**High-Level Design Document for Insurance Aggregator Application**

Version: 0.1.0 Updated: 04/03/2024

**SUBMITTED BY**

Haripriya Narravula

Preethi Ponnipenta

Vinitha Rajendran

Rizwan Mohammad

Jagadeeshwari S V

Sangeetha M

Sindhu Chinnathambi

Divyanshu Kumar Singh

Mounika M

**SUBMITTED TO:**

ALCHEMY TEAM





|  |  |  |
| --- | --- | --- |
| **S.No.** | **Table of Content** | **Page no.** |
| 1. | Introduction | 3 |
|  | 1.1 Purpose | 3 |
|  | 1.2 Scope | 3 |
|  | 1.3 Architectural Overview | 4 |
| 2. | Key objectives and goals of the system | 5 |
|  | 2.1 Core objectives | 5 |
|  | 2.2 Stakeholders | 5 |
| 3. | Requirements | 5 |
|  | 3.1 Functional Requirements | 5 |
|  | 3.1.1 User registration and Authentication | 5 |
|  | 3.1.2 Insurance policy search and comparison | 6 |
|  | 3.1.3 Policy details and information | 6 |
|  | 3.1.4 Code generation and purchase | 6 |
|  | 3.1.5 User dashboard | 7 |
|  | 3.1.6 Notifications and alerts | 7 |
|  | 3.1.7 Admin panel | 8 |
|  | 3.1.8 User registration and login | 8 |
|  | 3.1.9 Profile management | 8 |
|  | 3.1.10 Admin modules | 8 |
|  | 3.1.11 Reports | 9 |
|  | 3.2 Non-functional requirements | 9 |
|  | 3.2.1 Performance | 9 |
|  | 3.2.2 Security | 9 |
|  | 3.2.3 Usability | 10 |
| 4. | Assumptions and prerequisites | 11 |
| 5. | High level design | 12 |
|  | 5.1 System components | 12 |
|  | 5.2 Communication | 13 |
|  | 5.3 Deployment | 13 |
|  | 5.4 Third party integration | 13 |
|  | 5.5 User acceptance criteria | 13 |
|  | 5.6 Risks and mitigation strategies | 14 |
| 6. | Conclusion | 15 |

**High-Level Design Document**

**1.Introduction:**

**1.1 Purpose:**

The purpose of this document is to define the requirements for the development of a web-based insurance aggregator application. The application aims to provide users with a centralized platform to compare, choose, and purchase insurance policies from various providers.

**1.2 Scope:**

The web-based insurance aggregator application will cover a range of insurance types, including but not limited to health insurance, auto insurance, home insurance, and travel insurance. It will facilitate the comparison of policy features, premiums, and terms from multiple insurance providers.

**1.3 Architecture Overview:**

An architectural overview for an insurance aggregator project involves designing a scalable and robust system capable of integrating with multiple insurance providers, handling user interactions, managing policies, and ensuring security and reliability.

**Architecture Diagram:**

**A diagram of a software system

Description automatically generated**

**2. Key Objectives and Goals of the System:**

**2.1 Core Objectives**

1. Provide users with a user-friendly platform for comparing insurance policies.

2. Enable users to make informed decisions by presenting detailed policy

information.

3. Streamline the insurance purchasing process through a secure and intuitive

interface.

4. Foster partnerships with insurance providers for a comprehensive policy

offering.

**2.2 Stakeholders**

- Users (Insurance seekers)

- Insurance Providers

- Regulatory Authorities (if applicable)

- Customer Support

**3. Requirements:**

**3.1 Functional Requirements**

**3.1.1 User Registration and Authentication**

- Users can create accounts, log in, and manage their profiles.

- Secure authentication processes, including password recovery and two-factor

authentication.

**3.1.2 Insurance Policy Search and Comparison**

**3.1.2.1 Search Functionality**

- Users can search for insurance policies based on parameters like coverage

type, premium range, and policy duration.

