**Student Assessment Submission and Declaration**

When submitting evidence for assessment, each student must sign a declaration confirming that the work is their own.

|  |  |  |  |
| --- | --- | --- | --- |
| Student name: **Rashed Hasan Qahah.** | | Assessor name: | |
| Issue date (1St Submission):  **4/12/2024** | Submission date (1St Submission):  **21/1/2024** | | Submitted on: |
| In case of resubmission | | | |
| Issue date (2nd Submission):  26/1/2024 | Submission date (2nd Submission):  28/1/2024 | | Submitted on: |
| Programme: Software Engineering | | | |
| Unit: 29 Application Programming Interfaces | | | |
| Assignment number and title: **1 City Explorer API Enhancement** | | | |

**Task 1:**

**Q1** • Explore the significance of Applications Programming Interfaces (APIs) and Software Development Kits (SDKs) in web application development. Define each and extract the distinctions between them.

**Answer:**

**Application Programming Interface (API),** an abbreviation for the Application Programming Interface, is a set of protocols, methods, procedures, and tools that allow different applications and programs to communicate with each other. So, in this context, it only means that any program has a specific function or purpose interface that refers to A contract or protocol that defines how two applications communicate with each other using requests and responses. Hence, API bundling is simply a way of communicating between different systems or applications. It also defines the methods and data formats applications can use to request and exchange information. Application programming interfaces (APIs) act as intermediaries, facilitating communication and integration between diverse software systems.

**Importance of APIs:**

First, interoperability in APIs allows for seamless interaction between different software systems, enhancing interoperability and data exchange. And also, the ability of different systems or software components to work together, exchange information, and use each other's functions smoothly. Application programming interfaces (APIs) play a crucial role in achieving interoperability by providing a standardized way for different applications to communicate and exchange data.

This is emphasized by the seamless interaction, which APIs certainly act as a bridge between various software systems, which also allows them to communicate in a unified manner. This facilitates seamless interaction, as applications can request and exchange information without complex dependencies.

Second, innovating is easier because APIs provide flexibility, allowing companies and organizations to expand by making connections with new business partners, offering new services to their existing markets, and ultimately, also accessing new markets that can generate huge profits and drive digital transformation.

So, in software development, innovation is a major driver of progress, and APIs play a pivotal and important role in promoting and improving innovation within the developer community. Which can explain the importance of APIs in encouraging and improving innovation through access to external functions. This means that APIs are neglected as gateways that enable and give developers access to external functions and services, usually outside the scope of their application or platform. By leveraging application programming interfaces (APIs), developers can take advantage of the capabilities of other software systems and integrate various features seamlessly and efficiently into their own applications without having to build everything from scratch.

Third, efficiency, which APIs certainly refer to as simplifying development processes by allowing developers and programmers to reuse existing code and services, reducing development time, cost and effort and enhancing and improving high productivity efforts. In this context, API efficiency also works towards the possibility of code reuse. APIs enable developers and programmers to access pre-defined functions or services without going into the details of the internal implementation of those services.

Also, instead of recreating specific functionality from scratch, developers can take advantage of APIs to reuse existing code modules. This certainly significantly reduces the time and effort needed to develop new features, as developers can rely on well-established and tested code. (Lutkevich, n.d.)

**An SDK, or software development kit**, is a comprehensive package of tools, libraries, documentation, and code samples provided to developers to simplify the process of creating applications for a specific software platform or framework. This downloadable toolkit includes libraries that perform common tasks for a specific platform, as well as sample code. Essentially, an SDK allows developers to build and maintain applications without starting from scratch, providing pre-built functions and components to achieve desired functionality.

So, I'll illustrate further, think of baking a cake: An SDK is like a ready-made cake mix, providing pre-made functions or components (ingredients) that contribute to achieving the desired result. Just as assembling a model car or airplane requires a kit containing parts, tools, and instructions, an SDK provides developers with the basic elements needed to create software applications. Whether targeting platforms like iOS or Android, an SDK enables developers to leverage pre-written code, app templates, and comprehensive documentation, simplifying the development process and enabling the creation of more sophisticated apps.

**Importance of SDK:**

First, ease of development, as software development tools SDK simplify the development process by providing pre-made components, which reduces the effort and time required to create applications or websites.

Software development kits SDK are essential for application development because they are easy to develop. It comes with pre-built components, such as libraries, modules, and tools, that are developed and tested by experts. These components speed up the development process, saving time and effort. Software development kits SDK also reduce development efforts, allowing developers to focus on unique aspects of their applications rather than repetitive tasks. This reduces development efforts, especially with regard to meeting tight project deadlines. Software development kits SDK also promote consistency in development practices across different applications and projects, ensuring uniformity in coding approaches and methodologies. This is essential to keep code readable, facilitate collaboration, simplify maintenance tasks, and seamlessly integrate additional features or updates into existing applications.

Secondly, SDKs play an important and crucial role that are seamlessly designed with specific platforms, ensuring more effective and optimal compatibility and performance. SDKs are therefore essential to ensure compatibility with specific platforms, enabling developers to create high-performance applications that integrate seamlessly with the target environment, use resources efficiently, and provide a consistent user experience across different devices. They are designed to integrate seamlessly with operating systems or development frameworks, allowing developers to harness the full potential of the underlying technology without compatibility issues.

SDKs are optimized to fit the target platform's architecture and specifications, ensuring efficient use of system resources such as memory, processing power, and network connectivity. It provides a consistent development environment, enabling developers to create applications that deliver uniform performance across different devices. SDKs often expose APIs that abstract the complexities of accessing platform-specific features and services, allowing developers to take advantage of advanced functionality without digging deep into the underlying platform. It also focuses on enhancing security and compliance with platform-specific standards, ensuring that applications built using the SDK adhere to security protocols and guidelines. Software development kits (SDKs) speed up development cycles by providing pre-built components, libraries, and tools specifically designed for the target platform.

Third, SDKs include comprehensive and important documentation that helps developers understand and improve available tools and resources and use them correctly, similarly, and efficiently

Therefore, SDK documentation is an important tool for developers, providing clear instructions, explanations, and examples of the tools and features within the kit. It helps developers understand components, their functions, and their relationships, enabling them to make informed decisions during development. The SDK documentation also includes illustrations and worked examples, allowing developers to learn by doing. It also provides error handling, debugging tips, release information, and migration guides for smooth transitions. It also provides guidance for improving performance, resource management, standards compliance, and security considerations. It also serves as a hub for community discussions and support, offering forums and FAQs for additional insights and problem solving. Overall, SDK documentation is a vital resource for developers, guiding them through the application development lifecycle and fostering a collaborative and informed environment. (WhatIs.com, n.d.)

Differences between APIs and SDKs:

**APIs:**

Firstly, the **functionality** that APIs focus on are primarily defining data methods and formats in order to effectively facilitate communication between different software systems, and communication that enables smooth communication and data exchange between applications.

Therefore, application programming interfaces (APIs) are a set of procedures or procedures that define the ways to implement software applications. It also defines data formats, ensuring a standardized structure for exchanging information. Application programming interfaces (APIs) facilitate communication between different software systems, enabling seamless communication between applications and data exchange. They enhance interoperability by providing a common interface for communication.

APIs contribute to standardization by defining standards and ensuring consistency in requesting, processing, and returning data across different software components. It creates an abstraction layer that protects the underlying complexity of software systems, allowing developers to interact with APIs without requiring detailed knowledge of the inner workings of the systems they communicate with. This abstraction simplifies the integration process, allowing developers to take advantage of external system functionality without getting bogged down in complex details. APIs also provide flexibility, allowing applications to adapt and evolve independently. In general, APIs act as a communication bridge between software components, facilitating seamless and standardized communication.

Second, **access**, which APIs provide specific access points that allow developers and programmers to interact with features or data of a software system. Also limited in scope, the access provided is often limited to specific functionality offered by the API.

APIs are specific entry points or interfaces that developers use to interact with a software system's features or data. These interfaces are well defined and allow developers to make requests or send instructions to the platform. APIs are designed for a specific purpose, ensuring that access is limited to a well-defined set of functionalities. These details prevent unnecessary or unauthorized interactions and maintain security and control of the system. For example, a social media API may provide access points for posting updates, retrieving user information, and searching for content, but it may not provide access to administrative functions or sensitive user data.

Third, **intermediary**, Application Programming Interfaces (APIs) act as mediators and facilitators of communication, facilitating communication between different software applications and also play the role of acting as connectors, allowing different software components to interact seamlessly.

APIs therefore play a crucial and important role as facilitators and connectors of communication in software ecosystems. It facilitates seamless interaction and data exchange between different components, creating a unified means of communication. APIs act as bridges that connect different systems to understand and interpret each other's requests and responses. It serves as a common language, ensuring that different software components can understand and respond appropriately. APIs also act as connectors, providing well-defined access points and protocols for seamless interaction between different software components. It acts as an intermediary layer, abstracting the complexities of the platforms, simplifying the development process and allowing developers to focus on using the functionality exposed by the API. In short, APIs simplify the integration of diverse software applications, ensuring efficient and standardized interaction in complex, interconnected software environments.

**Software Development Kits (SDKs):**

First, **functionality**, a comprehensive toolkit whose software development kits (SDKs) include a wide range of tools, including libraries, documentation, sample code, and various resources, providing a comprehensive toolkit for application development. Comprehensive support, going beyond just connection definitions, provides a complete set of Utilities to help developers, ensuring that developers have the resources needed to create powerful and efficient software.

They also provide practical examples and learning aids to help developers learn how to implement specific features or functionality using the SDK. SDKs provide a comprehensive toolset, providing end-to-end support throughout the development lifecycle, from initial coding through testing, debugging, and deployment. They take a holistic approach that addresses design, user experience, security, and performance optimization.

Software development kits (SDKs) provide a variety of resources tailored to meet different development needs, allowing developers to choose and adapt resources to their specific projects. The availability of diverse resources allows developers to efficiently customize their applications, meeting unique project requirements. Overall, SDKs are comprehensive resources that improve efficiency, encourage best practices, and support developers throughout the full software development journey.

Second, **access**, which software development kits (SDKs) provide a comprehensive set of tools and resources for building applications on a specific and powerful platform. This is done on a wide scale, and access is not limited to specific jobs. Developers have access to a wide range of tools for developing applications.

So, SDKs are a coordinated package of essential tools for developing applications, including libraries, documentation, code samples, and utilities. They offer a wide range of tools, allowing developers to use various tools designed for different aspects of the development process. SDKs provide comprehensive support covering all phases of application development, including coding, testing, and debugging. It is platform-specific, providing tools that integrate seamlessly with a specific platform, ensuring optimal development. Software development kits SDKs often come with extensive documentation, which facilitates learning and enhances skills. It also supports team collaboration, allowing different team members to leverage specific tools based on their roles and responsibilities. The wide range of tools contributes to increased development efficiency, allowing developers to choose the tools best suited to their tasks. Overall, access to the SDK provides a versatile and comprehensive toolset for efficiently developing applications on specific platforms.

Third, **comprehensive**, is lifecycle support. Software development kits (SDKs) are comprehensive toolkits that support developers and programmers throughout the entire application development lifecycle. Beyond Communication, it extends beyond just communication and includes tools for programming, testing, debugging, and other aspects of development.

They provide guidance and resources for planning and designing applications, and facilitate rapid prototyping and design iteration. SDKs also provide libraries that simplify the coding process, providing pre-built functions, modules, or components. They also provide templates and code examples, serving as educational resources for developers.

SDKs provide testing and debugging support, ensuring the reliability and functionality of the developed applications. It also provides integrated debugging tools to identify and resolve issues efficiently. Software development kits (SDKs) are designed to integrate seamlessly with the target platform or technology stack, ensuring that the application is able to leverage its full potential

SDKs also provide deployment tools, simplifying the process of packaging applications, managing dependencies, and ensuring compatibility with the target environment. They also provide release support, allowing developers to manage their applications and update them in line with the latest features and security patches.

SDKs extend beyond communication by providing tools, resources, and utilities for planning, coding, testing, debugging, integration, deployment, maintenance, and updates. This comprehensive approach contributes to the efficiency and success of the development process.

**Q2** • Determine diverse APIs offered by platforms for various purposes. Choose one platform and identify its array of APIs.

**Answer:**

I've chosen a platform LocationIQ, which is a versatile platform that provides a range of APIs catering to location-based services and geospatial applications. Their offerings encompass several APIs, making it a comprehensive solution for developers seeking location intelligence. One prominent API is the LocationIQ Geocoding API, which enables users to convert addresses into geographic coordinates and vice versa. This is particularly useful for applications requiring precise location information. Additionally, the platform offers the LocationIQ Maps API, which allows developers to integrate dynamic and interactive maps into their applications, enhancing user experiences. The LocationIQ Routing API facilitates efficient route planning and navigation by providing optimal directions between different points. These APIs empower developers to incorporate advanced geospatial functionalities, making LocationIQ a valuable resource for mapping, navigation, and location-based insights projects. It's important to explore each API's documentation and terms of use to ensure they align with specific project requirements.

**1- Search / Forward Geocoding:**

Forward Geocoding is the process of converting human-readable addresses (like "Statue of Liberty, New York") into geographic coordinates (latitude and longitude). This conversion allows you to place markers on a map, tidy mismatched or incomplete addresses, optimize delivery routes, enhance customer profiling, and offer location-specific promotions.

**Additional and detailed information:**

Purpose: The primary purpose is to translate textual location information into precise geographic coordinates.

Applications:

Mapping: Placing markers on a map for locations specified by users.

Address Cleanup: Resolving incomplete or inaccurate addresses for better accuracy in applications.

Route Optimization: Enhancing navigation systems and delivery route planning.

Business Intelligence: Profiling and targeting customers based on location.

**LocationIQ Search API:**

What it Does? the LocationIQ Search API provides a forward geocoding service. It takes a free-form query or structured parameters representing an address and returns the corresponding geographic coordinates along with additional details about the location.

**The Benefits:**

Accuracy: The LocationIQ Search API ensures accuracy by providing precise latitude and longitude information for a given location. This is crucial for applications that require exact geographic coordinates to mark locations on maps or for precise navigation purposes. Users can rely on the API to deliver accurate and reliable location data, contributing to the overall precision of their applications.

Versatility: The API is versatile in handling various query formats, offering flexibility to developers. It accommodates both free-form queries, where users can input addresses in a natural language format ("Statue of Liberty, New York"), and structured queries, allowing the input of distinct address components. This versatility makes the API suitable for a wide range of applications, from handling user-friendly search queries to more structured and specific location data requirements.

Additional Information: Beyond providing basic geographic coordinates, the LocationIQ Search API furnishes additional detailed data about the location. This includes comprehensive address components such as country, state, city, postal code, and more. Moreover, it offers quality metrics that provide insights into the reliability and accuracy of the geocoding result. This rich set of additional information empowers developers to enhance user experiences, perform address validation, and make informed decisions based on the context of the location.

Customization: The API allows customization of the response format and level of detail according to the developer's specific needs. Developers can tailor the response to their application requirements, choosing the desired output format (e.g., JSON), and specifying the extent of information they need. This customization feature ensures that developers receive the relevant data without unnecessary overhead, optimizing the efficiency of their applications and streamlining the integration of geocoded information into their systems.

**Endpoint and Usage:**

Endpoint:

US Region:

<https://us1.locationiq.com/v1/search?key=YOUR_ACCESS_TOKEN&q=SEARCH_STRING&format=json>

Europe Region:

<https://eu1.locationiq.com/v1/search?key=YOUR_ACCESS_TOKEN&q=SEARCH_STRING&format=json>

**How to Use LocationIQ Search API:**

Access Token:

Before proceeding, ensure you have your API Access Token. If you don't have one, the guide provides assistance in creating a new one or finding an existing token.

**Make a Request:**

Choose the appropriate endpoint based on the user's location (US or Europe).

Include the required parameters in the API call:

**key:** Your API Key obtained during the registration.

**q:** Free-form query string representing the location you want to geocode (e.g., "Empire State Building").

**format:** Specify the output format, commonly set to JSON.

**Example Request - Request GET:**

'https://us1.locationiq.com/v1/search?key=YOUR\_ACCESS\_TOKEN&q=Empire%20State%20Building&format=json'

Replace YOUR\_ACCESS\_TOKEN with your actual API Access Token.

**Response:**

Upon making the request, you will receive a JSON response containing detailed information about the queried location. Key details include:

**lat:** Latitude of the location.

**lon:** Longitude of the location.

**display\_name:** Complete address breakdown.

**Additional information like** place\_id, licence, boundingbox, class, type, etc.

Implement in Your Application:

Extract the obtained coordinates (lat and lon) and use them in your application.

Common applications include placing markers on a map or utilizing the geographic coordinates for other location-specific functionalities.

**Additional Tips:**

Endpoint Selection: Choose the endpoint (US or Europe) based on the majority of your user base for faster response times.

URL Encoding: Ensure that the address string is URL encoded, especially if it contains special characters or spaces.

API Rate Limits: Be aware of the API rate limits. If you make frequent requests, consider caching results to reduce redundant calls. Retry failed requests after a short while.

Handling Multiple Results: Acknowledge that the API can return multiple results for ambiguous queries. Ensure your application can handle and appropriately display multiple geocoding results.

**For More Details:**

Refer to the LocationIQ API Reference – ([Search / Forward Geocoding](https://docs.locationiq.com/docs/search-forward-geocoding)) for an exhaustive list of parameters and options, allowing you to customize your geocoding requests according to your application's needs.

**2- Matrix API**

Matrix API is a service that calculates the duration of the fastest route and the corresponding distances between all pairs of supplied coordinates. It provides a matrix of travel times and distances, offering valuable information for route optimization and analysis. Unlike straight-line distances, these values consider the actual road network and provide more realistic estimates for travel.

The Matrix API is a powerful tool that computes the duration and distances between all pairs of specified coordinates. Its primary functionality lies in providing accurate travel times and distances, making it a valuable asset for route planning and optimization in various applications.

**Key Features and Benefits:**

* Computes Duration and Distances:

The API calculates both the travel duration and distances between multiple sets of coordinates. This information is crucial for understanding the time it takes to travel between different locations and the corresponding distances.

* Route Planning:

Users can leverage the Matrix API to plan routes efficiently. By obtaining accurate travel times, applications can suggest optimal paths for navigation, helping users save time and make informed decisions about their journeys.

* Optimization:

Businesses and logistics services can benefit from the optimization capabilities of the Matrix API. It enables the evaluation of various routes, allowing for the selection of the most time-efficient and cost-effective paths for deliveries, transportation, or other activities.

**Endpoint and Usage:**

Endpoint:

The endpoint for the Matrix API is based on the region. For the US region, it could be:

[https://us1.locationiq.com/v1/matrix/driving/{coordinates}?key=<YOUR\_ACCESS\_TOKEN>&sources={elem1};{elem2};..&destinations={elem1};{elem2};...&annotations={duration|distance|duration,distance}](https://us1.locationiq.com/v1/matrix/driving/%7bcoordinates%7d?key=%3cYOUR_ACCESS_TOKEN%3e&sources=%7belem1%7d;%7belem2%7d;..&destinations=%7belem1%7d;%7belem2%7d;...&annotations=%7bduration|distance|duration,distance%7d)

**Example Request:**

Below are examples of different requests:

**3x3 duration matrix:**

https://us1.locationiq.com/v1/matrix/driving/-0.127627,51.503355;-0.087199,51.509562;-0.076134,51.508037?key=<YOUR\_ACCESS\_TOKEN>

**1x3 duration matrix:**

https://us1.locationiq.com/v1/matrix/driving/-0.127627,51.503355;-0.087199,51.509562;-0.142001,51.501284?sources=0&key=<YOUR\_ACCESS\_TOKEN>

**3x3 duration and distance matrix:**

https://us1.locationiq.com/v1/matrix/driving/-0.127627,51.503355;-0.087199,51.509562;-0.142001,51.501284?annotations=distance,duration&key=<YOUR\_ACCESS\_TOKEN>

**Request Parameters:**

**sources:** Specifies source locations. Default is "all."

**destinations:** Specifies destination locations. Default is "all."

**annotations:** Defines what matrix values to return (duration, distance, or both).

**Response:**

The response includes a matrix of durations, distances, and details about sources and destinations.

**Benefits:**

* Accurate travel times and distances for route optimization.
* Essential for applications involving logistics, transportation, and delivery services.
* Supports informed decision-making for planning and resource allocation.

**Example Request for Amman, Jordan:**

Assuming you want to calculate the distance and duration matrix for locations in Amman, Jordan, you need to replace the coordinates in the API request with the ones for Amman. For example, using coordinates for Amman:

<https://us1.locationiq.com/v1/matrix/driving/35.9454,31.9552;35.8770,31.9632;35.9344,31.9529?annotations=distance,duration&key=pk.275ee52d3e9678c8b98d3bd8551da457>

**In this request:**

35.9454,31.9552: Represents the first location in Amman.

35.8770,31.9632: Represents the second location in Amman.

35.9344,31.9529: Represents the third location in Amman.

Finally, the Matrix API is a powerful tool for obtaining detailed information about travel between multiple locations, enabling developers to create more efficient and optimized routing solutions.

**3- Static Maps Overview:**

Static maps are standalone images - in JPG or PNG format, that can be displayed on web and mobile devices without the need for a mapping library or API. The LocationIQ Static Maps API allows users to generate static map images by specifying parameters such as location, zoom level, size, map theme, and markers.

**Static Maps API:**

**API Description:**

Functionality: The Static Maps API generates standalone map images with specified parameters.

Benefits: It provides a quick and simple way to display location-based information without the need for complex mapping libraries. This API is useful for embedding maps in websites, mobile apps, or other platforms where dynamic maps may not be necessary.

The Static Maps API from LocationIQ allows developers to create static map images by making simple HTTP requests. Instead of embedding interactive maps with complex functionalities, this API is designed to generate static images for scenarios where a simple visual representation of a location is sufficient.

**Key Features:**

* Standalone Map Images: The API returns complete map images in JPEG or PNG format that can be easily embedded in web pages, mobile apps, or other applications.
* Parameter Customization: Users can specify various parameters such as the center coordinates, zoom level, map size, format, map type (theme), and markers to customize the appearance of the static map.
* Quick Integration: Developers can integrate static maps without the need for extensive mapping libraries or complex configurations, making it a convenient solution for applications with simpler map visualization needs.

**Static Maps Endpoint and Usage:**

**Endpoint:**

https://maps.locationiq.com/v3/staticmap

**Request Parameters:**

* Key: Your LocationIQ API access token.
* Center: Coordinates (latitude, longitude) specifying the center of the map.
* Zoom: Zoom level of the map.
* Size: Size of the map image in pixels.
* Format: Format of the map image (JPEG or PNG).
* Maptype: Specifies the map theme/style (streets, dark, light).
* Markers: Allows the addition of markers to the map.

**Example Request:**

https://maps.locationiq.com/v3/staticmap?key=pk.275ee52d3e9678c8b98d3bd8551da457&center=40.7128,-74.0060&zoom=16&size=600x400&format=jpg&maptype=streets

**Benefits:**

* Easy integration: Simple HTTP request to generate static map images.
* Lightweight: Standalone images for quick loading.
* Customization: Users can customize the map's appearance using parameters.
* Marker support: Add markers to highlight specific locations on the map.

https://maps.locationiq.com/v3/staticmap?key=pk.275ee52d3e9678c8b98d3bd8551da457&center=31.9632,35.9304&zoom=11&size=600x400&format=jpg&maptype=streets

**Explanation of Parameters:**

* Center: Coordinates for the center of the map in Amman, Jordan (latitude 31.9632, longitude 35.9304).

The Static Maps API is a convenient tool for applications and websites that require static map images for visualization purposes. (locationiq.com, n.d.)

**Q3** • Assess the criticality of security in API usage. Analyse the strengths and weaknesses of security concerns associated with APIs.

**Answer:**

The importance of security in using an API:

APIs (Application Programming Interfaces) play a crucial and important role in modern software development by enabling communication and exchange of data and information between different applications. While APIs provide efficiency, flexibility, and effectiveness, their use presents enormous security challenges that require careful and significant consideration. Therefore, the importance of security in using an API is multifaceted and can be evaluated through several key important aspects.

First, the importance of security in using the API: data confidentiality. Their importance lies in the fact that APIs act as channels for transferring sensitive information, from user credentials to financial data. The importance of data confidentiality lies in protecting this information from unauthorized access, and ensuring the privacy and reliability of user data. In the absence of strong measures such as encryption and access controls, sensitive data becomes vulnerable to interception by malicious entities. The consequences extend beyond privacy violations, to include legal consequences for organizations that fail to adequately protect user information.

For example, for mobile banking application and API communications. Which mobile or hardware banking application interacts with the server API to retrieve the user's account details, including account balances and transaction history.

Implementing data confidentiality measures in APIs is critical to maintaining user trust and compliance. By establishing secure communication channels using protocols like TLS and encrypting sensitive information, organizations can demonstrate their commitment to user trust and legal compliance. Data protection regulations such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) impose strong data confidentiality measures, and failure to comply can result in serious legal consequences. Continuous monitoring and regular security audits help organizations stay ahead of potential risks. In conclusion, data confidentiality is a cornerstone of the importance of API security, demonstrating a commitment to user trust and legal compliance. (Indusface, 2021)

Second, authentication and authorization are in API security, through identity verification, which authentication ensures that entities interacting with the API are who they claim to be. It includes validating user credentials, such as usernames, passwords, API keys, or tokens. And also, permissions control, which determines the authorization actions that authenticated users are allowed to perform. Which includes checking whether the user has the necessary permissions to access specific resources or perform certain operations.

I will explain more here, Preventing Unauthorized Access is that strong authentication prevents unauthorized entities from accessing the API. Without this, malicious or unauthorized actors could enter, posing a major security threat. Restricting access to authorized actions Delegation ensures that even authenticated users can only perform actions they are explicitly authorized to perform. This limits potential misuse of the API.

For example, Cloud Storage API is a system that enables users to upload and download files. Authentication is crucial to this process, as users must provide a valid API key or OAuth tokens to authenticate themselves. Authorization defines user roles and permissions, and access controls restrict access to specific actions. The API verifies the user's identity using their API key or OAuth token, ensuring they are legitimate and have the right to interact with the storage service. Access controls restrict access to specific users, such as deleting another user's file. These mechanisms collectively protect sensitive operations, preventing unauthorized access and potential misuse of the API. Implementing a comprehensive authentication and authorization framework is essential to maintain API integrity and security across domains. (Kong, 2023)

Third, the importance of security in using an API: data integrity, which ensures that the integrity of API data ensures that information remains accurate and unchanged during transfer between systems. It involves protecting data from unauthorized modification, corruption, or tampering. Also, without strong integrity guarantees, malicious actors can manipulate data in transit, leading to severe consequences such as misinformation, financial loss, or a compromise of the entire system.

