**Project Design Phase-II**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 26 June 3035 |
| Team ID | LTVIP2025TMID59601 |
| Project Name | orderonthego: your on-demand food ordering solution |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

**Table-1**

**:**

**Components**

**&**

**Technologies:**

**S.No**

**Component**

**Description**

**Technology**

1.

User

Interface

How

user

interacts

with

application

e.g.

Web

UI,

Mobile

App,

Chatbot

etc.

HTML,

CSS,

JavaScript

/

Angular

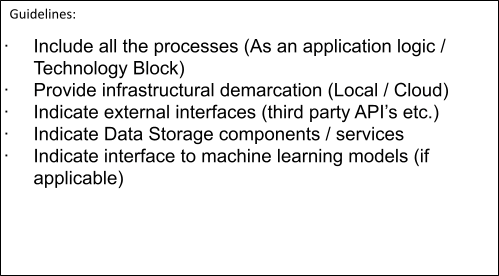
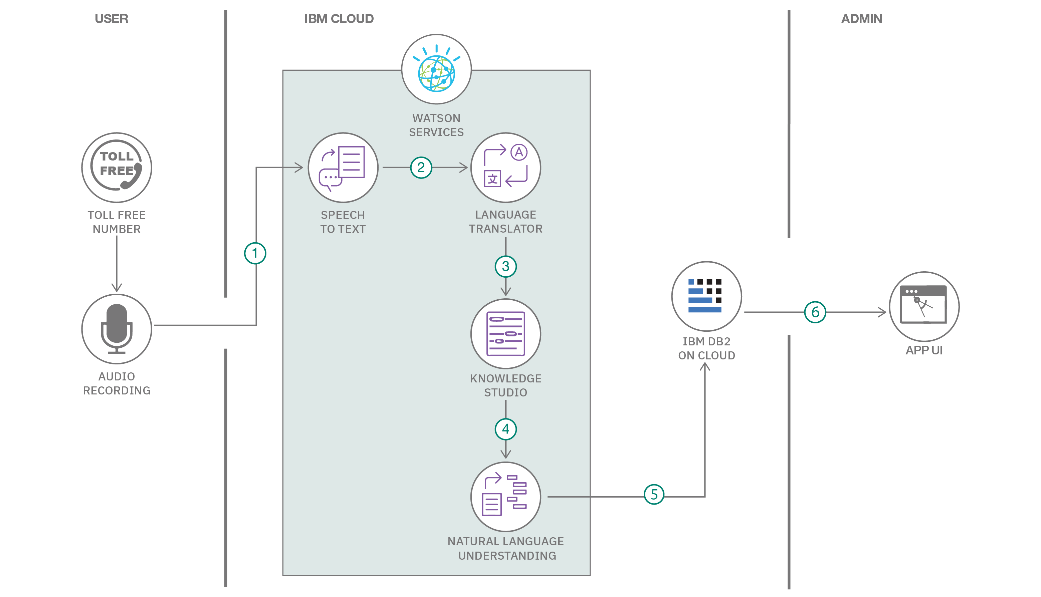
Js

/

React

Js

etc.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2. | Application Logic-1 | Logic for a process in the application | Node.js (Express.js Framework) | |
| 3. | Application Logic-2 | Logic for a process in the application | Not applicable for this project | |
| 4. | Database | Data Type, Configurations etc. | MongoDB (NoSQL) | |
| 5. | Cloud Database | Database Service on Cloud | Not applicable for this project | |
| 6. | File Storage | File storage requirements | Local Filesystem (or Cloud Storage if implemented later) |
| 7. | External API-1 | Purpose of External API used in the application | Not explicitly used for this project |
| 8. | External API-2 | Purpose of External API used in the application | Not explicitly used for this project |
| 9. | Machine Learning Model | Purpose of Machine Learning Model | Object Recognition Model, etc. |
| 10. | Infrastructure (Server / Cloud) | Application Deployment on Local System / Cloud | Local (for development), Cloud (e.g., AWS EC2, Heroku, or similar for deployment) |

**Table-2: Application Characteristics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Description** | **Technology** |
| 1. | Open-Source Frameworks | List the open-source frameworks used | Express.js, Mongoose, bcryptjs, jsonwebtoken, Chart.js, Bootstrap, Font Awesome |
| 2. | Security Implementations | List all the security / access controls implemented, use of firewalls etc. | JWT (JSON Web Tokens) for authentication and authorization, Password Hashing (bcryptjs), HTTPS (in production environment), Basic input validation. |
| 3. | Scalable Architecture | Justify the scalability of architecture (3 – tier, Micro-services) | 3-tier architecture (Frontend, Backend, Database). Can be scaled by deploying multiple backend instances behind a load balancer, and MongoDB's scaling features (e.g., sharding). |
| 4. | Availability | Justify the availability of application (e.g. use of load balancers, distributed servers etc.) | Achieved through potential future deployment on cloud services with load balancing and auto-scaling groups for the backend, and MongoDB's replica sets for database redundancy. |
| 5. | Performance | Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN’s) etc. | Efficient API design, optimized database queries (Mongoose), client-side rendering for responsiveness, future potential for Caching (Redis), CDN for static assets. |

**References:**

<https://mongoosejs.com/docs/>

<https://github.com/dcodeIO/bcrypt.js>