**CSE303 - Software Design & Architecture**

**Bloom Taxonomy Level: <*Applying*>**

**Section:** A

**Project Name:** *SmartWebCraft*

**Group Members:**

|  |  |
| --- | --- |
| **Name** | **ID** |
| Aoun Haider | FA21-BSE-133 |
| Muhammad Umer Atiq | FA21-BSE-007 |
| Muhammad Talha Shafiq | FA21-BSE-152 |
| Affan Ahmad | FA21-BSE-127 |
| Hussain Iqbal | FA21-BSE-109 |
| Muhammad Asad | FA21-BSE-052 |

**Note:** Each members’ work has been mentioned in [FA21-BSE-XYZ] Format before the start of work.

[FA21-BSE-133]

**Introduction:**

The need for web applications that are not just functional but also aesthetically pleasing and responsive has never been greater in the quickly changing digital landscape of today. To fulfill this requirement, we present the cutting-edge project *SmartWebCraft.*

By providing programmers with an all-inclusive platform that enables them to realize their ideas with unmatched ease and speed, *SmartWebCraft* transforms the web application development process. With the help of cutting-edge technologies like artificial intelligence, machine learning, and visual programming, "SmartWebCraft" enables programmers to construct complex online applications with little coding knowledge.

**Features:**

1. **Visual Programming Interface:**

Gone are the days of tedious coding – "SmartWebCraft" provides a user-friendly drag-and-drop interface reminiscent of renowned IDEs like IntelliJ and Microsoft Visual Studio. Programmers can effortlessly assemble components such as buttons and text fields, customizing them to suit their specific requirements with unparalleled ease.

1. **Enhanced Responsiveness:**

Unlike existing solutions, "SmartWebCraft" doesn't stop at mere visual design. With seamless integration with Bootstrap 5 and Tailwind CSS, programmers gain access to a vast library of pre-built, responsive components, ensuring that their creations adapt flawlessly to various screen sizes and devices.

1. **AI-Driven Suggestions:**

Our project harnesses the power of AI to offer intelligent suggestions during the development process. Machine learning algorithms analyze user actions, recommending optimal component combinations to enhance both responsiveness and user experience, thus streamlining the development workflow.

1. **Resource Optimization:**

SmartWebCraft is engineered to be lightweight and resource efficient. AI algorithms manage system resources intelligently, optimizing performance and minimizing energy consumption. Features such as automatic system sleep and cloud backup ensure data integrity and efficient resource utilization.

1. **Real-time Collaboration:**

Facilitate real-time collaboration among team members working on the same project, allowing simultaneous editing and instant feedback. Incorporate features like live chat, collaborative editing, and version control to enhance teamwork and productivity.

1. **Automated Testing & Quality Assurance:**

Implement automated testing frameworks and tools to perform comprehensive testing of web applications, including unit tests, integration tests, and UI tests. Offer built-in test suites, test case generators, and code coverage analysis to ensure the quality and reliability of applications.

**Principle Design Decisions:**

1. **Interaction:**
   1. **User Interface:**

* SmartWebCraft aims to provide a visually intuitive interface, resembling popular IDEs like Intellij and Microsoft Visual Studio, where programmers can seamlessly drag and drop components to build web applications.
* The UI will be divided into distinct sections for ease of navigation, such as a component library, canvas area for designing, properties panel for customization, and a preview mode to visualize the final output.
* To enhance usability, keyboard shortcuts and context menus will be implemented for power users who prefer faster interaction methods over drag and drop.
* The UI will be customizable, allowing users to arrange panels according to their workflow preferences. They can save custom layouts for future use.
  1. **User Input:**
* SmartWebCraft will support various modes of user input, including mouse, keyboard, touch, and stylus (for touch-enabled devices).
* Drag and drop functionality will be intuitive, allowing users to select components from the library and drop them onto the canvas. Real-time feedback will indicate valid drop zones and potential conflicts.
* Input validation will ensure that users adhere to constraints and guidelines while designing, preventing common errors and promoting best practices in web development.
* Context-aware input suggestions will be provided based on the user's actions and the current state of the project. For example, when adding a new component, relevant options and configurations will be presented dynamically.
  1. **Feedback Mechanism**
* SmartWebCraft will employ feedback mechanisms to keep users informed about the status of their actions and the state of the application.
* Visual cues such as color changes, tooltips, and animations will provide real-time feedback during interactions, indicating success, errors, or warnings.
* Interactive tutorials and tooltips will guide users through the interface and functionalities, especially for first-time users or those exploring advanced features.
* Comprehensive error messages and logs will be available to assist users in troubleshooting issues, along with suggestions for resolution or alternative approaches.
  1. **Error Handling**
* Proactive error prevention will be prioritized through intuitive design and input validation, reducing the occurrence of errors during interaction.
* When errors do occur, clear and informative messages will be displayed, pinpointing the cause of the problem and suggesting corrective actions.
* SmartWebCraft will implement an undo/redo mechanism to allow users to revert erroneous actions and restore previous states of the project.
* In cases of critical errors or system failures, SmartWebCraft will gracefully handle the situation by saving user progress, providing recovery options, and initiating appropriate error reporting processes.

