SMOGON FORUM MAIN FEATURES

Here are the main features the forums should have :

1. **User accounts:** Users should be able to create and manage their own accounts, including setting up profiles and updating their personal information.

2. **Threads and posts:** The core of any forum is the ability to create threads (topics) and make posts (replies) within those threads. Users should be able to create new threads and make posts, as well as edit or delete their own posts.

3. **Search:** Users should be able to search for specific threads or posts within the forum.

4. **Moderation:** There should be tools in place for moderators to manage the content of the forum, including the ability to delete or edit posts, ban users, and create rules for the forum.

5. **Notifications:** Users should be able to receive notifications when someone responds to one of their posts or mentions them in a thread.

6. **User profiles:** Each user should have their own profile page that displays their activity on the forum, including the threads and posts that they have created.

7. **Image and file attachments:** Users should be able to attach images and files to their posts.

8. **Team creation :** Users should be able to publish team composition from team text form

9. **Rate Posts :** Users should be able to upvote posts

SMOGON FORUM MAIN ROUTES

Here is an exhaustive list of the routes for the API :

1. User :

```
GET /api/users                          # Retrieves a list of all users
GET /api/users/:id                      # Retrieves the user with by ID
POST /api/users                         # Creates a new user
PUT /api/users/:id                      # Updates an existing user
DELETE /api/users/:id                   # Deletes a user

POST /api/users/:id/login               # Logs a user in

GET /api/users/:id/threads              # Retrieves all threads by user ID
GET /api/users/:id/posts                # Retrieves all posts created by user ID
PUT /api/users/:user_id/roles/:role_id  # Assoc User ID by role ID
```

2. Roles :

```
GET /api/roles                          # Retrieves a list of all roles
GET /api/roles/:id                      # Retrieves the role by ID
POST /api/roles                         # Creates a new role
PUT /api/roles/:id                      # Updates an existing role
DELETE /api/roles/:id                   # Deletes a role

GET /api/roles/:id/users                # Retrieves all users by role ID
```

3. Forums :

```
GET /api/forums                         # Retrieves a list of all forums
GET /api/forums/:id                     # Retrieves the forum by  ID
POST /api/forums                        # Creates a new forum
PUT /api/forums/:id                     # Updates an existing forum
DELETE /api/forums/:id                  # Deletes a forum

GET /api/forums/:id/subforums           # Get all subforums by forum ID
GET /api/forums/:id/threads             # Get all threads by forum ID
```

4. Subforums

   GET /api/subforums **# Retrieves a list of all subforums**
   GET /api/subforums/:id **# Retrieves the subforum by ID**
   POST /api/subforums **# Creates a new subforum**
   PUT /api/subforums/:id **# Updates an existing subforum**
   DELETE /api/subforums/:id **# Deletes a subforum**

   GET /api/subforums/:id/threads **# Retrieves all threads by subf ID**

5. Threads :

   GET /api/threads **# Retrieves a list of all threads**
   GET /api/threads/:id **# Retrieves the thread by ID**
   POST /api/threads **# Creates a new thread**
   PUT /api/threads/:id **# Updates an existing thread**
   DELETE /api/threads/:id **# Deletes a thread**

   GET /api/threads/:id/posts **# Retrieves a list of all posts of a thread**
   GET /api/threads/:id/likes **# Retrieves a list of all likes of a thread**

6. Posts :

   GET /api/posts **# Retrieves a list of all posts**
   GET /api/posts/:id **# Retrieves the post by ID**
   POST /api/posts **# Creates a new post**
   PUT /api/posts/:id **# Updates an existing post**
   DELETE /api/posts/:id **# Deletes a post**

   GET /api/posts/:id/likes **# Retrieves a list of all likes of a post**
   POST /api/posts/:id/likes **# Allows a user to like a post**
   DELETE /api/posts/:id/likes/:like_id **# Allows a user to unlike a post**

7. Likes :

GET /api/likes/post/:post_id	**# Retrieves all likes by post ID**
GET /api/likes/thread/:thread_id	**# Retrieves all likes by thread ID**
GET /api/likes/:id	**# Retrieves the like by ID**
POST /api/likes	**# Creates a like**
DELETE /api/likes/:id	**# Deletes a like**

8. BannedUsers :

GET /bannedusers	**#Retrieves all banned users**
GET /bannedusers/:id	**#Retrieves a single banned user by ID**
POST /bannedusers	**#Creates a new banned user**
PUT /bannedusers/:id	**#Updates an existing banned user**
DELETE /bannedusers/:id	**#Deletes an existing banned user**

SMOGON FORUM TECHNICAL ASPECT

Using C# and .NET 6.0 as the backend for a mobile application developed with React Native can be a good choice for several reasons :

1. Cross-platform compatibility: C# and .NET are known for their cross-platform compatibility. By using C# as the programming language for the backend, you can easily build and run your application on multiple platforms, including Windows, macOS, and Linux.

2. Performance: C# and .NET are known for their performance and scalability. The runtime and framework are optimized to provide fast and efficient performance, which can be beneficial for a mobile application that needs to handle a high volume of requests and data.

3. Productivity: C# is an expressive and powerful programming language that allows developers to write clean, readable, and maintainable code. The .NET framework provides a rich set of libraries and tools that can help increase developer productivity. The combination of these makes it easy for developers to quickly develop, test, and deploy features for your application.

4. Large developer community: C# and .NET have a large and active developer community, which means that you will have access to a vast amount of resources, tutorials, and libraries that can help speed up development and reduce the learning curve.

5. .Net 6.0 : The latest version of .net provides with a set of new features such as Blazor WebAssembly, Single file applications, improved performance and more. With new versions, it is easier to implement new functionalities, improve the existing ones, and take advantages of new technologies, which can help you to create more innovative and attractive apps that meet the latest trends and user expectations.

Using React Native as the frontend for an API developed with C# and .NET 6.0 can be a good choice for several reasons :

1. Cross-platform compatibility: React Native is a framework for building mobile applications using JavaScript and React. By using React Native, you can develop applications that can run on multiple platforms, including iOS and Android, without the need to write separate code for each platform.

2. Performance: React Native provides a high-performance experience by utilizing native components and APIs, which allows you to achieve near-native performance. This is particularly beneficial for mobile applications that require a lot of user interaction and need to respond quickly to user input.

3. Productivity: React Native allows developers to build rich and responsive user interfaces with the use of reusable components. This can help developers to write clean, maintainable, and efficient code. In addition, React Native has a rich ecosystem of libraries and tools that can help increase developer productivity.

4. Large developer community: React Native has a large and active developer community which means that you will have access to a vast amount of resources, tutorials, and libraries that can help speed up development and reduce the learning curve.

5. Access to native features: React Native allows you to access the full range of native APIs, such as camera, geolocation, and others, and use them inside your React Native application. This allows you to create more powerful and engaging applications with features that match the functionality of native applications.

By using these two technologies together, you can leverage the strengths of each to create a high-performance, cross-platform, and scalable mobile application. Furthermore, this combination allows you to reuse the same codebase for backend and frontend with different languages and environments, which can help to simplify the development process, increase productivity and overall maintainability.