Insurance Comparison, Recommendation & Claim Assistant

Project Statement:

This project lets users compare policies, get tailored recommendations, and manage claims with guided workflows and fraud checks.

Key Features:

- Policy comparison & premium calculators
- Personalized policy recommendations
- Guided claim filing with uploads
- Real-time claim status tracking
- Fraud detection (rules-based)

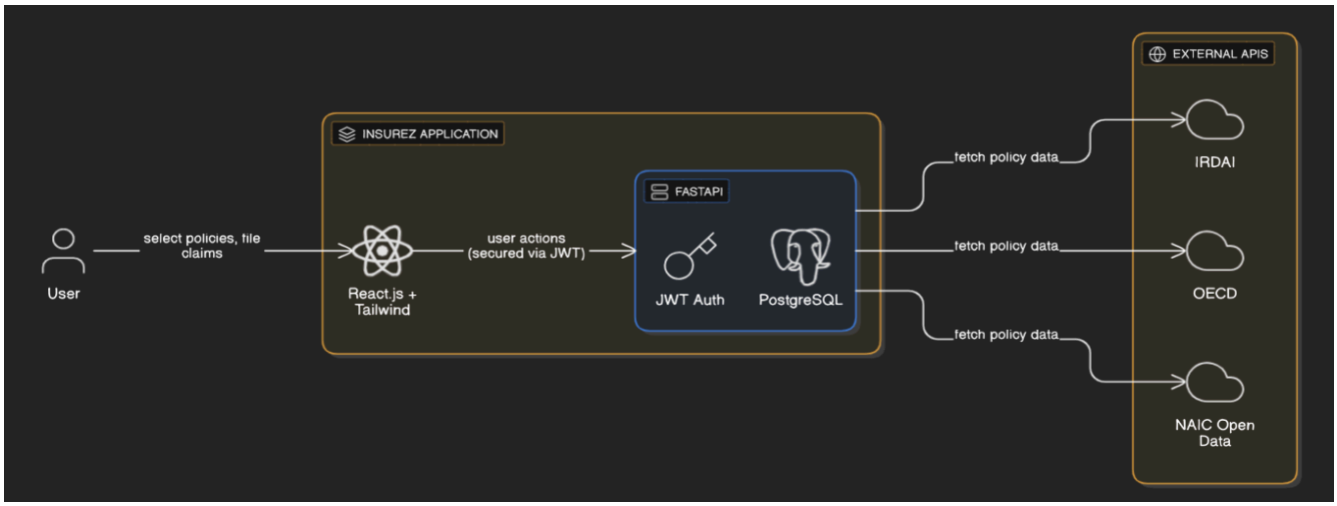
Tech Stack:

- Frontend: React.js + Tailwind CSS
- Backend: FastAPI
- Database: PostgreSQL
- Authentication: JWT (access + refresh)
- **Extras:** S3 for claim docs, Celery for notifications; Integrations (IRDAI/OECD/NAIC data where applicable)

Modules:

- Module A: Auth, Profile & Preferences
- Module B: Policy Catalog, Compare & Quote
- Module C: Recommendation Engine
- Module D: Claims (filing, documents, tracking)
- Module E: Fraud Rules & Admin Analytics

Architecture Diagram:



8-Week Milestone Plan

Milestone 1: Weeks 1–2 – Foundations & Catalog

Week 1: Auth, Users/Providers/Policies schema, seed sample policies
Week 2: Policy browse & compare UI, calculators

Expected Output:

Users can browse/compare policies with pricing fields.

Milestone 2: Weeks 3–4 – Recommendations

Week 3: Collect user preferences/risk profile
Week 4: Score policies per user (Recommendations table)

Expected Output:

Personalized shortlists with rationale.

Milestone 3: Weeks 5–6 – Claims

Week 5: Claim filing wizard with uploads (S3)
Week 6: Claim status tracking; notifications (Celery emails)

Expected Output:

End-to-end claims workflow with document storage.

Milestone 4: Weeks 7–8 – Fraud & Analytics

Week 7: Fraud rules engine (duplicate docs, suspicious timing, amounts) → FraudFlags
Week 8: Admin dashboards, export, QA & deployment

Expected Output:

Flags on risky claims, admin oversight, production build.

Expected Project Outcome:

By Week 8, users can compare & buy policies, file and track claims, and admins can monitor risk via fraud flags and analytics.

Database Schema:

- **Users:** id (INT, PK), name (VARCHAR), email (VARCHAR, UNIQUE), password (VARCHAR), dob (DATE), risk_profile (JSONB), created_at (TIMESTAMP)
- **Providers:** id (INT, PK), name (VARCHAR), country (VARCHAR), created_at (TIMESTAMP)
- **Policies:** id (INT, PK), provider_id (FK to Providers.id), policy_type (ENUM: 'auto', 'health', 'life', 'home', 'travel'), title (VARCHAR), coverage (JSONB), premium (NUMERIC), term_months (INT), deductible (NUMERIC), tnc_url (VARCHAR), created_at (TIMESTAMP)
- **UserPolicies:** id (INT, PK), user_id (FK to Users.id), policy_id (FK to Policies.id), policy_number (VARCHAR), start_date (DATE), end_date (DATE), premium (NUMERIC), status (ENUM: 'active', 'expired', 'cancelled'), auto_renew (BOOLEAN)
- **Claims:** id (INT, PK), user_policy_id (FK to UserPolicies.id), claim_number (VARCHAR), claim_type (VARCHAR), incident_date (DATE), amount_claimed (NUMERIC), status (ENUM: 'draft', 'submitted', 'under_review', 'approved', 'rejected', 'paid'), created_at (TIMESTAMP)
- **ClaimDocuments:** id (INT, PK), claim_id (FK to Claims.id), file_url (VARCHAR), doc_type (VARCHAR), uploaded_at (TIMESTAMP)
- **Recommendations:** id (INT, PK), user_id (FK to Users.id), policy_id (FK to Policies.id), score (NUMERIC), reason (TEXT), created_at (TIMESTAMP)
- **FraudFlags:** id (INT, PK), claim_id (FK to Claims.id), rule_code (VARCHAR), severity (ENUM: 'low', 'medium', 'high'), details (TEXT), created_at (TIMESTAMP)
- **AdminLogs:** id (INT, PK), admin_id (FK to Users.id), action (TEXT), target_type (VARCHAR), target_id (INT), timestamp (TIMESTAMP)