[FA21-BSE-052]

* 1. **Adaptability and Responsiveness**
* The interaction design of SmartWebCraft will prioritize adaptability and responsiveness across devices and screen sizes.
* The UI will be designed using responsive principles, ensuring optimal viewing and interaction experiences on desktops, tablets, and smartphones.
* Touch-friendly gestures and interactions will be supported for touchscreen devices, enhancing usability for users on the go.
* Accessibility features such as screen reader compatibility, keyboard navigation, and high contrast modes will be integrated to ensure inclusivity and compliance with accessibility standards.
  1. **Collaborative Features**
* SmartWebCraft will support collaborative features to facilitate teamwork and project management among developers. Real-time co-editing capabilities will enable multiple users to work on the same project simultaneously.
* Collaborators will be able to view each other's changes in real-time, with visual indicators highlighting areas of activity and potential conflicts.
* Communication tools such as comments, chat, and video conferencing integrations will be available within the interface, allowing team members to discuss designs, share feedback, and resolve issues without leaving the platform.
  1. **Integration with External Tools and Services**
* SmartWebCraft will integrate seamlessly with external tools and services commonly used in web development workflows.
* Version control systems like Git will be integrated to enable efficient collaboration, code management, and version tracking within projects.
* Continuous integration and deployment (CI/CD) pipelines can be configured to automate the build, test, and deployment processes, streamlining the development lifecycle.
* Third-party APIs and services for analytics, monitoring, and performance optimization can be integrated to enhance the functionality and capabilities of SmartWebCraft.

[FA21-BSE-127]

**BEHAVIOUR:**

SmartWebCraft Architecture Pattern: Model-View-Controller (MVC)

The MVC architectural pattern divides the system into three main components: Model, View, and Controller. This separation of concerns helps manage complex applications by organizing the code in a way that increases reusability and maintainability. Here's how it can be applied to SmartWebCraft:

**a) Separation of Concerns**

*1. Model:*

- Description: Represents the data and business logic. It handles data storage, retrieval, and business rules.

- Responsibilities: Manage the state of the application, interact with the database (MongoDB), and enforce business rules.

- Components: Database schema, data access objects (DAO), business logic classes, AI modules for suggestions and optimizations.

*2. View:*

- Description: Represents the UI of the application. It displays the data to the user and sends user commands to the controller.

- Responsibilities: Render the user interface using HTML, CSS (Bootstrap 5 & Tailwind CSS), and JavaScript.

- Components: HTML templates, CSS stylesheets, JavaScript files, front-end frameworks (React, Angular, Vue.js).

*3. Controller:*

- Description: Acts as an intermediary between the Model and the View. It receives user input, processes it (using the Model), and returns the result to the View.

- Responsibilities: Handle user inputs, call Model methods, and decide which View to render.

- Components: HTTP request handlers, routing logic, API controllers.

