INSURANCE SAMADHAN

Business Concept :-

Insurance Samadhan: India's Trusted Platform for Resolving Insurance Complaints

Insurance Samadhan is India's most trusted platform dedicated to resolving insurance-related complaints. Specializing in helping customers secure their rightful insurance claim settlements, the platform provides expert guidance and support throughout the resolution process.

Fee Structure:

- A **one-time registration fee** of ₹799 is charged when customers file their complaints. However, if the case remains unresolved, the registration fee is fully refunded.
- Upon the **successful resolution** of a case, Insurance Samadhan charges a **15% success fee** of the claim amount, plus **GST**.

Expert Teams: Insurance Samadhan boasts a robust team of professionals, including:

- **Mailing team** for handling communications.
- **Ombudsman team** for addressing disputes.
- **Legal team** and **advocates** to manage legal proceedings, especially in cases involving insurance fraud.

In cases where fraud by an insurance company is detected, Insurance Samadhan takes necessary **legal action** to ensure justice for the customer.

Leaders :-

Deepak Bhuvneshwari Uniyal (Co-Founder & CEO)

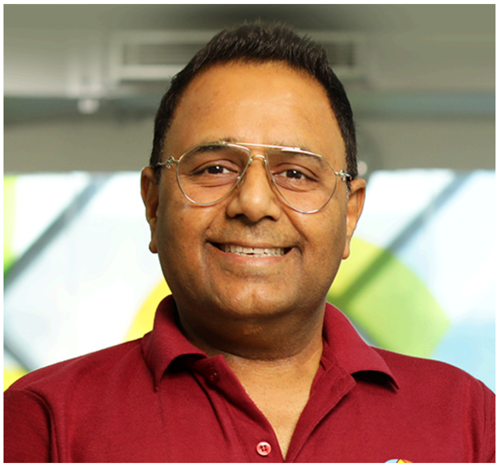
Deepak comes with 17 years of insurance expertise out of 26 years of work experience. He is dedicated to enhancing transparency and improving customer experience in the Indian insurance industry by leveraging his robust connections and adept problem-solving skills. He has a positive standing and a respected reputation among CXOs in the Insurance and Finance sectors.

Shilpa Arora (Co-founder & COO)

Shilpa Arora is the soul of InSa, crazily passionate about helping customers resolve their insurance grievances and ensuring they receive fair and timely settlements. She has a total of 28 years of working experience in Pharma and Insurance, which comes very handy in her day-to-day routine in leading the operations. Shilpa's expertise and dedication are vital in achieving Insurance Samadhan's mission to revolutionize the Indian insurance industry.

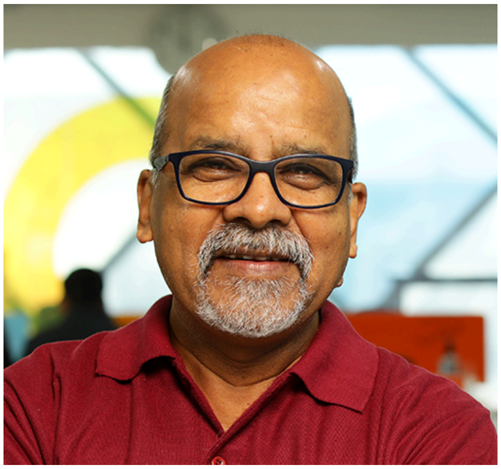
Ravi Mathur(Co-founder & CTO)

With a strong background in technology and innovation of 14 years, Ravi leads the development and implementation of the platform's cutting-edge technology solutions. His expertise and vision have been instrumental in creating an efficient and user-friendly platform that enables customers to resolve their insurance issues quickly and effectively. Ravi was chosen one in 40 under 40 in 2021 by Fortune INDIA.



Sanjay Aggarwal (Co-founder & Legal Head)

Sanjay Aggarwal has over 28 years of legal expertise and has founded multiple companies. With a vast amount of legal knowledge and experience, he ensures that the platform operates in compliance with the legal framework while safeguarding its customers' interests. His impeccable ability to deal with the most complex situations greatly supports the InSa family.