- Advanced search filters for specific coverage features.

**3.1.2.2 Comparison Tools**

- Side-by-side comparison of policy features, coverage, and premium rates.

- Visual aids (charts, graphs) for a clear understanding of policy differences.

**3.1.3 Policy Details and Information**

**3.1.3.1 Policy Descriptions**

- Detailed information on each insurance policy, including coverage details and

exclusions.

- Plain language explanations of insurance jargon.

**3.1.3.2 Reviews and Ratings**

- User-generated reviews and ratings for each insurance policy.

- Integration with social media for sharing experiences.

**3.1.4 Quote Generation and Purchase**

**3.1.4.1 Quote Generation**

- Users can generate instant quotes based on their selected coverage options.

- Clear breakdown of premium components.

**3.1.4.2 Seamless Purchase Process**

- Secure payment gateway integration for policy purchases.

- Digital document signing for policy acceptance.

**3.1.5 User Dashboard**

**3.1.5.1 Policy Management**

- Overview of purchased policies, renewal dates, and coverage details.

- Policy renewal reminders.

**3.1.5.2 Claim Submission**

- Users can initiate and track insurance claims through the platform.

- Guidance on the claims process.

**3.1.6 Notifications and Alerts**

**3.1.6.1 Personalized Alerts**

- Notifications for policy renewals, new offers, and updates.

- Email and in-app notifications.

**3.1.7 Admin Panel**

**3.1.7.1 Provider Management**

- Add, update, and remove insurance providers.

- Monitor provider performance and user reviews.

**3.1.7.2 Content Management**

- Manage and update policy details, descriptions, and educational content.

- Ensure compliance with regulatory changes.

**3.1.8 User Registration & Login**

- Registration: Sign up using personal details

- Login: Access using credentials, possibly with added security.

**3.1.9 Profile Management**

-View and update personal details

-View policy details like policy number, coverage, premiums, and renewal

dates.

**3.1.10. Admin Modules:**

- Policy + User Onboarding

- Work on policies CRUD activities

- Work on user CRUD activities

- Work on claims CRUD activities

**3.1.11. Reports**

- User on-boarding reports

- Claim processing reports

- Policy Reports

- Policy renewal alerts.

- Coverage details and benefits.

**3.2. Non-Functional Requirements**

**3.2.1 Performance**

**3.2.1.1 Response Time**

- Ensure response times for search, comparison, and purchase processes are

within 3 seconds.

**3.2.1.2 Scalability**

- The system should accommodate a 30% increase in concurrent users over the

next year.

**3.2.2 Security**

**3.2.2.1 Data Encryption**

- Implement encryption for user data and payment transactions.

- Regular security audits and vulnerability assessments.

- Secure data storage

- Secure Data Communication

**3.2.2.2 Compliance**

- Adhere to data protection regulations and industry standards.

**3.2.3 Usability**

**3.2.3.1 Intuitive Interface**

- User testing for interface usability.

- Accessibility features for diverse user needs.

**3.2.3.2 Mobile Responsiveness**

- Ensure a seamless experience on various devices, including smartphones and

tablets.

**4. Assumptions and Prerequisites:**

**4.1 Technology Stack:**

* Sprint Boot
* MySQL
* NodeJS
* AngularJS
* Junit
* Java - 18
* Jira Software
* Docker
* Jenkins
* AWS
* Figma & Draw.io for design
* Reporting/Visualization Tool

Technology Stack Frontend: HTML, CSS, JavaScript (AngularJS)

Backend: Java (Spring Boot)

Database: MySQL

**4.2 Infrastructure requirements:**

• Web Servers

• Application Servers

• Database

• Caching

• Load Balancing

• Content Delivery Network (CDN)

• Security

• Monitoring and Logging

• Scalability

• High Availability

• Backup and Disaster Recovery

• Scalable Storage

• API Integration

• Containerization and Orchestration

**4.3 Data Privacy and Compliance:**

**4.3.1 Data Encryption & Encryption**

- Implement encryption for user data and payment transactions.

