**Text

Description automatically generated with low confidence**

**CS301: IT Solution Architecture**

AY2021/2022 Term 2

**Project Proposal: Project B**

**Spend Transaction Processing**

**Group 4**

**Team Members:**

|  |  |  |
| --- | --- | --- |
| Charmaine Tan | [rxtan.2018@scis.smu.edu.sg](mailto:rxtan.2018@sis.smu.edu.sg) | 01374766 |
| Chen Sujuan | [sujuan.chen.2018@scis.smu.edu.sg](mailto:sujuan.chen.2018@scis.smu.edu.sg) | 01370038 |
| Er Soon Hao | [soonhao.er.2019@scis.smu.edu.sg](mailto:soonhao.er.2019@scis.smu.edu.sg) | 01364060 |
| Nicolas Wijaya | [nicolasw.2018@scis.smu.edu.sg](mailto:nicolasw.2018@scis.smu.edu.sg) | 01332027 |
| Renata Dharma | [renatad.2018@scis.smu.edu.sg](mailto:renatad.2018@scis.smu.edu.sg) | 01372220 |
| Sim Sheng Qin | [sqsim.2018@scis.smu.edu.sg](mailto:sqsim.2018@scis.smu.edu.sg) | 01340058 |

## **GitHub Team name, team members (name and matriculation)**

* + Github Name: [project-2021-22t2-g1-project-2021-22t2-g1-team4](https://github.com/cs301-itsa/project-2021-22t2-g1-project-2021-22t2-g1-team4)
  + Team Members (as per cover page)

## **Background and Business Needs**

Provide the background context of the project. Include a summary of the business needs and drivers for this application. A short summary (at most half a page) is sufficient, design should be the focus of your proposal.

Ascenda requires a platform that helps to process loyalty points for their customers in near real-time. The business needs include

* An interface to view transactions per customer per card
* An API endpoint to retrieve all transaction details so that transaction details can be displayed for the customers
* An API endpoint to receive files daily
* A file transfer dropbox to receive files daily
* A backend to store the transaction details

## **Stakeholders**

|  |  |  |
| --- | --- | --- |
| **Stakeholder** | **Stakeholder Description** | **Permissions  (if not applicable, write N.A.)** |
| Ascenda Loyalty | Provides banks with a turn-key solution for a loyalty points processing platform. | Read/Write permission to API Gateway  Write access to RDS |
| Banks | Submit customers transaction details daily | Write access to RDS |
| Customers (Bank users) | Spend and receive benefits for each eligible transaction | Read permission to CloudFront |

## **Use Cases for Architectural Significant Requirements (ASRs)**

|  |  |
| --- | --- |
| **Use Case Name** | Spend Transaction & Exclusion Processing |
| **Use Case ID** | 1 |
| **Brief Description** | This use case specifies what the client and customers can perform on our system to manage the spend transactions. |
| **Actors** | |
| **Primary Actor** | Client (Ascenda) |
| **Secondary Actor(s)** | Customer, Banks |
| **Flow of Events** | |
| **Main Flow (MF)** | |
| **1.1** | UPLOAD TRANSACTION FILES via file processing  We assume that files can be dragged and dropped into a central dropbox or posted via API.  The banks can drag and drop their transaction files. The process should operate in the background. We expect a file of 1 million records that will be processed by the end of the day.  As new customers join the platform and more customers spend more over time, we expect that the file processing will receive files periodically over a day. |
| **1.2** | CONVERT CURRENCY  If the transaction’s currency is not in SGD, convert to SGD. |
| **1.3** | APPLY EXCLUSIONS UNDER A CARD PROGRAM  All transactions will be first checked against exclusions by using each transaction’s category codes (MCC). Certain spend types will be excluded from the card program. Transactions are marked with a note to indicate the reason for exclusion. |
| **1.4** | CONVERT TRANSACTION DETAILS TO BENEFITS  All transactions will be converted to a benefit if eligible using the applicable categories. If no benefit was converted, then reference the base earn rate and convert if eligible. |
| **1.5** | VIEW ALL TRANSACTION DETAILS  Customers will view a list of all transactions, remaining balances and benefits earned for each card they own. |
| **Alternate Flow (AF)** | |
| **1.6** | UPLOAD TRANSACTION FILES via API processing  We expect 1 million requests per day, or 700 requests per minute. The files will be sent round the clock (24/7). The flow will follow from 1.2. |
| **Pre-conditions** | Customers must have a valid card |
| **Post-conditions** | Card is valid and Ascenda Loyalty writes the transaction and converts benefits to the database. |

