## **Table of Contents**

**Introduction**

1.1 Purpose of Document

1.2 Scope

* 1.3 Target Audience

**System Overview**

2.1 Background

2.2 Objectives

2.3 Features

* 2.4 Technology Stack

**User Requirements**

3.1 Job Seekers

3.2 Employers

* 3.3 Recruiters

**Functional Specifications**

4.1 Registration and User Profiles

4.2 Job Search and Recommendations

4.3 Job Posting and Application

4.4 Resume Builder and Profile Enhancement

4.5 Messaging and Communication

4.6 Notifications and Alerts

* 4.7 Payment and Subscription

**Non-functional Requirements**

5.1 Performance

5.2 Security

5.3 Scalability

5.4 Usability

* 5.5 Accessibility

**System Architecture**

6.1 High-Level Architecture

6.2 Database Design

6.3 Server-Side Logic

6.4 Front-End Components

* 6.5 Integration Points

**Data Flow Diagram**

7.1 User Registration and Profile Creation

7.2 Job Search and Application

7.3 Messaging and Communication

* 7.4 Payment Processing

**User Interface Design**

8.1 Home Page

8.2 User Dashboard

8.3 Job Listing Page

8.4 Job Details Page

8.5 User Profile Page

8.6 Employer Dashboard

8.7 Recruiter Dashboard

* 8.8 Messaging Interface

**Testing Strategy**

9.1 Unit Testing

9.2 Integration Testing

9.3 System Testing

* 9.4 User Acceptance Testing

**Deployment Plan**

10.1 Development Environment

10.2 Testing Environment

* 10.3 Production Environment

**Maintenance and Support**

11.1 Bug Tracking and Resolution

* 11.2 Updates and Enhancements

**Conclusion**

12.1 Summary

* 12.2 Future Enhancements

## **1. Introduction**

### **1.1 Purpose of Document**

This document outlines the design of employerspick.com, a platform connecting job seekers with employers and recruiters.

### **1.2 Scope**

The scope includes user roles, functionalities, system architecture, and the design rationale.

### **1.3 Target Audience**

Developers, designers, testers, and other stakeholders involved in the development and maintenance of employerspick.com.

## **2. System Overview**

### **2.1 Background**

getmeajob.com is a leading job portal facilitating job search, recruitment, and talent acquisition.

### **2.2 Objectives**

To provide an intuitive, secure, and efficient platform for job seekers, employers, and recruiters.

### **2.3 Features**

* Job Search and Recommendations
* Job Posting and Application
* Resume Builder
* Messaging and Communication
* Notifications
* Subscription Plans

### **2.4 Technology Stack**

* Frontend: Angular
* Backend: Node.js, Express
* Database: MongoDB
* Authentication: JWT
* Messaging: WebSocket
* Payment: Stripe

## **3. User Requirements**

### **3.1 Job Seekers**

* Create and manage profiles
* Search for jobs
* Receive job recommendations
* Apply for jobs

### **3.2 Employers**

* Post and manage job listings
* Search for candidates
* Communicate with candidates
* View applicant profiles

### **3.3 Recruiters**

* Search for talent
* Connect with job seekers
* Manage recruitment processes

## **4. Functional Specifications**

### **4.1 Registration and User Profiles**

* Secure registration process
* Detailed user profiles
* Resume upload and parsing

### **4.2 Job Search and Recommendations**

* Advanced search filters
* Personalized job recommendations

### **4.3 Job Posting and Application**

* Easy job posting process
* Job application tracking
* Application history for job seekers

### **4.4 Resume Builder and Profile Enhancement**

* Intuitive resume builder
* Profile completeness tracking
* Tips for profile enhancement

### **4.5 Messaging and Communication**

* Real-time messaging
* Interview scheduling
* Notifications for new messages

### **4.6 Notifications and Alerts**

* Email and in-app notifications
* Job application status updates
* Subscription renewal reminders

### **4.7 Payment and Subscription**

* Secure payment gateway integration
* Subscription plans for employers and recruiters
* Invoices and billing history

## **5. Non-functional Requirements**

### **5.1 Performance**

* Fast response times
* Scalable architecture

### **5.2 Security**

* SSL encryption
* Secure user authentication
* Data encryption

### **5.3 Scalability**

* Horizontal scalability
* Load balancing

### **5.4 Usability**

* Intuitive UI/UX
* Responsive design

### **5.5 Accessibility**

* Compliance with accessibility standards
* Screen reader compatibility

## **6. System Architecture**

### **6.1 High-Level Architecture**

* Client-Server architecture
* Microservices for scalability

### **6.2 Database Design**

* MongoDB for flexible data storage

### **6.3 Server-Side Logic**

* Node.js for server-side logic
* Express.js for API development

### **6.4 Front-End Components**

* Angular for dynamic UI components

### **6.5 Integration Points**

* Third-party integrations for messaging and payments

## **7. Data Flow Diagram**

### **7.1 User Registration and Profile Creation**

[Diagram]

### **7.2 Job Search and Application**

[Diagram]

### **7.3 Messaging and Communication**

[Diagram]

### **7.4 Payment Processing**

[Diagram]

## **8. User Interface Design**

[Wireframes and UI mockups]

## **9. Testing Strategy**

### **9.1 Unit Testing**

* Test individual components

### **9.2 Integration Testing**

* Test interactions between components

### **9.3 System Testing**

* Test end-to-end functionality

### **9.4 User Acceptance Testing**

* Involve users to validate the system

## **10. Deployment Plan**

### **10.1 Development Environment**

* Local development setup

### **10.2 Testing Environment**

* Staging environment for testing

### **10.3 Production Environment**

* Deployment to live servers

## **11. Maintenance and Support**

### **11.1 Bug Tracking and Resolution**

* Use of issue tracking tools

### **11.2 Updates and Enhancements**

* Regular updates based on user feedback

## **12. Conclusion**

### **12.1 Summary**

* Recap of key design elements

### **12.2 Future Enhancements**

* Potential features for future development