For example, an e-commerce API to process orders and update inventory. Because of its importance in the e-commerce environment, maintaining accurate transaction data is critical to financial and operational health.

The API can use cryptographic hash functions or digital signatures to ensure the integrity of order details and inventory updates. These techniques generate a unique checksum or signature of the data, which can be verified by the receiving system. For example, when processing a request, the API calculates the hash using a secure hash function such as SHA-256. The receiving system then recalculates the hash based on the received data, ensuring that the data has not been tampered with during transmission. This ensures that order details, pricing information and stock updates are protected from unauthorized modifications, thus maintaining systems reliability and trustworthiness.

Fourth, the importance of security in using an API: Managing the API lifecycle. Throughout the API lifecycle, which includes stages from design to decommissioning, incorporating strong security measures is essential. This comprehensive approach includes consistent efforts such as regular updates, patching vulnerabilities, and secure API retirement. I will explain further that neglecting security considerations at any stage of the API lifecycle poses significant risks. Outdated security protocols may introduce vulnerabilities that threat actors can exploit, leading to a data breach, unauthorized access, or service outage. A proactive security approach is critical to address emerging threats and maintain a resilient API environment.

For example, a social media API evolves through multiple releases. The social media API, which evolves through different versions, faces dynamic security challenges due to changing user requirements and emerging threats. Regular security audits, updates, and downtime planning are essential to address new vulnerabilities and maintain a secure API environment.

Regular security audits are therefore crucial in identifying and patching potential vulnerabilities in APIs. Regular updates are necessary to patch known vulnerabilities and incorporate the latest security practices. Planning for downtime helps users move to newer, more secure releases. Secure retirement of an API is vital when an API reaches the end of its lifecycle. Best practices for API lifecycle security include integrating security from the initial design phase, conducting regular audits to identify vulnerabilities, implementing a robust patch management process, educating API users on security best practices, and monitoring and incident response. By integrating security measures from the design phase to retirement, organizations can effectively mitigate risks, adapt to evolving threats, and provide a secure and reliable API environment for users. (Postman API Platform, n.d.)

**Analyze the strengths and weaknesses of security concerns associated with APIs:**

**Strength point:**

First, secure authentication mechanisms, which secure authentication mechanisms are fundamental to API security. OAuth 2.0 and API keys are powerful methods that verify the identity of entities accessing an API. Designed specifically for authorization, OAuth 2.0 enables secure token-based authentication. API keys provide a simpler yet effective way to control access.

For example, in the context of the banking and finance API, OAuth 2.0 is used to authenticate the user. Users initiating orders must provide a valid access code obtained through a secure authorization process. Additionally, multi-factor authentication, such as an SMS code, is enforced, adding an extra layer of identity verification.

Therefore, OAuth 2.0 is a widely adopted security mechanism for application programming interface (API) access, providing high security guarantee. It provides a unified authorization framework, allowing APIs to control access permissions based on the scope of the access token. This ensures that users only access resources they are explicitly authorized to use. OAuth 2.0's token-based approach reduces the risk of sensitive credentials being exposed during API interactions. It is scalable and well suited for managing user permissions in a dynamic environment. Including multi-factor authentication, such as an SMS code, improves the user experience by ensuring security without creating any friction for users familiar with the additional verification step. Overall, OAuth 2.0 and multi-factor authentication improve the overall security posture of an API. (Frontegg, n.d.)

Second, regular security audits and penetration testing. Conducting regular security audits and penetration testing is a key force in maintaining the security posture of the API. Security audits include a comprehensive and complete examination of the API design, implementation, and infrastructure to identify vulnerabilities, while penetration testing simulates real-world attacks to assess system resilience.

For example, a cloud service provider, which hosts critical applications and APIs, therefore recognizes the importance of ongoing security. To strengthen and improve its defense mechanisms, the provider constantly participates in penetration tests.

I'll explain more Regular security audits and penetration testing are essential for organizations to identify vulnerabilities and remediate them before they can be exploited by malicious actors. This proactive approach helps organizations stay ahead of potential threats and mitigates the risk of exploitation. Security audits also include a comprehensive assessment of security controls, including access controls, encryption, and authentication mechanisms. Penetration testing evaluates the effectiveness of these controls in real-world attack scenarios, ensuring robust security in all aspects of APIs.

Regular security audits allow organizations to adopt a continuous improvement mindset, allowing them to adapt and enhance their security measures as new threats emerge. This includes incorporating evolving security practices in response to emerging threats in the dynamic cybersecurity landscape.

Penetration testing also contributes to incident response preparedness, helping organizations understand how well they can detect, respond to, and recover from security incidents. For organizations operating in regulated industries, regular security audits are essential to maintaining compliance with industry standards and regulations. The proactive nature of these activities is critical in keeping API infrastructures resilient in the face of an ever-changing threat landscape. (BreachLock\_Labs, 2022)

Third, data encryption, through data encryption for strong, more protective and effective security, which encrypts data during transmission and at rest is an essential practice in information security. It ensures the confidentiality and integrity of sensitive data, and provides a strong defense against various security threats. Therefore, for data transmission, data is encrypted during transmission, which includes securing the information as it is transmitted between the client (requesting party) and the server (API). This is usually achieved through the use of HTTPS (Hypertext Transfer Protocol Secure).

Data-at-rest encryption also involves protecting information when it is stored in databases or other forms of persistent storage. Prevents unauthorized access to data even if the storage medium is compromised.

For example, a healthcare API encrypts patient records, and consider a healthcare API that handles patient records, which are highly sensitive and subject to privacy regulations. Which API ensures that HTTPS is used during data transfer. All communications between clients (such as healthcare applications or systems) and the API take place over a properly secure, encrypted, and efficient channel.

Patient records stored in the API database are protected using passive encryption mechanisms. This involves encrypting data before it is written to the storage system, making it unreadable without the appropriate decryption key.

Finally, encryption of patient data ensures confidentiality, integrity, and compliance with healthcare regulations. Encrypted data is incomprehensible even if intercepted, and is protected from unauthorized modifications. It also reduces the risk of legal and financial consequences. Encrypting data in transit and at rest is a comprehensive approach to data security, addressing vulnerabilities during communication and storage. Comprehensive protection ensures that patient records are protected not only during interactions with healthcare applications but also in databases. The power of cryptography lies in its ability to adapt to different scenarios and industries, making it the cornerstone of building secure API ecosystems. (docs.aws.amazon.com, n.d.)

**Weaknesses point:**

Bad coding in API development Bad coding practices in API development refer to inefficient, insecure, or poorly structured code. It includes a range of issues, including incorrect input validation, lack of proper error handling, and vulnerability to common exploits such as SQL injection or buffer overflow.

Therefore, the impact of bad coding on the security of the API is that it has exploitable vulnerabilities, and bad coding leads to the emergence of security vulnerabilities that malicious parties can exploit. For example, incorrect input validation may lead to injection attacks, compromising data confidentiality and integrity.

Also, ineffective coding can unintentionally increase the attack surface of an API. Unnecessary functionality, poorly implemented authentication mechanisms, or exposed sensitive information contribute to a wide range of potential exploits.

Bad coding can lead to security misconfigurations, exposing default settings, debugging information, and sensitive credentials. This can lead to unauthorized access or privilege escalation, posing a significant security risk to the APIs. One scenario is an e-commerce API with insufficient input validation, resulting in customer details being exposed. This vulnerability increases the risk of a data breach or service outage.

Mitigation strategies include regular code reviews and audits, enforcing secure coding practices such as input validation and output encoding, and providing ongoing education on secure coding principles. By prioritizing code quality and security, organizations can strengthen the cybersecurity posture of their APIs and protect against potential exploits. Therefore, a comprehensive approach is needed to address this weakness in API security.

Second, Lack of Proper Monitoring and Logging in API Security, which insufficient monitoring and logging represents a serious weakness in API security. Monitoring and logging are essential components of a strong security infrastructure, providing insight into the operational aspects of an API. When these mechanisms do not exist or are inadequately implemented, the ability to detect and respond to security incidents is severely compromised.

Monitoring includes real-time monitoring of API activities, performance, and behavior. It helps to immediately identify anomalies, suspicious patterns or deviations from normal operation. Logging also includes the systematic recording of events, transactions, and interactions within the API. It serves as a historical record that can be analyzed for security audits, incident investigations, and compliance purposes.

For example, failed authentication attempts, and the API is a target for malicious actors trying to gain unauthorized access. Without proper monitoring, there is no real-time visibility into authentication attempts.

Therefore, the absence of proper monitoring and logging of API security can lead to delayed threat identification, limited visibility, reduced incident response effectiveness, and regulatory compliance challenges. This delay allows malicious actors to exploit vulnerabilities over a long period, reducing the efficiency of the response process. In regulated industries, insufficient logging may result in non-compliance with data protection and security standards, leading to legal consequences and damage to the organization's reputation. To address this vulnerability, organizations must invest in comprehensive monitoring tools, enhance logging practices, automate alerting systems, regularly review logs, and conduct compliance audits. By addressing these issues, organizations can significantly improve their ability to detect, respond to, and mitigate security incidents, strengthening the overall security posture of their API. (owasp.org, n.d.)

Third, Poorly Managed API Lifecycle, neglecting security considerations throughout the API lifecycle, from design to decommissioning, can lead to a poorly managed API ecosystem. The API life cycle includes different stages, such as design, development, testing, deployment, maintenance, and decommissioning. Security must be integrated at every stage to ensure ongoing protection of sensitive data and resources.

As an example, consider an application programming interface (API) used by a financial institution that evolves through multiple versions to accommodate new features and functionality. For various reasons, including lack of time or lack of awareness, the development team neglects regular security audits and updates for older versions of the API.

As newer security protocols and encryption standards emerge, the lack of updates means that deprecated API versions continue to use outdated and potentially weak security measures.

The API lifecycle is an important aspect of cybersecurity, as it contains vulnerabilities that can be exploited by malicious actors. Regular security audits, planning for immediate release and patching are key to mitigating these risks. Regular audits help identify and remediate vulnerabilities, while downtime planning encourages users to move to newer, more secure versions. Education and security awareness are also essential to maintaining a strong security posture against evolving threats. (Wentowski, 2023)

**Task 2:**

**Q1** • Investigate the existing web application “**ExploreWonders**” to explore its potential extension through APIs. Delve into the application's scope and determine the relevant APIs, considering the web as the development platform.

**Answer:**

We will discuss the contents of the ExploreWonders website, The Topbar section of the ExploreWonders page serves as a prominent header containing essential information and quick access to social media platforms. In this section, users are provided with key contact details, reinforcing the brand's credibility and facilitating communication. The left side of the Topbar features the company's physical address, phone number, and email address, presented in a clear and readable format. This information contributes to building trust and making it convenient for users to reach out.

On the right side of the Topbar, a set of social media icons is displayed, allowing users to connect with ExploreWonders through various online platforms. The use of well-known social media icons such as Twitter, Facebook, LinkedIn, Instagram, and YouTube indicate the brand's active presence in the digital space. Each icon is accompanied by a small, outlined button, creating a visually cohesive and easily clickable set of links. This design encourages users to explore and engage with ExploreWonders on different social channels.

The Topbar's background is styled in a dark color, providing a clear visual demarcation from the rest of the page and enhancing the visibility of the displayed information. The layout is responsive, adapting to different screen sizes and ensuring a consistent user experience across devices. Overall, the Topbar is a strategic section that combines essential contact details with social media integration, emphasizing accessibility and user engagement for ExploreWonders.

**The Navbar section** of the ExploreWonders page is a crucial part of the website's front-end design, combining navigation elements with an attention-grabbing hero header. Here's a detailed analysis of its components:

Navbar (Navigation Bar):

The navigation bar is positioned at the top of the page, contained within a responsive container. The ExploreWonders logo or site name ("Tourist") is displayed on the left side of the navbar.

Navigation links include "Home," "About," "Services," and "Contact," providing users with easy access to key sections of the website. A "Register" button is placed on the right side of the navbar, serving as a prominent call-to-action for user registration. The main headline, "Enjoy Your Vacation with Us," is presented in a large and attention-grabbing.

A subheading below the main headline provides additional context, describing the services or benefits briefly. A search input field with a placeholder ("Eg: Jordan") is provided, allowing users to search for specific information or destinations. A "Search" button adjacent to the search input field provides a clear call-to-action for initiating the search.

The Main section is contained within a container and is designed to be visually appealing, encouraging users to explore the site further. Animations are incorporated into the Main section for a dynamic and engaging user experience. The "slideInDown" animation adds a subtle visual effect to the text elements, creating a sense of fluidity and modernity. The entire structure is designed to be responsive, ensuring a seamless user experience across different devices and screen sizes.

Overall, this section effectively combines navigation functionality with an aesthetically pleasing hero header, providing users with a clear path to navigate the site and encouraging them to engage with the featured content. The use of animations adds a touch of sophistication and modern design to enhance the overall user experience.

**The About section** of the ExploreWonders website has been carefully designed to provide users with a comprehensive introduction to the platform. It starts with responsive design.

Includes photo on left side. This image is visually appealing. The “About Us” section also contains textual content. It begins with a small section title (“About Us”) with a white background color and body text color. Next, a welcome headline introduces users to the word with special emphasis on the word highlighted in the primary color. The following paragraphs provide a brief overview of the company, describing its characteristics and services. The text is organized, showcasing key points and features such as first-class flights, carefully selected hotels, 5-star accommodations, the latest vehicle models, 150 distinct city tours, and 24/7 service. Each feature is accompanied by a stylish arrow icon and text.

In short, the About section effectively combines visually appealing design, engaging animations, and informative content to introduce users to the ExploreWonders platform, emphasize its key features and invite them to learn more.

**The Services section** on the ExploreWonders website is designed to showcase and elaborate on the range of services offered by the platform. The section begins with a visually appealing and centered introduction, featuring a white background and a text center with a title "Services" and a subsequent heading "Our Services."

The main service offerings are displayed in a horizontal "boxes" format, divided into rows and columns. Each service is represented by a dedicated box, and there are a total of four columns in each row. Services are displayed with a combination of icons, titles, and descriptive paragraphs to provide users with a quick overview of each service. The structure is structured and follows a consistent pattern for each service.

Icon: An icon representing the nature of the service ("WorldWide Tours," hotel for "Hotel Reservation," user for "Travel Guides," cog for "Event Management").

Title: A heading indicating the type of service offered ("WorldWide Tours," "Hotel Reservation," "Travel Guides," "Event Management").

Description: A brief paragraph that provides additional details about the service, highlighting key features or benefits. The description is consistent across all services, emphasizing a professional and engaging tone.

**When you go** to the Navigation Bar section and click on “About Us” it will take you to the “About Us” page.

The Services section effectively uses a clean and organized layout to present the core offerings of ExploreWonders. The combination of icons, titles, and descriptive content aims to capture the user's attention and communicate the diverse set of services provided by the platform. The consistent design and animation enhance the overall user experience and contribute to a visually engaging presentation of services.

**The Destination section** of the ExploreWonders website is a visually engaging and informative part that showcases popular travel destinations. The section begins with a centered heading, "Popular Destination," which creates emphasis and guides the user's attention. The background and text colors contribute to a visually appealing design.

The destinations are presented in a grid layout, with each destination having its own container. For each destination, there is a combination of a clickable image, a discount label in red ("% OFF"), and a label indicating the destination's name in blue. These elements provide users with a quick overview of the destination, its attractiveness, and any ongoing promotions.

The images used for each destination are large, high-quality, and captivating, showcasing the beauty of the locations. The images are accompanied by discount labels, adding a promotional element to encourage user engagement.

The section features destinations such as Thailand, Malaysia, and Australia, each with its own image, discount label, and name label. The "zoom In" animation applied to each destination enhances the visual experience by adding a subtle, eye-catching effect as the user scrolls down.

Additionally, the use of absolute positioning for the discount and name labels over the images ensures a clean and consistent design. The combination of bold font styles and contrasting colors (red and blue) makes the text easily readable against diverse background images.

The Destination section effectively combines visual appeal, promotional elements, and concise information to entice users to explore popular travel destinations. The use of animations adds a dynamic touch to the user experience, making this section a key component of the website's overall presentation and engagement strategy.

The ExploreWonders page includes a dedicated section for presenting travel packages. This section is designed to showcase different travel options with captivating visuals and essential details.

**The Package Section** is visually appealing and well-structured, allowing users to easily browse through different travel packages. The use of animations, clear information hierarchy, and call-to-action buttons enhances user interaction and encourages exploration. The section effectively balances aesthetics with functionality, providing a seamless experience for potential travelers.

The "ExploreWonders" website's Booking Section is designed to facilitate online tour bookings. The section is aesthetically presented within a container, animation effect. It encompasses key elements to engage users in the booking process.

The left side of the Booking Section features engaging text content presented in a visually appealing manner. A bold heading "Booking" followed by a subheading "Online Booking" captures attention. The content emphasizes the simplicity and comfort of the booking experience, encouraging users to explore further. A "Read More" button is strategically placed, offering a pathway to more detailed information.

On the right side, a well-structured booking form is presented. The heading "Book A Tour" is prominently displayed, reinforcing the user's intention. The form includes several fields to collect the necessary information:

* Name and Email Fields: Capture user identification details.
* Date & Time Picker: Facilitates the selection of preferred date and time, enhancing user convenience.
* Destination Dropdown: Provides a selection of destinations, allowing users to specify their preferences.
* Special Request Text Area: Allows users to communicate any specific requirements or requests.
* "Book Now" Button: Serves as a clear and prominent call-to-action, initiating the booking process.

The Booking Section effectively combines compelling content with a user-friendly form, creating a seamless and visually pleasing experience for users interested in booking tours through the "ExploreWonders" website.

**The Process section** on the ExploreWonders page is designed to guide users through three easy steps involved in the travel process. The layout is centered and visually appealing.

The overall design and content strategy in the "Process" section aim to simplify the travel process into three digestible steps, using a combination of icons, concise text, and a consistent visual theme to enhance user understanding and engagement. The clean and structured layout contributes to a positive user experience, making it easy for visitors to comprehend and follow the outlined steps.

The "Travel Guide" section of the ExploreWonders website is designed to showcase the team of travel guides available to assist users. The section begins with a title and a subtitle that states "Meet Our Guide." The layout is structured within a container and includes responsive design elements for various screen sizes.

Each travel guide is presented in a card-like format within a row. The information for each guide includes an image, social media links, full name, and designation. The design is visually appealing, with a clean and modern look.

Title and Subtitle:

The section starts with a clear and prominent title ("Travel Guide") and a subtitle ("Meet Our Guide").

The card includes an image of the guide, social media buttons (Facebook, Twitter, Instagram), full name, and designation.

Social media buttons are positioned below each guide's image, allowing users to connect with the guides on Facebook, Twitter, and Instagram.

Overall, this section serves the purpose of introducing travel guides to users, providing a glimpse into their personalities through images and social media links. The clean design and use of animations enhance the visual appeal, making it engaging for visitors to explore the team of guides available for their travel experiences.

**The Testimonial section** of the ExploreWonders website is designed to showcase feedback and reviews from clients. It begins with a centered title, "Testimonial," and a subtitle, "Our Clients Say!!!" for emphasis.

Within the container, a text center is featured with a white background. The section starts by presenting a series of testimonials in a carousel format, allowing users to scroll through client feedback.

The Testimonial section aims to build trust and credibility by showcasing positive client experiences. The use of consistent design elements, client avatars, and well-organized content contributes to an engaging and visually appealing presentation of client testimonials. This section serves as a valuable tool for potential clients to gain insights into the quality and satisfaction of the services offered by ExploreWonders.

The Footer Section of the ExploreWonders webpage is a comprehensive and well-organized component that serves various purposes. It is designed to provide essential information, navigation links, and additional features to enhance user engagement. Let me explain the contents of the footer section:

The company section provides a brief overview of the company, including links to important pages, contact information, privacy policy, terms and conditions, FAQs, and helpful resources. The contact section displays contact information, including physical address, phone, email, and social media links. The gallery section showcases images related to travel packages. The newsletter section encourages subscription, with a description, input field, and signup button. The copyright section displays copyright information and additional links. Additional navigation links at the bottom of the footer enhance user navigation.

The Footer Section is designed not only for navigation but also for user interaction and engagement. It provides a comprehensive set of links and features that contribute to a positive user experience and ensure that users have access to important information and resources.

**The Services section** of the ExploreWonders webpage is a visually appealing and informative segment that showcases the various offerings provided by the platform. The section begins with utilizing a large container for content presentation.

At the top, a stylish heading emphasizes the theme with a background in white and primary text color, creating a visually distinct "Services" title. Following this, a larger heading introduces the section as a whole, stating "Our Services."

The main content consists of a grid layout with four columns, each representing a different service offered by ExploreWonders. Each service is presented in a dedicated column and follows a consistent structure:

Service Icon: A visually striking icon representing the nature of the service. Icons include a globe for "WorldWide Tours," a hotel for "Hotel Reservation," a person for "Travel Guides," and a cog for "Event Management."

Service Title: A clear and concise title for each service, Titles include "WorldWide Tours," "Hotel Reservation," "Travel Guides," and "Event Management."

Service Description: A brief but informative paragraph follows each title, providing a succinct description of the corresponding service. The descriptions highlight the key aspects of each service, such as the global reach of tours, hotel reservations, travel guide services, and event management.

This section effectively communicates the range of services offered by ExploreWonders through a combination of visually appealing icons, clear headings, and concise yet informative descriptions. The consistent and well-organized layout contributes to a positive user experience, allowing visitors to quickly grasp the breadth of services available on the platform.

Also on the Services page, there is a Testimonial section within the page, and I have explained its details.

**The Contact section** of the ExploreWonders website is a comprehensive component designed to facilitate user interaction and communication. It comprises various elements, each serving a distinct purpose.

The section features a visually appealing header and introduction, contact information blocks, Google Maps integration, and a contact form. The header provides essential contact details, while the contact information blocks use icons to represent different types of contact. The rightmost column includes a contact form for user inquiries.

The Contact section is strategically structured, offering users multiple channels to connect with ExploreWonders. It seamlessly combines static information (contact details), dynamic content (Google Maps), and interactive features (contact form). The section effectively balances aesthetics with functionality, catering to users seeking to contact the website for various purposes.

Integrating and adding the **Current Weather API** into ExploreWonders can have many benefits, benefiting the site itself, its visitors as well as customers. The Current Weather API adds an enhanced user experience, creating a dynamic and information-rich element to the site, enhancing the overall user experience. Users can access real-time weather information for their desired travel destinations, making ExploreWonders an even more valuable and user-friendly platform.

Comprehensive travel information, ExploreWonders becomes a one-stop destination for travel-related information, including current weather conditions. Visitors can easily plan their trips by accessing not only travel recommendations but also updated weather details, ensuring a more comprehensive travel planning experience. Also increasing user engagement, users are more likely to engage with the site due to the inclusion of a relevant and interesting feature such as real-time weather updates. Displaying current weather conditions alongside travel destinations encourages users to spend more time on the site, explore different destinations and perhaps discover new places to visit.

It also gives a competitive advantage. The ExploreWonders website gains a competitive advantage by providing an additional advantage and value compared to other travel sites. Providing real-time weather information sets ExploreWonders apart in the travel industry, attracting users who prioritize having all necessary information in one place.

The integration of the existing weather API on ExploreWonders improves its features and competitiveness, providing users with real-time information about weather conditions. This allows users to make informed travel decisions, save time, and plan activities based on the current weather. This also increases confidence in travel plans, reduces uncertainty and ensures a more enjoyable trip. A visual representation of weather conditions adds to the overall appeal of the site, making it more attractive to users. Overall, integrating the existing weather API enhances ExploreWonders competitiveness.

**1- I will be using** this **"Current Weather API** " to add an Explore corner in the ExploreWonders website.

**Overview of the current weather API:**

**API description:**

Which Current Weather API provides real-time weather data for a specific location. It provides details such as temperature, humidity, wind speed, and more, allowing developers to incorporate up-to-the-minute weather information into their apps.

Its function is to retrieve the current weather conditions for a specific location. It also returns data in JSON or XML format.

Among its features, it gives real-time information, through which developers can access the latest weather updates for a particular location. Dynamic applications, which enable the creation of applications that respond to current weather conditions. User engagement which enhances the user experience by providing relevant and timely weather information.

**Endpoint URL:**

http://api.weatherapi.com/v1/current.json?key=180d3e8aa32945f2968170915232012&q=LOCATION

**Requesting parameters** for the current weather API, which the current weather API requires specific parameters to retrieve accurate and relevant weather information. Below are the details of the parameters required for this API:

1. key (Required): The API key is a unique identifier associated with your account, providing authentication to access the WeatherAPI.com services.

Example My API Key: key=180d3e8aa32945f2968170915232012

1. q (Required): The query parameter specifies the location for which you want to retrieve current weather data. It can accept various formats, including city name, coordinates, zip code, and more.

