Title: User Login Sequence Diagram

User -> BlogWebsite: Request Homepage

BlogWebsite -> User: Provide Homepage

User -> BlogWebsite: Clicks "Login" Button

BlogWebsite -> User: Provide Login Form

User -> BlogWebsite: Submit Username and Password

BlogWebsite -> AuthenticationSystem: Verify Credentials

activate AuthenticationSystem

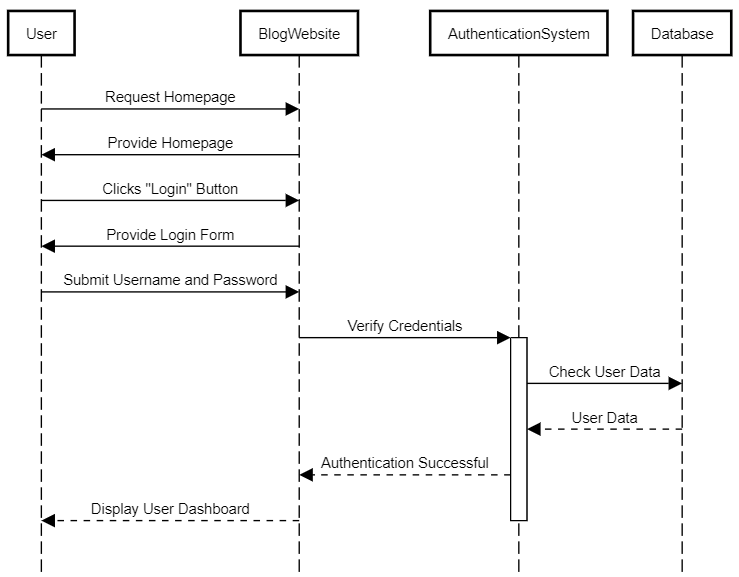
AuthenticationSystem -> Database: Check User Data

Database --> AuthenticationSystem: User Data

AuthenticationSystem --> BlogWebsite: Authentication Successful

BlogWebsite --> User: Display User Dashboard

deactivate AuthenticationSystem



Title: User Comment Submission Sequence Diagram

User -> BlogPost: View Blog Post

User -> BlogPost: Enters Comment

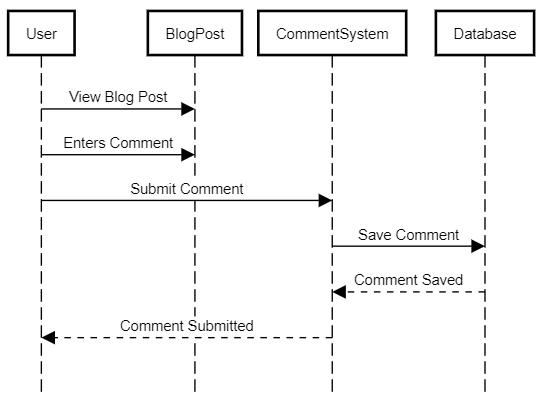
User -> CommentSystem: Submit Comment

CommentSystem -> Database: Save Comment

Database --> CommentSystem: Comment Saved

CommentSystem --> User: Comment Submitted

deactivate CommentSystem



Designing a schema for a blog website using a PostgreSQL database involves defining tables and relationships that ensure efficient querying and data management. Here's an example schema with optimizations in mind:

-- Users Table

CREATE TABLE Users (

user\_id SERIAL PRIMARY KEY,

username VARCHAR(50) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

password\_hash VARCHAR(128) NOT NULL,

date\_registered TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Posts Table

CREATE TABLE Posts (

post\_id SERIAL PRIMARY KEY,

title VARCHAR(255) NOT NULL,

content TEXT NOT NULL,

publish\_date TIMESTAMPTZ DEFAULT CURRENT\_TIMESTAMP,

author\_id INT REFERENCES Users(user\_id)

);

-- Categories Table

CREATE TABLE Categories (

category\_id SERIAL PRIMARY KEY,

name VARCHAR(50) NOT NULL

);

-- Post\_Category Junction Table

CREATE TABLE Post\_Category (

post\_id INT REFERENCES Posts(post\_id),

category\_id INT REFERENCES Categories(category\_id),

PRIMARY KEY (post\_id, category\_id)

);

-- Indexes

CREATE INDEX idx\_posts\_author ON Posts(author\_id);

CREATE INDEX idx\_post\_category\_post\_id ON Post\_Category(post\_id);

CREATE INDEX idx\_post\_category\_category\_id ON Post\_Category(category\_id);

In this schema:

* SERIAL is used to automatically generate unique IDs.
* TIMESTAMPTZ is used for timestamp with time zone data type.
* UNIQUE constraint is applied to the email field to ensure uniqueness.
* Foreign key relationships maintain data integrity.
* Indexes are added to optimize common queries.

The created indexes help improve query performance by allowing the database to quickly find relevant data. However, the choice of indexes should be based on the specific queries you plan to perform most frequently.

Remember, optimization can be more intricate depending on the scale of your application, traffic, and usage patterns. As your application grows, you might need to consider other optimizations like partitioning, caching, and denormalization.

Before implementing this schema, ensure you thoroughly test it with your expected workloads to ensure it meets your performance requirements.