PROJECT SCOPE DOCUMENT

Customer Income Capture Microservice

CLIENT	**Enterprise banking client**
TYPE	Professional employment, backend engineering team
MY FOCUS	Designed and shipped the microservice end to end, plus the Apigee security and CI/CD around it
STATUS	Live, in production
STACK	Java, Spring Boot, MongoDB, Apigee, Jenkins CI/CD, Hadoop/Hive, SQL, Kubernetes, PCF, AWS ECS, Lambda
SOURCE	Private client repository, available for walkthrough on request

1. Overview

A new microservice extending the client's core banking application to capture customer declared income directly at the point of credit card activation. Previously, income data came from credit application forms, often 3 months or more out of date, or from third party income surveys that could take up to 6 months to return results. This service adds a direct, first party data entry point at exactly the moment the customer is engaged, closing that gap.

The service was designed and shipped end to end as part of a professional engagement with an enterprise banking client, along with the Apigee gateway security and CI/CD pipeline that support it.

2. Problem Statement

Before this service, the bank had two ways to learn a customer's income: the original credit application form, or a third party income survey run after the fact. Application data went stale quickly. By the time a customer activated their card, the income figure on file could be 3 months old or more, and did not reflect any change in employment or earnings since the application was submitted.

Third party income surveys were more current but slow and costly. A survey could take up to 6 months to return results, and each one carried a vendor fee. Relying on this data left offer targeting and credit review working from outdated or delayed information, at a real cost to the business.

3. In Scope

Income capture flow

- Direct income capture step added to the credit card activation flow
- New Spring Boot microservice handling the capture request and validation

- Document shaped storage in MongoDB to support variable income fields by employment type

Apigee security

- Authentication, rate limiting, and routing configured at the Apigee gateway layer
- Gateway level security policy, kept separate from application business logic

CI/CD

- Jenkins pipeline for automated build and deployment
- Migration of the deployment target from Pivotal Cloud Foundry to Kubernetes and AWS ECS and Lambda

Pipeline extension

- Extension of the existing Hadoop/Hive batch pipeline with new SQL queries to include the captured income data
- No new pipeline or tooling introduced

Tableau integration

- Captured income data flowing into the existing enterprise datalake
- Existing business analyst Tableau dashboards updated to reflect the new data, with no analyst retraining required

4. Out of Scope

- Credit decisioning engine: this service captures income, it does not decide credit outcomes
- Customer identity and authentication systems, beyond the Apigee gateway policies applied to this service's endpoints
- Card issuance and card manufacturing systems

5. Key Architecture Decisions

First party income capture at the activation moment

The existing process relied on stale credit application data, 3 months or more old, or expensive third party income surveys with a 6 month turnaround. The decision was to add a direct capture step at card activation, when the customer is already engaged and motivated to complete the process, rather than sending a separate survey later.

MongoDB as the transactional store

Income data at card activation is document shaped, not relational. Each activation event has a variable set of income fields depending on the customer's employment type. MongoDB was chosen as the transactional store to accommodate that variability without requiring schema migrations for each new income type.

Extending the existing Hadoop/Hive pipeline rather than building a new one

The existing Hadoop/Hive batch pipeline was extended with new SQL queries, instead of building a parallel data pipeline for the new income data. This kept the data flowing to the same enterprise datalake and Tableau dashboards the business already used, with no new tooling or analyst retraining required.

Apigee for gateway security rather than in service auth

Authentication, rate limiting, and routing were handled at the Apigee gateway layer rather than inside the Spring Boot service itself. This kept the microservice focused on business logic and meant security policy changes could be applied at the gateway without a service redeployment.

6. Outcomes

> - 61% customer adoption of the new income capture flow at card activation
> - 20% reduction in external data vendor costs, from fewer paid income surveys needed
> - 12% lift in targeted offer conversion in the initial release, from fresher first party income data feeding targeting models
> - Manual review time cut from 3 to 6 months down to real time
> - Extended existing Hadoop/Hive pipelines so captured income reached business analyst Tableau dashboards, not just the offer targeting system

7. Integration Footprint

Service	Role	Auth method
Apigee	API gateway, authentication, rate limiting, routing	API key / OAuth policies
Jenkins	CI/CD, automated build and deployment	Internal pipeline
MongoDB	Transactional store for captured income data	Internal service credentials
Hadoop/Hive	Batch pipeline feeding the enterprise datalake	Internal service credentials
Kubernetes	Container orchestration for microservice deployment	Internal

Service	Role	Auth method
AWS ECS/Lambda	Scalable workload hosting post PCF migration	IAM roles
Tableau	Business analyst dashboards consuming datalake data	Internal service credentials

8. My Role

Within the engagement team, my specific responsibilities on this service:

- Designed and built the Spring Boot microservice end to end, from the income capture endpoint to MongoDB persistence
- Configured Apigee gateway policies for authentication, rate limiting, and routing
- Built and maintained the Jenkins CI/CD pipeline, including the migration from Pivotal Cloud Foundry to Kubernetes and AWS
- Extended the Hadoop/Hive batch pipeline with new SQL queries to bring captured income into the enterprise datalake
- Led code reviews for the service
- Ran stakeholder calls with client side leads to confirm requirements and review progress

---

Source code is in a private client repository. Akash Ramesh is available to walk through implementation in detail during any interview or technical review.