**b) Transformation of Functional Requirements into Components**

*1. Drag-and-Drop Functionality:*

- Controller: DragDropController

- Model: DragDropModel (handles logic for drag-and-drop actions)

- View: DragDropView (UI components for drag-and-drop interface)

*2. AI Suggestions:*

- Controller: AISuggestionController

- Model: AISuggestionModel (implements ML algorithms for UI suggestions)

- View: AISuggestionView (displays AI suggestions to the user)

*3. Backend Management (PHP and MongoDB):*

- Controller: BackendController

- Model: BackendModel (handles interactions with PHP scripts and MongoDB)

- View: BackendView (admin UI for managing backend)

*4. Cloud Support and Caching:*

- Controller: CloudController

- Model: CloudModel (manages cloud storage and caching logic)

- View: CloudView (UI for cloud settings and cache management)

[FA21-BSE-052]

**c) Assigning Collection of Components to Modules/Subsystems**

*1. UI Module:*

- Components: DragDropView, AISuggestionView, BackendView, CloudView

- Subsystems:

- Responsiveness and Accessibility: Utilizes Bootstrap 5 and Tailwind CSS

- Interactive Elements: JavaScript and front-end framework integration

*2. Business Logic Module:*

- Components: DragDropModel, AISuggestionModel, BackendModel, CloudModel

- Subsystems:

- AI and Machine Learning: Handles AI-based suggestions and optimizations

- Backend Management: Manages business rules and data integrity

*3. Controller Module:*

- Components: DragDropController, AISuggestionController, BackendController, CloudController

- Subsystems:

- Request Handling: Routes and processes HTTP requests

- User Input Processing: Mediates between user actions and model updates

**Summary**

By applying the MVC architectural pattern, SmartWebCraft can achieve a clear separation of concerns, transforming functional requirements into well-defined components, and organizing these components into cohesive modules and subsystems. This structure enhances the manageability, scalability, and maintainability of the SmartWebCraft platform.

**FA21-BSE-152**

**Non-Functional Requirements Recap:**

1. The system shall store user information including username, email, and encrypted passwords.
2. The system shall provide functionality to manage a catalog of products including adding, updating, and deleting product information.
3. The system shall track order history including order details, status, and delivery information.
4. The system shall allow users to update their profile information including name, address, and contact details.
5. The system shall integrate with payment gateways for processing online transactions securely.

**Design Decision 1: Adopt a Microservices Architecture**

* **Description:** Implement the Smart Web Craft project using a microservices architecture, where each service is responsible for a specific business function (e.g., user management, product catalog, order tracking, profile management, payment processing).
* **Supported Non-Functional Requirements:**
  + **User Information Storage:** A dedicated User Management Service will handle user data storage with encrypted passwords.
  + **Catalog Management:** A Product Catalog Service will manage product information, supporting adding, updating, and deleting operations.
  + **Order Tracking:** An Order Management Service will track order history and details.
  + **Profile Management:** A Profile Management Service will handle user profile updates.
  + **Payment Integration:** A Payment Service will securely integrate with payment gateways.
* **Design Principles:**
  + **Scalability:** Each microservice can be scaled independently based on demand.
  + **Modularity:** Clear separation of concerns makes the system modular and easier to manage.
  + **Security:** Each service can have tailored security measures, enhancing overall system security.
  + **Reliability:** Fault isolation ensures that a failure in one service does not affect others.
  + **Privacy by Design:** Each service can be designed with privacy considerations specific to its data.

**Design Decision 2: Implement Horizontal Scaling and Load Balancing**

* **Description:** Use horizontal scaling and load balancing to distribute incoming requests across multiple instances of each microservice.
* **Supported Non-Functional Requirements:**
  + **User Information Storage, Catalog Management, Order Tracking, Profile Management, Payment Integration:** All services benefit from distributed load and improved performance.
* **Design Principles:**
  + **Scalability:** Supports increased traffic by adding more instances.
  + **Reliability:** Load balancing increases fault tolerance and availability.
  + **Security:** Load balancers can add an additional layer of security by filtering traffic.
  + **Efficiency:** Ensures optimal resource utilization and response times.

**Design Decision 3: Implement Robust Authentication and Authorization Mechanisms**

* **Description:** Use industry-standard protocols like OAuth 2.0 and JWT for secure authentication and authorization across all services.
* **Supported Non-Functional Requirements:**
  + **User Information Storage:** Secure login and access management.
  + **Profile Management:** Secure access to profile updates.
  + **Payment Integration:** Secure transactions and access control.
* **Design Principles:**
  + **Security:** Ensures secure user authentication and authorization.
  + **Reliability:** Reduces the risk of unauthorized access, enhancing system stability.
  + **Privacy by Design:** Ensures that user data is accessed only by authorized entities.