**Description:**

* q=Amman (City name)
* q= 31.9539, 35.9106 (Latitude and Longitude)
* q=10001 (US zip code)
* q=auto:ip (IP address lookup)
* q=iata:DXB (IATA code for airports)

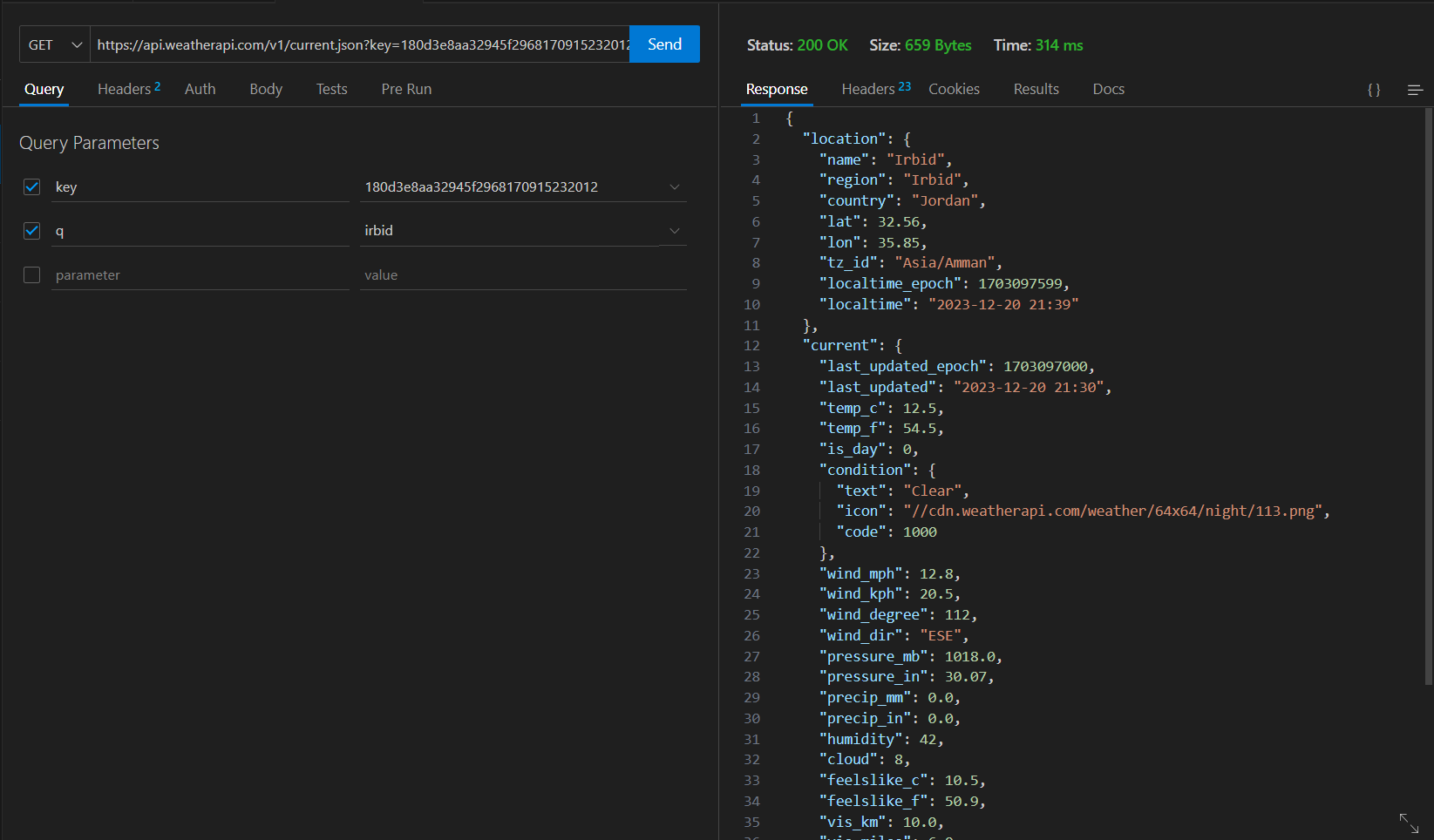
**How to Use:**

**Example Request:**

Assuming you want to get the current weather for **Irbid**:

https://api.weatherapi.com/v1/current.json?key=180d3e8aa32945f2968170915232012&q=irbid

**Example Response:**



**Experience and format of the API that will appear to the user on the ExploreWonders website:**

**Current weather API:**

User interaction, where users can explore real-time weather conditions for wonders around the world. This is done by asking the user to type the name of the country or city in the search bar in the “Explore Corner” section.

Visual representation, the user interface displays temperature, humidity, and wind speed in an easy-to-use manner by having the user write the name of the country or city in the search section. Icons and graphics represent weather conditions for quick understanding.

**2- I will use this** "**Unsplash API**" to add an Explore corner to the ExploreWonders website.

**Unsplash API Overview:**

**API description:**

The Unsplash API serves as a gateway for me as a developer to tap into a huge repository of high-quality photos and images. These resources are freely usable, making them valuable for a variety of applications.

Implementing the “Search Images by Country or City” feature will certainly greatly improve the user experience, allowing visitors to explore and discover wonders from specific countries.

Also, the Unsplash API provides access to a wide variety of high-quality, professionally captured photos. This extensive repository includes different categories, allowing ExploreWonders visitors to discover amazing images related to wonders from around the world.

Improved user experience with search functionality the search functionality in the Unsplash API allows ExploreWonders visitors to customize their explorations. By implementing the “Image Search by Country or City” feature, users can easily find and enjoy the wonders of the chosen country. This enhances the overall user experience by providing relevant and visually attractive content. By integrating the Unsplash API, ExploreWonders can enrich its content with high-quality images, making exploring wonders more immersive and engaging. This visual appeal contributes to providing a memorable and enjoyable experience for website visitors.

**Search function:**

One notable feature and benefit is the search function, allowing me to retrieve images based on specific parameters. This means that users can query images relevant to their needs, enhancing the versatility of the API.

**Endpoint URL:**

https://api.unsplash.com/search/photos?query=[Country or City]&client\_id=bafav7h4vq6gtFO4iXzQtwinqfKhSA6KVWQ5UATp5a8

**Authentication:**

To interact with the API, users must authenticate themselves by including their unique API key in the request. In this case, my API key is [**bafav7h4vq6gtFO4iXzQtwinqfKhSA6KVWQ5UATp5a8**]**.**

**Endpoint Details and Usage:**

**Endpoint Explanation:**

The /search/photos endpoint is specialized for searching photos based on user-defined queries. It's structured to return a list of photos matching the specified search terms.

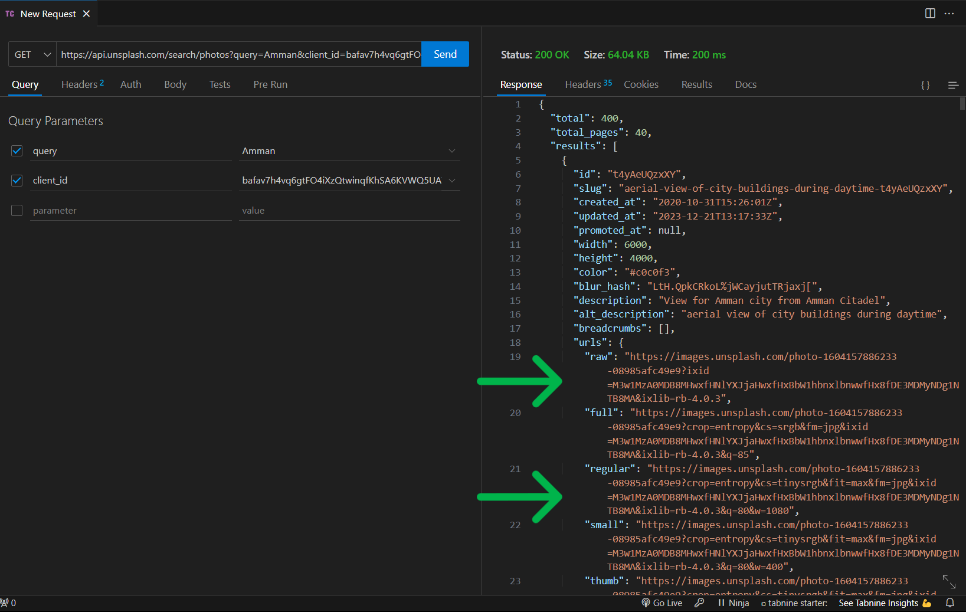
**How to Use:**

A GET request is made to the endpoint, and the query parameter is included to define the search terms. For example, the request GET:

https://api.unsplash.com/search/photos?query=Amman&client\_id=bafav7h4vq6gtFO4iXzQtwinqfKhSA6KVWQ5UATp5a8 searches for photos related to "Amman."

**Example Request:**

https://api.unsplash.com/search/photos?query=Amman&client\_id=bafav7h4vq6gtFO4iXzQtwinqfKhSA6KVWQ5UATp5a8



**Parameters and Descriptions:**

**Parameters:**

* query (Required): Specifies the search terms for finding photos.
* client\_id (Required): The Unsplash API access key required for authentication.

**Descriptions:**

* query: This parameter defines the search terms, as shown in the example with "Amman," indicating a search for photos related to the city.
* client\_id: This is the user's API key, ensuring authentication and permission to access the Unsplash API.

**Benefits:**

* Vast Collection: Users gain access to an extensive collection of high-quality, freely-usable images, providing a valuable resource for various projects.
* Search Functionality: The API's search functionality enables efficient retrieval of photos based on specific search terms, enhancing the precision of image results.
* Versatility: The API's utility extends across a broad spectrum of applications, including web development, content creation, and any other context where high-quality images are needed.
* Security Considerations: Developers should handle their API keys securely, and they need to be aware of any rate limits or usage policies set by Unsplash to ensure responsible and compliant use of the API.

**Experience and format of the API that will appear to the user on the ExploreWonders website:**

User Interaction, whose Explore Photos feature invites users to discover wonders visually through high-quality images. Users can enter the name of the country or city in the search bar to find related images or allow the platform to suggest popular curiosities, which will be in the “Explore Corner” section.

Which will display image search results in a visually attractive grid format. Clicking on the image reveals clear details of images of cities and countries in a wonderful, attractive and high-resolution manner.

**3- I will use this** "**Holidays API**" to add an Explore corner to the ExploreWonders website.

**Holidays API overview:** Holidays API is a service and feature that provides comprehensive holiday information for over 230 countries, regions and territories globally. It covers data for previous years as well as future calendars, and provides details on public holidays, religious dates, and various other categories.

So ExploreWonders visitors can greatly benefit from integrating the Holidays API into the website. Here are several ways the Holidays API improves the user experience on the ExploreWonders website:

* Holiday Information for Countries Visitors can access detailed holiday information for more than 230 countries, regions and territories around the world. Users can also explore and plan their visits to wonders in specific countries by considering local holidays.
* The API provides comprehensive data, including historical and future calendars, covering previous years and calendars up to 2030. Users can also plan their trips in advance by checking upcoming holidays in specific regions.
* The API covers different holiday categories, such as public holidays, religious dates, public holidays, and more. Through which visitors can learn about the different cultural and religious events that are celebrated in specific areas.

ExploreWonders now provides accurate scheduling and global holiday coverage by integrating the Holidays API. This feature improves functionality and provides a more comprehensive travel experience. The user-friendly interface makes it easier for visitors to plan trips, and align their travel plans with local celebrations. This add-on also enhances visitor satisfaction by providing vacation-related insights. Overall, ExploreWonders provides a more personalized and informative travel experience.

**API Details:**

**Endpoint:**

https://api.api-ninjas.com/v1/holidays?

**HTTP Method:** GET

**Endpoint Details and Usage:**

**Endpoint Explanation:** The /v1/holidays endpoint is used to retrieve a list of holiday entries for a given country and year. The response includes details such as the holiday name, date, day of the week, and the type of holiday.

**How to Use:** To make a request, a GET request is sent to the endpoint with the required parameters, including the country (name or ISO code), year (between 2010 and 2030), and optional type filter to narrow down the holiday types.

**Sample Request URL:**

https://api.api-ninjas.com/v1/holidays? X-Api-Key=yBgXLgi2xsi8mrK422zDsQ==nYjEJj5Lqbx2fQd9country=CA&year=2021&type=public\_holiday

**Parameters and Descriptions:**

**Required Parameters:**

* country (Required): The country name or ISO 3166-2 country code.
* year (Required): The calendar year between 2010 and 2030.

**Optional Parameters:**

**type (Optional):** Holiday type filter with possible values such as major\_holiday, public\_holiday, observance, national\_holiday, etc.

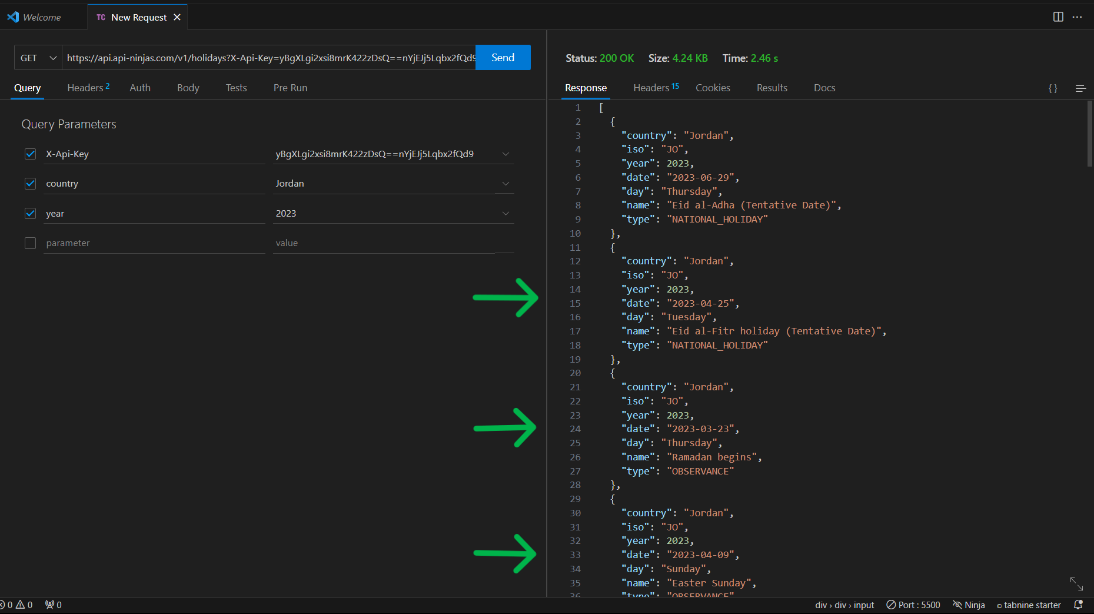
**Headers:**

**X-Api-Key (Required):** API Key associated with your account for authentication.

**Sample Request URL:**

https://api.api-ninjas.com/v1/holidays?X-Api-Key=yBgXLgi2xsi8mrK422zDsQ==nYjEJj5Lqbx2fQd9&country=Jordan&year=2023

**Sample Response:**

****

**This Holidays API is** valuable for applications, websites, or platforms that require accurate and up-to-date holiday information for planning and scheduling purposes.

**The experience and format of the API that will appear to the user on the ExploreWonders website:**

User interaction, the “**Explore Holidays**” section on the “**Explore Corner**” page allows users to plan great visits around local celebrations. Users enter the country and year to discover public holidays, cultural events, and festivals in the country search bar. They also enter the year in the input bar. Optionally, the user can type the type of vacation. This will be in the “**Explore Corner**” section.

The calendar interface displays holidays in text and a beautiful design that displays the information to the user, allowing users to see the temporal context of their wonderful visits.

**4- I will use this** "**Forecast API** " to add an Explore corner to the ExploreWonders website.

Explore Wonders users can gain significant benefits from the Forecast API capability in multiple ways. Users planning to visit the Explore corner listed on Explore Wonders can check the 14-day weather forecast. This helps them plan their trips more effectively by taking into account the weather conditions during their visit. Weather-Based Activities Travelers with a particular interest in activities such as hiking, beach vacations, or outdoor events can take advantage of comprehensive weather information to synchronize their plans with favorable weather conditions. And also, local insights, which the Application Programming Interface (API) provides daily forecasts and detailed hourly weather data. Users can access comprehensive, educated information about temperature fluctuations, wind speed and precipitation around the clock, enabling them to make more accurate decisions.

Weather Alerts the API provides weather alert data, providing customers with quick updates regarding noteworthy weather events or modifications. This function ensures that customers are informed of meteorological conditions that may affect their trip arrangements. Enhancing the Visual Experience To enhance its visual appeal, Explore Wonders can include weather-related images in its platform. For example, showing photographs of wonders under different weather conditions or providing content tailored to the current weather.

Local suggestions can be provided to users by leveraging forecast data. For example, if the weather forecast calls for rain, the platform can offer indoor masterpieces or attractions in the vicinity. Personalized User Experience Integrating meteorological data enhances the individualized nature of the user experience. Users of the Explore Wonders website have the ability to customize their tours by taking into account current and upcoming weather conditions for their chosen locations.

Using the Forecast API, Explore Wonders can improve the vacation planning experience by providing a more comprehensive, user-friendly platform. This integration makes Explore Wonders a convenient, one-stop destination for travel enthusiasts interested in exploring wonders around the world.

The Forecast API enables and provides weather forecast information, including weather forecasts and alerts for up to 14 days, in JSON or XML format. The returned data is organized as a forecast object, containing astronomy data, daily weather forecasts, and weather information at hourly intervals for a given city.

**API Details:**

**Endpoint:**

https://api.weatherapi.com/v1/forecast.json?q=${countryInput}&key=180d3e8aa32945f2968170915232012&days=7

**Functionality:**

It returns up to 14 days weather forecast and also weather alerts. It provides detailed information such as max/min temperature, average temperature, wind speed, rainfall and more.

Includes astronomy data (sunrise, sunset, moonrise, moonset) and hourly weather forecasts.

**Endpoint URL:**

https://api.weatherapi.com/v1/forecast.json?q=${countryInput}&key=180d3e8aa32945f2968170915232012&days=7

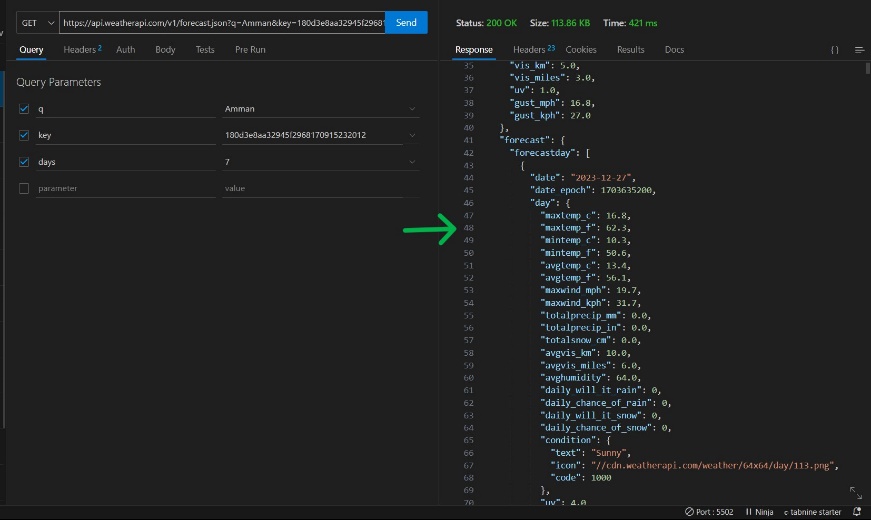
**Parameters:**

* city (Required): The name of the city for which the forecast is requested.
* country (Required): The country code for the specified city.
* days (Optional): Determine the number of days in a weather forecast by the specific endpoint or method used.
* format (Optional): The format of the response, either JSON or XML.

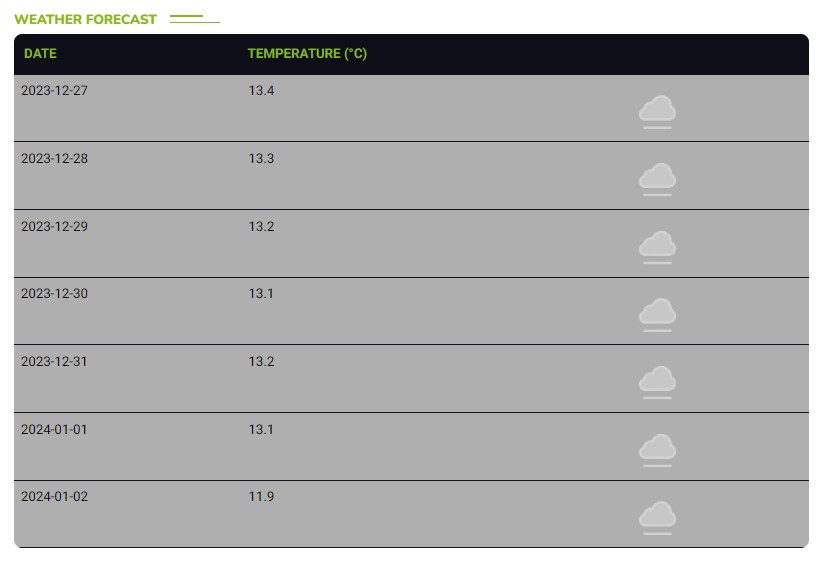
**Sample Request URL:**

https://api.weatherapi.com/v1/forecast.json?q=Amman&key=180d3e8aa32945f2968170915232012&days=7

**Sample Response:**



In the “Explore Corner” section, when the user searches for the city of Amman, these results will appear in the table, which are the temperature and dates for the seven days of the week.



**Q2** • Devise an original design for the application requested by your manager. Analyse this design by outlining the requirements and identifying the API(s) to be incorporated. Justify your selection of the chosen API(s).

**Answer:**

I have chosen to create a program and website for movies, and I have designed, implemented, and analyzed the requirements for the site and what features the user will get by visiting the site, through my use of application programming interfaces. Therefore, I will analyze and discuss the aesthetic design and contents of my website, which I have named. **"WatchWise".**

First, I will explain and analyze the contents and parts of the site, so the movie search section (the <Search> component), which is organized as a section of a series of containers, with the function of searching for movie information, displaying the movie’s promotional video, and an area for displaying search results.

The title and search bar, which is the section of Navbar, was designed so that the user can write the name of the movie he wants, and through it, the search results for the movie he chose will appear, which is the video trailer for the movie and information about it. Therefore, the Navbar contains the logo of the movie website, which is called “WatchWise,” and also three words, which are navigation links “Movie Features,” “Movies,” “Find Celebrity,”and “Contact,” through which the user can click on them. It automatically takes him to the part he wants. The section is designed with a prominent title (“About the Movie”) and a search bar. With which users can quickly recognize the purpose of the section and start searching for movie information easily.

The search icon is placed next to the input field for a visual indication. The search icon improves ease of use by providing a clear visual indication of the search function. Users can recognize and interact with the search feature intuitively.

The movie results container is well-organized, containing image, movie details, and sections for easy navigation. The user benefit is the pleasant visual experience of easily scanning provided movie details. Fade animation adds a visually appealing element to the presentation, making the experience more engaging. The show content is presented in a clean and organized manner, promoting easy reading and understanding of basic movie information. The design choices, including fonts, layout structure, visual cues, and animations, contribute to a modern, easy-to-use interface, improving the overall user experience.

The header or Navbar section contains the site logo, navigation links, and search bar. The navigation links are designed as a horizontal menu, and the search bar is integrated into the navigation menu. The design is clean and straightforward, with a minimalist style. Using a simple color palette and clear design contribute to a stylish aesthetic. The logo is prominently displayed, contributing to brand recognition. Using a transform feature on the logo while scrolling or interacting can provide a subtle interactive element.

Navigation links are easy to read, with clear font size and style. Using micro-conversion in the search portlet and button provides a visual cue to users that these elements are interactive. The navigation adapts well to different screen sizes, as shown by the percentage-based width and height of certain elements.

The search form is part of the navigation, allowing users to enter a search query for movies. The search bar is seamlessly integrated into the navigation, maintaining a cohesive design. Using a form element and search button with a magnifying glass icon is a common and well-known design pattern. The search and button inputs are also designed to be responsive, with a specific width and height. Transformation is used for positioning, which contributes to a consistent design across different devices.

User Benefits The clean and straightforward design of the header and navigation menu ensures that users can quickly find and access different sections of the site, enhancing the overall navigation experience. The prominent display of the logo contributes to brand recognition. Users can easily associate the logo with the site, which enhances the site's identity and user trust. Integrating the search bar into the navigation system, along with the use of a recognizable magnifying glass icon, makes it easier for users to start searches. This feature benefits users by providing a quick and efficient way to find specific movies. Responsive design ensures the site is accessible and visually appealing across different devices, contributing to a positive user experience for both desktop and mobile users.

Movie Posters Section (main container) The Posters section uses a dynamic background image fetched from the movie database, creating an immersive visual experience. A linear gradient overlay is applied to the background image. This overlay potentially enhances the readability of the text and provides a stylistic touch. User Benefit Users benefit from the aesthetic presentation of movie posters. A dynamic background adds an attractive visual element to the site, making it more attractive.

Trailer Viewing (Modal Viewing), YouTube component: The site uses the YouTube component to embed movie trailers and play them seamlessly within the interface. Button Design The on and off buttons are designed with an overlapping gradient, contributing to a consistent and attractive design.

Users can easily watch movie trailers directly on the site, enhancing the overall user experience. Film details and overviews are presented in a well-organized and focused layout, maintaining visual consistency. The trailer plays and close buttons are visually appealing with a layered design. Users benefit from comprehensive movie information, including details and overviews, enhancing their understanding of the specific movie.

Trailer Viewing (Modal Viewing), YouTube component: The site uses the YouTube component to embed movie trailers and play them seamlessly within the interface. Button Design The on and off buttons are designed with an overlapping gradient, contributing to a consistent and attractive design. Users can easily watch movie trailers directly on the site, enhancing the overall user experience.

Film details and overviews are presented in a well-organized and focused layout, maintaining visual consistency. The trailer plays and close buttons are visually appealing with a layered design. Users benefit from comprehensive movie information, including details and overviews, enhancing their understanding of the specific movie.

The design of the search results section and the “No Movies Found” section contribute to an attractive and easy-to-use interface. Search results are displayed in a central container, where each movie includes a poster image and relevant details. This facilitates efficient information retrieval and benefits the user. The “Sorry, no movies found” message is designed in clear typography, ensuring clear communication and easy navigation. The combination of dynamic background images, well-designed components, and clear messaging improves the overall user experience.

Features Section, Design Aesthetic the Features section is organized inside a container and includes a title ("About Movie") as well as a search bar and an icon to launch movie searches. Organizing the feature section into a container provides an organized layout. Adding a search bar and associated icon involves focusing on user interaction and engagement.

Search bar and icon, design aesthetic: The search bar contains a placeholder (“Search movie information”) and is accompanied by a stylized search icon in SVG format. The use of an SVG search icon indicates attention to detail in the visual design. Placeholder text guides users to the intent of the entry, and a clickable icon improves user interaction. This design choice contributes to a clean, modern aesthetic.

Movie Scores Section the Movie Scores section includes a box containing movie information, an image, and various details presented in an organized format. An organized layout with an image, followed by detailed information, creates a visually appealing presentation. Fade animation, implemented through AOS (Animate on Scroll), adds a subtle and elegant touch, making movie details appear seamless.