Shailesh Kumar (Co-founder & Insurance Head)

Shailesh is a distinguished insurance professional and passionate consumer rights activist. He is very passionate about empowering policyholders and promoting fairness in the industry. Shailesh has 23 years of experience in the insurance domain out of his total 35 years of working career. His deep knowledge of the products in the industry and the t&c adds tremendous value in getting the resolution to the policyholder. At 65 years of age, he is the youngest team member in InSa among most colleagues who are in their mid 20's

Language Sync :-

Insurance Type
Policy Type
Insurance Company
Claim Amount
B2C/B2B
Lead
Lead Status
Complaint
Complaint Status
Ombudsman
IGMS
Legal
Source
Manager
Partner
Admin
Executive
Expert

Core Business Logic :-

At Insurance Samadhan, customers can register their case through the web or Polifyx by providing basic details such as policy type, claim amount, policy number, uploaded documents, and company type. Once the details are submitted, an executive verifies the information. If the details are invalid, the case is rejected; if valid, it is assigned to an expert. The expert reviews the documents, and if they meet the requirements, the case is accepted. Before acceptance, the customer must pay a ₹799 registration fee, which is refunded if the case is deemed invalid.

After acceptance, the case moves to the mailing phase, where a formal complaint is sent to the insurance company. If the company refuses to respond or denies the claim, the case is escalated to the ombudsman level. If the issue is still unresolved, it proceeds to the legal phase, where it is assigned to an advocate, and hearing dates are sent to the company at scheduled intervals.

If the company agrees to pay, the case is marked as settled, and the company compensates the customer. In case the customer refuses to pay the agreed-upon 15% fee to Insurance Samadhan, legal action will be taken against the customer, following a process similar to that applied to the company.

Individuals can join Insurance Samadhan as agents through the partnership program by registering. As agents, their role is to collect cases from customers. The compensation depends on the agent's level: partners receive 5.5% of the claim amount, associates receive 4.5%, and advisors receive 2.5%. Partners can also recruit associates and advisors under them. When an associate or advisor collects cases, a portion of the payment is also shared with the partner and associate.

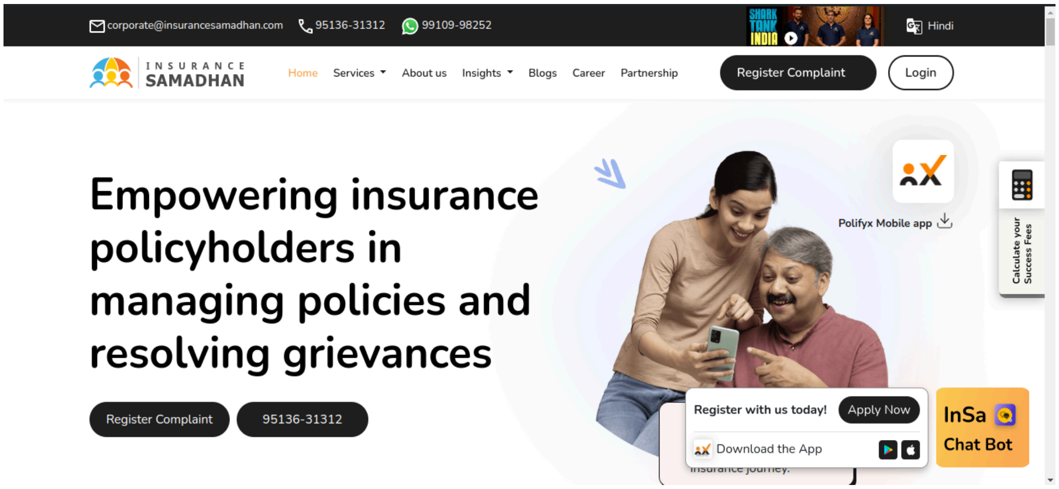
Partners at Insurance Samadhan are required to sign a contract when joining the partnership program, ensuring their agreement to the terms before they begin collecting cases and earning commissions.

Insurance Samadhan features a partner portal where partners can access customer details and register cases. They can fill in necessary information, upload relevant documents, and check their payout reports to track their earnings and performance.