|  |  |
| --- | --- |
| **Use Case Name** | Campaigns Management |
| **Use Case ID** | 2 |
| **Brief Description** | This use case will allow the Client (Ascenda) to manage campaign data, including retrieval and display. Send notifications to the customers and retrieve bonus points from the activity page. |
| **Actors** | |
| **Primary Actor** | Client (Ascenda) |
| **Secondary Actor(s)** | Banks, Customer |
| **Flow of Events** | |
| **Main Flow (MF)** | |
| **2.1** | CREATE PROMOTIONAL CAMPAIGNS  The bank merchant managers will create new promotional campaigns by inputting the details of the campaign. The site should validate the inputs to ensure that the inputs are valid. |
| **2.2** | SELECT customers  The system retrieves all eligible customers from the Customer Relationship Management System and appends them into this promotion campaign form. The bank merchant selects the customers and submits the form. |
| **2.3** | NOTIFY USER OF BONUS BENEFITS  The system will consolidate all eligible customers and notify them via SMS/Email of the bonus benefit. |
| **2.4** | DISPLAY INFORMATION ON ACTIVITY PAGE  Upon customer login, the bonus campaign will be displayed on the activity page which reflects the user points. |
| **2.5** | APPLY BENEFIT CAMPAIGN  The system will compile to see if each transaction under the customer's card will fulfil a campaign and issue the benefits to the card. The use case ends. |
| **Alternate Flows (AF)** | |
| **3.1.** | NOTIFICATION BROADCAST FAILS  At MF 2.3 NOTIFY USER OF BONUS BENEFITS and 2.5 APPLY BENEFIT CAMPAIGN, when the SMS or Email broadcast fails, the system will make another attempt to process this task. In the event, it is still unsuccessful, it will log the error message into the systems log and inform the Bank Merchant of the error code and a short description of the error. The use case ends. |
| **3.2** | APPLY REWARD CAMPAIGN FAILS  At MF 2.5 APPLY BENEFIT CAMPAIGN, the redemption fails. The system will detect which stage of the redemption is causing the failure of the transaction, create a system log and reattempt the transaction. The use case continues at MF 2.5. |
| **3.3** | APPLY REWARD CAMPAIGN FAILS REPEATEDLY  At AF 3.2 APPLY BENEFIT CAMPAIGN, the system detects that the transaction fails the second time. The system will decline this transaction and create a system log. The use case ends. |
| **Pre-Conditions** | The users who are the end customers and the client must be first connected online and logged in before they can use this function. Only eligible customers will be able to apply for a benefit campaign function. |
| **Post-Conditions** | All inputs, processed tasks outcome and errors should be displayed to the users. All errors should be logged into the system for audit purposes. |

## **Proposed Budgets**

### Development Budget

|  |  |  |
| --- | --- | --- |
| **Activity Name** | **Description** | **Cost** |
| Analysis and Design | Project analysis and architecture design | Man-hours: 5 |
| Creating of UI | Creating UI for customers to view transaction and reward | Man-hours: 3 |
| Creating UI for banks to upload transaction file | Man-hours: 3 |
| Creating UI for client (Ascenda) to manage campaign data | Man-hours: 3 |
| Backend logic development | Developing logic for file and transaction processing | Man-hours: 20 |
| Implement service logic for each use case | Creating endpoints for the frontend access | Man-hours: 10 |
| Production environment configuration | Services setup in AWS | Man-hours: 10 |
| Implementing application to AWS | Man-hours:10 |
| System Testing | Performance testing and security testing | Man-hours: 10 |