**Design Decision 4: Implement Automated Backup and Disaster Recovery**

* **Description:** Set up automated backup routines and robust disaster recovery plans for all critical data and services.
* **Supported Non-Functional Requirements:**
  + **User Information Storage, Catalog Management, Order Tracking, Profile Management, Payment Integration:** Ensures data integrity and availability in case of failures.
* **Design Principles:**
  + **Reliability:** Regular backups and disaster recovery plans ensure minimal data loss and downtime.
  + **Security:** Secure backups protect against data breaches.
  + **Compliance:** Ensures compliance with data protection regulations.

**Design Decision 5: Utilize Comprehensive Monitoring and Logging**

* **Description:** Implement comprehensive monitoring and logging across all services to track performance, detect issues, and ensure auditability.
* **Supported Non-Functional Requirements:**
  + **User Information Storage, Catalog Management, Order Tracking, Profile Management, Payment Integration:** Enables tracking and auditing of all critical operations.
* **Design Principles:**
  + **Reliability:** Continuous monitoring helps detect and address issues proactively.
  + **Security:** Logs provide an audit trail, enhancing security and compliance.
  + **Transparency:** Detailed logs and monitoring data ensure transparency and accountability.
  + **Scalability:** Monitoring tools scale with the system, providing consistent insights.

By implementing these design decisions, the Smart Web Craft project ensures that each of the identified non-functional requirements is supported by the design principles, resulting in a robust, scalable, secure, reliable, and compliant system architecture.

[FA21-BSE-007]

**Customize Component:**

1. Customize Appearance
2. Edit Parameters
3. Add CSS Style
4. Define Custom Animation
5. Use Shared Component
6. Import Asset
7. Specify Component Rendering across different screens
8. Access pre-built Library
9. Define Environmental Effects
10. Create Interaction Components
11. Enable built-in Accessibility
12. Integration of 3rd Party Components
13. Preview Component
14. Test Component
15. Version Control
16. Share Component Design
17. Localization
18. Compatibility
19. Define Custom Breakpoints
20. Access Documentation
21. Backup Components
22. Define Custom Event Handler
23. Arrange Components
24. Resize Component
25. Specify Icons
26. Enable Colour-blindness Components
27. Control Components Accessibility
28. Create Parent Components
29. Create Custom Shortcuts
30. Simulate Component Appearance
31. Integration of Analytics
32. Define Custom Tooltips
33. Create Visualization Components
34. Create Visual Design
35. Reuse Component Code
36. Change Component Theme
37. Format Component Code
38. Optimize Component Performance
39. Component Drag & Drop
40. Component Debugging
41. Delete Component

**AI powered design assistant:**

1. User Interface Component Analyzer
2. Layout Optimization Module
3. Style Consistency Checker
4. Responsiveness Analyzer
5. Accessibility Improvement Module
6. UI Pattern Recognition
7. Design Mistake Detector
8. User Preference Learner
9. Trend Awareness Integrator
10. Context-based Recommendation Engine
11. Hierarchy and Emphasis Analyzer
12. UX Enhancement Proposer
13. Real-time Feedback Provider
14. Preview Mode Activator
15. A/B Testing Facilitator
16. Color Palette Generator
17. Font Pairing Suggester
18. Image Recommender
19. Icon Selection Assistant
20. Usability Issue Identifier
21. Error Prevention Advisor
22. Accessibility Rule Enforcer
23. User Preference Recorder
24. AI Suggestion Override Mechanism
25. User Feedback Integrator
26. Emotional Design Investigator
27. Personalization Explorer
28. Trend Forecasting Investigator
29. Resource Usage Sharer
30. Data Management Integrator
31. Backend Compatibility Checker
32. Documentation and Training Provider
33. Performance Monitor
34. Explainability Investigator
35. User Acceptance Tester
36. Security Implementation
37. Scalability Analyzer
38. Error Handling and Recovery System
39. Maintenance and Updates Planner
40. Version Control System