Core Technical Aspects :-

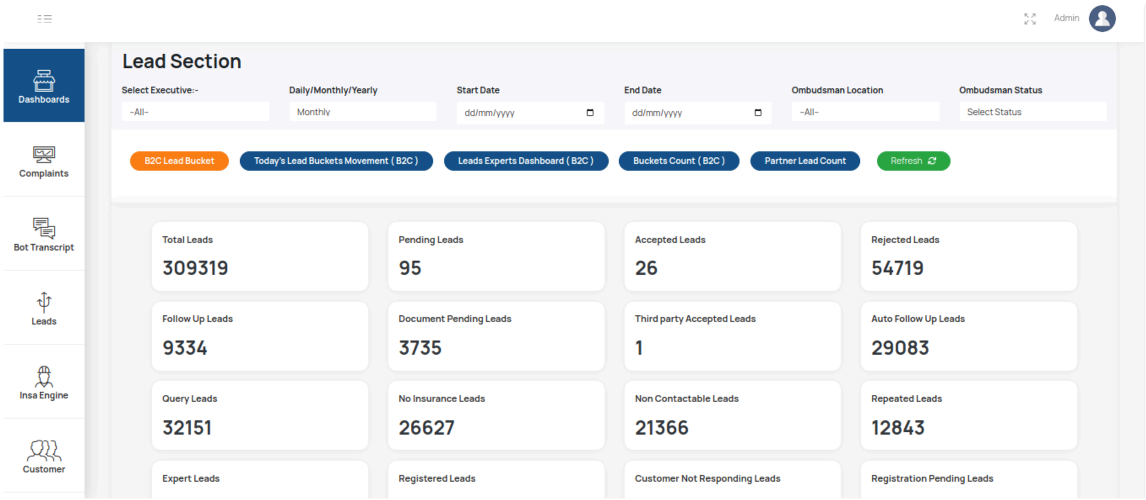
1. Technology Stack
   - Backend : NodeJS v18
   - Frontend : ReactJS
   - Database: MongoDB
   - Authentication : JWT token

2. Project Structure
   - Main Website : Here users can directly register their complaint and thereafter users can login and redirect to the user's portal to see complaint status, progress and personal details etc.



Empowering insurance policyholders in managing policies and resolving grievances

   - Admin Portal :
     - Admin
     - Executive
     - Expert



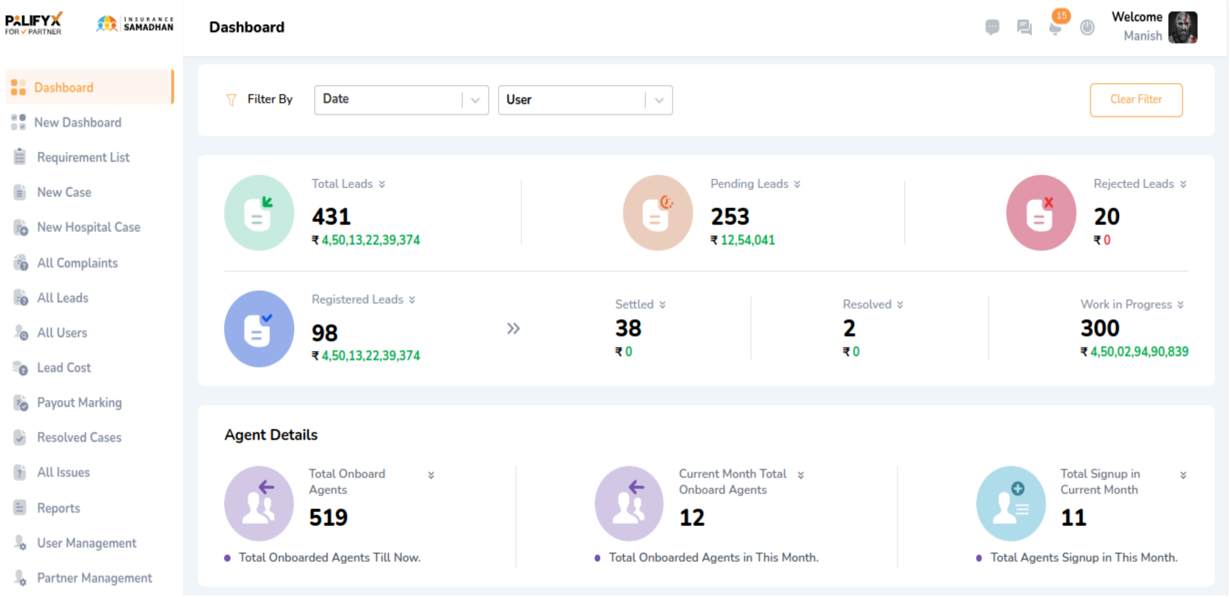
Lead Section

Select Executive:- -All-
Daily/Monthly/Yearly Monthly
Start Date dd/mm/yyyy
End Date dd/mm/yyyy
Ombudsman Location -All-
Ombudsman Status Select Status

