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

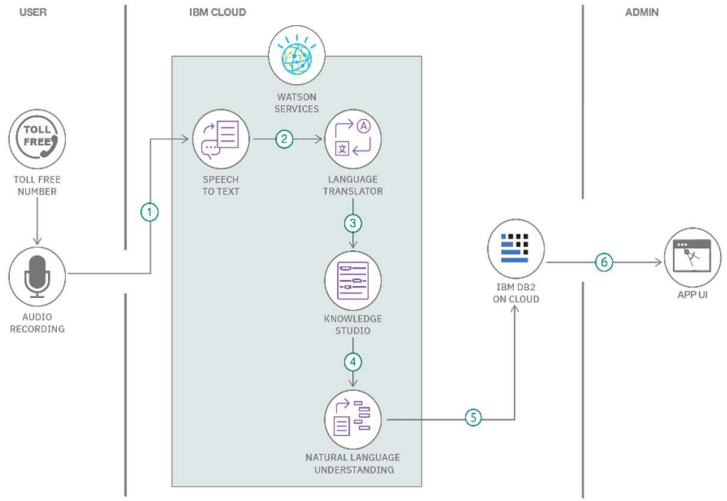
Date	26-06-2025
Team ID	LTVIP2025TMID19414
Project Name	BookNest
Maximum Marks	4 Marks

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/



Guidelines:

Include all the processes (As an application logic / Technology Block)
Provide infrastructural demarcation (Local / Cloud)
Indicate external interfaces (third party API's etc.)
Indicate Data Storage components / services
Indicate interface to machine learning models (if applicable)

S.No	Component	Description	Technology

1.	User Interface	Interface through which users operate the system	HTML, CSS,ReactJS+Vite/Bootstrap, CSS etc.
2.	Application Logic-1	Core processing and control logic of the application	JavaScript.
3.	Database	Data schema and setup configuration	MongoDB, Mongoose.
4.	File Storage	Mechanism for managing and storing files	MongoDB Cluster storage.
5.	External API-1	Integration of third-party services for specific tasks	(Specify as per use, e.g., Payment Gateway, Email API, etc.)
6.	External API-2	Additional external services integrated into the system	Specify as per use, e.g., Maps API, SMS API, etc.)

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	The UI leverages component-based libraries like React (via Vite), styled with Tailwind and Bootstrap. Axios handles API requests. Backend runs on Express framework over Node.js.	ReactJS (Vite bundler), TailwindCSS, Bootstrap UI, Axios (HTTP client), Express.js on Node.js
2.	Security Implementations	User credentials are hashed with bcrypt. Secure cross-origin requests are ensured via CORS. Input sanitation protects against code injection.	bcrypt (password hashing), CORS (cross-origin policy), express-validator, Helmet.js (optional)
3.	Scalable Architecture	. Built using a modular (layered) structure with isolated frontend, backend, and data layers. Docker can be used for deployment and scalability.	Modular Node.js services (Docker-capable, optionally microservice-based)
4.	Availability	. Hosted on cloud platforms (e.g., AWS, Render) enabling flexible resource allocation. Traffic distribution is managed via load balancing tools.	Cloud hosting (AWS, Render, etc.), Load Balancer / Nginx (optional)
5.	Performance	. Uses Axios for optimized HTTP calls, static content is cached using CDNs. MongoDB supports efficient read/write operations under heavy load.	Axios (HTTP requests), MongoDB (NoSQL), CDN (e.g., Cloudflare), Compression techniques

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