**Requirement Gathering and Analysis Phase**

**Technology Stack (Architecture & Stack)**

|  |  |
| --- | --- |
| Date |  |
| Team ID |  |
| Project Name |  |
| Maximum Marks |  |

**Table-1: Components & Technologies for Movie App (Frontend)**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology/Tool(s)** |
| 1 | **User Interface** | How user interacts with the application (UI) | HTML, CSS, JavaScript, React.js |
| 2 | **Application Logic-1** | Logic for handling the app's internal functionality | JavaScript (React Components, React Hooks, Redux, etc.) |
| 3 | **External API-1** | API to fetch movie data (e.g., movie details, genres, etc.) | The Movie Database (TMDb) API, or OMDB API |
| 4 | **External API-2** | API for authentication or user login (optional) | Firebase Authentication, OAuth, or custom API for login |
| 5 | **External API-3** | API for search functionality (movie search, filters) | The Movie Database (TMDb) API (search queries) |
| 6 | **External API-4** | API for ratings or reviews (optional) | Yelp API, IMDb API, or TMDb ratings API |
| 7 | **External API-5** | API for movie trailers or videos | YouTube API, TMDb API (for embedded trailers) |
| 8 | **External API-6** | API for location-based movie showtimes (optional) | Google Maps API, Fandango API (if applicable) |
| 9 | **File Storage** | File storage requirements (for caching, media) | LocalStorage, IndexedDB (for offline storage) |
| 10 | **Cloud Database** | Database for storing user preferences, ratings, etc. | Firebase Realtime Database, Firestore (if necessary) |
| 11 | **Infrastructure (Server / Cloud)** | Deployment of front-end app in production | Netlify, Vercel, AWS S3, GitHub Pages, or any static hosting |

**Table-2: Application Characteristics for Movie App (Frontend)**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Characteristics** | **Description** | **Technology/Tool(s)** |
| 1 | **Open-Source Frameworks** | List the open-source frameworks used for the front-end | React.js, React Router, Redux (optional for state management) |
| 2 | **Security Implementations** | Security measures to protect data and interactions (client-side) | HTTPS, Content Security Policy (CSP), OWASP security best practices, Input validation (prevent XSS) |
| 3 | **Scalable Architecture** | Scalability considerations (app performance under increasing users) | Stateless app (frontend), CDN for static assets, Lazy Loading with React, Code Splitting |
| 4 | **Availability** | Justify availability of the application for users in different regions | Static hosting (Netlify, Vercel), use of global CDN (Cloudflare) |
| 5 | **Performance** | Performance design considerations (app speed, caching, request handling) | Caching strategies (localStorage, IndexedDB), Lazy Loading, CDN for faster static asset delivery |