SATeLite

1. Project Vision & Idea
SATeLite is a cloud-native platform designed to automate fiscal compliance, provide financial diagnostics, and detect fraud for SMEs and accountants, with a focus on adapting to different countries' regulations (starting with Mexico/SAT). The platform leverages AI/ML for document processing, classification, and analytics.

Key Goals:

Automate SAT (Mexican tax authority) compliance for SMEs and accountants.
Use AI for financial health diagnostics and fraud detection.
Provide a seamless, modern, and secure user experience.
Be scalable, cloud-based, and easy to adapt to other fiscal environments.

2. System Architecture
Frontend: React.js (Vite, TailwindCSS), deployed via AWS Amplify.
Backend: Python microservices (AWS Lambda), API Gateway, Step Functions, Textract, Comprehend, SageMaker.
Data: DynamoDB, PostgreSQL (RDS), S3 for document storage.
Authentication: Amazon Cognito.
CI/CD & Hosting: AWS Amplify.
Security: IAM, KMS, Macie, WAF.

See:
System Architecture Diagram
Flow Diagram

3. Development Process
3.1. Planning & Design
User Stories:
As an SME, I want to upload my fiscal documents and get instant AI-driven analysis.
As an accountant, I want to manage multiple clients' documents and compliance.
As an admin, I want to monitor system health and user activity.
Wireframes & UI Design:
Designed in Figma, focusing on clarity, accessibility, and modern UX.
Component Structure:
Modular, with pages for onboarding, dashboard, document upload, AI processing, classification, and declaration.
3.2. Frontend Implementation
Stack:
React.js (Vite), TailwindCSS, shadcn/ui for UI components, react-dropzone for file uploads, react-router-dom for routing.
Key Pages/Flows:
Auth: /login, /signup — secure login, registration, forgot password.
Onboarding: Guided onboarding for both accountants and SMEs.
Dashboard: Main workspace, with tabs for document upload, analysis, SAT reports, and settings.

Document Upload: Drag-and-drop interface, supports XML, PDF, XLSX.
AI Processing: Triggers backend AI/ML workflows, displays results (deductions, errors, alerts).
Classification: NLP-based categorization of transactions.
Declaration: Generates and allows download of tax declarations, with notification options.
Component Example:
FiscalDashboard in /src/pages/pymes/dashboard/Home.jsx handles file uploads, triggers AI analysis, and displays results.
TaxDeclarationPage in /src/pages/declaration/Home.jsx simulates the declaration generation and download flow.
Routing:
Centralized in /src/routes/AppRoutes.jsx, with clear separation for accountant and SME flows.

3.3. Backend Implementation

Stack:
Python (FastAPI or Flask for local dev), AWS Lambda for serverless deployment.
AWS Step Functions to orchestrate document processing (Textract OCR, Comprehend NLP, SageMaker ML).
API Gateway exposes RESTful endpoints.
S3 for document storage, DynamoDB/PostgreSQL for structured data.
Endpoints:
/upload: Accepts documents, triggers processing pipeline.
/analyze: Returns AI/ML results.
/classify: Returns NLP-based transaction categories.
/declaration: Generates and returns tax declaration files.
Security:
All endpoints protected via Cognito JWT tokens.
Data encrypted at rest and in transit.

3.4. Integrations & AI/ML

Textract: Extracts data from invoices/receipts.
Comprehend: Classifies and tags transactions.
SageMaker: Custom models for fraud detection, financial health scoring.

3.5. Testing

Frontend:
Component and integration tests (Jest, React Testing Library).
Backend:
Unit and integration tests for API and ML pipeline.
End-to-End:
Manual and automated flows to verify user journeys.

3.6. Deployment

Frontend:
Deployed to AWS Amplify, with CI/CD from GitHub.
Backend:
Lambda functions deployed via AWS SAM or Serverless Framework.
Database:
DynamoDB and RDS provisioned via CloudFormation.

4. Codebase Structure

CopyInsert

```
talent-hackathon-2025/
│
├── backend/
│   ├── api/              # API endpoints (Python, Flask/FastAPI)
│   ├── core/             # Core business logic
│   ├── requirements.txt  # Python dependencies
│   ├── manage.py         # Entrypoint for dev
│   └── tests/            # Backend tests
│
├── frontend/
│   ├── public/           # Static assets (logo, etc.)
│   ├── src/
│   │   ├── assets/       # Images, icons
│   │   ├── components/   # Shared React components (UI, Button, etc.)
│   │   ├── pages/        # Page-level React components
│   │   │   ├── accountant/
│   │   │   ├── auth/
│   │   │   ├── cassification/
│   │   │   ├── declaration/
│   │   │   ├── login/
│   │   │   ├── process/
│   │   │   ├── pymes/
│   │   │   └── singup/
│   │   ├── routes/       # AppRoutes.jsx (routing config)
│   │   ├── utils/        # Helper functions
│   │   └── main.jsx      # App entrypoint
│   ├── package.json      # JS dependencies
│   └── tailwind.config.js  # TailwindCSS config
│
├── README.md             # Project documentation
└── ...                   # CI/CD, infra, etc.
```

5. Key Features & User Flows

5.1. Authentication & Onboarding

Secure login/registration using Cognito.

Role-based onboarding for accountants and SMEs.

5.2. Document Upload & Processing

Drag-and-drop upload for XML, PDF, XLSX.

Files sent to backend, stored in S3.

AI pipeline extracts, analyzes, and classifies data.

5.3. Analysis & Reporting

Real-time display of deductions, errors, and SAT alerts.

Quick summaries and detailed reports.

Classification of transactions using NLP.

5.4. Tax Declaration

Generate tax declarations from processed data.

Download as PDF.

Option to notify via SMS.

6. Prototype & Demo

Frontend:
Fully navigable UI with simulated AI analysis and declaration flows.
Responsive, modern design using Tailwind and shadcn/ui.
Backend:
Simulated endpoints for AI/ML analysis.
Ready for integration with AWS services.
Testing:
Manual and automated tests for critical flows.

7. Next Steps

Integrate real AWS AI/ML services.
Expand multi-country compliance support.
Add dashboard analytics and admin panel.
Enhance security and monitoring.

8. How to Run Locally

Frontend:
bash
CopyInsert
cd frontend
npm install
npm run dev

Backend:
bash
CopyInsert
cd backend
pip install -r requirements.txt
python manage.py runserver

9. Contributors

AWSome Team