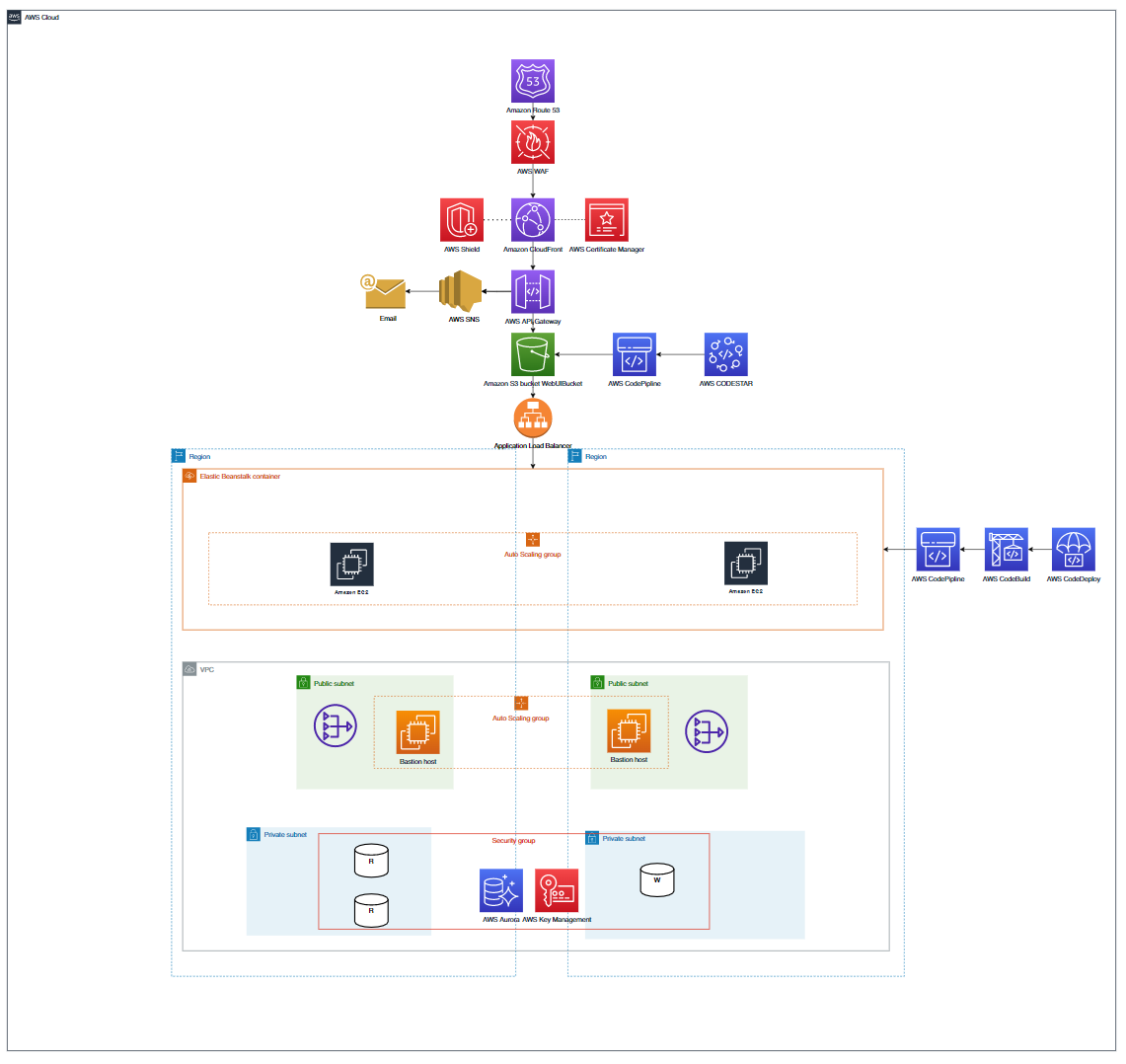
### Production Budget[[1]](#footnote-1)

