Based on the provided image, the assignment appears to involve building a RESTful API for a blogging application with features like CRUD operations for posts, user accounts, comments, and possibly voting. Here's a brainstorming session with ideas for projects/services your team can implement:

---

### \*\*1. Multi-Role Blogging Platform\*\*

\*\*Services:\*\*

1. \*\* API Gateway \*\*

2. \*\***User** Management Service\*\*

- \*\*Features\*\*: User management (CRUD), login/logout, roles (admin, user), messaging to Post service.

- \*\*Technologies\*\*: Spring Security, JWT for authentication, Hibernate, Kafka.

3. \*\***Post** Management Service\*\*

- \*\*Features\*\*: CRUD operations for posts (create, edit, delete, read), ~~tagging, and category assignment~~, subscribing User service’s messages.

- \*\*Technologies\*\*: Hibernate, JPA for database interaction, Kafka.

To maximize your grade based on the provided grading rubric and the 1st project idea (Multi-Role Blogging Platform), here's how you can enhance each aspect:

---

### \*\*Project Management (15%)\*\*

- \*\*Maximize Grade:\*\*

1. \*\*Clear Roles\*\*: Divide responsibilities explicitly:

- One team member handles User Management and Authentication. (**Tin**)

- The other manages Post Management and Gateway API. (**Son**)

~~2. \*\*Frequent Updates\*\*: Use tools like Trello or Jira to track progress and share updates with your instructor weekly.~~

---

### \*\*Hibernate (25%)\*\*

- \*\*Maximize Grade:\*\*

1. \*\*Advanced Mapping\*\*: Use advanced Hibernate features such as:

- \*\*Inheritance mapping\*\* for user roles (e.g., admin, author, reader).

- \*\*Collections mapping\*\* for comments and votes.

2. \*\*Custom Queries\*\*:

- Use \*\*JPQL\*\* and \*\*Criteria API\*\* for complex queries (e.g., fetching top-voted posts or comments).

- Implement pagination with `Query` objects.

3. \*\*Cascading\*\*: Use cascading for related entities (e.g., deleting a post deletes its comments and votes).

4. \*\*Caching\*\*:

- Add a second-level cache (e.g., Ehcache) to improve performance for frequently accessed entities like posts and users.

---

### \*\*Spring (25%)\*\*

- \*\*Maximize Grade:\*\*

1. \*\*Spring Features\*\*:

- \*\*Transactions\*\*: Implement transactional methods for operations that involve multiple entities, such as posting with comments.

- \*\*Security\*\*: Add \*\*Spring Security\*\* with JWT for token-based authentication and role-based authorization.

- \*\*Validation\*\*: Use `@Valid` and custom validators for inputs like usernames and post content.

2. \*\*Spring Testing\*\*:

- Write unit tests for services and controllers using \*\*Spring Boot Test\*\* and \*\*MockMvc\*\*.

- Add integration tests for REST APIs.

3. \*\*Internationalization (I18N)\*\*:

- Support multiple languages for error messages and UI text using `MessageSource`.

---

### \*\*Integration (25%)\*\*

- \*\*Maximize Grade:\*\*

1. \*\*Microservices Architecture\*\*:

- Break the project into distinct services (User, Post, Comment/Vote) and connect them via REST APIs.

- Use tools like Postman for testing API integration.

2. \*\*API Gateway\*\*:

- Implement a \*\*gateway service\*\* (e.g., Spring Cloud Gateway) for routing client requests to respective services.

3. \*\*Event-Driven Communication\*\*:

- Use \*\*RabbitMQ\*\* or \*\***Kafka**\*\* for inter-service communication (e.g., a new user -> notify Post service to update database, ~~notifying users of new comments or votes~~).

4. \*\*Documentation\*\*:

- Use \*\*Swagger/OpenAPI\*\* to document your APIs.

---

### \*\*Presentation Delivery (10%)\*\*

- \*\*Maximize Grade:\*\*

1. \*\*Clarity\*\*: Prepare a concise walkthrough of each service and demonstrate its working via APIs.

2. \*\*Visualization\*\*:

- Use tools like Lucidchart for diagrams showcasing architecture.

- Demonstrate APIs live with Postman or a frontend client.

3. \*\*Engagement\*\*: Highlight innovative features (e.g., second-level caching or I18N).

---

Adding features like advanced Hibernate mappings, Spring Security, microservices with an API gateway, and event-driven communication will align your project with the rubric's "Excellent" (A) standards. Let me know if you'd like more details or guidance on any of these features!