234",
  "username": "johndoe",
  "email": "johndoe@example.com",
  "passwordHash": "hashedpassword",
  "createdAt": "2024-01-01T12:00:00Z",
  "updatedAt": "2024-01-01T12:00:00Z"
}
```

3. Data Model

Note Model

A note will be associated with a user and will contain the following properties:

Note Model Properties:

- **id**: Unique identifier for the note (UUID).
- **userId**: Identifier for the user who owns the note (foreign key reference to the User model).
- **title**: Title of the note (optional but useful for better organization).
- **content**: The text content of the note.
- Status: The status of Note ("in progress", “done”, “archived")
- **createdAt**: Timestamp when the note was created.
- **updatedAt**: Timestamp when the note was last updated.

3. Data Model

Note Database Schema

Column	Type	Constraints
id	INT	Primary Key/Auto Increment
userId	INT	Foreign Key References User (id)
title	Varchar(50)	Null
content	Text(500)	Not Null
status	Enum(“in progress”,” done”."archived")	Not Null
createdAt	Timestamp	NotNull
UpdatedAt	Timestamp	Not NUll

3. Data Model

Note Model

JSON example:

```
{
  "id": "note-5678",
  "userId": "user-1234",
  "title": "Meeting Notes”,
  “status”: “in progress”,
  "content": "Discuss the project roadmap and milestones.",
  "createdAt": "2024-06-18T12:00:00Z",
  "updatedAt": "2024-06-18T12:00:00Z"
}
```

4. RESTful API

Notes Web App

POST /api/signin: Provides access token to user
- ○ Response: 200 OK with a JSON array of notes.

POST /api/signup: Register a new User
- ○ Response: 200 OK with a JSON array of notes.

GET /api/notes: Retrieve a list of notes for the authenticated user.
- ○ Response: 200 OK with a JSON array of notes.

POST /api/notes: Save a new note.
Request Body: JSON object with content.
Response: 201 Created with the created note object.

- **DELETE /api/notes/{id}**: Delete a note by ID.
    - Response: 204 No Content.

- **PUT /api/notes/{id}**: Update a note by ID.
  Request Body: JSON object with content.
  Response: 201 Created with the created note object.

5. Web Server

Business Logic

- Ensure that each note operation (create, retrieve, delete) is performed only by the authenticated user to maintain data security and integrity.
- All endpoints needs to send in the header the authentication token to validade each request.
- Validate note content before saving to ensure it is not empty.

Data Persistence

- Use a relational database (e.g., PostgreSQL) to store notes, with a schema designed to store note data and relationships between users and their notes.