Movie Details Movie details are displayed in a box with carefully designed content, including bold, blue text for emphasis.

Using different text styles and colors enhances readability and draws attention to important information. Including various details such as genre, cast, title, description, year and country contributes to a comprehensive and visually rich user experience.

The website design prioritizes readability and intuitive interaction with the search bar and associated icons. Its structured design and progressive animation enhance the user experience. The Movie Scores section presents movie details in an attractive manner, while comprehensive information about selected movies enhances the site's value to movie lovers. Including different movie details improves the overall user experience.

Section title (“Find Celebrity”): The section title is prominently displayed with a bold, clear font (“section\_\_content\_title”). The use of one focused heading indicates the purpose of this section. Users can quickly understand the basic function of this section, which is to find information about celebrities.

Search Input and Icon The search input field is accompanied by a search icon (“icon”). Placing the icon inside an input field is a common design pattern for the search function. The input field contains a routing placeholder. The design suggests an intuitive and familiar search interaction. Users can easily familiarize themselves with the search function and start searching for celebrity information.

Celebrity Information Display Celebrity information is presented in a visually appealing manner. Each piece of information is placed in a container with a consistent design (“section\_\_box”). The design uses animation (fading) to enhance the visual experience. Users can quickly scan and digest information about celebrities, including nationality, name, date of birth, age, and occupation.

Displaying images, the celebrity image is included in a custom container ("section\_\_box\_img"). The use of images adds a personal touch to the information display. Users get a visual representation of celebrities, which helps in quick recognition and communication. It enhances the overall user experience by making the interface more attractive and informative.

The design includes responsive elements, ensuring that the design adapts to different screen sizes. Users can access the site seamlessly across different devices, which contributes to a positive user experience and makes information accessible to a wider audience.

The design aesthetic focuses on clarity, visual appeal, and user interaction. Users benefit from an intuitive search experience, a visually appealing information display, and responsive design for easy access across devices. The use of animation adds a layer of sophistication to the overall user interface.

Feedback Section, Users can easily define the purpose of this section - to collect feedback. Form entries are clearly labeled, making it easy for users to submit their information. The left face animation adds a touch of creativity, is likely to grab the user's attention and make the comments section more engaging. If the overall site is designed to be responsive, users can provide feedback seamlessly across different devices.

Footer Section The footer is contained within a <div> with the class "container1." It has an organized design consisting of three main sections: logo, copyright information, and social media icons. Using a container helps organize and structure your footer content. The three distinct sections make it easier for users to recognize and navigate through the different elements.

**To achieve the functional requirements** described for this WatchWise movie app, you will need to integrate different APIs to fetch movie data, including search results, movie details, and trailers. Below are details of the requirements and APIs:

**Functional requirements:**

Search Movies Requirements Users must be able to search for movies by entering a query in the search input field.

API Considerations the TMDB (Movie Database) API provides a comprehensive database of movie information and includes a search endpoint. You can use the Search Movies endpoint to fetch related movies based on the user's search query.

Displaying Movie Information, Requirements The app must display information about each movie, including the title, background image, and other details.

API Considerations the TMDB API can be used for this requirement as well. You can use the Movie Details endpoint to fetch detailed information about a specific movie.

Play Movie Trailer Requirements Users must be able to play the official trailer of the selected movie. API Considerations To play movie trailers, you may want to consider integrating with the YouTube Data API. You can use this API to search for videos and get the video ID of the official trailer. Then, you can embed the YouTube video player into your app.

Select Movie Requirements Users must be able to select a movie from the displayed list. API Considerations This functionality does not necessarily require new API integration, as it is more about handling user interactions within the application. You can use the TMDB API to fetch additional details about the selected movie when the user makes a selection.

The integration process includes searching for movies, viewing movie information, playing movie trailers, and selecting a movie. The user enters a search query, requests to TMDB's "Find Movies" endpoint, and the application retrieves the video ID and embeds it in the YouTube video player. The application then fetches additional details from the TMDB API and updates the displayed information.

**Non-Functional Requirements:**

Performance, requirements that the application must load and display, movie information efficiently. API integration to optimize requests to the Movie Database API for minimum latency.

Requirements: Response time must be improved to fetch movie details and trailers.

Application Programming Interface (API) integration which implements caching mechanisms and ensures efficient processing of trailer requests through the YouTube/Vimeo API.

User Interface, Requirements The user interface must be intuitive and easy to navigate.

API integration, no specific API integration; Ensure user-friendly front-end design. Requirements, movie details and trailers must be presented in a visually appealing manner.

Application Programming Interface (API) Integration: Focus on front-end design principles to present information attractively.

Search Accuracy, Requirements The search function must provide accurate and relevant results. Application Programming Interface (API) integration and also leverage the search capabilities of the Movie Database API to obtain accurate results.

Trailer availability, requirements for which the application must safely handle, cases where a trailer is not available. API integration and implementation of error handling in cases where the YouTube/Vimeo API does not return a trailer.

Responsiveness, the requirements for which the application must be responsive and work well on various devices and screen sizes.

API integration for which there is no specific API integration; Ensure a responsive front-end using responsive design techniques.

Error handling, the requirement that meaningful error messages be provided for data fetch or trailer playback issues. API Integration Implemented error handling for responses from the Movie Database API and YouTube/Vimeo API.

Protect, requirements that API keys and sensitive information are, securely handled. API Integration: Which implements secure methods for storing and transmitting API keys.

Scalability, the requirements that an application must handle a large number of users. API integration and consider scalability of the Movie Database API and implement server-side scalability measures.

Browser compatibility, requirements for which the application must be compatible with major web browsers. API Integration There is no specific API integration; Cross-browser testing to ensure compatibility.

**Functional Requirements Analysis:**

Requirements that users must be able to search for movie information. API Integration the Open Movie Database (OMDb) API (http://www.omdbapi.com/) is used.

* // Example of an API request
* const movieUrl = `http://www.omdbapi.com/?i=tt3896198&apikey=e346d95b&s=${movieInput}`;

**Features:** This is done by searching the input for movie information (<input id="movieInput">). View movie results including Picture, Genre, Cast, Title, Descraption, Year and Country. By processing the events to start the search by pressing the "Enter" key.

**Celebrity search section:**

Requirements: Users must be able to search for information about celebrities. API Integration: The Celebrity Info API (https://api.api-ninjas.com/v1/celebrity) is used.

* // Example of an API request
* const CelebrityUrl = `https://api.api-ninjas.com/v1/celebrity?X-Api-Key=yBgXLgi2xsi8mrK422zDsQ==nYjEJj5Lqbx2fQd9&name=${celebrityInput}`;

**Features:**

Search for celebrity information (<input id="searchCelebrity">). View celebrity details including name, date of birth, profession, age and nationality. Event Processing To start the search by pressing the "Enter" key.

**User Feedback Section:** Requirements, which users must be able to provide feedback.

**Features:** Feedback form with input fields for name and email (<form class="formFeedback">). Input validation for name and email fields. Provide feedback.

**Choose an API:**

* Movie Information: API: Open Movie Database (OMDb) API.
* Endpoint: http://www.omdbapi.com/
* API key: e346d95b

**Usage example:**

* const movieUrl = `http://www.omdbapi.com/?i=tt3896198&apikey=e346d95b&s=${movieInput}`;

**Celebrity information:**

* API: Celebrity Information Application Programming Interface.
* Endpoint: https://api.api-ninjas.com/v1/celebrity
* API key: yBgXLgi2xsi8mrK422zDsQ==nYjEJj5Lqbx2fQd9

**Usage example:**

* const CelebrityUrl = `https://api.api-ninjas.com/v1/celebrity?X-Api-Key=yBgXLgi2xsi8mrK422zDsQ==nYjEJj5Lqbx2fQd9&name=${celebrityInput}`;

**Choose an API:**

* Movie Information: API: Open Movie Database (OMDb) API.
* Endpoint: http://www.omdbapi.com/
* API key: e346d95b

**Usage example:**

* const movieUrl = `http://www.omdbapi.com/?i=tt3896198&apikey=e346d95b&s=${movieInput}`;

**Celebrity information:**

* API: Celebrity Information Application Programming Interface.
* Endpoint: https://api.api-ninjas.com/v1/celebrity
* API key: yBgXLgi2xsi8mrK422zDsQ==nYjEJj5Lqbx2fQd9

**Usage example:**

* const CelebrityUrl = `https://api.api-ninjas.com/v1/celebrity?X-Api-Key=yBgXLgi2xsi8mrK422zDsQ==nYjEJj5Lqbx2fQd9&name=${celebrityInput}`;

**User Benefits:**

Movie Search So users can quickly find information about movies, including details such as genre, actors, and plot. Visual representation with movie posters adds to the user experience.

Celebrity search, Users can collect information about their favorite celebrities, including date of birth, profession, and nationality. Including a photo of a celebrity would enhance the user experience further.

Providing Feedback Users have a channel to provide feedback, enhance user engagement and allow for continuous improvement of the platform.

Consistent Design A consistent design across different sections of the site provides users with a seamless experience and makes navigation intuitive.

**Non-functional requirements:**

Performance: Asynchronous fetching of movie and celebrity data improves performance by allowing the application to continue working without waiting for the data retrieval process to complete. Animations, such as the fade-in effect, contribute to a visually appealing user experience. However, it is important to note that excessive or resource-intensive animations may have a slight impact on performance, especially on resource-constrained devices.

Ease of Use: User interactions are designed to be intuitive, with movie and celebrity searches triggered by pressing the “Enter” key. This is a popular style, easy to use and matches user expectations. Including visual feedback through animation improves the overall user experience by providing a clear indication of the system's response to user actions. This contributes to a more attractive and user-friendly interface.

Protection: API keys are visible in the code (apikey=e346d95b and X-Api-Key=yBgXLgi2xsi8mrK422zDsQ==nYjEJj5Lqbx2fQd9). Although API keys are necessary to access external services, they must be handled carefully to prevent unauthorized access. API keys should never be exposed in client-side code. It must be stored securely on the server, and server-side logic must handle interactions with external APIs to avoid revealing sensitive information.

Maintainability: The code structure and separation of concerns are reasonable, but there is room for improvement. Consider organizing functions and patterns into separate files or modules to promote a modular and maintainable code base. Establishing clear conventions for organizing code, such as using meaningful functions and variable names, can make the code base more readable and easier to maintain over time.

Accessibility: Accessibility is crucial to ensuring that all users, including people with disabilities, are able to use and interact with the application. Although the code does not explicitly address accessibility, it is important to include appropriate ARIA attributes for interactive elements. For example, adding ARIA labels to search input fields can improve the accessibility of search functionality for users who rely on assistive technologies.

Scalability: The provided code does not explicitly address scalability considerations. Depending on expected traffic and usage patterns, improvements may be necessary. Strategies such as caching, implementing pagination for large result sets, and optimizing database queries can be considered to enhance the scalability of the application.

Cross-browser compatibility: Also, the code does not explicitly deal with cross-browser compatibility. Testing the application on different browsers and adjusting styles or functionality as needed is important to ensure a consistent user experience across different browser environments.

Mobile Responsiveness: Ensuring your design and layout are responsive to different screen sizes and devices is crucial to reaching a wider audience. Responsive design practices, such as using media queries and fluid layouts, must be implemented to create a seamless experience on mobile devices.

Error handling: There is limited error handling in the provided code. Robust error handling is critical to handle scenarios such as failed API requests, network issues, or unexpected data formats. Also, implementing error messages and logging mechanisms can help diagnose problems and provide a better user experience by handling errors safely.

Code documentation: Adding comments to explain complex logic, functions, or components can greatly improve code understanding, especially for developers who maintain or extend the code base. Therefore, comprehensive documentation, including high-level overviews and inline comments, contributes to improved code maintainability and collaboration among team members.

* const MOVIE\_API = "https://api.themoviedb.org/3/";
* const SEARCH\_API = MOVIE\_API + "search/movie";
* const DISCOVER\_API = MOVIE\_API + "discover/movie";
* const API\_KEY = "25371891a57c62e6ae5d894eba150721";

**Discussion and analysis of API selection:**

The API chosen for this project is the Movies Database (TMDb) API. This API was chosen due to its comprehensive, well-organized and efficient data related to movies. It offers a wide range of information, including details about movies, TV shows, and more. The base API URL is set as https://api.themoviedb.org/3/, and specific endpoints such as search/movie and discover/movie are used to retrieve movie information.

**Key Features:**

Fetch Movies: The fetchMovies function is responsible for retrieving the movie list. It uses the axios library for asynchronous requests to the TMDb API based on a search query (SEARCH\_API) or to discover popular movies (DISCOVER\_API).

Render Movies: The renderMovies function maps the movie list and displays the movie component for each movie. This function will likely handle displaying movie data in the app.

Play Trailers: This application allows users to play trailers for selected films. To achieve this, it uses the Youtube component from the “react-youtube” library, allowing YouTube videos to be embedded and controlled within a React application.

Search Function: Users have the ability to search for movies using the search bar. This triggers a new call to the TMDb API with the provided search query, allowing the application to fetch and display relevant movie information dynamically.

Responsive Graphic: The app layout is designed to be responsive, ensuring a seamless user experience across different screen sizes and devices.

Justification Choosing TMDb API TMDb API is widely known and used in developing applications related to movies and TV shows. Its popularity ensures that developers are able to tap into a large and active community for support and resources.

Well-Documented Architecture the TMDb API is well-documented, providing clear and comprehensive instructions on how to access and use the available data. This documentation is essential for developers to understand API endpoints, request parameters, and response formats.

Comprehensive Data TMDb offers a rich collection of movie-related data, including details, photos, videos, and more. This comprehensive nature of the data matches the requirements of a movie-related application, ensuring that users have access to a wide range of information about movies and TV shows.

Accuracy and Up-to-date Information TMDb is known for providing accurate and up-to-date information. This is essential for movie-related apps as users expect the latest details about movies, including release dates, cast, trailers, and reviews.

**axios for HTTP Requests:**

Commonly used in React: axios is a widely adopted library for making HTTP requests in React applications. Its popularity within the React community makes it a suitable choice for handling asynchronous data fetching. Developers are likely familiar with syntax and usage patterns.

Simplicity and Efficiency: Axios provides a simple and efficient way to implement asynchronous operations. Its syntax is concise, and it offers features such as request and response interceptors, making it versatile for different use cases, including application programming interface (API) communication.

**"react-youtube" library for embedding YouTube trailers:**

Integration and controlling which explains the decision to use the “react-youtube” library as a strategic choice to seamlessly integrate and control YouTube videos within the React environment. This is crucial for movie-related applications where playing trailers is a common and expected feature.

Convenience The library will potentially simplify the process of embedding YouTube videos, handling player controls, and managing the overall user experience. This convenience is valuable to developers, as it allows them to focus on the core features of the app rather than dealing with the complexities of YouTube embeds.

Finally, the TMDb API was chosen for its popularity, well-documented and efficient architecture, and the breadth of movie-related data it provides. This matches the requirements of the film-related application, ensuring accurate, up-to-date and comprehensive information. Axios was chosen to serve HTTP requests because of its common use in React applications, its simplicity, and its effectiveness in handling asynchronous data fetching.

The 'react-youtube' library is included to provide a convenient solution to embed and control YouTube trailers within a React app, enhancing the user experience.

The combination of the TMDb API and selected libraries provides a solid foundation for movie-related application development, providing rich data and powerful tools for implementation.

**Endpoint and use:**

* Base URL: https://api.themoviedb.org/3/
* Endpoints:
* Search endpoint: /search/movie
* Description: Allows searching for movies based on a query.
* Parameters: Accepts parameters like api\_key for authentication, search term query, etc.
* Example request: <https://api.themoviedb.org/3/search/movie?api_key=25371891a57c62e6ae5d894eba150721&query=Inception>
* Discover endpoint: /discover/movie
* Description: Retrieve a list of movies based on various criteria, including release date, popularity, and more.
* Parameters: Similar to a search endpoint, allowing customization of the query.
* Example request: <https://api.themoviedb.org/3/discover/movie?api_key=https://api.themoviedb.org/3/search/movie?api_key=25371891a57c62e6ae5d894eba150721&query=Inception&sort_by=popularity.desc>

Finally, to use the API, developers need to include their unique API key in each request. The API key is obtained by registering on the TMDb website.

**Parameters:**

api\_key (required): This is a key parameter for authenticating requests. It is a unique key associated with the developer's TMDb account and must be included in every request to access the API.

SEARCH\_API parameters:

Query (optional): This parameter is used when searching for movies. It represents a search query, allowing users to find movies based on specific keywords or phrases.

DISCOVER\_API parameters:

sort\_by (optional): This parameter is used to specify sorting criteria for the list of detected movies. In the example, it is not explicitly set in the code, but can be added based on application requirements. For example, sort\_by=popularity.desc displays popular movies.

SEARCH\_API or DISCOVER\_API is determined dynamically based on whether the search key is valid or not. If search key is not empty, SEARCH\_API is used; Otherwise, DISCOVER\_API is used.

The parameters object is passed to the axios get method, which contains the parameters required for the API call. In this case, api\_key is always included, and the query is only included if the search key is not empty.

This architecture allows the application to fetch movies either by search query or by discovering trending movies, depending on the user's interaction with the search bar.

Additionally, the fetchMovie function makes a separate API call to obtain detailed information about a specific movie by appending the videos parameter to the request. This parameter is used to retrieve video-related information, including trailers.

So, follow best practices to handle asynchronous operations in React application using async/await. The fetched movie data is then used to update the state of the component, causing it to replay and update the UI accordingly.

I have chosen the APIs on the movie site “**WatchWise”**, **omdbapi** and **Ninjas** (**API)s**:

**OMDb API:**

OMDb (Open Movie Database) API is a RESTful web service that provides a large amount of information related to movies and TV. Includes data about movies, TV shows, actors, and more.

**Features:**

* Search by ID or Title: You can query information about a specific movie or TV show using its IMDb ID or title.
* Filter Type: Select the type of result to be returned, such as Movie, Series, or Episode.
* Year Filter: Filter results based on year of release.
* Plot Length: Choose between short or full plot descriptions.
* Data Format: Receive data in JSON or XML format.
* Callback support: Implement JSONP callback function.
* Specify Version: Include API version (reserved for future use).

**Endpoint:**

Base URL: **http://www.omdbapi.com/**

Poster API: **http://img.omdbapi.com/**

**Parameters:**

**By ID or Title:** i (IMDb ID) or t (title), with additional options for type, year, plot, data type, callback, and version.

**By Search:** s (movie title), with options for type, year, page, data type, callback, and version.

**API Ninjas:**

**What it is:** The Celebrity API from API Ninjas is a RESTful web service that provides information about famous individuals worldwide. It includes details such as net worth, profession, birthday, height, and nationality.

**Features:**

* Search by Name: Query celebrities by their name.
* Additional Parameters: Filter by net worth, nationality, height, and more.
* Headers: Requires the X-Api-Key header for authentication.

**Endpoint:**

URL: https**://api.api-ninjas.com/v1/celebrity**

**Parameters:**

**name (optional):** Name of the celebrity for the search.

**Additional optional parameters:** min\_net\_worth, max\_net\_worth, nationality, min\_height, max\_height.

**X-Api-Key (required):** API Key associated with your account.

**API Details and Benefits:**

**OMDb API:**

What it Does: Provides detailed information about movies, TV shows, actors, and related data.

**Benefits:**

* Comprehensive Data: Access extensive movie and TV show details.
* Popular Source: Widely used for its comprehensive and reliable movie-related information.
* Flexible Querying: Search by ID, title, or perform searches with various parameters.

**API Ninjas (Celebrity API):**

What it Does: Offers information about famous individuals, including net worth, profession, birthday, height, and nationality.

**Benefits:**

* Rich Celebrity Data: Provides diverse details about celebrities from various fields.
* Versatile Searching: Search by name and apply filters for net worth, nationality, and height.
* API Key Security: Requires an API key (X-Api-Key) for authentication.
* Endpoint and Usage Details for Provided Code:

**OMDb API Endpoint:**

Base URL: http:**//www.omdbapi.com/**

Poster API URL: **http://img.omdbapi.com/**

Usage in Code: The getMovies function uses the SEARCH\_API to fetch a list of movies based on user input.

The getCelebrity function uses the Celebrity API endpoint to get information about a celebrity based on user input.

**API Parameters:**

**For movie search:** s (title), type, y (year), r (data type), page, callback, v (version).

**For poster API:** s (title), type, y (year), r (data type), callback, v (version).

**API Ninjas (Celebrity API) Endpoint:**

URL: **https://api.api-ninjas.com/v1/celebrity**

**Usage in Code:**

The getCelebrity function utilizes this endpoint to search for and retrieve celebrity information.

**API Ninjas (Celebrity API) Parameters:**

name (celebrity name), min\_net\_worth, max\_net\_worth, nationality, min\_height, max\_height.

**X-Api-Key header:** Required for authentication.

In summary, these APIs provide valuable data for movie-related information and details about celebrities, and the provided code demonstrates how to use them in a web application.

It is also important for me to review and discuss the importance and benefits of the application programming interfaces that I used on the **WatchWise** movies site, through which they effectively enhance the users’ experience of the site, the most important of which are:

**TMDb Movies API:**

Benefits for users: Users can access a wealth of information about the films, including details about the cast and crew, reviews, ratings, and more. TMDb is also updated frequently, ensuring users get the latest data about movies and TV shows.

Users can view posters, wallpapers, and trailers, enhancing the overall movie watching experience. The API supports search and discovery features, allowing users to find specific movies or discover new ones based on various criteria. With detailed information about genres, keywords and trending films, users can customize their own movie recommendations.

**OMDb API:**

Benefits to Users: Users can get detailed movie information, including plot summaries, cast details, release years, and more. The API provides poster images, enhancing the visual appearance of movie-related applications. Users can search for movies using their IMDb IDs or titles, providing flexibility in how they access the information. Users can filter results based on the type of content they're interested in, whether it's a movie, series, or episode. Simple HTTP requests allow movie data to be retrieved quickly and easily.

**NinjasAPI (Celebrity API from Ninjas API):**

Benefits for users:

Users can get comprehensive details about famous individuals, including net worth, profession, date of birth, height, and nationality.

Users can search for celebrities by name, making it easy to find information about their favourite personalities. Users can filter results based on net worth, nationality, and height, allowing for more targeted searches.

Height and Net Worth Details, The API provides specific details such as height and net worth, which can be interesting for users interested in these aspects of celebrities.

Up-to-date data: Similar to TMDb, the API aims to provide accurate and up-to-date information about celebrities.

Finally, TMDb Movies API, omdbapi, and ninjasAPI are a set of APIs that provide a comprehensive set of features for users interested in movies and celebrities. These features include personalized experiences, enhanced recommendations, visual appeal, and educational value. Users can explore a wide range of movies, watch trailers, and meet their favourite celebrities, all in one app. Developers can use sophisticated recommendation algorithms to suggest movies based on users' favourite actors or directors, creating an engaging user experience. Access to poster images, wallpapers, and celebrity photos enhances the app's visual appeal. These APIs also provide educational insights into the world of movies and the lives of celebrities, making it a versatile and engaging platform for users.

**Reasons for choosing APIs:**

TMDb API:

TMDb provides comprehensive and up-to-date information about movies, making it ideal for viewing a variety of movies on the site. Its search and discovery features allow users to find specific movies or explore popular movies easily.

OMDb API:

The OMDb API complements TMDb by providing additional details about movies, including plot summaries and movie-specific details. The API provides images for posters, which enhances the visual appearance of the site.

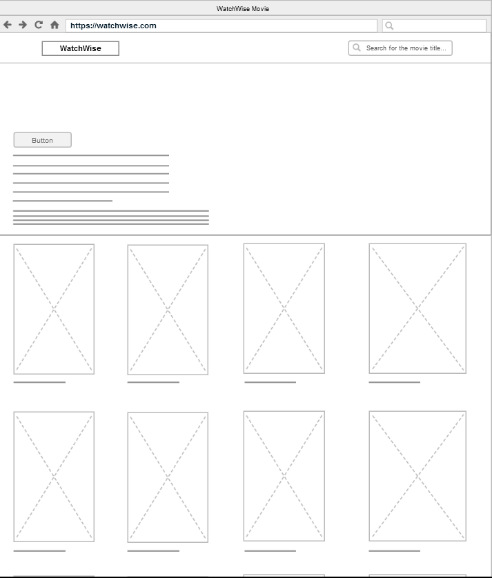
Ninjas API (Celebrity API):

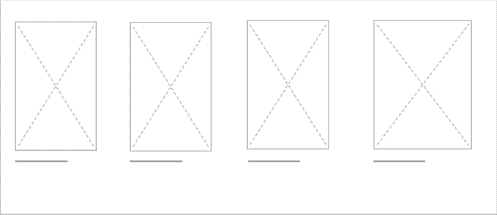
The Celebrity API adds a unique feature by providing detailed information about celebrities, including net worth, profession, and date of birth. With which users who are interested in movies and celebrities can enjoy a more engaging experience through a set of these APIs.

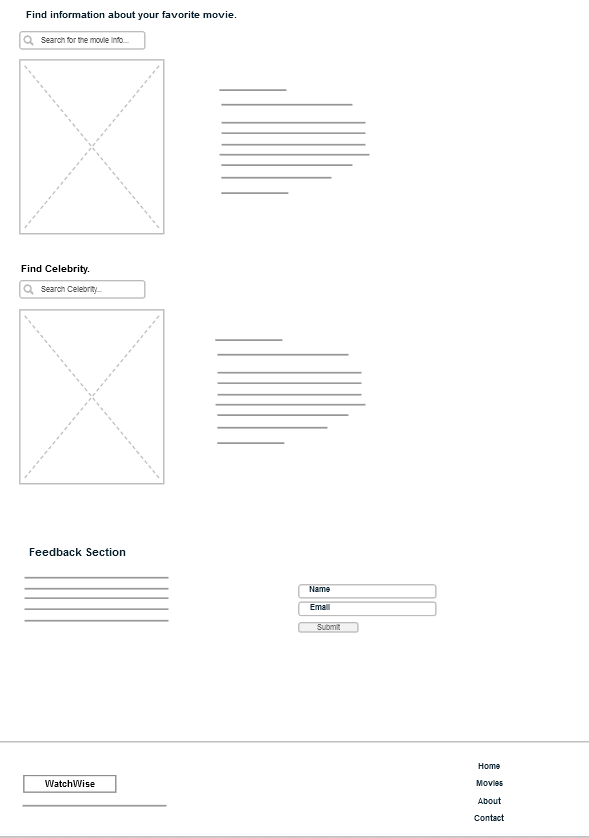
The chosen APIs are well suited for creating a feature-rich movie related website. The TMDb API provides a wide range of movie data, OMDb supplements it with specific details, and the Ninjas API adds a unique touch by providing comprehensive celebrity information. This combination ensures that users have access to a diverse and engaging range of content on the website.