|  |  |  |
| --- | --- | --- |
| **Hardware/Software/ Service** | **Description** | **Cost** |
| AWS EC2 T3.micro with VPC and ALB | EC2 for 1 AZ in 2 different regions | ~ 128.41 USD/month |
| AWS Aurora MySQL | db.t3.medium with 20 GB database storage and backup storage | ~ 62.81 USD/month |
| AWS S3 | S3 Standard 20GB/month, S3 Glacier 5TB/month | ~ 18.99 USD/month |
| Amazon CloudFront | Global CDN to deliver website and API call at high speed with low latency | 1TB Free Tier for data transfer to internet and 10 million http/https requests |
| AWS Route 53 | Cloud DNS web service, 1 hosted zone | ~ 10.50 USD/month |
| AWS Certificate Manager | Public SSL/TLS certificates provisioned through AWS Certificate Manager are free | No cost |
| AWS API Gateway | 30million API call per month, including free tier | ~ 20 USD/month |
| AWS Key Management | Securely Generate and Manage AWS Encryption Keys | ~ 7 USD/ month |
| AWS SNS | Notification service with estimated 30 million request/month | ~ 14.50 USD/month |
| AWS CodeBuild | Estimated 12 builds per month  Estimated 10mins build duration | ~ 2.40 USD/month |
| AWS CodeDeploy | Number of on-premise instances (4), Number of deployments (4 per month) | ~ 0.32 USD/ month |
| AWS CodePipeline | Number of active pipelines used per account per month (1) | No Cost |
| AWS Web Application Firewall (WAF) | Number of Web Access Control Lists (Web ACLs) utilized (1 per month) | ~ 5 USD/ month |
| Total per month | | ~ 268.92 USD/ month |

## **Quality Attributes**

|  |  |  |
| --- | --- | --- |
| **Category** | **Quality attribute** | **How it can be achieved** |
| Performance | 100 concurrent logins with response time of less than 10 seconds each during normal operation | Use load balancers, which will equally distribute the load to the different servers available. This will ensure that there will not be a single server that is overloaded, which can cause a reduction in performance and hence delays |
| Performance / Scalability | 700 transactions requests can be processed per minute for both file uploads and API calls | AWS EC2 auto-scaling can be implemented, whereby the EC2 will detect a surge in traffic and auto scale up by adding more EC2 instances. After the traffic reduces, it will also auto scale down accordingly  Aurora can be configured to scale up horizontally automatically to handle a much greater number of requests simultaneously. It will then scale down subsequently when there is minimum usage |
| Availability | All pages are down for less than 1 hour (RTO) during server failover operation | Use of multiple servers (active-active or active-passive) and load balancer, which will detect that a request to a particular server is failing and will handle the fault by routing the request to a secondary server.  The deployments are distributed across at least 2 availability zones |
| Availability/ Maintainability | Zero downtime during deployments | Achieved using AWS CodePipeline, AWS CodeBuild and AWS CodeDeploy for continuous deployments, resulting in little to no downtime during updates. Health checks are conducted using AWS ElasticBeanstalk. If errors are detected in the newly deployed update, the pipeline will identify these errors and the new update will not be pushed to production. |
| Security | Minimal portion of customer data are affected when 1 part of the deployment is compromised | Ensure passwords are hashed before storing in the database so hackers who gain access to the database will not see the password in plaintext  Separate different functionalities of the application to smaller microservices with different authentication details  Separate the data for each bank in the database |
| Security | AWS Shield provides Distributed Denial of Service (DDoS) protection to AWS services and application  AWS WAF prevents attacks by allowing you to configure traffic rules that allow, block and monitor traffic.  AWS Certificate Manager allows for the provisioning of SSL/TLS certificates. | AWS Shield inspects incoming network flow and monitors application traffic to Elastic IP address, Load Balancers, CloudFront.  AWS WAF can block popular web exploits such as cross site scripting and SQL injections.  SSL/TLS certifications allow for secure network communications over the internet as well as on private networks. |
| Reliability | AWS Aurora is fault tolerant and self healing hence provides higher site reliability  AWS Aurora can be synced across Availability Zone hence meeting the Availability attribute | AWS Aurora is designed to recover from a crash almost instantaneously and is able to perform crash recovery on parallel threads.  AWS provides up to 15 read or write replicas hence it can be scaled horizontally and vertically to meet the application requirement I/O operation.  An AWS Aurora global database serving as the primary DB instance can be spanned across regions and synchronously replicated across hence providing data redundancy. |