**Drag and drop UI:**

1. Intuitive Drag and Drop Interface
2. Pre-built UI Component Library
3. Resizable and Movable Components
4. Grouping and Ungrouping of Components
5. Customizable Component Properties
6. Creation of Responsive Layouts
7. Design Preview for Various Resolutions and Orientations
8. Snapping Guidelines for Alignment
9. Undo and Redo Functionality
10. Save and Load Projects
11. Import and Export of Project Files
12. Real-time Collaboration with Version Control
13. Keyboard Shortcuts for Common Actions
14. Addition of Custom CSS Styles
15. Support for Nested Components and Layouts
16. Locking Certain Components
17. Search Functionality for Adding Components
18. Addition of Interactive Elements
19. Support for Third-Party Plugins and Extensions
20. Component Alignment Options
21. Grid System for Precise Positioning
22. Creation of Reusable Templates and Components
23. Addition of Custom Scripts and JavaScript Functionality
24. Built-in Accessibility Features
25. Setting Constraints on Components
26. Provision of Tooltips and Help Documentation
27. Definition of Interactions and Animations
28. Creation of Multi-page Designs
29. Importing Existing Designs
30. History Panel for Navigation
31. Publishing Designs to Web Servers or Hosting Platforms
32. Integration of Version Control Systems
33. Integration of Analytics and Tracking Codes
34. Pre-designed Themes and Templates
35. Previewing Designs on Different Devices and Browsers
36. Creation of Dynamic Content Using Server-side Scripting Languages
37. Exporting Designs as HTML, CSS, and JavaScript Files
38. Provision of Tutorials and Guides
39. Ability to Share Designs with Clients or Team Members for Feedback
40. Regular Updates Based on User Feedback and Industry Trends

[FA21-BSE-052]

**Security:**

1. Authentication Component
2. Access Control Component
3. Encryption Component
4. Intrusion Detection Component
5. Communication Security Component
6. Session Management Component
7. Password Policy Component
8. Error Handling Component
9. File Upload Component
10. Logging Component
11. Session Persistence Component
12. API Authentication Component
13. Database Access Control Component
14. CORS Protection Component
15. Remote Access Component
16. Data Transmission Security Component
17. API Endpoint Protection Component
18. System Update Management Component
19. Account Management Component
20. Backup and Recovery Component

**User Authentication:**

1. User Registration Component
2. User Login Component
3. Password Reset Component
4. Password Change Component
5. Account Deletion Component
6. User Roles Component
7. Role Permissions Component
8. Two-Factor Authentication Component
9. Session Management Component
10. Secure Login Component
11. Social Login Component
12. User Profile Management Component
13. Password Strength Meter Component
14. Remember Me Functionality Component
15. Account Verification Component
16. Account Lockout Component
17. IP Address Tracking Component
18. Session Hijacking Prevention Component
19. Idle Session Timeout Component
20. Secure Password Storage Component

[FA21-BSE-109]

**5.IMPLEMENTATION**

**Frameworks and Libraries:**

**Backend:**

***Express.js (Node.js):***

Description: A minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

Key Features:

Routing: Defines how an application’s endpoints (URIs) respond to client requests.

Middleware: Functions that execute during the lifecycle of a request to the server.

Templates: Dynamically renders HTML on the server side.

Usage: Ideal for building RESTful APIs due to its lightweight nature and ease of use.

***Django (Python):***

Description: A high-level Python web framework that encourages rapid development and clean, pragmatic design.

Key Features:

ORM (Object-Relational Mapping): Maps database schemas to Python objects.

Admin Interface: An automatic administrative interface for managing application data.

Security: Built-in protection against common security threats.

Usage: Suitable for complex applications requiring robust backend logic and a rapid development cycle.

**Frontend:**

***React.js:***

Description: A JavaScript library for building user interfaces, maintained by Facebook.

Key Features:

Component-Based Architecture: Allows building encapsulated components that manage their state.

Virtual DOM: Improves performance by minimizing direct DOM manipulation.

Hooks: Functions that let you use state and other React features without writing a class.

Usage: Excellent for building dynamic, high-performance user interfaces.

***Vue.js:***

Description: A progressive framework for building user interfaces.

Key Features:

Reactivity System: Automatically tracks and updates data changes.

Single-File Components: Combines HTML, JavaScript, and CSS in a single file.

Flexibility: Can be used as a library to enhance existing projects or as a full-featured framework.

Usage: Ideal for both small and large-scale applications due to its scalability and ease of integration.

**Database:**

***MongoDB:***

Description: A NoSQL database that uses a document-oriented data model.

Key Features:

Schema-less: Flexible document structure that can be modified on the fly.

Scalability: Supports horizontal scaling through sharding.

Performance: Optimized for read and write operations with indexes and replication.

Usage: Suitable for applications requiring flexible data models and rapid iterations.

***PostgreSQL:***

Description: A powerful, open-source object-relational database system.