- Regular security audits and vulnerability assessments.

- Secure data storage

- Secure Data Communication

**4.3.2 Compliance**

- Adhere to data protection regulations and industry standards.

**5. High-Level Design:**

**5.1 System Components:**

- Frontend (AngularJS): Provides the user interface for interacting with the system.

- API Gateway (NodeJS/Express): Handles incoming requests, routes them to the appropriate microservice, and provides a unified interface to the clients.

- Microservices (Java/Spring): Individual services responsible for specific functionalities such as user management, policy search, comparison, quote generation, purchase, etc.

- Database (MySQL/MongoDB): Stores user data, policy information, and other relevant data.

- CI/CD (Jenkins): Implements Continuous Integration and Continuous Deployment pipelines for automated testing and deployment.

- AWS (Deployment): Hosting platform for deploying the system.

**5.2 Communication:**

- Frontend communicates with the API Gateway via RESTful APIs.

- CI/CD pipeline ensures smooth integration, testing, and deployment of changes.

**5.3 Deployment:**

- The system will be deployed on AWS using services like EC2, S3, RDS, etc.

- Docker containers can be used for containerization to ensure consistency across different environments.

**5.4 Third Party Integration:**

* Insurance Provider APIs: Integrate with APIs provided by various insurance companies to fetch quotes, policy details, coverage options, and other relevant information. These APIs allow users to compare different insurance products and select the one that best fits their needs.
* Payment Gateways: Incorporate payment gateways to facilitate premium payments and policy purchases securely.
* Identity Verification Services: Integrate identity verification services to authenticate users and prevent fraud during the registration and policy purchase process.

**5.5 User Acceptance Criteria:**

**5.5.1 Registration and Authentication:**

- Users can successfully register and log in with valid credentials.

- Two-factor authentication is functional.

**5.5.2 Search and Comparison:**

- Users can find relevant insurance policies using search filters.

- Side-by-side comparison results in a clear understanding of policy differences.

**5.5.3 Quote Generation and Purchase:**

- Users can generate accurate quotes based on selected coverage.

- The purchase process is completed without errors.

**5.5.4 User Dashboard:**

- Policies are accurately displayed in the user dashboard.

- Users receive timely notifications and alerts.

**5.6. Risks and Mitigation Strategies:**

**5.6.1 Risks:**

- Integration Challenges: Potential difficulties in integrating with diverse

insurance providers.

- Regulatory Compliance: Changes in regulatory requirements affecting the

platform.

**5.6.2 Mitigation Strategies:**

- Integration Challenges: Conduct thorough testing with each insurance

provider during the development phase.

- Regulatory Compliance: Establish a dedicated team to monitor and

implement compliance changes promptly.

**6. Conclusion:**

The development of an insurance aggregator represents a significant opportunity to revolutionize the insurance industry by providing users with a convenient, transparent, and efficient platform for comparing and purchasing insurance policies from multiple providers. Through careful planning, meticulous design, and strategic integration of third-party services, an insurance aggregator can offer a seamless user experience while driving business growth and innovation in the industry.

As the insurance aggregator continues to evolve and expand its offerings, it must remain agile and adaptable to changes in market dynamics, technological advancements, and regulatory requirements. By staying ahead of the curve and embracing innovation, an insurance aggregator can position itself as a leader in the industry, driving positive change and delivering value to both users and insurance providers alike.

In summary, the development of an insurance aggregator represents a transformative opportunity to reshape the insurance landscape, empower users with greater choice and control, and drive efficiency and competitiveness in the market. With a clear vision, robust strategy, and relentless commitment to excellence, an insurance aggregator can unlock untapped potential and pave the way for a more accessible, inclusive, and responsive insurance ecosystem.