Second Certainty

A Tax Liability Management Tool

Cesaire Tobias

2025-05-08

Table of contents

1 Project Description

The Second Certainty Tax Tool addresses one of life's two certainties—taxes—by providing individuals and small businesses with an intuitive platform for managing tax liabilities throughout the fiscal year. Traditional tax management often occurs reactively at year-end, leading to unexpected liabilities, missed deductions, and financial stress. Our application transforms this approach by implementing a proactive, year-round tax management system.

The tool continuously calculates estimated tax liabilities based on income streams, identifies potential deductions, forecasts quarterly payments, and provides optimization strategies—all in real-time. In today's complex financial landscape, where gig economy participation, investment income, and small business ownership have complicated personal taxation, our solution brings clarity and control to users' tax situations.

By empowering users with ongoing visibility into their tax position, the Second Certainty Tax Tool reduces financial anxiety, prevents costly surprises, and helps maximize legitimate tax advantages throughout the year.

2 Technical Requirements

Our application leverages modern technology frameworks to deliver a secure, responsive, and user-friendly experience:

- **Frontend**: React 18.2 (with React Router 6.10 for navigation, Axios 1.3.5 for HTTP requests, and Tailwind CSS 3.3.1 for styling)
- **Backend**: FastAPI 0.110.1 (with Pydantic 2.6.0 for data validation, SQLAlchemy 2.0.40 for ORM, and Python-Jose 3.3.0 for JWT authentication)
- **Database**: PostgreSQL 15 (selected for its robust handling of financial data, transactional integrity, and comprehensive indexing capabilities)
- **Data Processing**: Pandas 2.0.0 and NumPy 1.24.2 (for efficient tax calculations and financial modeling)

- **Visualization**: D3.js 7.8.4 and Recharts 2.5.0 (for interactive data visualization of tax liabilities and projections)
- **Web Scraping**: BeautifulSoup4 4.12.3 and HTTPX 0.27.0 (for automating tax data collection from SARS)
- **Authentication**: Password hashing with Passlib 1.7.4 and Bcrypt 4.1.2, JWT tokens with Python-Jose 3.3.0
- **Database Migrations**: Alembic 1.13.1 (for managing database schema changes)
- **Testing**: Pytest 7.4.4 with pytest-asyncio 0.23.5 and pytest-cov 4.1.0 (90% code coverage achieved)
- **CI/CD**: GitHub Actions for continuous integration and AWS CodeDeploy for continuous deployment

The architecture follows a microservices approach where the tax calculation engine, user data management, and document generation are separated for scalability and maintainability. We implemented secure API gateways with rate limiting to ensure system integrity against potential abuse.

2.1 Recent Infrastructure Improvements

We have recently enhanced the project's infrastructure to ensure maximum stability and maintainability:

- **Comprehensive Test Suite**: Implemented extensive unit and integration tests for all core components, focusing on the tax calculation engine, data scraper functionality, and API endpoints
- **Database Migration Framework**: Set up Alembic for systematic database schema management and version control
- **Environment Configuration**: Enhanced the configuration system with a well-documented environment variable setup
- **Dependency Management**: Updated the requirements.txt file to explicitly specify versions for all dependencies to prevent compatibility issues
- **Source Control Optimization**: Improved .gitignore to exclude development artifacts and ensure clean repository management

3 Application Features

The Second Certainty Tax Tool offers comprehensive functionality to address all aspects of proactive tax management:

- **Intelligent User Onboarding**: Guided setup process that collects relevant financial information and tax situation details, requiring minimal tax expertise from users
- **Income Stream Tracking**: Integration with multiple income sources including traditional employment, freelance work, rental income, dividends, and capital gains
- **Real-time Tax Liability Dashboard**: Dynamic visualization of current tax position, projected year-end liability, and comparison with previous years

- **Deduction Discovery Engine**: AI-powered analysis that identifies potential tax deductions based on spending patterns and user-specific situations
- **Quarterly Payment Management**: Automated calculation of estimated quarterly tax payments with reminder systems and payment tracking
- **Document Management System**: Secure storage for tax-relevant documentation, receipts, and financial statements with OCR capabilities
- **Tax Calendar**: Personalized calendar with important tax dates, filing deadlines, and custom reminders
- **Tax Scenario Modeling**: What-if analysis tool for exploring potential financial decisions and their tax implications
- **Multi-year Tax Strategy Planning**: Long-term tax optimization recommendations based on user's financial goals and life events
- **Secure Data Export**: One-click generation of organized financial records for tax preparation or professional review
- **Tax Professional Collaboration Portal**: Secure channel for sharing relevant information with accountants or tax advisors
- **Automatic Tax Rate Updates**: SARS website scraping to ensure tax calculations use the most current rates and thresholds

4 Technical Architecture

Our application follows a layered architecture that ensures separation of concerns and maintainability:

4.1 API Layer

The API layer is implemented using FastAPI and provides the interface for client applications. It includes:

- **Authentication Routes**: User registration, login, and token management
- **Tax Calculation Routes**: Endpoints for calculating tax liabilities and provisional tax
- **Data Management Routes**: Endpoints for managing income sources and expenses
- **Tax Data Routes**: Endpoints for retrieving tax brackets, rebates, and thresholds

4.2 Business Logic Layer

The core business logic includes:

- **Tax Calculator**: The computation engine that applies South African tax rules
- **Data Scraper**: Automated collection of tax data from the SARS website
- **Authentication System**: JWT-based authentication with secure password hashing

4.3 Data Layer

The database architecture uses SQLAlchemy ORM to interact with a PostgreSQL database:

- **User Profiles**: Personal and authentication information
- **Income Sources**: Various income streams for users
- **Expenses**: Tax-deductible expenses
- **Tax Data**: Tax brackets, rebates, thresholds, and medical credits by tax year
- **Tax Calculations**: History of tax calculations for audit and reference

4.4 Testing Infrastructure

The testing infrastructure ensures reliability and correctness:

- **Unit Tests**: Tests for individual components (90+ tests)
- **Integration Tests**: End-to-end tests for complete workflows
- **Mock Services**: Simulated external services for consistent testing
- **Continuous Integration**: Automated test execution on every commit

5 Deployment

The Second Certainty Tax Tool is deployed and accessible at: https://secondcertaintytax.financial-tools.com

Our deployment strategy utilizes AWS infrastructure to ensure security, reliability, and scalability:

- Frontend assets are served through Amazon CloudFront CDN for optimal global performance
- Backend services are containerized using Docker and orchestrated with Amazon ECS
- Database operations run on Amazon RDS PostgreSQL instances with automated backups and point-in-time recovery
- We maintain separate development, staging, and production environments with comprehensive CI/CD pipelines

The deployment process presented several challenges, particularly around securing sensitive financial data. We implemented end-to-end encryption, rigorous access controls, and comprehensive audit logging to address these concerns. Additionally, we established a blue-green deployment approach to ensure zero-downtime updates when implementing new features or security patches.

Performance optimization was another significant challenge, as tax calculations involving multiple income sources and deductions can be computationally intensive. We addressed this through strategic caching, database query optimization, and background processing of non-time-sensitive calculations.

6 Development Workflow

We follow a structured development workflow that ensures code quality and reliable releases:

6.1 Version Control

- **Git Workflow**: Trunk-based development with feature branches and pull requests
- **Commit Guidelines**: Semantic commit messages following the Conventional Commits specification
- **Code Reviews**: Required peer review for all changes before merging

6.2 Testing Approach

- **Test-Driven Development**: Critical components are developed using TDD methodology
- **Automated Tests**: Every feature is accompanied by appropriate tests
- **Coverage Requirements**: Minimum 85% code coverage maintained across the codebase

6.3 Continuous Integration/Continuous Deployment

- **Build Pipeline**: Automated builds triggered on every push to the repository
- **Testing Stage**: Automated test execution across multiple Python versions
- **Deployment Stage**: Automated deployment to appropriate environment based on branch

6.4 Documentation Practices

- **Code Documentation**: Comprehensive docstrings for all functions and classes
- **API Documentation**: Auto-generated OpenAPI specification with detailed endpoint descriptions
- **Change Documentation**: Detailed changelogs maintained for all releases

7 Code Repository

Our application's source code is maintained in a GitHub repository with comprehensive documentation and structured contribution guidelines:

https://github.com/ces0491/second-certainty

The repository includes detailed setup instructions for local development environments, API documentation, and test suites. We follow a trunk-based development workflow with feature branches and pull request reviews to maintain code quality.

7.1 Project Structure

```
second-certainty/

  app/
     api/         # API routes and dependencies
     core/        # Core business logic
     db/          # Database models and migrations
     models/      # SQLAlchemy models
```

```
    schemas/    # Pydantic schemas
    utils/      # Utility functions

docs/           # Documentation
scripts/        # Utility scripts
tests/          # Comprehensive test suite
.env.example    # Environment variable template
alembic.ini     # Database migration configuration
requirements.txt # Project dependencies
README.md       # Project documentation
```

7.2 Getting Started

To set up the development environment:

1. Clone the repository:

```
git clone https://github.com/ces0491/second-certainty.git
cd second-certainty
```

2. Create a virtual environment:

```
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

3. Install dependencies:

```
pip install -r requirements.txt
```

4. Set up environment variables:

```
cp .env.example .env
# Edit .env with your configuration
```

5. Run database migrations:

```
alembic upgrade head
```

6. Seed initial data:

```
python scripts/seed_data.py
```

7. Start the development server:

```
uvicorn app.main:app --reload
```