Schema Design and Indexing in MongoDB

Part 1: Schema Design

Design the schema for the following collections:

- Users: Each user has a name, email, and a list of blog posts they have written.
- Posts: Each post has a title, content, author (reference to the user), comments, and tags.
- Comments: Each comment has a user_id (who made the comment), text, and a timestamp.
- Tags: Each tag has a name and can be associated with multiple blog posts.

  Questions to Consider:

  – Should comments be embedded within the posts, or stored as a separate collection?

  – Should tags be referenced or embedded within the posts?

Part 2: Implement the Schema

- Write Python code using PyMongo to implement the schema and insert sample data. Below is a starting point for the users and posts collections. You will need to complete the schema for comments and tags.

```
from pymongo import MongoClient

# Connect to MongoDB
client = MongoClient("mongodb+srv://<username>:<password>@cluster0.mongodb.net/test
db = client['blog_platform']

# Insert sample users
users = db['users']
users.insert_many([
    {"name": "Alice", "email": "alice@example.com"},
    {"name": "Bob", "email": "bob@example.com"}
])

# Insert sample posts with comments and tags (decide whether to embed or reference)
posts = db['posts']
posts.insert_many([
    {
        "title": "How to Use MongoDB",
        "content": "This is a guide to using MongoDB.",
```

```
        "author": "Alice",
        "comments": [
            {"user_id": "Bob", "text": "Great post!", "timestamp": "2024-09-12T10:0
        ],
        "tags": ["MongoDB", "Database"]
    }
])
```

- Add comments and tags based on your design choice (embedded or referenced).

Part 3: Indexing for Performance

- Query Optimization: Write a query to fetch all posts by a specific author and optimize the query using an index.
- Query Comments: Write a query to find all comments made by a specific user and create an appropriate index to improve performance.

Part 4: Refactoring for Performance

- Consider a scenario where the number of comments per post grows large. Would you still embed comments or reference them in a separate collection? Refactor your schema accordingly.
- Create an index on the comments collection to optimize fetching comments by user_id.

Part 5: Submit Your Solution

- Submit a Jupyter notebook containing your MongoDB schema, queries, and indexing solutions.
- Include comments in your code explaining why you chose to embed or reference data.
- Test the performance of queries with and without indexes and explain the differences in query times.