**Requirements Engineering Document**

**Business Requirements:**

Current modern financial transaction fulfillment, and banking requires fulfillment in bulk at the end of each business day. This is a constraint that limits the flow of transactions, and results in bottlenecks and untimely business transaction fulfillments due to the monolithic and highly coupled nature of their architecture. In order to solve this issue, many banking businesses set out to build an event driven, cloud-based system for processing financial transactions. This paradigm processes significant changes in states as events in order to facilitate efficiency and modularity.

We are tasked with creating a small-scale EDA prototype for the Fall semester utilizing Amazon Web Services with the requirements: 1) Create new user accounts 2) Initiate and fulfill transactions. In addition to this, the storing of financial and user information in secure databases must be considered as well as the connection between event producers and consumers in order to improve the overall processing efficiency for financial transactions.

**Functional and Non-Functional Requirements:**

*Functional-*

Main priority according to Mehul (CapitalOne Mentor)

1. Create Account
2. Create and fulfill Transaction

Authorization

Merchant Fulfilled Transaction

Pending Transaction

Approve/Decline Transaction

Update Balance

*Non-functional-*

We want our consumers to also be producers. We want an event filter in the middle of the architecture that processes each event.

Security- IAM Permissions

Reliability- Event Driven, AWS

Performance- Highly decoupled, multiple lambdas

Maintainability- SAM CLI, or AWS UI

Scalability- Very scalable, create Lambdas as appropriate

Usability- CLI as user interface for now, build UI with PHP for next semester possibly, react for front end possibly too

**Technical Requirements:**

Amazon Web Services provides us the functionality to build an event driven system. Our software, using AWS DynamoDB, complies with the security necessary for sensitive information of credit card and user information through AWS cloud-based databases and the security measures implemented there. Furthermore, CloudWatch logs are provided to ensure that no events go astray, we are able to manage and track everything. Architecturally, AWS allows us to scale our operation, adding whatever functions are needed, with for example, AWS Lambda, a compute service that allows us to integrate and add more services to our system one piece at a time depending on the nature of what needs to be installed.

**Prioritize each Requirements:**

Objectively speaking, prioritized based on the design process, the most important requirements are the business requirements. We fleshed out our architecture based off of our CapitalOne mentors needs and requirements that were in context to improving the financial systems implemented today. Then Functional, Non-functional as these are guidelines, we set for amongst ourselves with input from our mentors to be able to follow an event driven architecture. Finally technical requirements were last because once we had everything agreed upon and structured, all we had to do was begin coding and implementing our afore-discussed design criteria which wasn’t necessarily too difficult.