Practical-2

Aim: Define requirement Gathering and technical requirement specification for selected project.

* Requirement Gathering for Job Portal

**1. Define the Purpose**

The purpose of the portal is to create a seamless platform where candidates can explore, apply for, and manage job applications for software development positions. It also helps employers post job listings, review applications, and manage hiring processes efficiently.

**2. Identify the Stakeholders**

Primary Stakeholders:

* Job seekers: Candidates looking for software development roles.
* Recruiters: HR personnel and hiring managers posting job openings and managing applications.

Secondary Stakeholders:

* Administrators: Platform operators who maintain and oversee the portal.
* Referrers: Existing employees who might refer potential candidates.

**3. Conduct Interviews**

For Job Seekers:

* What features do you find most useful when searching for a job?
* What difficulties have you faced using existing job portals?
* What additional features would you like to see?

For Recruiters:

* How do you currently manage job postings and applications?
* What challenges do you face in sorting and shortlisting candidates?
* Would you prefer features like automated matching or scheduling interviews?

For Administrators:

* What is essential for platform monitoring and analytics?
* How often will updates or maintenance be required?

**4. Services and Questionnaires**

For Job Seekers:

* Preferred notification method: Email, SMS, or in-app?
* Feedback mechanism on the application process.

For Recruiters:

* Rank the importance of features: resume parsing, applicant tracking, job posting customization, etc.

**5. Observations**

Observe real-world scenarios where current job portals are used:

* Analyze the application flow of existing job portals like LinkedIn, Glassdoor, or Indeed.
* Identify bottlenecks in user journeys (e.g., slow profile setup or poor search results).

**6. Prototype**

* Wireframe Designs: Create sketches of user interfaces, including the job seeker dashboard, recruiter dashboard, and admin panel.
* User Flow Diagrams: Show how a user navigates through the portal, from registration to job application.
* Use tools like Figma or Adobe XD for designing interactive prototypes.

**7. Document Requirements**

Functional Requirements:

* For Job Seekers:
  + Profile creation and resume upload.
  + Advanced job search with filters (location, experience, skills).
  + Application tracking and status updates.
* For Recruiters:
  + Job posting templates.
  + Resume screening and shortlisting.
  + Messaging and interview scheduling
* For Admins:
  + Monitoring and analytics dashboards.
  + User management (e.g., ban or activate accounts).

Non-Functional Requirements:

* High platform availability and uptime.
* Data security and privacy compliance.
* Mobile-friendly interface.

**8. Prioritize Your Requirements**

High Priority:

* User registration and login.
* Job posting and application management.
* Search and filtering functionality.

Medium Priority:

* Resume parsing and automated recommendations.
* Notifications and reminders.

Low Priority:

* Gamification elements like badges for profile completion.
* Social media integration.

**9. Review and Validate**

* Stakeholder Review: Share prototypes and requirements with stakeholders to gather feedback.
* Validation:
  + Conduct usability testing with a group of job seekers and recruiters.
  + Verify that all critical features align with business objectives and user needs.
* **Technical requirement specification**

**1. System Architecture**

A 3-tier architecture is suitable for scalability and maintainability:

* Presentation Layer: Handles the user interface (UI) and user experience (UX) through a web or mobile application.
* Application Layer: Contains business logic, such as user authentication, job recommendations, and application tracking.
* Data Layer: Manages data storage and retrieval, including user profiles, job listings, and application histories.

**2. Technology Stack**

* Frontend:
  + Frameworks: React.js or Angular
  + Styling: Tailwind CSS or Bootstrap
  + State management: Redux or Context API
* Backend:
  + Language: Node.js or Python (Django/Flask)
  + Framework: Express.js for Node.js
  + APIs: RESTful or GraphQL
* Database:
  + Relational: PostgreSQL or MySQL for structured data (e.g., job listings, user profiles).
  + NoSQL: MongoDB for flexible data storage (e.g., activity logs).
* Authentication & Security:
  + OAuth 2.0, JWT (JSON Web Tokens) for secure authentication.
  + HTTPS for encrypted communication.
* Hosting:
  + Cloud services: AWS, Azure, or Google Cloud.
  + CDN: Cloudflare or AWS CloudFront for fast content delivery.

**3. Maintainability, Scalability, and Security**

* Maintainability:
  + Modular code with reusable components.
  + CI/CD pipelines for frequent updates and bug fixes.
* Scalability:
  + Horizontal scaling for handling increased traffic.
  + Load balancing (e.g., Nginx or AWS Elastic Load Balancer).
* Security:
  + Regular vulnerability assessments.
  + Encryption for sensitive data (e.g., passwords, personal details).
  + Role-based access control (RBAC) for different user permissions.

**4. Database Design**

* Tables for Relational Database:
  + Users: Stores user details (e.g., userID, name, email, password).
  + JobListings: Stores job details (e.g., jobID, title, company, description).
  + Applications: Tracks job applications (e.g., appID, userID, jobID, status).
  + Admins: Stores admin credentials and permissions.
* Indexes: Optimize search queries for job listings and applications.
* Relationships:
  + One-to-many: A user can apply to many jobs.
  + One-to-many: A job can receive applications from many users.

**5. Interfaces**

* Frontend Interfaces:
  + Job Seeker Dashboard: Profile, search, application status.
  + Recruiter Dashboard: Job postings, applicant list, communication tools.
* Backend Interfaces:
  + APIs for CRUD operations on users, jobs, and applications.
  + Admin Panel: Manage users, job listings, and analytics.

**6. Performance Requirements**

* Page load time: Under 2 seconds for optimal UX.
* Concurrent users: Support for 10,000+ users initially, scalable to 1M.
* Uptime: 99.9% availability.

**7. Deployment Plan**

1. Development Environment Setup: Configure local and staging environments.
2. Code Deployment: Use CI/CD pipelines for automated deployment.
3. Server Configuration: Set up web and database servers on cloud platforms.
4. Domain Configuration: Link the portal to a registered domain.
5. Testing: Conduct end-to-end testing on staging before production.
6. Go Live: Roll out to production with a soft launch (beta users).
7. **Maintenance and Support**

* Scheduled Updates: Regular feature updates and performance optimization.
* Monitoring Tools: Use tools like Datadog or New Relic for real-time monitoring.
* Customer Support: Offer email/chat support for users and recruiters.

1. **Documentation**

User Manual:

* How to create an account, search for jobs, and apply.
* Tips for optimizing profiles for better job recommendations.

Technical Manual:

* API documentation for integration with other systems.
* Database schema and query optimization guidelines.

Troubleshooting Manual:

* Common issues (e.g., password recovery, application submission errors).
* Error codes and resolutions.