B2C Lead Bucket | Today's Lead Buckets Movement (B2C) | Leads Experts Dashboard (B2C) | Buckets Count (B2C) | Partner Lead Count | Refresh

Total Leads 309319
Pending Leads 95
Accepted Leads 26
Rejected Leads 54719
Follow Up Leads 9334
Document Pending Leads 3735
Third party Accepted Leads 1
Auto Follow Up Leads 29083
Query Leads 32151
No Insurance Leads 26627
Non Contactable Leads 21366
Repeated Leads 12843
Expert Leads
Registered Leads
Customer Not Responding Leads
Registration Pending Leads

- Partner Portal/Polifyx Partner :
  - Partner
  - Associate
  - Advisor



Dashboard

Welcome Manish

Filter By Date User Clear Filter

Total Leads 431 ₹ 4,50,13,22,39,374
Pending Leads 253 ₹ 12,54,041
Rejected Leads 20 ₹ 0
Registered Leads 98 ₹ 4,50,13,22,39,374
Settled 38 ₹ 0
Resolved 2 ₹ 0
Work in Progress 300 ₹ 4,50,02,94,90,839

Agent Details

Total Onboard Agents 519
• Total Onboarded Agents Till Now.
Current Month Total Onboard Agents 12
• Total Onboarded Agents in This Month.
Total Signup in Current Month 11
• Total Agents Signup in This Month.

- Polifyx :
  - User/Customer

Security Parameters :-

- Always use JWT token for any user login flow. (Admin, Executive, Expert, Partner Agents, Customer, etc.)
- Always stores user's email and mobile number in encrypted form in DB (Using crypto package for encryption of data)
- OTP based login system form customer.
- Role wise access of panel tabs (Access control).

Absolute Todo :-

- Never hard code any value in code.
- Cross verify your code changes before push to git.
- Use of logger in every try catch block + manage history in database.

➔ Key Features of Insurance samadhan's `LoggerService` :

Log Levels:

- There are four log methods: `info`, `debug`, `warn`, and `error`. Each method formats the log data, adds metadata, and sends it either to the console or over a socket.

Handling Errors:

- When the `message` is an `Error` object, it extracts and logs the error's `message` and `stack`.

Metadata:

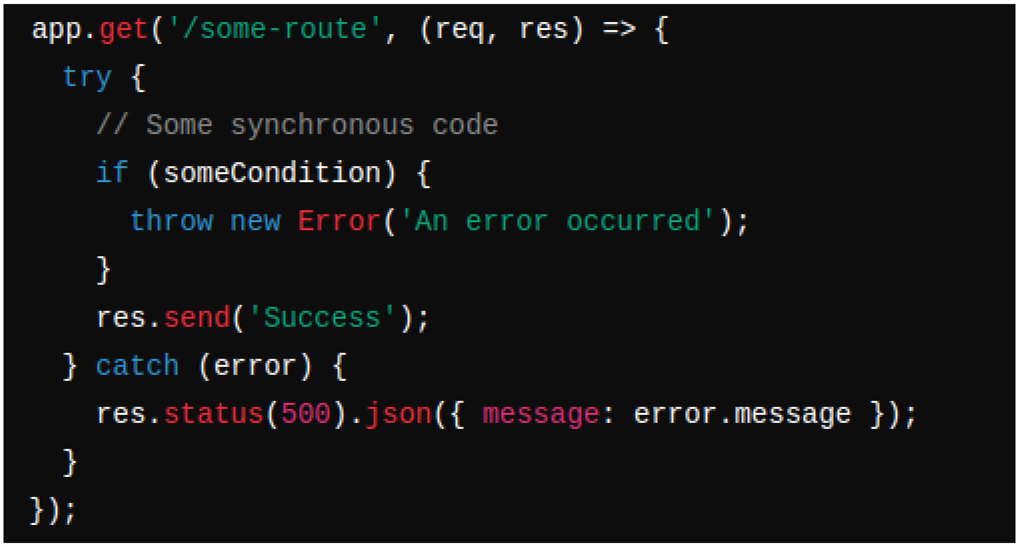
- The `metaData` object can be passed into each method. If the `message` has an `isOperational` flag, this flag is appended to the metadata.

Dynamic Log Output:

- If `DEBUG_MODE` is set to `"true"`, the logs are sent to the console using `console.log()`. Otherwise, logs are sent over a socket (presumably to some logging service or aggregator).