**Q2** • Construct a wireframe diagram for the manager-requested application, incorporating the previously selected APIs.

**Answer:**







**Explanation of the details of the wireframe interface diagram:**

Which Director's Request Movie App Wireframe Interface Diagram includes a user-friendly design that leverages the power of The Movie Database (TMDb) API, OMDb API, and ninjasAPI. The primary goal is to provide a smooth and engaging experience to users who want to explore movies, view details, and discover information about celebrities.

**Home page:**

* Head section:
* Logo and app name.
* Navigation bar containing sections for movies, celebrities, and search.

Search bar:A prominent search bar in the middle of the page for users to search for movies.

**Featured Films Section:** A dynamic carousel showing popular and trending movies fetched from TMDb.

Film Section:Movie cards: A grid layout displays movie cards with basic details (poster, title, year of release). Clicking on a movie card will expand it to reveal more information (OMDb API).

Filter options:Filtering options for users to explore movies based on categories like 'Popular', 'Top Rated', 'Upcoming' etc.

research results: Displays search results based on user queries, using the TMDb API.

**Movie details:**

Header with navigation: Back button for navigation.

Film title and year of release.

Movie details: High quality movie poster.

Detailed information about the movie (genre, running time, release date).

Split the trailer with the embedded YouTube player using the "react-youtube" library.

Cast and crew:Displays a list of the main cast and crew members involved in the film.

**Celebrity Section:**

Celebrity cards:A grid layout displays celebrity cards and names. Clicking on a celebrity's card will expand it to reveal more information.

Celebrity details page:

It displays in-depth details about selected celebrities, including net worth, date of birth, profession, age, and nationality. It uses data fetched from ninjas API.

**Feedback section**: which collects users’ opinions and comments about their experience with the site and looks at improving it continuously.

**Footer section**: which shows the WatchWise website logo and also the ownership rights.

This wireframe interface diagram envisions a visually appealing, user-centric application that seamlessly integrates the power of TMDb, OMDb, and ninjasAPI. Users can easily explore movies, discover details of celebrities, and enjoy rich multimedia experience, making it a comprehensive and attractive platform for movie lovers.

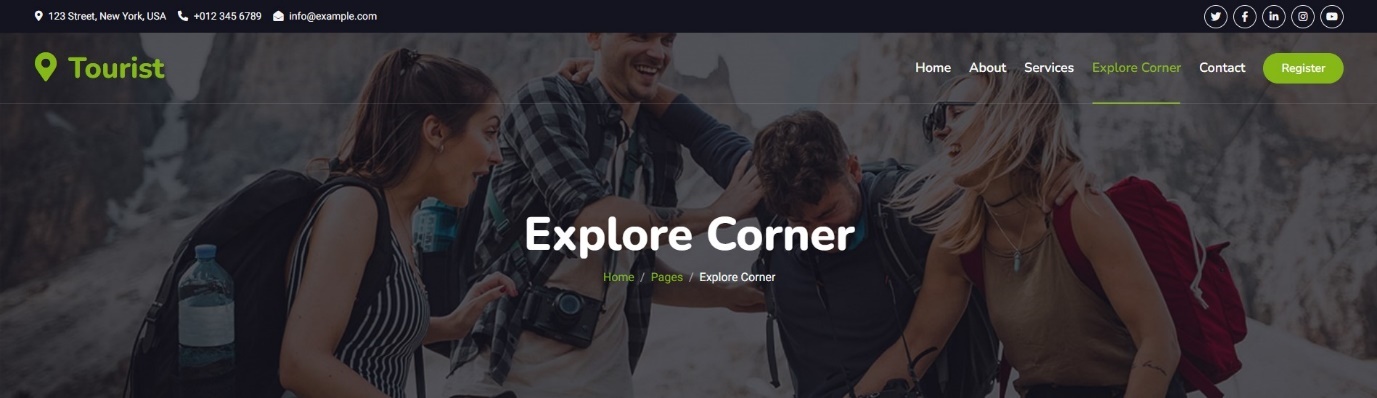
**Task 3:**

**Q3** • Implement the selected APIs from task 2.1 within the "**ExploreWonders**" web application to establish the **City Explorer** **Corner**.

**Answer:**First, I will explain how to implement the Explore Corner section for the **ExploreWonders** website, through the page sections, and what is the function of each section in the page. I will discuss the results of implementing the APIs that I used and the way they are presented to users:

**Top bar:** through which contact information and social media links are displayed in a dark bar at the top of the page. The information includes address, phone number, and email. Social media icons are used for Twitter, Facebook, LinkedIn, Instagram and YouTube.

**Navigation section:** The navigation bar that contains links to different pages of the site (Home, About, Services, Explore Corner, Contact). It also contains a section containing a title (“Explore Corner”) that I created and implemented, a navigation bar that indicates the path (Home > Pages > Discovery Corner) and a background, which gives an aesthetic to the page.

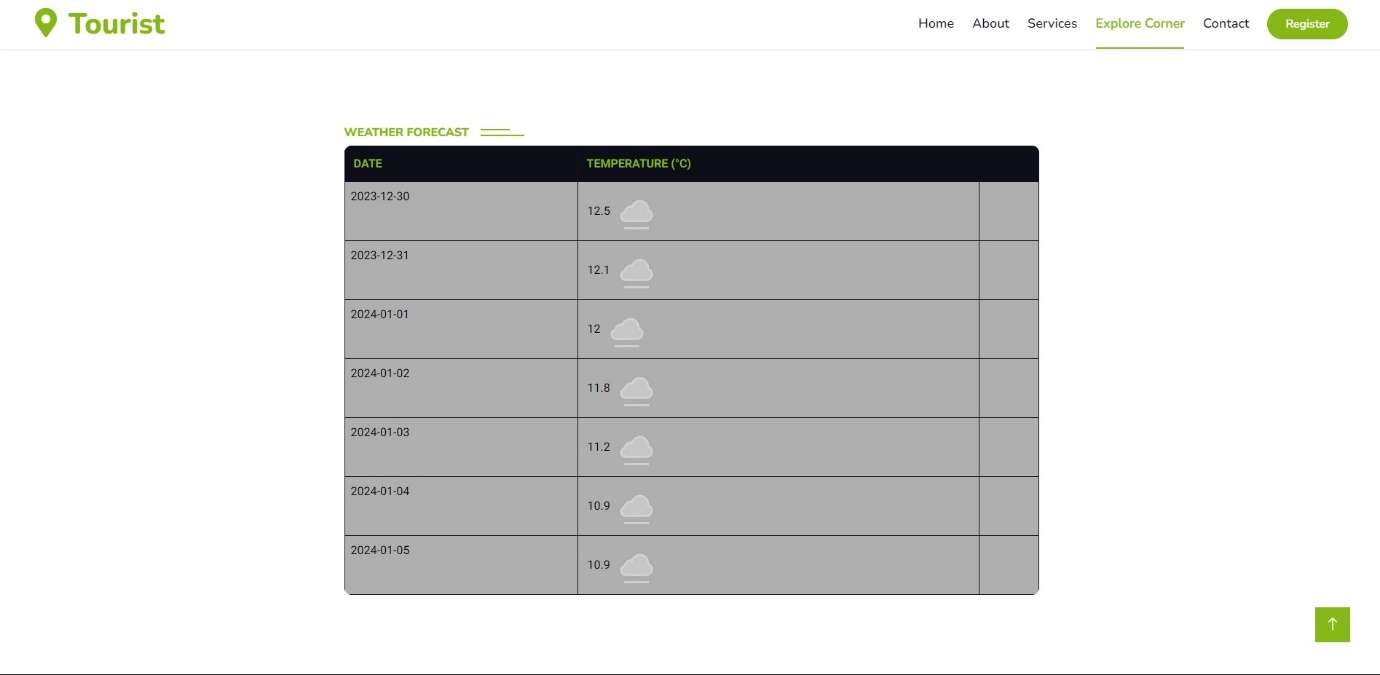


**Service Section:** In which users can search for holidays using a form that contains entries for the country and year. The results are displayed in holiday schedules and weather forecasts. The various input fields are designed using Bootstrap for a clean look. JavaScript functions are used to handle the search and display the results dynamically. From which users can retrieve a list of holidays for a particular country and year. Dynamic content of the Holiday Calendar Search Results table. The information includes the date, holiday name, and type. The table is designed for clarity, with large text and colourful backgrounds.



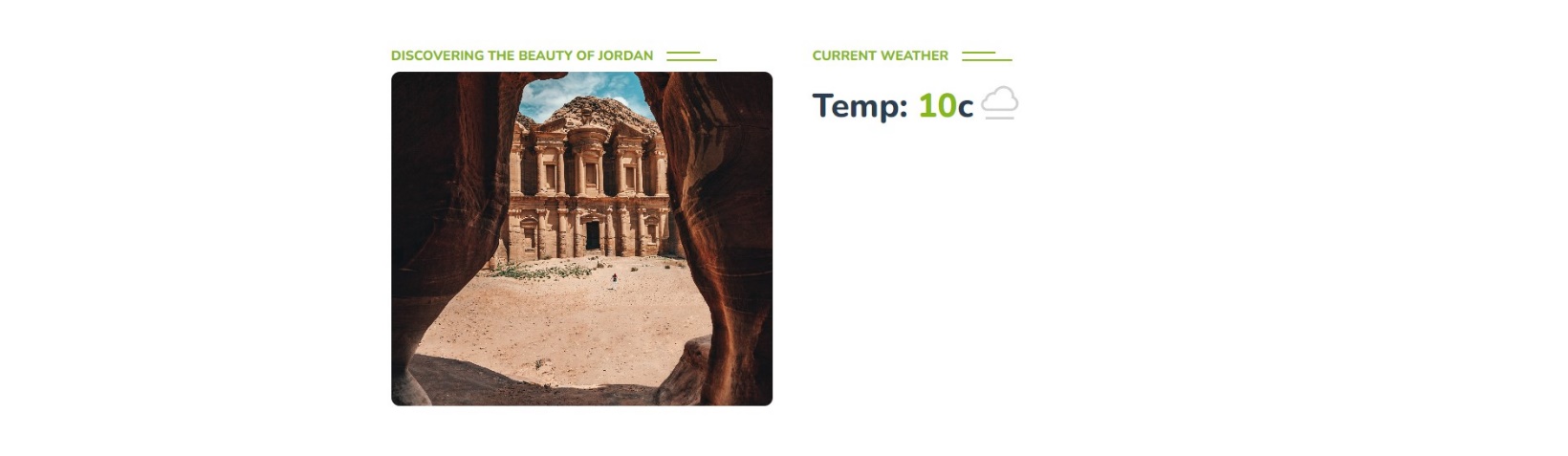


Also, based on the user entering the country or city he wants, search results will appear for a feature that benefits the user by displaying search results for seven-day weather conditions for the country or city the user wants and bringing the 7-day weather forecast for the specified destination. Dynamic population of Weather Forecast table with date and average temperature. The forecast includes an icon representing the weather for each day.



A section that displays the beauty of the destination with a picture of the country that the user wants and information, by having the user enter the name of the country and year when entering in the holiday search section for the country or city that the user wants. Dynamically fetches an image related to the searched country, enhancing the visual appeal. The photo is displayed in the "Discover Beauty" section. It promotes the user to the destination he wants.

Current weather: Displays the current temperature and weather icon for the selected destination. Advantages: It displays the current temperature and weather icon for the selected destination by having the user enter the name of the country or city and the year in the Holidays section, which will show a direct current weather forecast for the country or city he chose.



Footer: which contains sections for the company, contact, gallery, and newsletter. Social media links and newsletter signup form are included. There is copyright information and a back to top button.



**Benefits to users of the APIs you have implemented and placed in the services provided by the site:**

**Holiday search:**

* Easy to plan: Users can easily search for holidays based on their preferred country and year, making travel planning easier.
* Comprehensive information: The service provides not only holiday schedules, but also weather forecasts, allowing users to plan their trips more comprehensively.
* Clean UI: Input fields designed with Bootstrap contribute to a clean and modern look, enhancing the overall user experience.

**Weather forecast:**

* Location-Specific Forecast: Users take advantage of location-specific weather forecasts, providing a seven-day forecast for the chosen country or city.
* Real-time updates: The dynamic number of weathers forecast table ensures that users get up-to-date and accurate information.
* Iconic representation: Weather icons provide a quick visual representation of the forecast, helping users understand weather conditions at a glance.

**Discover the beauty section of the country:**

* Visual Appearance: Users can visually explore the beauty of their destination of desired countries and cities through dynamically fetched images.
* Engaging experience: Displaying images of a destination enhances user engagement and fosters a sense of excitement about the chosen location.
* Personalized Content: Section content is designed based on user input, creating a personalized and immersive experience.

**Current weather:**

* Instant Updates: Users receive instant information on the current temperature and weather icon for their selected destination.
* Relevance to travel plans: Direct integration with the vacation search section ensures that users get real-time weather updates, which helps in making travel decisions.
* Convenience: Users can quickly check current weather conditions without navigating to a separate page, providing a seamless experience.

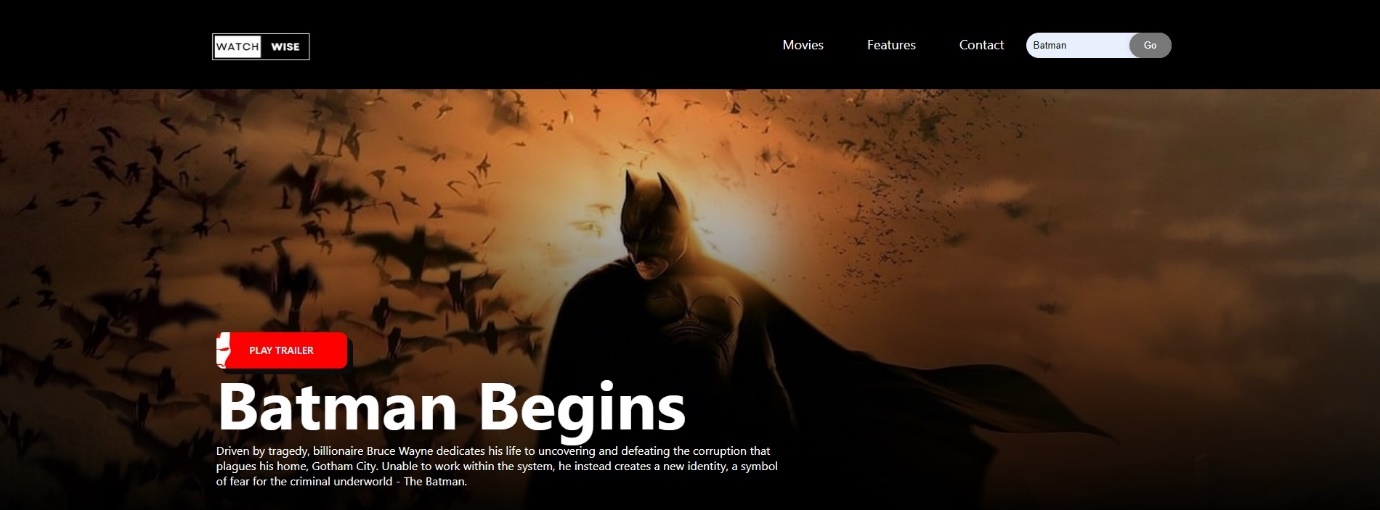
**Q2 •** Employ HTML, CSS, and JavaScript to produce the web-based application designed in task 2.2. Implement the chosen API or range of APIs.

**Answer:**

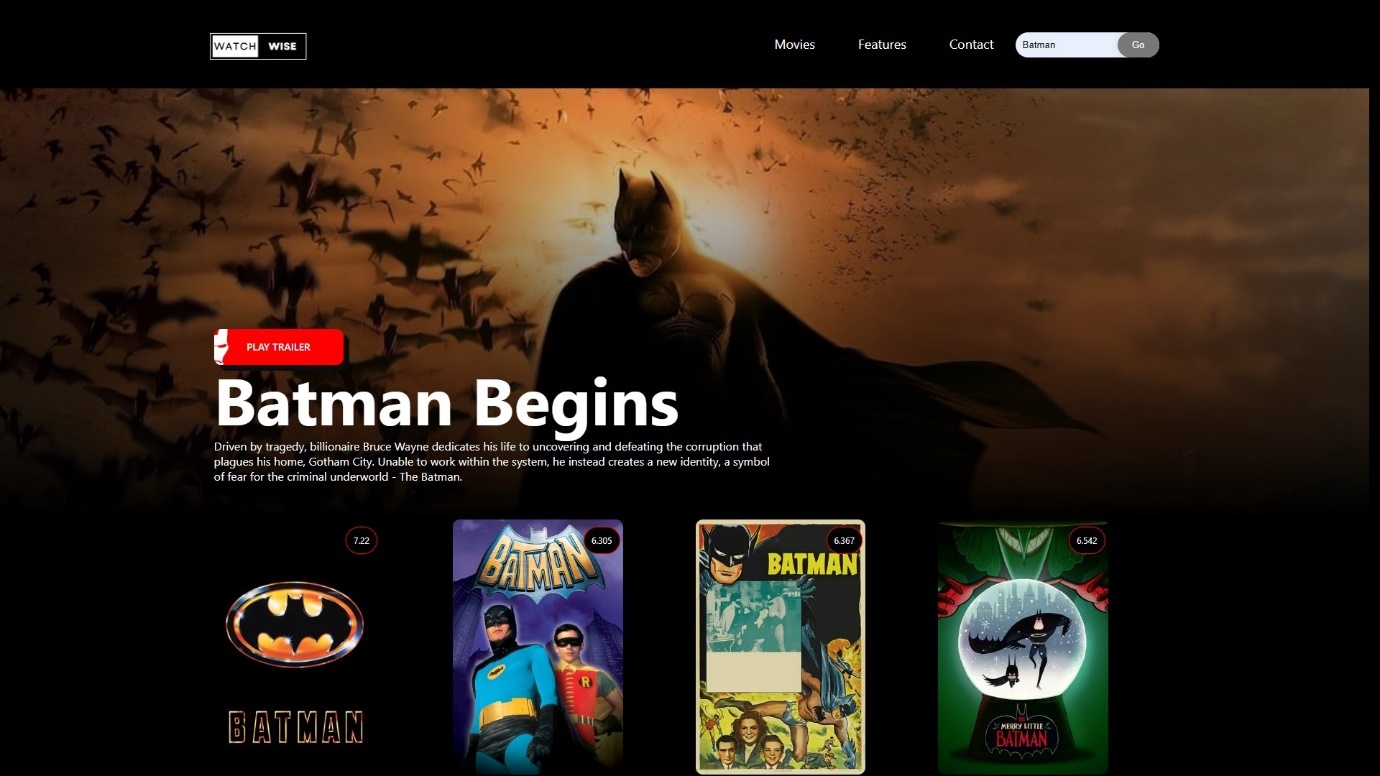


In the design and implementation of the **WatchWise** movie website, I used React, which is interactive. Node.js is a fast, secure, and scalable JavaScript framework. It provides a great user and developer experience. Therefore, I will explain how to implement the **WatchWise** movie website, through the page sections, and what is the function of each section on the page. I will discuss the implementation results of the APIs I used and how they are presented to users:

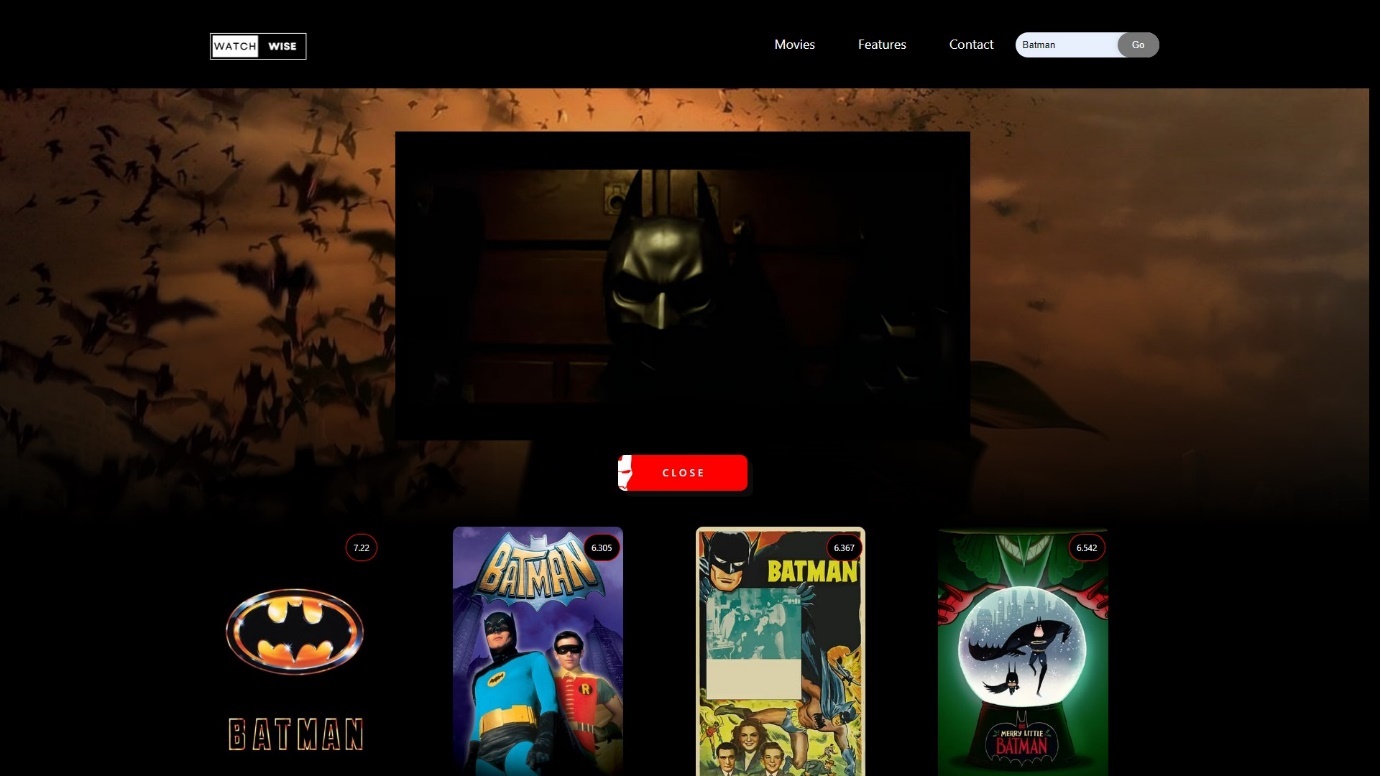
In the Navbar or Header section: The logo is displayed in the upper left corner of the page. A navigation bar containing links to different sections of the page (Movies, Features, Contact). A search bar is included for users to search for the movies they want.



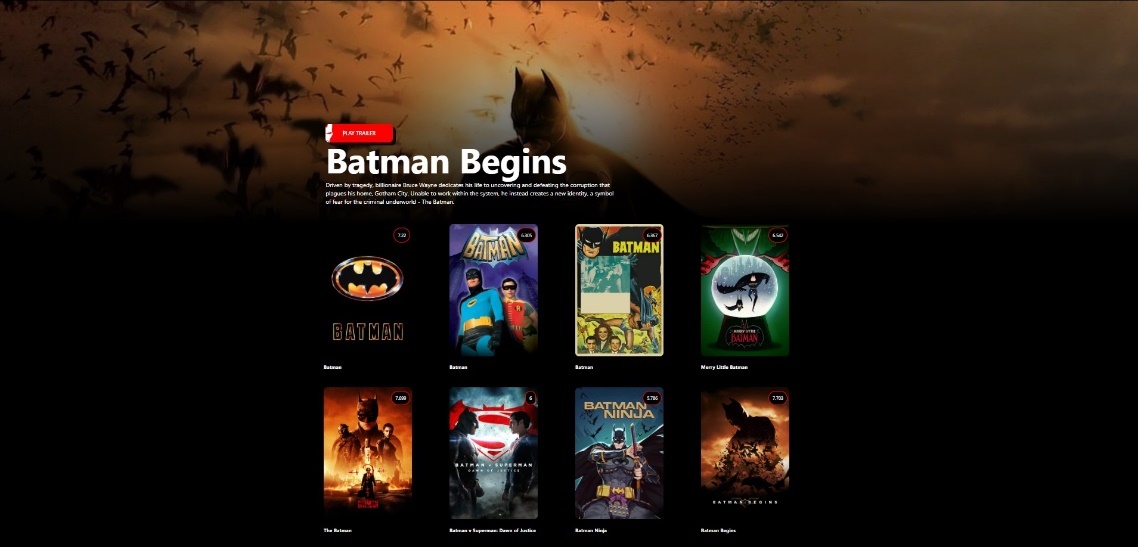
Main Section: In a main section with a background image that changes based on the selected movie. The background is created using the background path of the currently selected movie from the Movie Database (TMDb). The main content also includes information about the selected movie, such as the title and overview. Trailer - If a trailer is available, there is a play button that allows users to play the trailer using the YouTube component.



YouTube API: The YouTube API is used to embed movie trailers and play them directly on the website. In the main section, if a trailer is available for the selected movie, the play button allows users to play the trailer using the YouTube component. The app fetches movie-related videos or trailers from the TMDb API, and the YouTube API handles the embedding and playback process.



