1■■ Why do we need the get_csrf_token endpoint?

Explanation:
Django’s CSRF protection uses a cookie named 'csrftoken' to verify that POST/PUT/DELETE requests come from your frontend a
But when React is separate from Django, React never automatically receives this cookie because no Django-rendered HTML pag
So you create this special endpoint that forces Django to send the 'csrftoken' cookie to the browser.

Code: Django view to set CSRF cookie
```python
from django.views.decorators.csrf import ensure_csrf_cookie
from django.http import JsonResponse

@ensure_csrf_cookie
def get_csrf_token(request):
    # This endpoint sets the CSRF cookie on the client
    return JsonResponse({'detail': 'CSRF cookie set'})
```

2■■ How do you call this from React?

Explanation:
When your React app loads, it must call the above endpoint once so the browser receives and stores the 'csrftoken' cookie.

Code: React useEffect to fetch CSRF cookie
```
import { useEffect } from 'react';

function App() {
  useEffect(() => {
    fetch('https://your-backend-url.com/api/get-csrf-token/', {
      credentials: 'include'  // Important to receive and save the cookie
    });
  }, []);

  return (
    // Your app JSX here
  );
}

export default App;
```

3■■ How do you read the CSRF cookie in React?

Explanation:
Now that the 'csrftoken' cookie is in the browser, React needs to read that cookie’s value to include it in the X-CSRFToken header

Code: Helper function to get the CSRF token from cookie
```
function getCookie(name) {
  let cookieValue = null;
  if (document.cookie && document.cookie !== '') {
    const cookies = document.cookie.split(';');
    for (let cookie of cookies) {
```

```
      cookie = cookie.trim();
      if (cookie.startsWith(name + '=')) {
        cookieValue = decodeURIComponent(cookie.substring(name.length + 1));
        break;
      }
    }
  }
  return cookieValue;
}

// Usage:
const csrftoken = getCookie('csrftoken');
```

4■■ How do you send the CSRF token in fetch POST requests?

Explanation:
Django expects every unsafe HTTP request (POST, PUT, DELETE) that uses session authentication to have the CSRF token ser

Code: Example POST fetch with CSRF token included

```
const csrftoken = getCookie('csrftoken');

fetch('https://your-backend-url.com/createUser/', {
  method: 'POST',
  credentials: 'include',  // send cookies (sessionid) with the request
  headers: {
    'Content-Type': 'application/json',
    'X-CSRFToken': csrftoken,  // <-- send the CSRF token here
  },
  body: JSON.stringify({
    username: 'john',
    email: 'john@example.com',
    password: '1234'
  }),
})
.then(response => response.json())
.then(data => {
  console.log('Success:', data);
})
.catch(error => {
  console.error('Error:', error);
});
```

Summary of the flow:
1. React calls /api/get-csrf-token/ → Django sends csrftoken cookie to browser
2. React reads the csrftoken cookie from document.cookie
3. React sends POST requests with header X-CSRFToken: csrftoken
4. Django verifies CSRF token and accepts the request