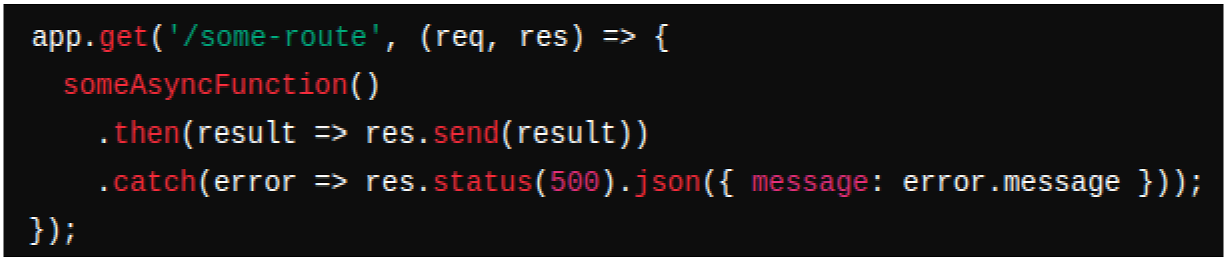
Error Handling :-

- Always use a middleware function for APIs.
- Use try-catch block for synchronous code.



```
app.get('/some-route', (req, res) => {
  try {
    // Some synchronous code
    if (someCondition) {
      throw new Error('An error occurred');
    }
    res.send('Success');
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});
```

- Handle promises & async/await.



```
app.get('/some-route', (req, res) => {
  someAsyncFunction()
    .then(result => res.send(result))
    .catch(error => res.status(500).json({ message: error.message }));
});
```

- Use validation for req.body data.

- ESLint is indeed a popular tool that helps developers maintain high code quality by identifying issues such as undefined variables, duplicate code, unused variables, unnecessary `console.log()` statements, and many more. Here's a detailed explanation of how ESLint works and how it can be integrated into the development workflow, particularly focusing on preventing code commits with linting errors

Use Case for ESLint in Node.js Projects:

1. **Identifying Code Issues**: ESLint scans your code for potential errors and stylistic issues such as:

    1.1. **Variables not defined**: If you try to use a variable that hasn't been declared, ESLint will catch that.

    1.2. **Duplicate code**: ESLint can detect duplicate blocks of code or unnecessary redundancy.

    1.3. **Unused variables**: It will warn you if you declare a variable or import a module and never use it.

    1.4. **Improper formatting**: Things like missing semicolons, extra whitespace, or inconsistent quotes (single vs double) will be flagged.

    1.5. **Console logs**: If your configuration disallows `console.log()` in production code, ESLint will remind you to remove them.

2. **Automating Code Checks with Pre-Commit Hooks**: ESLint can be combined with **Git pre-commit hooks** to enforce code quality before code is committed. This way, developers can't push code until all ESLint errors or warnings are fixed. Here's how to set it up

    2.1. Install ESLint as a Development Dependency:

```
npm install eslint --save-dev
```

3. **Add Pre-Commit Hook Using Husky and lint-staged:**

To prevent committing code with ESLint errors, you can use tools like **Husky** and **lint-staged**. These tools help run ESLint checks automatically when you try to commit code.

Husky: Manages Git hooks, like pre-commit hooks.

lint-staged: Runs linters (e.g., ESLint) only on the files that are staged for commit.

```
npm install husky lint-staged --save-dev
```

4. **Configure Husky and lint-staged:**

    In your `package.json`, add the following scripts:

```
{
"Husky":
      { "hooks":
      { "pre-commit": "lint-staged" }
 }, "lint-staged":
     { "*.js": [ "eslint --fix", "git add" ]
      }
 }
```

5. **Running ESLint:**

```
npx eslint .
```

    To automatically fix issues (like formatting errors), use:

```
npx eslint --fix .
```

Incident Book :-

- There is a waive-off functionality where, if a customer is unable to pay the registration fees or has a genuine reason, an expert or executive can waive off their case. A confirmation email is then sent to the admin (Shilpa Ma'am), and when she accepts the case via a link in the email, the case is approved for further processing.

  Once, while working on this module for testing purposes, I commented out the function code that sends the email to the admin. After completing my task, I pushed the code to git, and unfortunately, it went into production. The next day, the admin stopped receiving any waive-off approval emails, so she approached the tech team to ask why this was happening and we had to resolve the issue immediately.

  The key takeaway is that before pushing your code to git, always ensure that it works correctly and cross-verify it line by line.