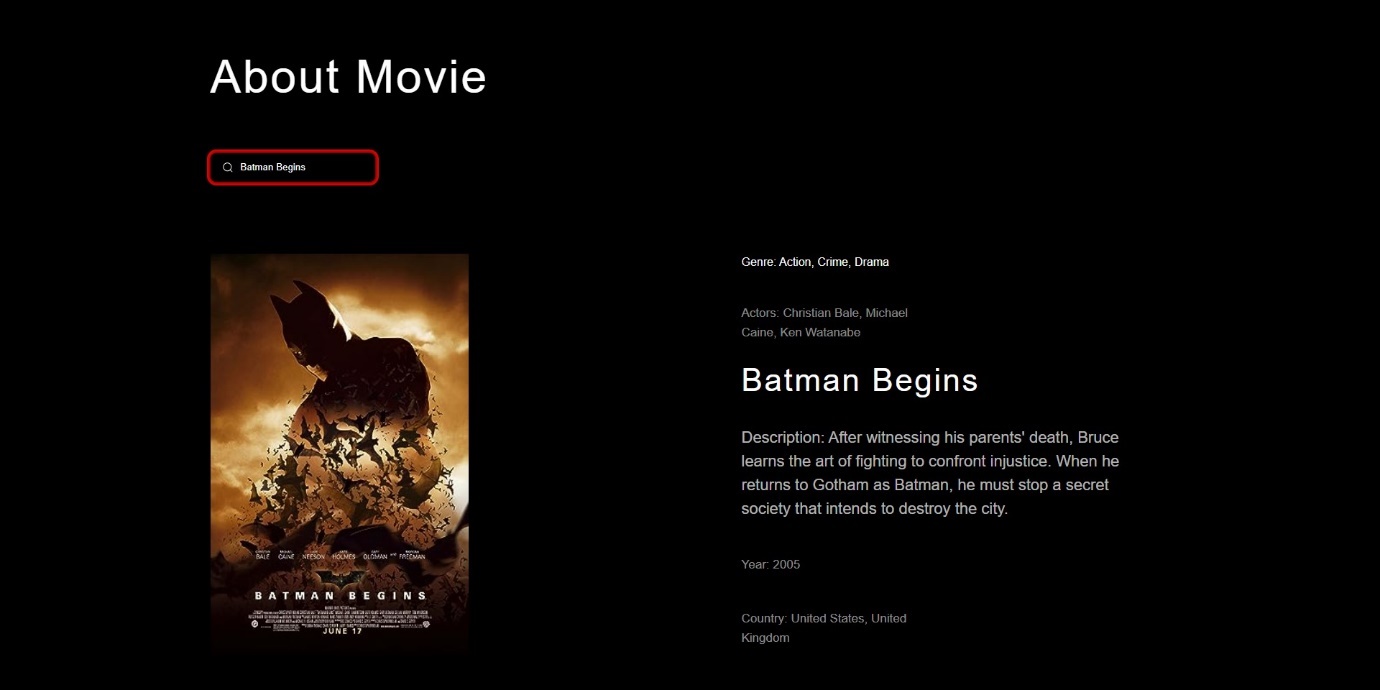
Movie List Section: This displays movie posters, which displays a list of movie posters. Each poster is a clickable element that allows users to select a movie, view its details, and watch a movie trailer.



The app fetches data from the Movie Database (TMDb) API for information about movies, including details and video clips (trailers).

The Movie Database (TMDb) API: The TMDb API is used to fetch information about movies, including details and video clips (trailers). The main section of the site, especially the background image, changes dynamically based on the selected movie. This is achieved by fetching the background path from TMDb. Movie details, such as title and overview, are sourced from TMDb to fill the main content section. Movie posters in the movie list section are fetched dynamically from TMDb.

Section about films: which begins the section with a prominent header displaying “About the film”. A search bar is provided through which the user can search for the movie information he wants, such as the type of movie, the date of the movie’s release, its name, the most prominent actors, and the country that produced the movie. Also, when the results of the search entered by the user are displayed based on the name of the movie, movie posters related to the name of the movie entered by the user will appear. Images are fetched dynamically from the Open Movie Database (OMDb) API or use a default movie image if it is not available. Movie details: Genre information, actors, title, description of the movie, year and country are displayed. Details are formatted in a way that is visually attractive and easy to read.



Open Movie Database (OMDb) API: The OMDb API is used to collect detailed information about movies, including genre, cast, plot, year of release, and country. The About Movies section allows users to search for movie information. Movie posters are displayed based on search results. Movie details such as genre, cast, title, description, year and country are sourced from OMDb API and presented in a visually attractive format.

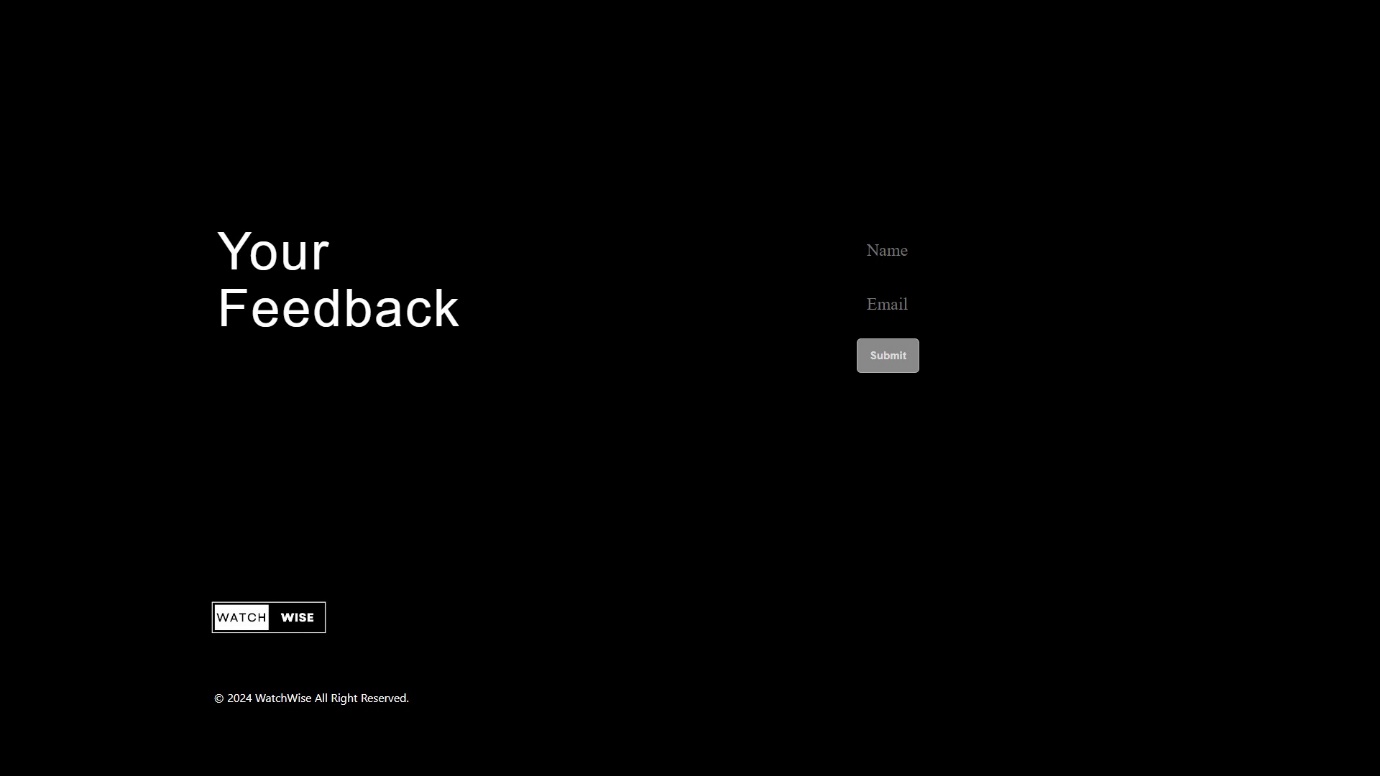
Find Celebrities Section: The section has a header titled “Find Celebrities”. A search bar is provided to search for information about celebrities. The user has to enter the name of the famous person or actor, and information about this actor or famous person will appear. The interactive search icon allows users to launch the search function. Celebrity information is displayed in a container with a similar layout to the Movies section. Each result includes a photo, celebrity details and related information.



Ninjas Celebrity API: Ninjas Celebrity API is used to fetch information about celebrities, including name, date of birth, profession, age, and nationality. The Celebrity Search section allows users to search for information about celebrities by entering the name of the famous person or actor. Celebrity details, including photo, are displayed based on search results from the Ninjas Celebrity API.

Feedback Section: A dedicated section for user comments has been included. The form includes fields for username, email, and a submit button. The model has a modern and minimalistic design. This allows users to submit their ideas and comments through their experience of visiting the **WatchWise** website, which means that we take their opinions into consideration and develop the website to effectively improve the users’ experience.

Finally, the footer section, which contains the site’s logo and also the text explaining the site’s ownership rights.



**Benefits and features that the user will get from adding and implementing APIs to the WatchWise website:**

Comprehensive movie information: Users can access detailed information about movies, including genres, actors, plots, years of release, and country of production. API Contribution: The OMDb API provides comprehensive details of the movie, which enhances the user's understanding of the content.

Dynamic Movie Posters: The visual representation of movies through dynamically fetched posters provides an attractive and eye-catching user interface. API Contribution the TMDb API provides dynamic movie posters, enhancing the visual appearance of the About Movies and Movie List sections.

Embedded Trailers: Users can watch movie trailers directly on the site, enhancing the overall movie watching experience. API Contribution the YouTube API is used to embed and play trailers, providing users with a convenient way to preview movies.

Celebrity Information: Users can find detailed information about celebrities, including their names, birthdays, profession, age, and nationality. The Ninjas Celebrity API contributes celebrity details, enriching the Celebrity Search section.

Interactive Search Function: Search bars in the “About Movies” and “Celebrity Search” sections allow users to interactively search for specific movie details or celebrity information. API Contribution APIs enable real-time searching and retrieval of relevant data, providing users with instant results.

User Feedback Section: Users can submit feedback, suggestions and comments, enhancing the feeling of user engagement and community engagement. API contribution Although not explicitly stated, if the feedback model involves server-side operations (for example, storing comments in a database), back-end APIs may play a role.

Modern and minimalistic design: The modern and minimalist design of the site contributes to a visually attractive and user-friendly experience. API Contribution Although not directly related to the APIs, the design may be influenced by the flexibility and data visualization capabilities provided by the movie-related APIs.

Ease of navigation: The navigation bar facilitates easy navigation between different sections of the site (Movies, Features, Contacts). APIs contribute to the content and functionality of these sections, ensuring a seamless experience.

Proprietary information: Users are clear about site ownership rights, which builds trust and transparency. Although proprietary rights are not directly related to APIs, APIs may play a role in back-end processes related to website management.

Finally, my WatchWise website leverages various APIs to provide users with a feature-rich and visually appealing experience, providing comprehensive movie details, interactive search functionality, and the ability to handle celebrity information and user comments. The integration of APIs enhances the website's capabilities, making it a valuable resource for movie fans.

**Task 4:**

**Q1** • Formulate a minimum of 3 distinct test cases for conducting white box testing on the "**ExploreWonders**" web application created in the culmination of task 3. Record the results for each test case.

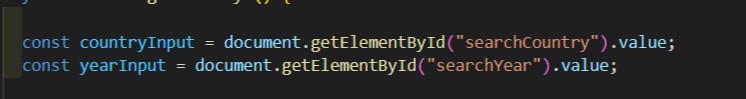
**Answer:**

**First, I will present the first test case, which is Test Case 1: Valid Input for Holidays API:**

The focus of this test case is to evaluate the behaviour of the “ExploreWonders” web application when providing valid input to the Holidays API. The goal is to ensure that the app is successful in fetching and displaying holidays when provided with a valid combination of country and year.

So, the primary goal of this test is to evaluate the application's ability to handle Holidays API responses appropriately and verify that it updates the user interface (UI) accordingly. By providing valid inputs, we aim to simulate a scenario where the application successfully interacts with the external API and seamlessly provides the retrieved vacation information.

**Retrieve User Input:**



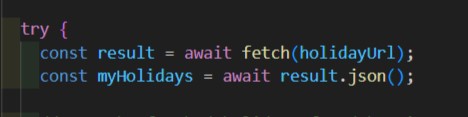
These lines fetch the values entered by the user for the country and year from the respective input fields in the web page.

**Build Holidays API URL:**



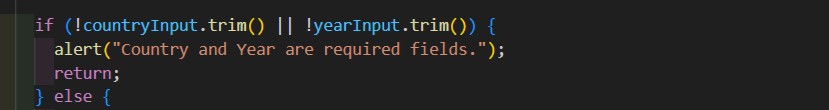
This line constructs the URL for the Holidays API using the entered country and year. It includes an API key for authentication.

**Fetch Holidays Data:**



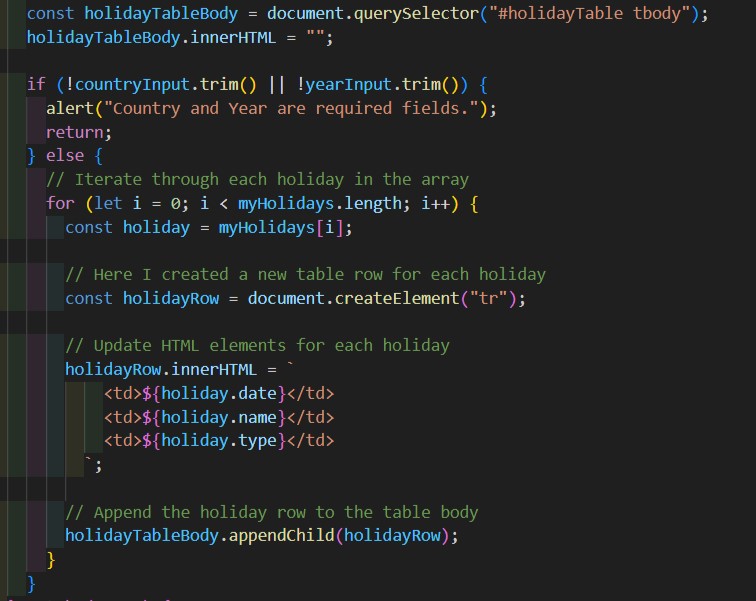
These lines use the fetch function to make an asynchronous request to the Holidays API. It then parses the JSON response into the myHolidays variable.

**Handle Empty Input:**



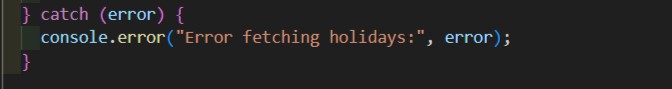
This part checks if either the country or year input is empty after trimming whitespaces. If so, it displays an alert message and exits the function.

**Update UI with Holiday Data:**



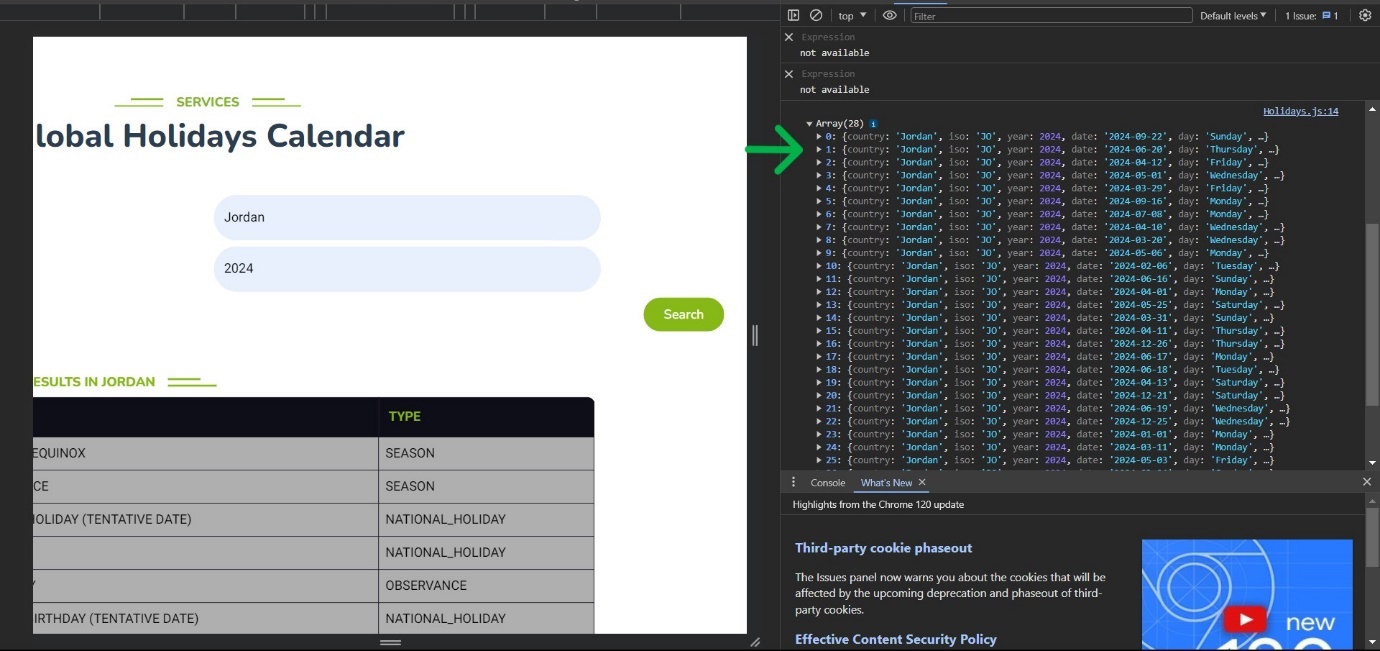
If the input is valid, the code proceeds to update the UI. It first clears the existing content in the holiday table. Then, for each holiday in the retrieved data, it creates a new table row (<tr>) and updates the HTML elements accordingly. Finally, it appends each row to the table body.

**Error Handling:**



In case of an error during the API request or JSON parsing, an error message is logged to the console.

When you have entered a valid country and year in the custom search bar. Then I pressed the "Search" button. I then verified that the API and application were successful in fetching and displaying holidays for the specified country and year. I ensured that the holidays offered were accurately displayed in the schedule.



**Result: Successful.**

The application successfully executed the test case by fetching and displaying holidays based on the valid country and year inputs provided. The UI has been updated with the correct holiday information in the table. This test case implies that the basic functionality of fetching and presenting holiday data works as intended when valid input is provided to the Holidays API. Subsequent test cases will explore different scenarios to comprehensively evaluate the application's robustness and user experience.

**Secondly, I will present the second test case, which is the revised test case 2: Error handling for the image search API (failed test case):**

So, the goal of this test case is to evaluate the behavior of the application when it encounters an error while calling the image search API. This test case specifically focuses on evaluating how the app responds to issues related to the Unsplash API, such as providing an incorrect API key. It aims to ensure that the application safely manages errors, and provides clear and meaningful feedback to the user in the event of API-related issues.

Description: Here I have temporarily modified the API key used for Unsplash API to an invalid key. You have entered a valid country and year in the search bar. I clicked the "Search" button.

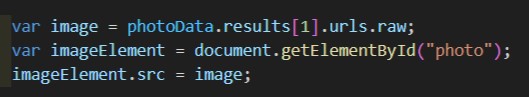


The searchPhotoUrl is constructed with the Unsplash API endpoint, including the query parameter for the country (${countryInput}) and the client ID



API Request:

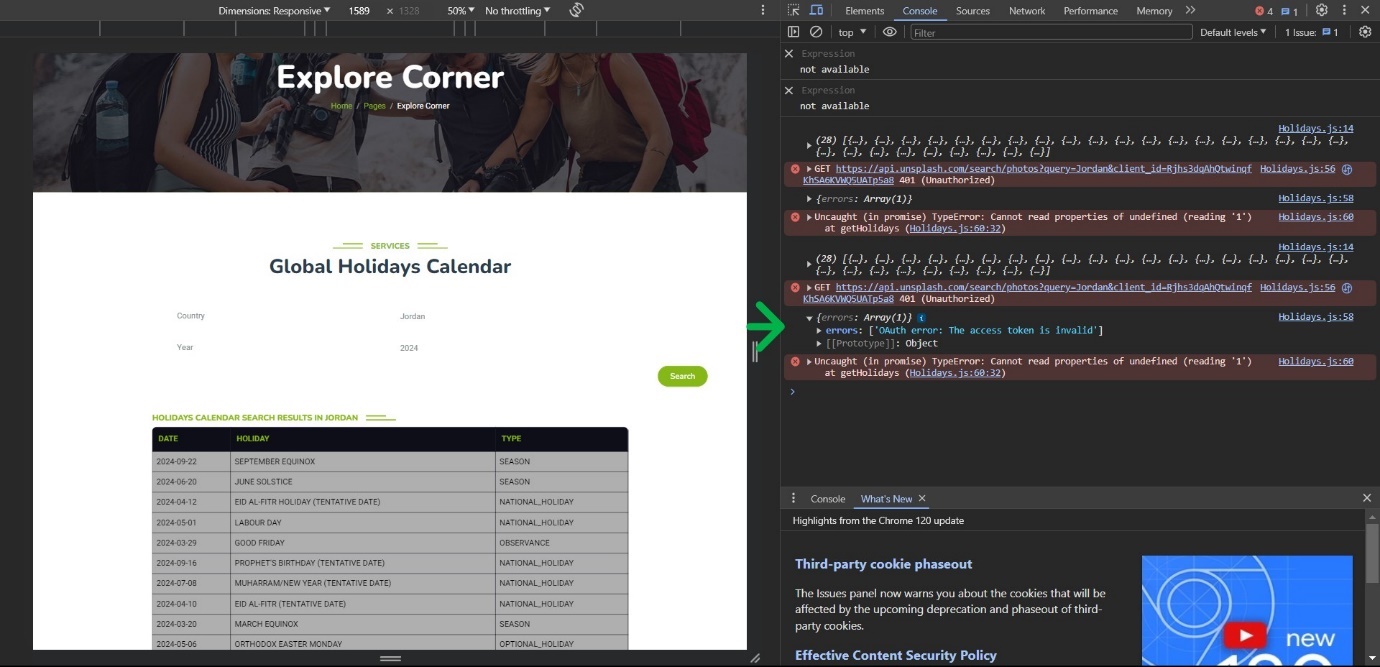
A fetch request is made to the Unsplash API using the constructed URL (searchPhotoUrl). The response is awaited, and the response.json() method is used to parse the response body as JSON. The resulting photo data is stored in the photoData variable.



An image URL is extracted from the second result of the Unsplash API response (photoData.results[1].urls.raw).

The image element on the web page, identified by the ID "photo," is located (document.getElementById("photo")).

The src attribute of the image element is set to the extracted image URL, effectively displaying the image on the UI.



**Result: failure.**

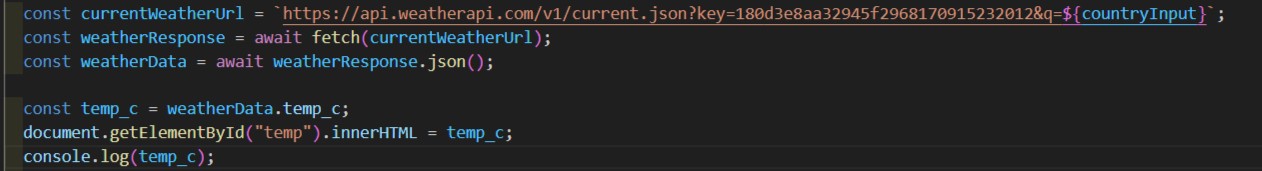
I checked the app through Inspect through the site's console, which gives an error message when encountering issues with the image search API.

**Thirdly, I will introduce the third test case, which is Test Case 3: Failure in fetching weather data:**

So, the goal of this test case is to evaluate the behavior of the application when there is a failure to fetch weather data from the Weather API.

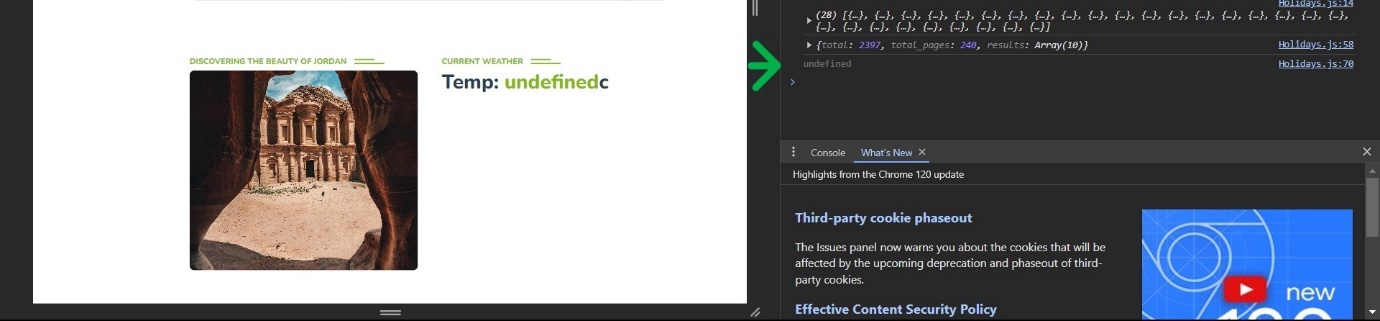
This test case critically examines how the application responds to errors or failures that may occur during the process of retrieving current weather information. It aims to ensure that the application safely manages various scenarios, including network issues, incorrect API keys, or any code execution errors that may arise during the Weather API call.

So many developers make the mistake of trying to directly access **WeatherData.temp\_c** without looking at the received **JSON data structure**. The JSON structure indicates that temp\_c is nested within the current property. Therefore, it should be accessed as **WeatherData.current.temp\_c**.

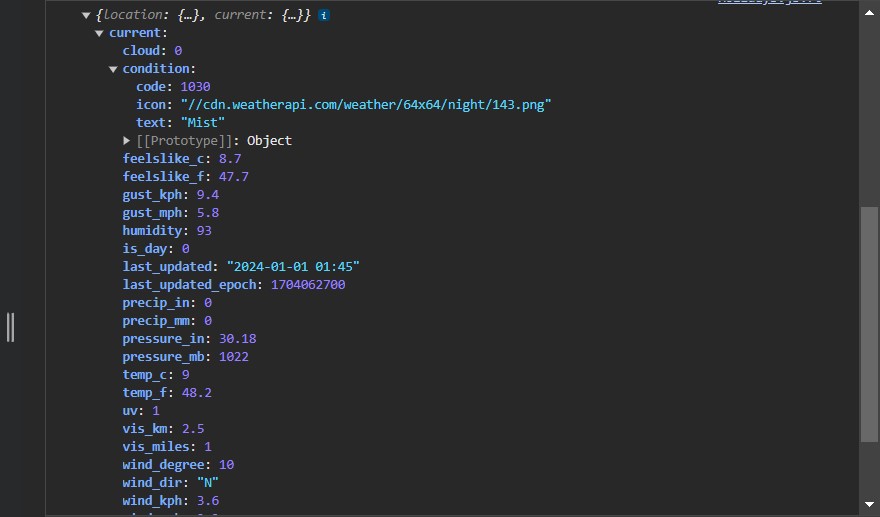


This Failure on the application response when trying to retrieve and display the current temperature (temp\_c) from the Weather API. The failure is primarily due to incorrect access to the temperature property within the JSON response.