Key Features:

ACID Compliance: Ensures reliable transactions.

Advanced SQL Support: Includes support for complex queries and data integrity.

Extensibility: Allows adding custom functions using different programming languages.

Usage: Preferred for applications needing complex queries, data integrity, and high reliability.

**Development Tools:**

**IDEs:**

***Visual Studio Code:***

Description: A free, open-source code editor developed by Microsoft.

Key Features:

Integrated Git: Built-in Git commands and GitHub integration.

Extensions: Wide range of plugins and extensions for different languages and frameworks.

Debugging: Advanced debugging capabilities directly within the editor.

Usage: Versatile for both frontend and backend development across various programming languages.

***PyCharm:***

Description: An integrated development environment (IDE) specifically for Python, developed by JetBrains.

Key Features:

Code Completion: Intelligent code completion and on-the-fly error checking.

Debugging: Robust debugging tools tailored for Python.

Django Support: Built-in support for Django development.

Usage: Highly recommended for Python projects, especially those using the Django framework.

**Version Control:**

***Git:***

Description: A distributed version control system.

Key Features:

Branching and Merging: Facilitates multiple development workflows.

Distributed: Every developer has a full copy of the repository.

Speed: Fast performance for most operations.

Usage: Essential for managing source code changes and collaboration in development projects.

***GitHub/GitLab:***

Description: Platforms for hosting and managing Git repositories.

Key Features:

CI/CD Pipelines: Automated build and deployment workflows.

Issue Tracking: Tools for managing project issues and feature requests.

Collaboration: Supports code reviews, pull requests, and team discussions.

Usage: Critical for team collaboration, code management, and continuous integration/continuous deployment (CI/CD).

**Filter Level:**

**Middleware:**

***JWT (JSON Web Tokens):***

Description: A compact, URL-safe means of representing claims to be transferred between two parties.

Key Features:

Stateless Authentication: Tokens are self-contained and can be verified without server-side sessions.

Security: Supports signing and encryption to ensure data integrity and confidentiality.

Interoperability: Can be used across different platforms and languages.

Usage: Ideal for secure user authentication and authorization in web applications.

***Express Middleware:***

Description: Functions that execute during the lifecycle of an HTTP request in an Express application.

Key Features:

Logging: Middleware like Morgan for request logging.

Error Handling: Custom error-handling middleware for catching and responding to errors.

Validation: Middleware for validating request data (e.g., using Joi).

Usage: Enhances the functionality of Express applications by modularizing common tasks.

***Redux Middleware:***

Description: Middleware for managing state in React applications.

Key Features:

State Management: Centralizes application state for predictable behavior.

Side Effects: Handles asynchronous actions, logging, and other side effects.

Integration: Works seamlessly with React and other libraries.

Usage: Essential for managing complex state logic in React applications.

Data Transformation:

***Lodash:***

Description: A JavaScript utility library delivering modularity, performance, and extras.

Key Features:

Array and Object Manipulation: Simplifies working with arrays, objects, strings, etc.

Functional Programming: Supports chaining and functional programming paradigms.

Performance: Optimized for high performance in common tasks.

Usage: Useful for data manipulation and transformation in both frontend and backend code.

***AJV (Another JSON Validator):***

Description: A JSON schema validator used to ensure data integrity.

Key Features:

Schema Validation: Validates JSON data against predefined schemas.

Performance: Fast and efficient validation process.

Customization: Supports custom keywords and formats.

Usage: Ensures data consistency and integrity across the application.

**Pipe Level:**

**API Tools:**

***Axios:***

Description: A promise-based HTTP client for the browser and Node.js.

Key Features:

Requests and Responses: Simplifies making HTTP requests and handling responses.

Interceptors: Allows intercepting requests or responses before they are handled.

Cancellation: Supports request cancellation.

Usage: Preferred for making HTTP requests from frontend to backend in web applications.

***Fetch API:***

Description: A modern interface for making HTTP requests in web browsers.

Key Features:

Promises: Uses promises for handling asynchronous operations.

Streaming: Supports streaming requests and responses.

Standardization: A standardized way to fetch resources.

Usage: Ideal for making network requests in modern web applications.

***GraphQL:***

Description: A query language for your API, and a server-side runtime for executing queries.

Key Features:

Flexible Queries: Clients can request exactly what they need.

