1. Frontend:

a. Design a simple, mobile-friendly user interface using HTML, CSS, and JavaScript (or a popular framework like React, Angular, or Vue.js). b. Implement a search form where users can enter a movie title or keyword.

c. Display search results in a list or grid format, showing relevant movie information such as title, poster image, and release year.

d. Ensure that the search results are updated dynamically as the user types, without requiring a page refresh.

2. Backend (Optional, if using a public API directly from the frontend):

a. Set up a basic backend to handle API requests (e.g., Node.js with Express, Django, Ruby on Rails, or Flask).

b. Implement caching or rate limiting as needed to optimize API usage and prevent exceeding the API limits.

3. API Integration:

a. Choose a public movie API, such as the Open Movie Database (OMDb) API or The Movie Database (TMDb) API.

b. Integrate the API into the frontend or backend, handling movie search requests and processing the API response.

c. Ensure proper error handling in case of API issues, such as invalid API keys, rate limits, or server errors.

4. Testing:

a. Write basic unit tests for key components, such as the search functionality and API integration. b. Provide instructions for setting up and running the tests.

5. Documentation: a. Write clear and concise documentation on how to set up, run, and use the movie search application.

b. Include any necessary information on obtaining and using an API key for the chosen movie API.

6. Deployment:

a. Containerize the application using Docker or a similar platform. b. Deploy the application on a cloud provider (e.g., AWS, Google Cloud Platform, or Microsoft Azure) or a platform-as-a-service provider (e.g., Heroku or Netlify).

c. Provide clear instructions for deployment and environment configuration.

It allows us to assess the your ability to work with APIs and create a functional, responsive user interface while interacting with external data sources.