Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	31 January 3035
Team ID	LTVIP2025TMID48209
Project Name	Heritage Treasures
Maximum Marks	4 Marks

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

Example: Order processing during pandemics for offline mode

Reference: https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing-during-pandemics/

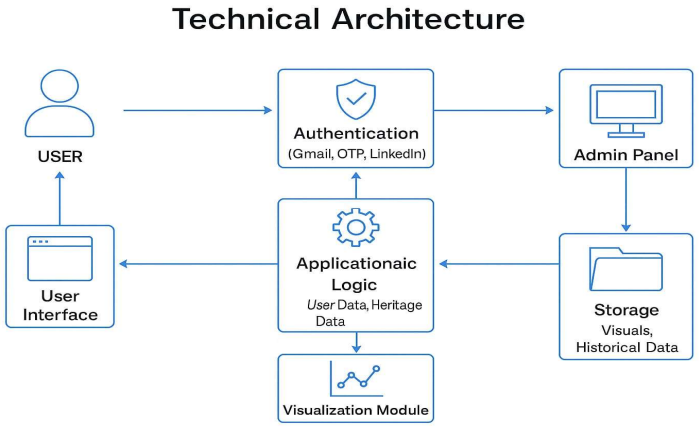

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How users interact with the application via web or mobile	HTML, CSS, JavaScript, React.js
2.	Application Logic-1	Logic for data visualization engine	Python, Tableau Extensions API
3.	Application Logic-2	Logic for data filtering and country-based insights	JavaScript/Node.js
4.	Application Logic-3	Logic for tracking endangered heritage sites	Python Flask (API calls)
5.	Database	Storage of UNESCO site metadata	MySQL, PostgreSQL
6.	Cloud Database	Scalable storage of structured data	Google Firebase / IBM Cloudant
7.	File Storage	File uploads for external CSV datasets or images	AWS S3 / IBM Cloud Object Storage
8.	External API-1	Geolocation API for mapping heritage sites	Mapbox API / Google Maps API
9.	External API-2	UNESCO Heritage site open data source	UNESCO API
10.	Machine Learning Model	Machine Learning Model Forecasting endangered sites based on risk patterns	Random Forest / Decision Tree
11.	Infrastructure (Server / Cloud)	Infrastructure (Server/Cloud) Hosting web app and backend services	Kubernetes, AWS EC2 / Google Cloud Run

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	React.js for frontend, Flask for backend, D3.js for charts	React.js, Flask, D3.js
2.	Security Implementations	HTTPS, API key restriction, OAuth 2.0, JWT authentication, Firewall setup	SSL, OAuth, JWT, OWASP practices
3.	Scalable Architecture	Microservices structure for DFD, ML, visualization, API layers	Docker, Kubernetes, Cloud Run

4.	Availability	Use of cloud-based load balancers and auto-scaling for uptime	AWS ELB, GCP Load Balancing
5.	Performance	Optimized response time, fast data rendering for large datasets, and asynchronous data loading to improve user experience	Redis caching, IndexedDB, Lazy Loading, Multithreading (Web Workers), GCP/AWS Performance Monitoring

References:

https://c4model.com/

https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/

https://www.ibm.com/cloud/architecture

https://aws.amazon.com/architecture

https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams-2d20c9fda90d