**Requirement Gathering and Analysis Phase**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date | 05-07-2024 |
| Team ID | SWTID1720010107 |
| Project Name | Freelancing Application – HireConnect |
| Maximum Marks |  |

**Technical Architecture:**

**Table-1 : Components & Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interface | How user interacts with application e.g.  Web UI, Mobile App, Chatbot etc. | ReactJS, ShadCN, TailwindCSS |
|  | Application Logic -1 | Authentication | ReactJS, bcrypt, NextAuth.js, JWT, mongoose |
|  | Application Logic-2 | Posting profiles and bounties | ReactJS, mongoose |
|  | Application Logic-3 | Chat functionality | ReactJS, mongoose |
|  | Database | Collections: Users, Tasks, Projects | MongoDB, mongoose |
|  | External API-1 | Handles the payments for the website. | Stripe Payments |
|  | External API-2 | Handles the indexing and searching of freelancers | Elasticsearch API |
|  | External API-3 | Handles the filtering based on location | Google Maps API |
|  | External API-4 | Handles fetching and categorising job titles and skills | LinkedIn API |

**Table-2: Application Characteristics:**

| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Open-Source Frameworks | MongoDB, Next.js, React.js, ShadCN, Tailwind CSS, Bcrypt | Technology of Opensource framework |
|  | Security Implementations | The architecture of the application is designed to be scalable using a microservices approach. This involves breaking down the application into small, independent services that can be developed, deployed, and scaled individually. Each service handles a specific piece of functionality and communicates with other services through APIs. | Bcrypt, JWT, Next Auth, NextJS, MongoDB |
|  | Scalable Architecture | The architecture of the application is designed to be scalable using a microservices approach. This involves breaking down the application into small, independent services that can be developed, deployed, and scaled individually. Each service handles a specific piece of functionality and communicates with other services through APIs. This allows for more granular scaling, as only the services experiencing high load need additional resources. | NextJS, MongoDB |
|  | Availability | High availability of the application is ensured through the use of load balancers and distributed servers. Load balancers distribute incoming traffic across multiple servers, preventing any single server from becoming a bottleneck and ensuring continuous service even if one or more servers fail. Additionally, deploying the application across multiple geographically distributed data centres minimizes downtime due to localized failures. However, for the sake of the project, we will only be deploying in one server | MongoDB Atlas |