## **Solution View**





**AWS Services used:**

AWS EC2 T3.micro with VPC and ALB

AWS Aurora MySQL

AWS S3

Amazon CloudFront

AWS Route 53

AWS Certificate Manager

AWS API Gateway

AWS Key Management

AWS SNS

AWS CodeBuild

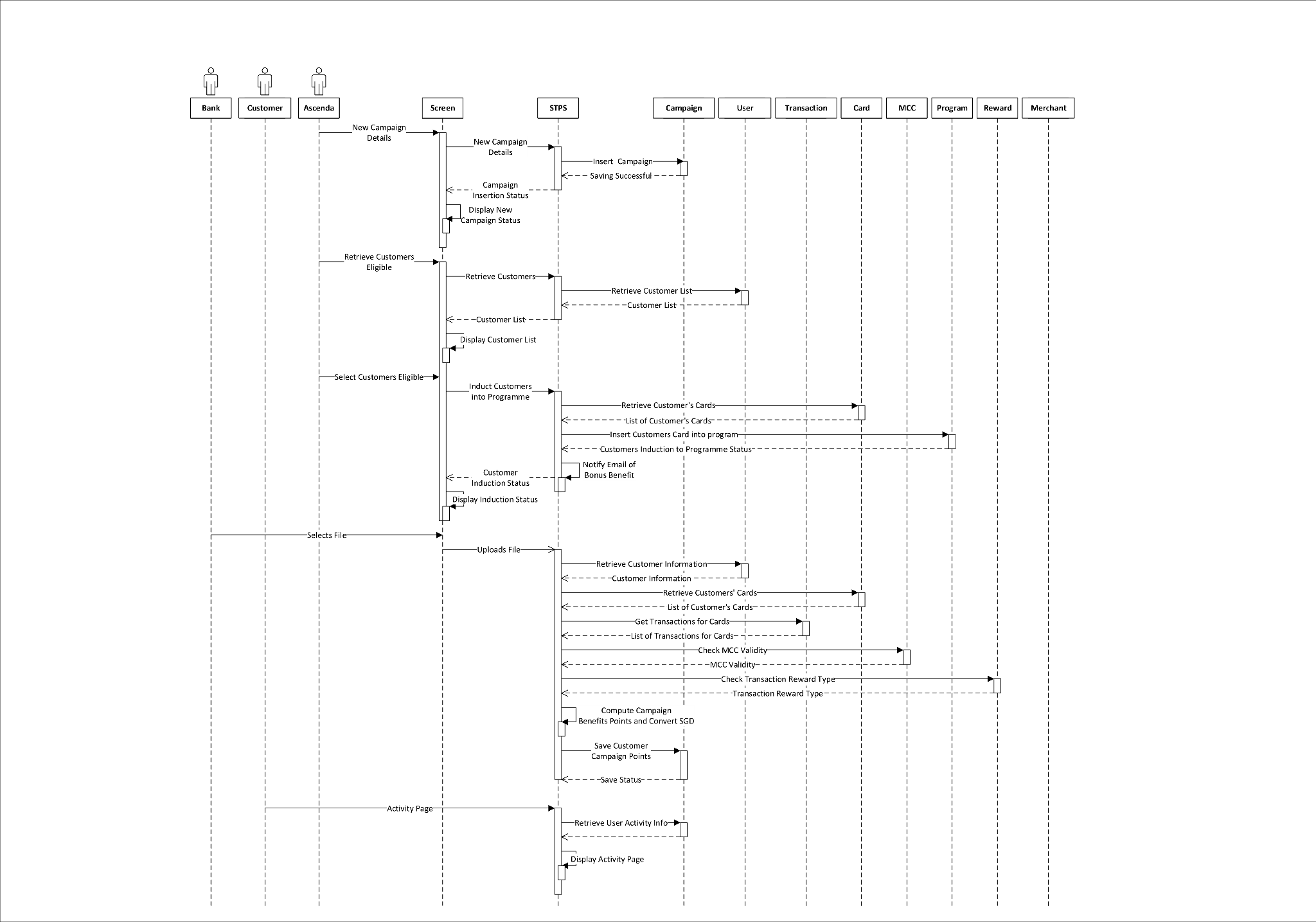
AWS CodeDeploy

AWS CodePipeline

AWS Web Application Firewall (WAF)

## **Sequence Diagram**

The sequence diagram details the methods from the control classes to the entity classes.