Processing JSON data: The goal is to extract temperature data from the JSON response. The application attempts to extract the current temperature (temp\_c) directly from the WeatherData object.



So, I chose the solution by writing console.log(temp\_c); This is to verify the status of the test and that the value was not retrieved correctly. “**Undefined**” appeared in the console. Therefore, I entered the name of the country and year, and the problem of the current weather appeared in the country that I entered.



For the Current Weather API, the JSON is structured in a way such that the temp\_c property is located in the current object, and from the current object you tap into the temp\_c property that is located there. which means that the correct path to it is Object -> current -> temp\_c

**Result: Failure.**

The application attempts to access temp\_c directly from weatherData, property. However, the actual structure of the JSON response might have a nested property, such as weatherData.temp\_c. As a result, the direct access to temp\_c leads to an error, and the displayed temperature is either incorrect or not displayed at all.

**Q3** • Enhance the application based on the testing results obtained in task 4.1.

**Answer:**

**So, I will address, improve and resolve errors in the code - errors in the second case of the image search API (failed test case):**

This test case focuses on evaluating the application's behaviour when an error is encountered during an image search API call. The failure occurs when a temporarily invalid API key is used for the Unsplash API. This is due to the application failing to handle an API key error appropriately. This is due to an error in the private key of the Image API.

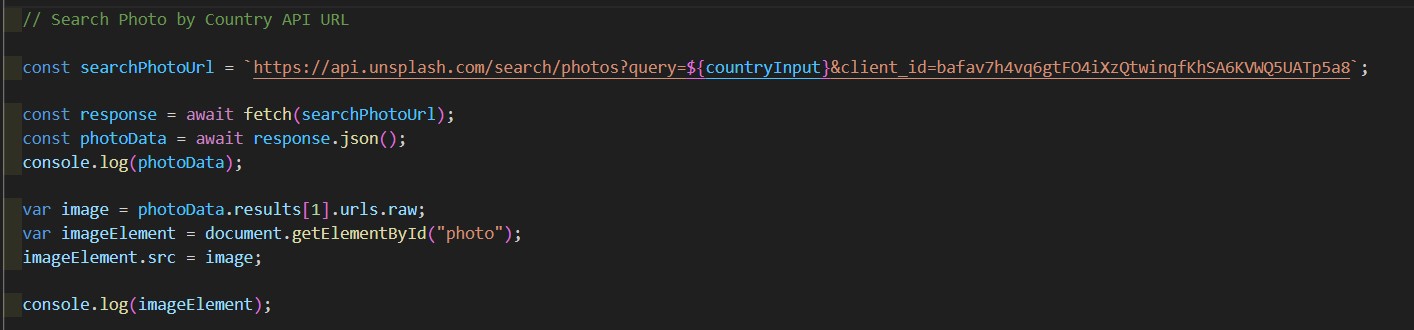
So here is an image that shows that the user interface key is wrong or incorrect.

There was an initial problem or issue related to an API key. An API key is often used for authentication purposes when making requests to external APIs.



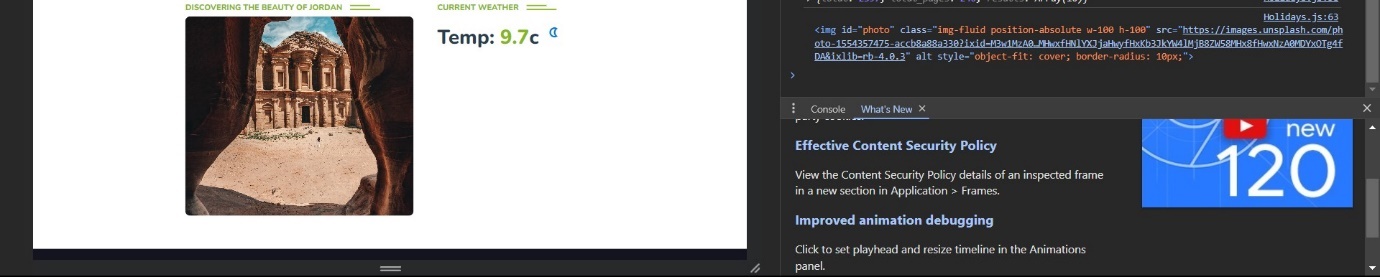
The solution here is that a special key must be set for the developer or programmer who sets a correct key and is authorized to access the data to fetch information and data from the Unsplash API website.

The issue was addressed by identifying and correcting the problem with the API key. This could involve ensuring the key is correctly formatted, has the right permissions, or is valid for the specific API being accessed.



Therefore, the image here shows that I modified the user programming interface key to a valid key that is available and authorized for one person, in order to collect the image data and show it to the user when he enters the name of the country or city and year.

So set up the console and error handling for the image search API. Therefore, console mode plays a crucial role in debugging and understanding the flow of your JavaScript code. When working with web applications, using the console is essential for recording information, errors, and debugging during development. Let's improve error handling in the Image Search API and explain the importance of console logs.



Following the correction of the API key, the application or script is now able to make successful API requests. The result of these requests is displayed in the console.

The console is a tool used in development environments to log information, and errors, and debug messages. In this context, it's being used to showcase the result obtained after resolving the API key issue.

The fact that the result is being displayed in the console suggests a successful operation. Developers often use console logs to verify that the expected data is being retrieved or processed correctly.

This statement reflects a typical part of the development workflow where issues are identified, addressed, and application behavior is verified through console logs.

Finally, this process indicates the success of your troubleshooting and error correction efforts. Correcting issues with API keys is a common task in API integration, and being able to see the desired result in the console is a positive result, indicating that the application is now working as intended.

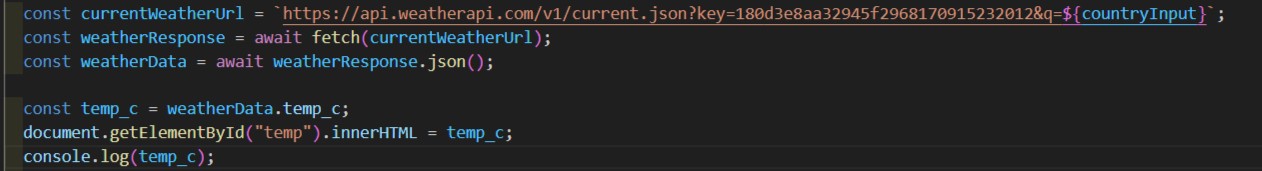
**Therefore, I will address, optimize and resolve bugs in the code - I will introduce the solution, processing and improvement of the third test case, which is Test Case 3: Failure to fetch weather data:**

The third test case for the Current Weather API has failed because the property temp\_c was not accessed properly. When going into the JSON data, WeatherData.temp\_c simply means that inside the Object, there is directly a property named "**temp\_c**" which is not true. the previous picture that was shown exactly how the temp\_c is accessed by first tapping into the WeatherData, from there you would tap into the current object, and inside the current object itself, there is a property named “**temp\_c**”. So, the mistake is quite clear.

To improve the application based on the test results obtained in Task 4.1, I will address the issue of incorrectly accessing the temp\_c property directly from the WeatherData object.

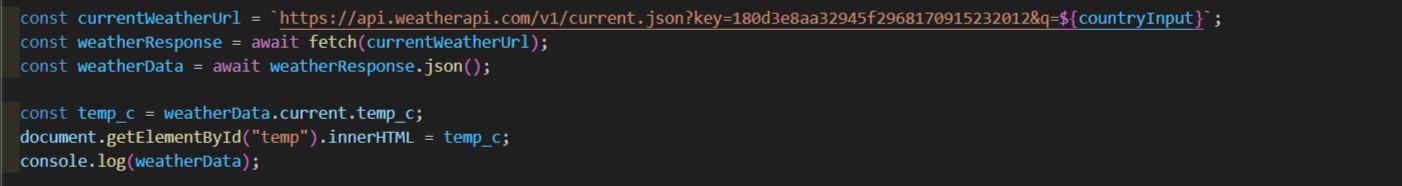
The problem here was that the temp\_c property was incorrectly accessed directly from the WeatherData object.

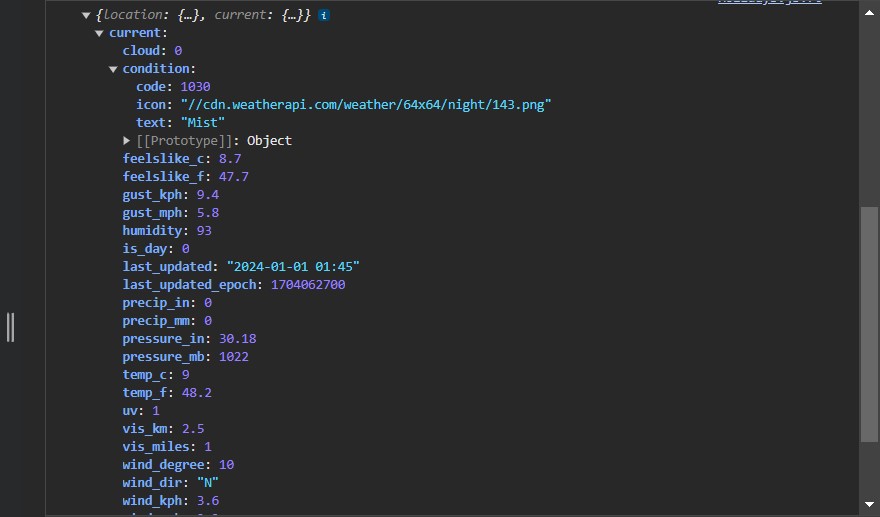
So many developers make the mistake of trying to directly access **WeatherData.temp\_c** without looking at the received **JSON data structure**. The JSON structure indicates that temp\_c is nested within the current property. Therefore, it should be accessed as **WeatherData.current.temp\_c**.



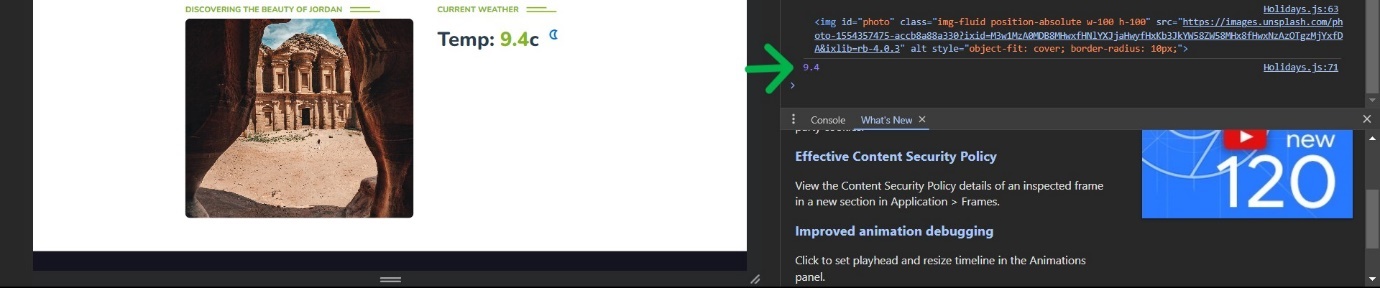
So, you accessed the temp\_c property correctly by navigating through the current object.

I used logging data during development and testing to understand the structure of the received JSON response and determine the correct property path.





The solution to this problem is that I as a developer have to go back to the JSON data and check the property name, and it's also better to just copy and paste the property name to avoid issues of the same nature as this one. What must be done for the code to work is that the correct nesting must be applied, and the correct property must be accessed.



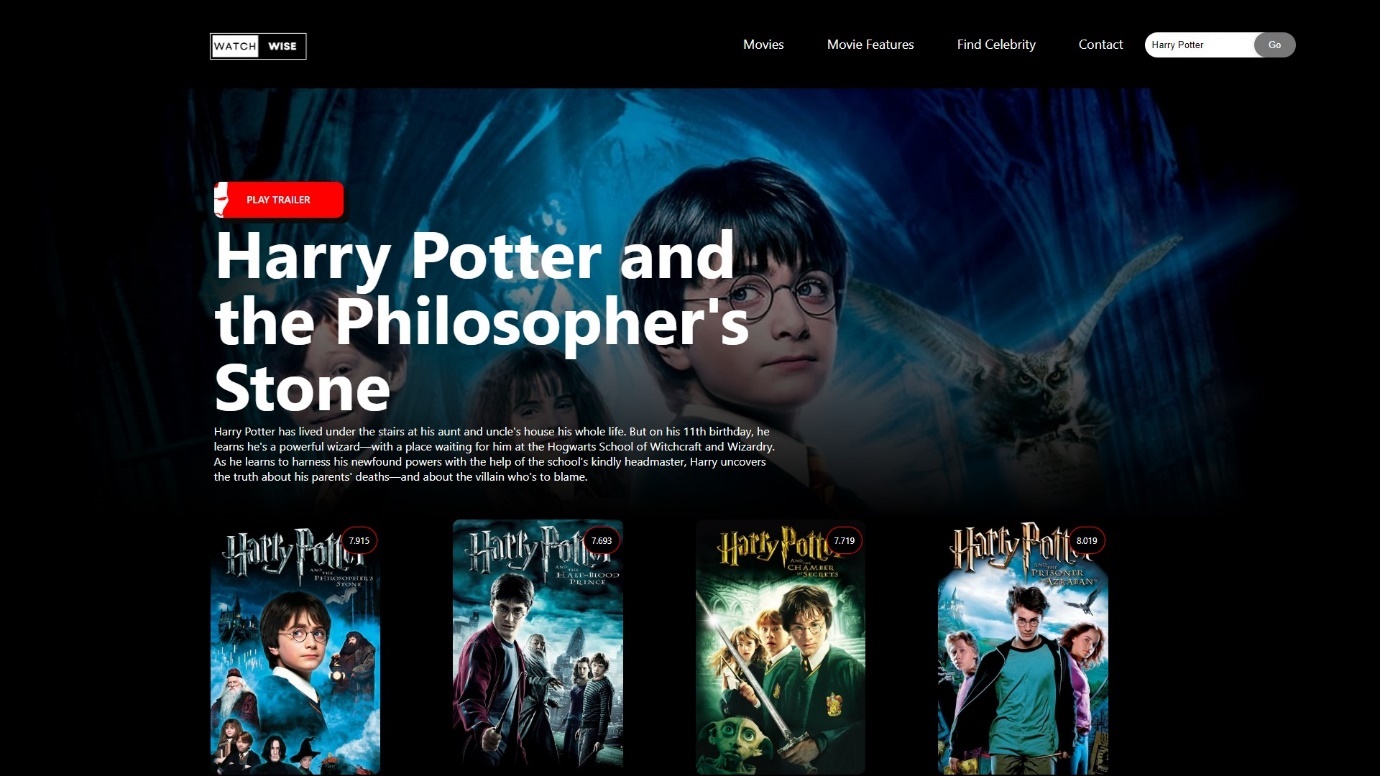
Finally, this solution works because this is the exact structure of its JSON data. JSON shows exactly how a developer can access a particular property from JSON and by following the right syntax, issues like this will definitely be reduced or eliminated.

**Q3** • Execute 'black box' testing for the APIs implemented in task 3.2, incorporating a minimum of 3 different test cases. Document each test case along with its result and make necessary adjustments.

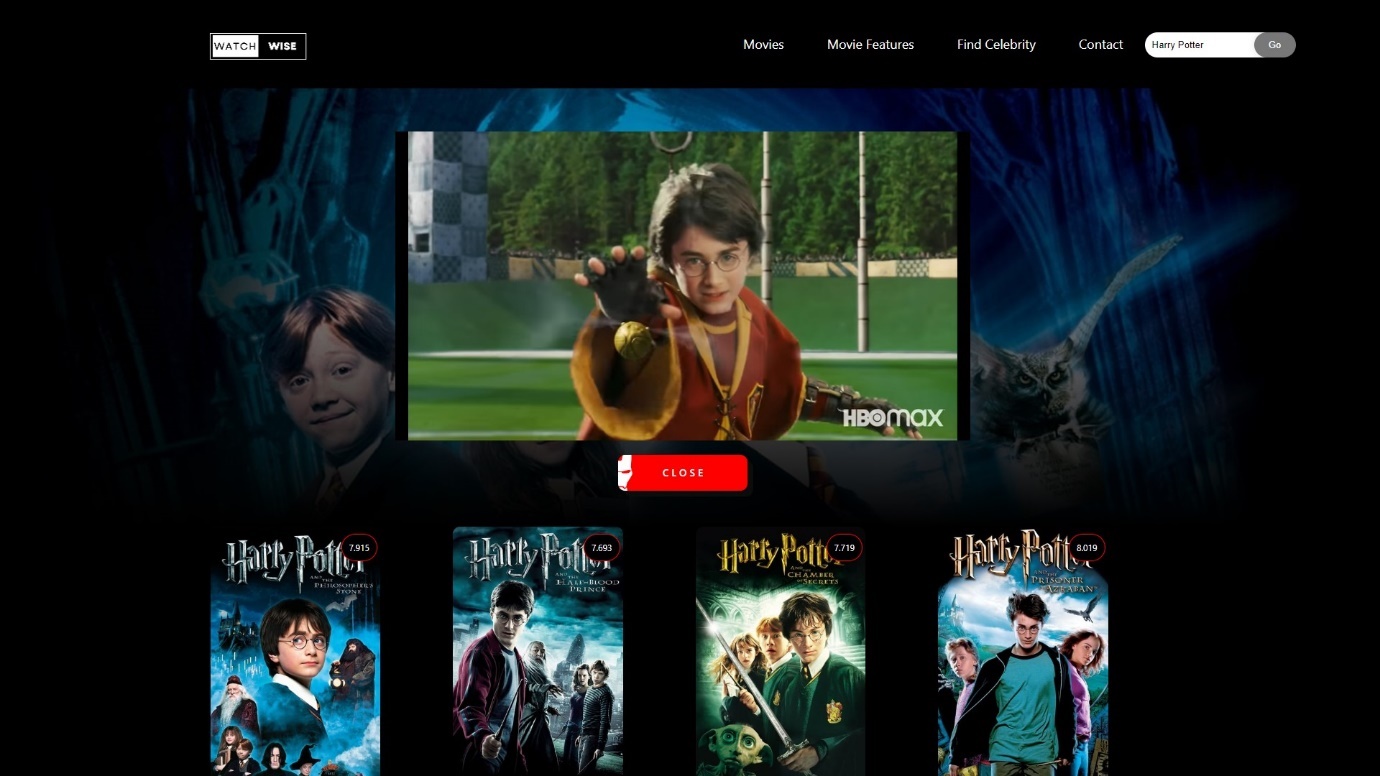
**Answer:**

The API responsible for fetching movie data from the Movie Database (TMDb). “**Black box**” testing involves evaluating the functionality of the system or website. This method allows me as a developer and software engineer to focus on expected output based on specific inputs, ensuring that the API works as intended. I'll be running about three distinct test cases, each designed to evaluate different aspects of the API's functionality. The goal is to document the results of these test cases, identify any discrepancies between actual and expected results, and make necessary modifications to enhance the robustness and reliability of the API. The following sections detail each test case, the inputs provided, the expected outputs, and subsequent adjustments made based on the test results.

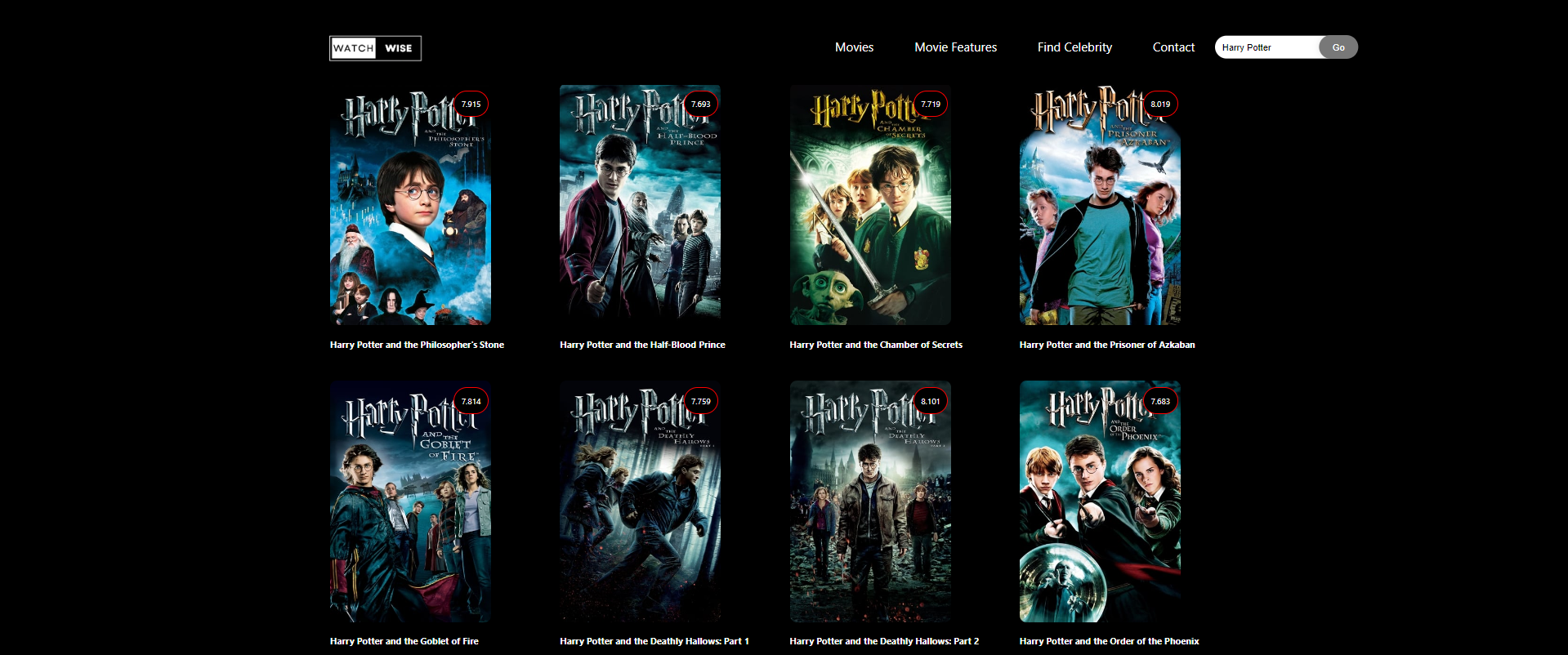
First, I will test the movie search section and display the results in the Watchwise website interface and display them to the user. So, I will search for a specific movie. I ensure that the API returns relevant movie posters and trailer information based on the search query.



Here I searched for the **Harry Potter movie** by going to the header and writing the name of the movie in the search section, then pressing Enter on the keyboard or Go in the search bar. You will notice that all the films related to the Harry Potter film series and a description of each film and title have appeared.



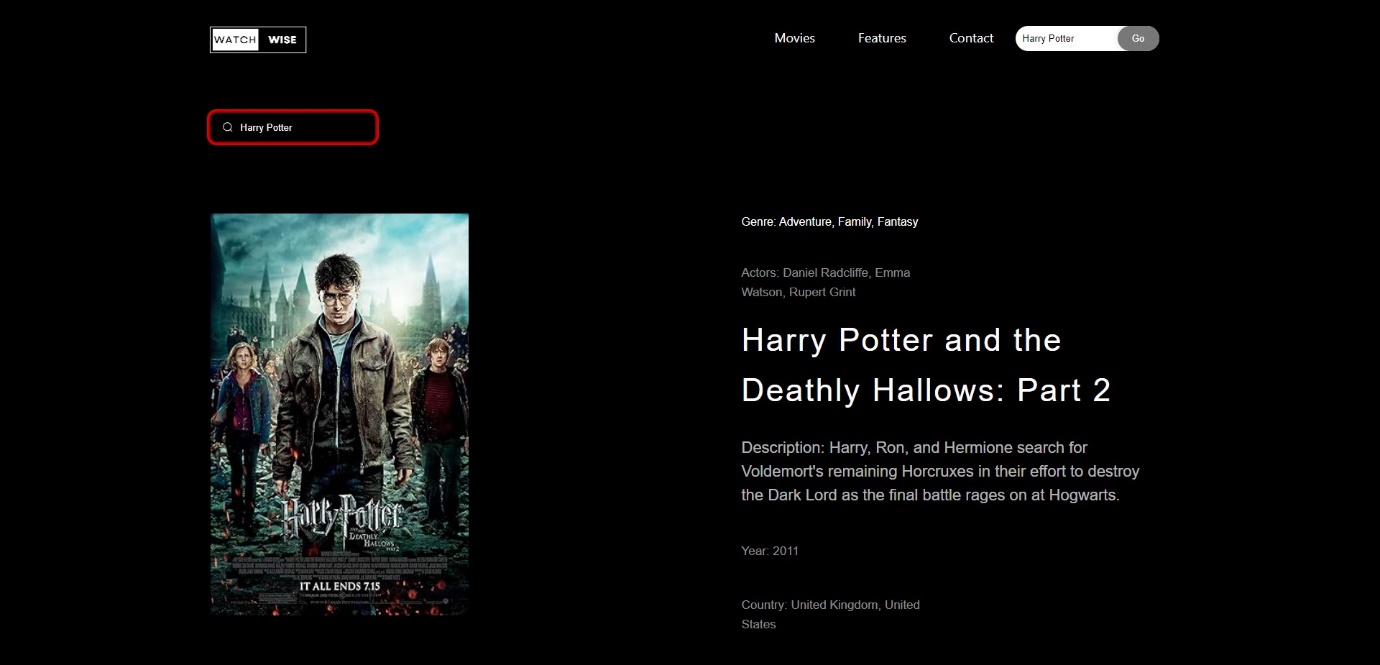
Here, after showing all the movie results, a trailer for each movie will appear when the user chooses the movie, he wants by clicking on the Play Trailer button above the movie title. The movie trailer will appear elegant and consistent, making the user experience great and effective at the same time. When the promotion is finished, the user can click on the close button below the trailer.