Type System: Strongly-typed schema to define the structure of the API.

Efficiency: Reduces over-fetching and under-fetching of data.

Usage: Suitable for applications needing efficient and flexible data retrieval.

Integration and Communication:

***WebSockets:***

Description: A protocol for full-duplex communication channels over a single TCP connection.

Key Features:

Real-Time Communication: Enables real-time data exchange between client and server.

Low Latency: Minimizes latency compared to traditional HTTP requests.

Bi-Directional: Supports bi-directional communication.

Usage: Crucial for real-time features like chat applications, live updates, and online gaming.

***RabbitMQ:***

Description: A message broker for communication between distributed systems.

Key Features:

Reliable Messaging: Ensures messages are delivered reliably and in order.

Flexible Routing: Supports different routing strategies for messages.

Scalability: Can handle high throughput with clustering and federation.

Usage: Useful for asynchronous communication and integrating microservices.

**Non-Functional Requirements:**

**Performance:**

***Nginx:***

Description: A high-performance web server and reverse proxy server.

Key Features:

Load Balancing: Distributes incoming traffic across multiple servers.

Caching: Caches static and dynamic content to improve response times.

SSL/TLS Termination: Offloads SSL/TLS encryption from backend servers.

Usage: Enhances the performance and reliability of web applications.

***Redis:***

Description: An in-memory data structure store, used as a database, cache, and message broker.

Key Features:

Caching: Caches frequently accessed data to reduce load on databases.

Data Structures: Supports strings, lists, sets, hashes, and more.

Persistence: Offers options for data persistence.

Usage: Speeds up data access and reduces latency in web applications.

**Scalability:**

***Kubernetes:***

Description: An open-source platform for automating deployment, scaling, and operations of application containers.

Key Features:

Orchestration: Manages containerized applications across multiple hosts.

Auto-Scaling: Automatically scales applications based on demand.

Service Discovery: Automatically assigns DNS names to services.

Usage: Facilitates scaling and management of containerized applications.

***Docker:***

Description: A platform for developing, shipping, and running applications in containers.

Key Features:

Containerization: Encapsulates applications and their dependencies in containers.

Portability: Ensures consistent environments from development to production.

Isolation: Provides process and network isolation.

Usage: Standardizes environments and simplifies deployment processes.

**Security:**

[FA21-BSE-052]

***OAuth 2.0:***

Description: An authorization framework that enables third-party applications to obtain limited access to user accounts.

Key Features:

Token-Based: Uses tokens to grant access without sharing credentials.

Scopes: Defines specific access permissions.

Flows: Supports multiple authorization flows for different use cases.

Usage: Secures access to APIs and services by external applications.

**Helmet.js:**

Description: A collection of middleware to help secure Express.js applications.

Key Features:

Security Headers: Sets HTTP headers to protect against common vulnerabilities.

XSS Protection: Enables Cross-Site Scripting (XSS) filters.

Content Security Policy: Helps prevent data injection attacks.

Usage: Enhances security by configuring appropriate HTTP headers.

**Reliability:**

***HAProxy:***

Description: A high-availability load balancer and proxy server.

Key Features:

Load Balancing: Distributes client requests across servers.

Health Checks: Monitors server health and reroutes traffic away from failed servers.

SSL Offloading: Offloads SSL processing from backend servers.

Usage: Ensures high availability and reliability of web services.

***AWS RDS:***

Description: A managed relational database service by Amazon Web Services.

Key Features:

Automated Backups: Provides automated backups and snapshots.

Replication: Supports multi-AZ deployment for high availability.

Scalability: Easily scales database instances up or down.

Usage: Manages relational databases with minimal operational overhead.

**Compliance:**

***Vault by HashiCorp:***

Description: A tool for securely accessing secrets and managing sensitive data.

Key Features:

Secret Management: Stores and controls access to secrets like API keys and passwords.

Encryption: Encrypts data at rest and in transit.

Access Control: Enforces strict access controls and audit logs.

Usage: Ensures secure storage and access to sensitive information.

***Open Policy Agent (OPA):***

Description: An open-source policy engine that unifies policy enforcement across the stack.

Key Features:

Policy Definition: Uses a high-level language to define policies.

Decoupled Policy: Separates policy decision-making from enforcement.

Integration: Works with a variety of systems and services.

Usage: Enforces policies and compliance requirements across microservices and applications.