## **Appendix**

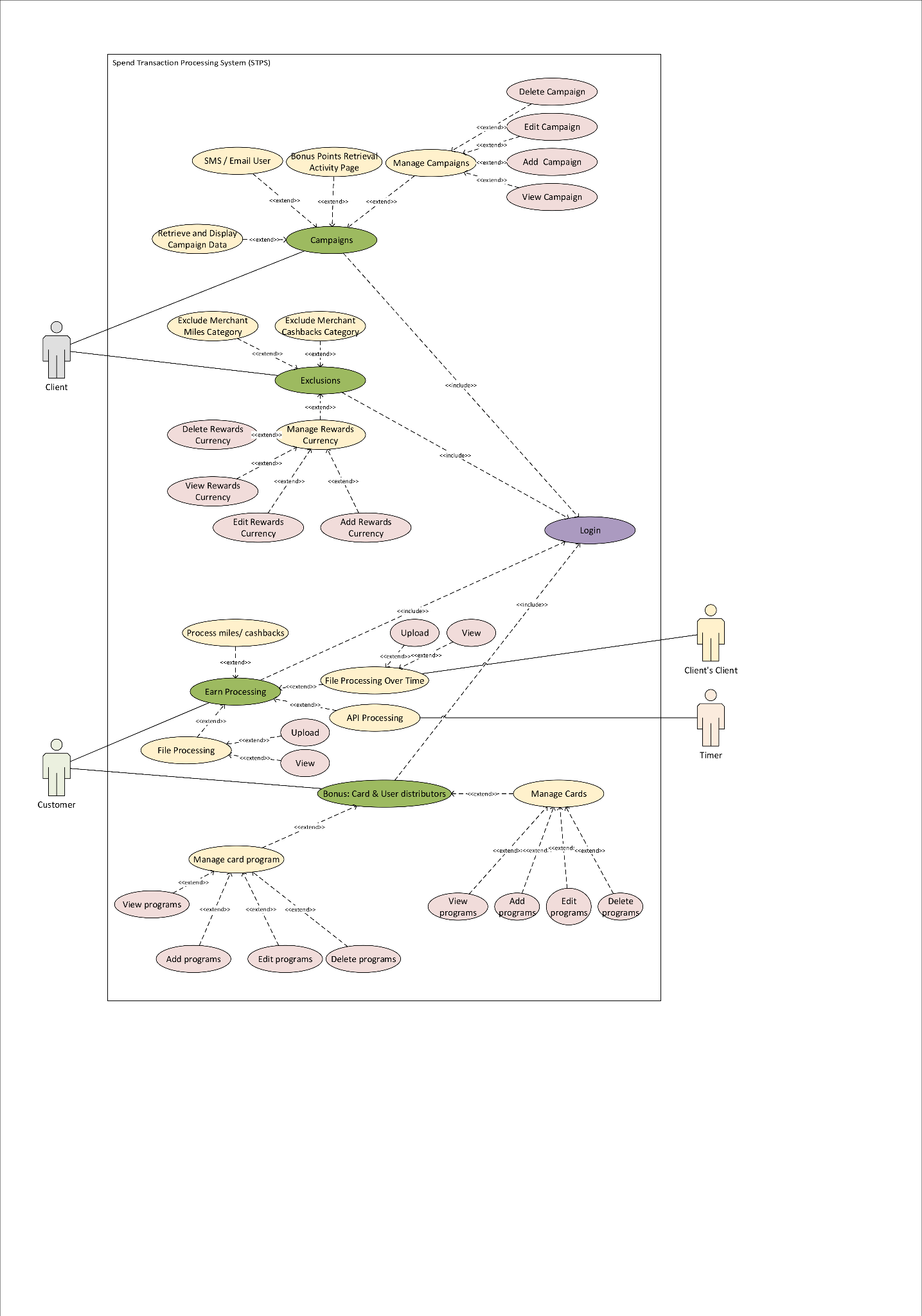
### Entity Relationship Diagram

Graphical user interface, application

Description automatically generated



### Use Case Diagram





1. <https://calculator.aws/#/estimate?id=4b2f817d41fe3bafc5f14ab3ee3c641015726afe> [↑](#footnote-ref-1)