Finally, here all the movie posters for the movie that the user searched for will appear in all the series of the movie he chose elegantly and attractively. When he clicks on the movie poster, he will be shown the title of the movie, a description of the movie, and its trailer.

**The second test case - searching for information or a summary about films**, how to display movie posters, and how to display movie information such as the type of movie, the title of the movie, the release date of the movie, the most prominent actors, and the country where the movie was created.

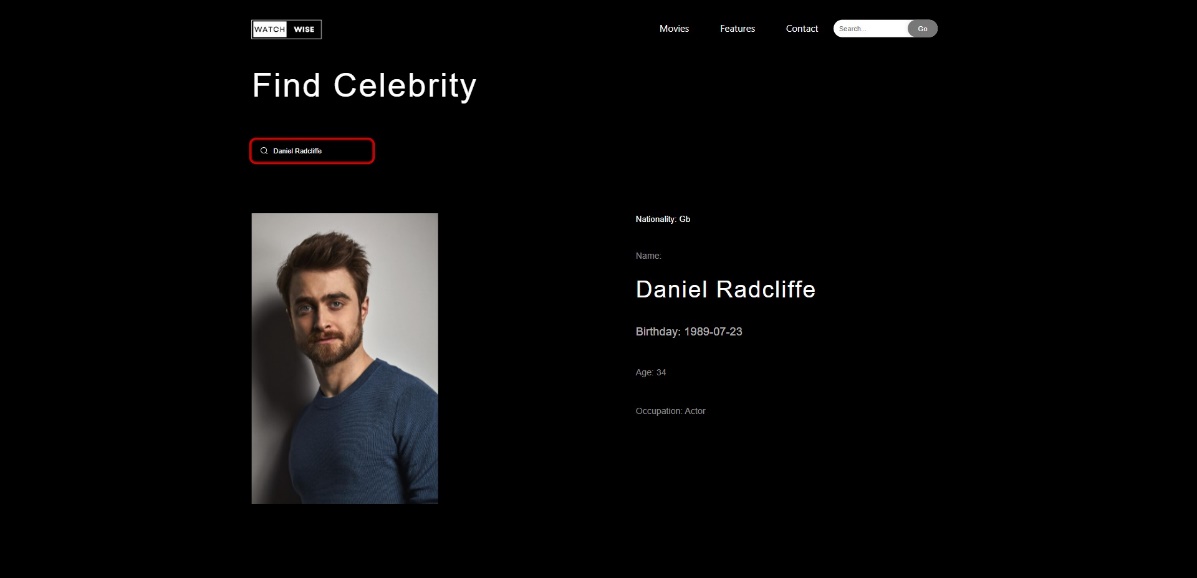
This code snippet represents the implementation of an API that interacts with the Open Movie Database (OMDb) to fetch and display information about movies. It is designed to handle user input, perform searches on the OMDb database, and present detailed information about each movie, including movie posters, genre, actors, title, plot description, release year, and country of origin.



First, through the search bar located below “About Movie,” there is a search bar through which the user can search for information about the movie he wants. When pressing Enter on the keyboard or the search tab in the search bar, upon triggering the search, the website sends a request to the Open Movie Database (OMDb) using the constructed API URL, which includes the user's input as the search query. the search results will appear to the user, which include the poster of the movie he wants and information about it, such as the name or title of the movie, the type of movie, the most prominent actors, the date of the movie’s release, and the country of origin of the movie, and that elegantly and effectively, which enhances the User experience on the **WatchWise** movie website.

**Thirdly, testing the third case of the API - which is searching for Celebrity’s** name and information about him, which includes his name, date of birth, age, nationality, and occupation.

**API Description: Api Ninjas - Celebrity API: The** Api Ninjas Celebrity API is a powerful tool that provides seamless access to a wealth of information about celebrities. It is designed to be utilized by developers and applications to fetch detailed data about a specific celebrity based on user queries. The API is well-suited for integration into platforms like the WatchWise movie website, enhancing user experiences by offering comprehensive details about various public figures.



First, through the search bar located below “**Find Celebrity**”, there is a search bar through which the user can search for information about the Celebrity he wants. This is done by entering the name of the Celebrity, and then the user presses Enter on the keyboard or the search tab in the search bar. When the search begins, the search results will appear to the user, which include a poster for the Celebrity he wants and information about him such as the Celebrity’s name or nationality, date of birth, age, and occupation, in an elegant and effective way, which enhances the user experience on the WatchWise movie website.

Finally, in short, the Ninjas Celebrity API has been integrated with the WatchWise movie site, allowing users to search for celebrity information. The website requests an Application Programming Interface (API), which retrieves detailed information about the celebrity. Data is presented dynamically on the **WatchWise** website, and real-time interaction is supported. This integration enhances the user experience by providing quick insights into celebrities' lives and complementing movie-related content.

**Q4** • Asses the chosen APIs for application development, critically analyzing their strengths and weaknesses. Additionally, supply you work with a data security report for the developed web application.

**Answer:**

In the field of website development and implementation, the integration of external APIs plays a pivotal role in enriching the functionality and content of applications. So, this analysis delves into the strengths and weaknesses of the different APIs used in Watchwise and ExploreWonders, and highlights their individual contributions and areas for improvement.

**TMDb API:**

**Strengths:**

Extensive Movie Database: TMDb features an extensive and comprehensive movie database, making it an outstanding app choice. This database serves as a rich source of information related to films, including details such as film synopses, genres, release dates, cast information, and more. This wealth of data provides a strong foundation for app content, ensuring a diverse and engaging user experience.

Simple Authentication: One of TMDb's strengths is its straightforward key authentication process. API key authentication is designed to be easy to use and uncomplicated. This simplicity simplifies the integration process for developers, allowing them to easily obtain the necessary authentication keys and integrate them into their applications. The simplicity of the authentication mechanism contributes to the efficiency of the development workflow, facilitating seamless access to TMDb's massive movie data.

**Weaknesses:**

Limited Requests: One notable weakness of the TMDb API, especially for free accounts, is potential limitations on the number of requests. Free accounts may be subject to price caps, meaning that if the app exceeds a certain number of requests within a certain time frame, further access may be restricted. This limitation can be a challenge, especially for applications with high traffic or frequent data retrieval needs. Developers must keep these limits in mind to prevent service interruptions or decreased performance.

Lack of error handling: The absence of robust error handling in the code creates a vulnerability in the reliability of the application. Error handling is critical to safely manage unexpected issues, such as network issues, server errors, or invalid requests, during application programming interface (API) interactions. Without proper error handling, the application may fail to respond to errors appropriately, resulting in potential disruptions to users. Incorporating error handling mechanisms such as try-catch blocks is essential to ensuring a smoother user experience by providing informative feedback and mitigating the impact of unexpected issues. Implementing such measures will enhance the overall flexibility of the application during interactions with the TMDb API.

**Open Movie Database (OMDb) API:**

**Strength:**

Movie and Celebrity Information: One of the notable strengths of the OMDb API is its versatility in providing both movie and celebrity information. This dual functionality enhances the richness of content within the application, catering to a broader range of user interests. By offering comprehensive details about movies and associated celebrities, the API contributes significantly to the application's depth and appeal.

Versatility in Content: OMDb's focus on both movies and celebrity data adds a layer of versatility to the application. Developers can leverage this breadth of information to create diverse and engaging features, such as linking movies to the profiles of the actors involved. This versatility opens up creative possibilities for developers to enhance user interaction and exploration.

Ease of integration: The OMDb API is designed with simplicity in mind, making it easy for developers to integrate into their applications. With straightforward endpoints and response structures, developers can quickly access and display movie and celebrity information, reducing the complexity of implementation and saving development time.

**Weaknesses:**

Exposure to the API key: One significant weakness lies in the direct embedding of the API key within the code. This practice poses a security risk as it exposes the key to potential misuse or unauthorized access. To mitigate this risk, best practices recommend storing API keys securely, such as using environment variables, to prevent unauthorized access and adhere to security standards.

Security Concerns: The exposure of the API key raises broader security concerns. If the API key falls into the wrong hands, it could be exploited for malicious purposes, leading to unauthorized access, data breaches, or misuse of the API. Implementing robust security measures, such as encryption and secure storage practices, is crucial to safeguard sensitive information.

Limited Rate Control: Description: OMDb's free-tier accounts might have limitations on the number of requests allowed within a specific time frame. Especially if it experiences high traffic or if there is a need for frequent API calls. Developers need to be mindful of these rate limits to ensure uninterrupted service.

The OMDb API is therefore versatile in providing movie and celebrity information, providing easy integration for developers. However, API key disclosure within the code and potential limitations on request rates are weaknesses that must be addressed to improve security and application performance.

**Ninjas Celebrity API:**

API Ninjas Celebrity API is an interface that provides detailed information about celebrities, aiming to enrich the content of the application. Here are three strengths and three weaknesses associated with this API:

**Strengths:**

Rich Celebrity Information: One of the core strengths of the API is its ability to provide detailed and comprehensive information about celebrities. This includes various aspects such as the celebrity's name, date of birth, profession, age, and nationality. This richness in data enhances the overall content quality of the app, providing users with engaging and informative details about their favorite public figures.

Content Enhancement: By incorporating the API, the application gains the ability to feature diverse and up-to-date content related to celebrities. This can include real-time updates on celebrity events, milestones, or any other relevant information. The API contributes to keeping the application's content dynamic, attracting users interested in celebrity-related content.

Versatility in Application Use Cases: The API's provision of detailed celebrity information makes it versatile for a range of application use cases. Whether the application is centered around entertainment, news, or any other domain that involves celebrities, the API's wealth of data can be leveraged to tailor content and features to a wide audience.

**Weaknesses:**

Limited Documentation: A notable weakness lies in the limited documentation accompanying the API. Lack of comprehensive documentation can pose challenges for developers attempting to integrate the Celebrity API into the application. Detailed documentation is crucial for understanding the API's endpoints, data structures, and potential error scenarios, which, in turn, facilitates smoother integration and maintenance.

Incomplete Error Handling: The absence of robust error handling in the API integration code is a significant weakness. Incomplete error handling can lead to unpredictable behavior when issues arise, making it challenging to diagnose and rectify problems promptly. Well-implemented error handling is essential for maintaining the application's reliability and providing a seamless user experience.

Challenges in Extendibility: The lack of error handling and detailed documentation can collectively hinder the extendibility of the Celebrity API integration. Without proper documentation and error management, developers may find it challenging to understand and extend the functionality of the API. This limitation can impede the application's ability to adapt to changing requirements or incorporate new features related to celebrity information.

While API Ninjas Celebrity API brings strengths in providing rich celebrity information and enhancing content, its weaknesses, such as limited documentation and incomplete error handling, highlight areas that need attention for a smoother integration process and improved maintainability of the application.

**Unsplash API:**

**Strengths:**

High-Quality Images: One of the primary strengths of integrating the Unsplash API is the access to a vast repository of high-quality images. Unsplash is renowned for its collection of stunning, royalty-free images contributed by a community of talented photographers. Leveraging these high-quality images enhances the visual appeal of the application, making it more aesthetically pleasing for users. This can lead to a more engaging and enjoyable user experience.

Extensive Image Variety: Unsplash offers a diverse range of images covering various themes, genres, and styles. This diversity allows developers to choose images that align with the application's content or theme, catering to different user preferences. The extensive variety ensures that developers have ample options to select images that resonate with the overall design and purpose of the application.

Community-Driven Content: Unsplash operates on a model where photographers contribute their work for free use. This community-driven approach ensures a constant influx of new and diverse content. The collaborative nature of Unsplash's content creation fosters a dynamic and ever-expanding library of images. Developers can benefit from this continuous stream of fresh, high-quality images, keeping the application's visual content up-to-date and engaging.

**Weaknesses:**

API Key Exposure: A notable weakness lies in the direct exposure of the Unsplash API key in the code. This practice poses a significant security risk. Exposing API keys in the code makes them vulnerable to unauthorized access and potential misuse. To address this weakness, it is advisable to adopt more secure practices, such as storing API keys in environment variables. This not only enhances security but also aligns with best practices for handling sensitive information in a codebase.

Potential Rate Limits: While Unsplash offers its API for free, there may be rate limits imposed on the number of requests developers can make within a given timeframe. Exceeding these limits could result in restricted access or temporary suspension of API usage. Developers need to be mindful of these rate limits and implement mechanisms to manage and control the frequency of API requests to avoid disruptions in service.

Dependency on External Service: Integrating the Unsplash API introduces a dependency on an external service. If, for any reason, Unsplash experiences downtime, undergoes changes in its API structure, or ceases to be available, it can impact the application's ability to fetch images. To mitigate this weakness, developers should consider implementing fallback mechanisms or caching strategies to ensure a graceful degradation of the application's functionality in the absence of the Unsplash service.

While the Unsplash API brings considerable strengths in terms of image quality, variety, and community-driven content, addressing weaknesses related to API key exposure, potential rate limits, and dependency on external services is crucial for maintaining the security and reliability of the application.

**WeatherAPI:**

**Strengths:**

Comprehensive Weather Information: The WeatherAPI integration enhances the application by providing users with up-to-date and accurate information about the current weather conditions and forecasts. This real-time data adds significant value to the user experience, especially for applications where weather information is relevant.

User Engagement: Weather is a dynamic and widely relevant aspect of daily life. By incorporating WeatherAPI, the application becomes more engaging for users, offering them timely and relevant information that can influence their decisions and activities. This feature contributes to increased user retention and satisfaction.

Diverse Data for Different Use Cases: WeatherAPI likely provides a range of data beyond just the current temperature, including details like humidity, wind speed, and precipitation forecasts. This diverse set of data allows developers to create a multifaceted weather feature within the application, catering to different user preferences and needs.

**Weaknesses:**

No Handling for Incomplete Data: One notable weakness in the implementation is the lack of checks for incomplete data in API responses. If, for any reason, the API fails to provide the expected data or returns incomplete information, the application may encounter issues. Proper error handling and checks for data completeness are crucial for ensuring a reliable and robust user experience.

Potential Disruption Due to Data Issues: Since the code doesn't address incomplete data, there's a risk of unexpected behavior or disruptions in the application when dealing with partial or missing information. This could lead to inaccuracies in the displayed weather data or, in extreme cases, break the functionality related to weather information.

**YouTube API:**

**Strengths:**

Efficient Data Fetching: The code utilizes asynchronous functions and the axios library to efficiently fetch movie data from the YouTube API. Using async/await ensures that the application isn't blocked during API calls.

Dynamic Content Rendering: The application dynamically renders movie content based on user interactions, such as searching for movies or discovering popular ones. This dynamic rendering enhances the user experience by providing relevant and up-to-date information.

Responsive UI with Scroll Event: The code includes a scroll event listener that dynamically adjusts the appearance of the header based on the user's scroll position. This responsiveness contributes to a more polished and visually appealing user interface.

**Weaknesses:**

Security: The API key is hardcoded directly in the code (const API\_KEY = "25371891a57c62e6ae5d894eba150721";). Hardcoding API keys poses a security risk as they can be exposed in the client-side code. It is recommended to use secure methods, such as storing keys on the server or using environment variables, to protect API keys.

Limited Error Handling: The code lacks comprehensive error handling for API requests. In case of network issues or API failures, there's no implementation of try-catch blocks to gracefully handle errors. Adding robust error handling would improve the application's reliability and user experience.

Overall, while the code effectively fetches and displays YouTube API data, addressing the weaknesses related to security, error handling, and code organization would contribute to the overall robustness and maintainability of the application.

The integration of external APIs in website development, as demonstrated in the **WatchWise** and ExploreWonders applications, serves as a crucial component for enhancing functionality and content richness. Each API selected brings unique strengths and weaknesses to the applications.

Starting with the TMDb API, its extensive movie database provides a diverse range of movie-related information, contributing to a rich user experience. The simple authentication process facilitates easy integration for developers. However, limitations on requests, especially for free accounts, and the absence of robust error handling in the code present challenges that developers need to address to prevent service interruptions and enhance reliability.

The OMDb API stands out for its versatility in providing both movie and celebrity information, broadening the scope of the application. While its ease of integration and focus on diverse content are strengths, the exposure of the API key and potential rate limits pose security and performance concerns that require attention.

The API Ninjas Celebrity API enriches content by offering detailed information about celebrities. The strength lies in its ability to provide comprehensive data, enhancing the app's appeal. However, limited documentation and incomplete error handling hinder the smooth integration of the API, emphasizing the need for improvements in these areas.

The Unsplash API contributes high-quality and diverse images, enhancing the visual appeal of the application. Its community-driven content model ensures a constant influx of fresh images. Yet, API key exposure, potential rate limits, and dependency on an external service pose security and reliability challenges that should be addressed for a more secure and stable application.

WeatherAPI strengthens the application by providing comprehensive and timely weather information, increasing user engagement. However, the lack of handling for incomplete data in API responses poses a reliability risk that needs attention for a seamless user experience.

Finally, the YouTube API efficiently fetches movie data, dynamically renders content based on user interactions, and enhances the user interface with a responsive header. Yet, the hardcoded API key and limited error handling require improvements for enhanced security and a more reliable application.

While each API brings valuable strengths, addressing weaknesses related to security, error handling, and documentation is essential for creating robust, secure, and user-friendly applications.

Integrating external APIs into website development, as demonstrated in my Watchwise and ExploreWonders movie website app that I added and implemented "Explore Corner APIs," is a critical component to enhancing functionality and richness of content. Each specific API provides unique strengths and weaknesses to applications.

Starting with the TMDb API, its extensive movie database provides a diverse range of movie-related information, contributing to a rich user experience. The simple authentication process facilitates easy integration for developers. However, limitations on requests, especially for free accounts, and the absence of robust error handling in the code present challenges that developers need to address to prevent service interruptions and enhance reliability.

The OMDb API stands out for its versatility in providing both movie and celebrity information, broadening the scope of the application. While its ease of integration and focus on diverse content are strengths, the exposure of the API key and potential rate limits pose security and performance concerns that require attention.

The API Ninjas Celebrity API enriches content by offering detailed information about celebrities. The strength lies in its ability to provide comprehensive data, enhancing the app's appeal. However, limited documentation and incomplete error handling hinder the smooth integration of the API, emphasizing the need for improvements in these areas.

The Unsplash API contributes high-quality and diverse images, enhancing the visual appeal of the application. Its community-driven content model ensures a constant influx of fresh images. Yet, API key exposure, potential rate limits, and dependency on an external service pose security and reliability challenges that should be addressed for a more secure and stable application.

WeatherAPI strengthens the application by providing comprehensive and timely weather information, increasing user engagement. However, the lack of handling for incomplete data in API responses poses a reliability risk that needs attention for a seamless user experience.

Finally, the YouTube API efficiently fetches movie data, dynamically renders content based on user interactions, and enhances the user interface with a responsive header. Yet, the hardcoded API key and limited error handling require improvements for enhanced security and a more reliable application.

While each API brings valuable strengths, addressing weaknesses related to security, error handling, and documentation is essential for creating robust, secure, and user-friendly applications.

Ensuring the security of your web applications is crucial in today's digital landscape. Protection against common vulnerabilities is essential to effectively protect sensitive data and maintain system integrity. In this comprehensive guide, I'll delve into the basic security measures every web developer should consider. Each point is accompanied by a detailed explanation and a practical example to illustrate its importance and implementation. By incorporating these best practices into the development process, you can create a more flexible and secure web application environment. A comprehensive data security report evaluates the protection measures in place, highlighting vulnerabilities and suggesting improvements.

**Data Security Report:**

**Data Encryption:**

Web Application: Ensure that data transmitted between users and the web application is encrypted using HTTPS, safeguarding sensitive information like user credentials. Example: Utilize SSL/TLS certificates to establish a secure connection between users and the web application.

APIs: Employ HTTPS to encrypt data in transit between clients and APIs, maintaining confidentiality during communication. For example, use API keys or tokens to control access to APIs securely, preventing unauthorized users from interacting with sensitive endpoints.

This approach is crucial for protecting user credentials and other sensitive data from interception. In the context of APIs, HTTPS is also employed to encrypt data in transit between clients and the APIs. This is often complemented with API keys or tokens, which control access to the APIs and maintain the confidentiality of the data exchanged, thus preventing unauthorized access to sensitive endpoints.

**Authentication and Authorization:**

Web Application: Evaluate the effectiveness of user authentication mechanisms, such as password policies and multi-factor authentication, to ensure secure access. For example, implement multi-factor authentication (MFA) to add an extra layer of security, requiring users to verify their identity through multiple means.

APIs: Assess the security of authentication methods like API keys or OAuth tokens used in APIs to control access to sensitive data. For example, implement OAuth 2.0 to enable secure, token-based authentication for API access, reducing the reliance on static API keys.

**Input Validation and Sanitization:**

Web Application:

Validate and sanitize user inputs to thwart common vulnerabilities like SQL injection and cross-site scripting (XSS) attacks. For example, server-side validation to check the format and length of user inputs and reject inputs that do not meet specified criteria.

APIs: Mitigate injection attacks by rigorously validating input parameters in API requests and rejecting invalid or malicious input. For example, implement input validation routines in the API code to ensure that data received from clients adhere to expected formats.

**Logging and Monitoring:**

Web Application: Establish a robust logging mechanism that records relevant security events without exposing sensitive information. For example, log unsuccessful login attempts, including details like IP addresses, to detect potential brute-force attacks.

APIs: Implement logging practices within APIs to capture security-related events and monitor API usage for abnormal patterns. For example, API requests and responses, emphasizing security-relevant details, to facilitate forensic analysis in the event of a security incident.

**Compliance with Data Protection Regulations:**

Web Application: Verify compliance with data protection regulations applicable to your domain, such as GDPR or HIPAA. For example, regularly review and update privacy policies to align with evolving data protection standards.

APIs: Ensure that APIs adhere to the same data protection regulations as the web application to maintain consistency in data handling. For example, implement mechanisms to honor user data deletion requests across both the web application and associated APIs.

**Secure File Handling:**

Web Application: If the application supports file uploads, assess measures for validating file types, checking for malware, and enforcing size limits. For example, uploaded files for malware using antivirus tools before allowing them to be processed or stored.

APIs: Evaluate security measures in APIs to prevent malicious file uploads, ensuring uploaded files undergo proper validation. For example, set maximum file size limits for API-based file uploads to prevent abuse and denial-of-service attacks.

**Dependency Management:**

Web Application: Confirm that dependencies, including libraries and frameworks, are regularly updated to patch known security vulnerabilities. For example, automated tools to track dependencies and receive alerts about security patches, facilitating timely updates.

APIs: Assess dependency management practices for APIs, ensuring that third-party libraries are regularly updated to address security issues. For example, regularly review and update API dependencies, considering the security impact of each update to prevent vulnerabilities.

**Incident Response Plan:**

Web Application and APIs: assessing the existence and effectiveness of an incident response plan is crucial for both web applications and APIs. Evaluate the existence and effectiveness of an incident response plan to ensure a coordinated and effective response to security breaches. For example, simulate a security incident to test the effectiveness of the incident response plan, identifying areas for improvement.

**References:**

* Lutkevich, B. (n.d.). *What Is an API (Application Program Interface)?* [online] SearchAppArchitecture. Available at: <https://www.techtarget.com/searchapparchitecture/definition/application-program-interface-API>.
* WhatIs.com. (n.d.). *What is software development kit (SDK)? - Definition from WhatIs.com*. [online] Available at: <https://www.techtarget.com/whatis/definition/software-developers-kit-SDK>.
* locationiq.com. (n.d.). *LocationIQ - Free Reverse Geocoding API, Geocoding API, Autocomplete API*. [online] Available at: <https://locationiq.com/>.
* Indusface. (2021). *What is API Security and Why is It Important? | Indusface Blog*. [online] Available at: <https://www.indusface.com/blog/what-is-api-security-and-why-is-it-important/>.
* Kong (2023). API Authentication vs. API Authorization: What’s the Difference? [online] Kong Inc. Available at: https://konghq.com/blog/engineering/api-authentication-vs-api-authorization [Accessed 19 Dec. 2023].
* Postman API Platform. (n.d.). What Is the API Lifecycle? Stages & Best Practices | Postman. [online] Available at: <https://www.postman.com/api-platform/api-lifecycle/>.
* Frontegg. (n.d.). API Authentication and Authorization: 6 Methods and Tips for Success. [online] Available at: https://frontegg.com/guides/api-authentication-api-authorization [Accessed 19 Dec. 2023].
* BreachLock\_Labs (2022). What is API Penetration Testing? [online] BreachLock. Available at: <https://www.breachlock.com/resources/blog/what-is-api-penetration-testing/>.
* <https://www.akamai.com/glossary/what-is-an-api-security-audit>
* docs.aws.amazon.com. (n.d.). Encrypting Data-at-Rest and -in-Transit - Logical Separation on AWS. [online] Available at: <https://docs.aws.amazon.com/whitepapers/latest/logical-separation/encrypting-data-at-rest-and--in-transit.html>.
* owasp.org. (n.d.). API10:2019 Insufficient Logging & Monitoring - OWASP API Security Top 10. [online] Available at: <https://owasp.org/API-Security/editions/2019/en/0xaa-insufficient-logging-monitoring/>
* Wentowski, M. (2023). *API Lifecycle Management: Phases, Challenges & Best Practices*. [online] Document360. Available at: https://document360.com/blog/api-lifecycle-management/ [Accessed 19 Dec. 2023].
* Palo Alto Networks. (n.d.). *What Is Web Application and API Protection?* [online] Available at: https://www.paloaltonetworks.com/cyberpedia/what-is-web-application-and-api-protection [Accessed 31 Dec. 2023].
* Digital Defense. (n.d.). *Web Application Security | Why Is It Important?* [online] Available at: https://www.digitaldefense.com/web-application-security/ [Accessed 31 Dec. 2023].
* What Is Web Application Security? | Web Security | Cloudflare. (n.d.). *Cloudflare*. [online] Available at: https://www.cloudflare.com/learning/security/what-is-web-application-security/.

‌

**Plagiarism**

Plagiarism is a particular form of cheating. Plagiarism must be avoided at all costs and students who break the rules, however innocently, may be penalised. It is your responsibility to ensure that you understand correct referencing practices. As a university level student, you are expected to use appropriate references throughout and keep carefully detailed notes of all your sources of materials for material you have used in your work, including any material downloaded from the Internet. Please consult the relevant unit lecturer or your course tutor if you need any further advice.

**Student Declaration**

|  |
| --- |
| **Student declaration**  I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.  Student signature: Rashed Hasan Qahah. Date: 25/1/2024 |