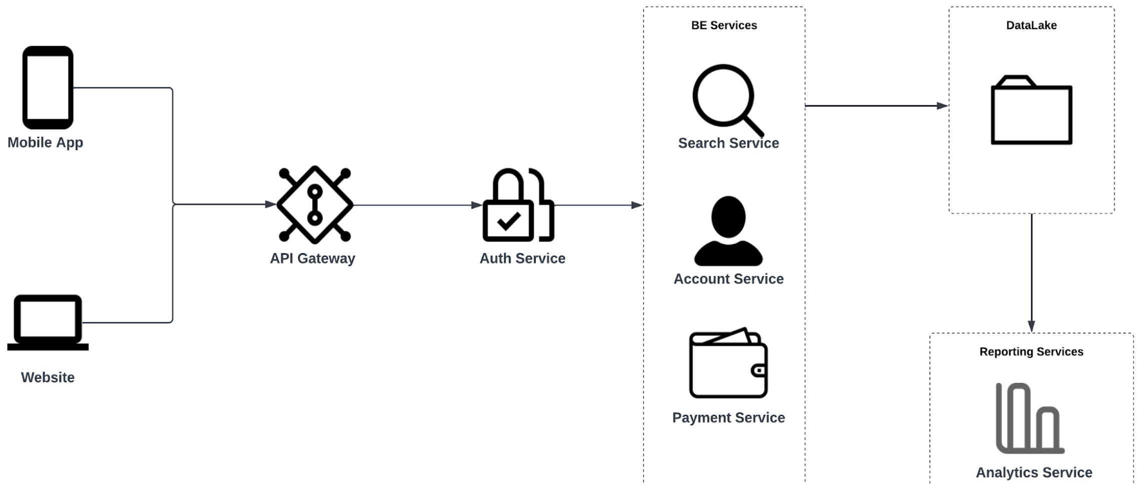
Requirement Gathering and Analysis Phase

Solution Architecture

Date	06 July 2024
Team ID	SWTID1719923176
Project Name	Project - Freelance Finder : Discovering Opportunities, Unlocking Potential.
Maximum Marks	3 Marks

Solution Architecture:



1. Frontend (Client Side)

- **Components**

  - User Interface: Pages for freelancers, clients, and admin panels.

  - Authentication: Login, registration, and user profile management.

  - Project Management: Listings, proposals, messaging, and payment functionalities.

  - Dashboard: Personalized views for managing projects, profiles, etc.

- **Technologies**

  - React: Frontend library for building interactive user interfaces.

  - Redux(optional): State management for managing application-wide state.

  - React Router: Navigation between different pages/components.

  - Axios: HTTP client for making API requests to the backend.

2.Backend (Server Side)

- Components

  - API Layer: Handles HTTP requests/responses.
  - Business Logic: Implements core application functionalities (e.g., project creation, user management).
  - Database Access: Manages interactions with the database.
  - Authentication: Handles user authentication and authorization.

- Technologies

  - Node.js: JavaScript runtime for server-side development.
  - Express.js: Web application framework for Node.js, simplifies API development.
  - MongoDB (or any other database): NoSQL database for storing user data, project details, etc.
  - Mongoose: Object Data Modeling (ODM) library for Node.js, simplifies interactions with MongoDB.
  - JWT (JSON Web Tokens): For secure authentication and authorization.
  - Passport.js (optional): Authentication middleware for Node.js.

3. Database

- MongoDb
  - Collections
    - Users: Stores freelancer and client information.
    - Projects: Stores project details, proposals, and status.
    - Messages: Stores communication history between users.

4. Integration with Third-party Services

- Payment Gateway: Integration with services like PayPal, Stripe for handling payments securely.
- Email Service: Utilize services like SendGrid, AWS SES for sending notifications and alerts.
- Cloud Storage: Services like AWS S3, Firebase Storage for storing file uploads (e.g., project documents).

5. Deployment and Scalability

- Deployment: Deploy frontend (React) and backend (Node.js + Express) separately or together on cloud platforms like AWS, Heroku, Azure.

- Scalability: Implement load balancing, caching strategies, database scaling techniques (e.gMongoDB sharding) as the user base grows.

6. Security Considerations

- HTTPS: Secure communication using HTTPS.

- Input Validation: Sanitize and validate user inputs to prevent injection attacks.

- Authorization: Implement role-based access control (RBAC) to restrict access to sensitive data and functionalities.

- Data Encryption: Encrypt sensitive data at rest (e.g., user passwords) and in transit (e.g., using TLS).