SMU

SINGAPORE MANAGEMENT UNIVERSITY

School of

Information Systems

Khoo Khim Boon Ernest 01337679

Woon Quan Austin 01315332

Wong Hejian Javier 01340105

Sherman Lee 01333200

Leonard Tan 01368820

Tan Jiale Brennan 01346536

1. Background & Business Needs

Ascenda facilitates loyalty currency transfers between multiple loyalty programs. Without Ascenda's platform, building such capabilities is often time-consuming and complex. The loyalty program has to build individual integrations into each third party loyalty program and each integration is often quite different from each other. Some loyalty programs may be based on the REST architecture while others thrive on more traditional methods such as the SOAP protocol or file based transfer. Ascenda is required to collaborate with the banks with regards to the following functions: Provide the latest information about the loyalty programs, validate membership for a given loyalty program, accept and process accrual information on behalf of the banks and provide transaction details and status when being polled by the banks. Additionally, Ascenda is required to collaborate with the loyalty programs to perform transfer fulfillments as well.

2. Stakeholders

Stakeholder	**Stakeholder Description**	**Permissions**
Banks	Banks are Ascenda's primary customers that depend on Ascenda to carry out a multitude of functions which includes information gathering , validation of membership, processing of information and liaising with the different loyalty programs.	Ascendas Middleware - Read: Get latest available loyalty Programmes - Write: Send information regarding transfer of points
Loyalty Programmes	Loyalty Programmes are the main vendors Ascenda works with to abstract out the complicated API integrations to other 3rd party loyalty programmes, perform seamless transfer and fulfil file retention policies in the correct file format.	Ascendas Middleware - Write: Send updated handback file to Ascenda's database to update transaction statuses - Read: Download accrual information
AWS	AWS provides Ascenda with the infrastructure and services to deploy and manage the applications. As an IT stakeholder, Ascenda is expected to use their services legally and perform all of the necessary security configuration and management tasks.	
Ascendas' Maintenance Team	Ascenda's Maintenance Team is responsible for the availability, scalability and security of the solution.	

Ascenda's Developmen t Team	Ascenda's Development Team is responsible for the development of features that fulfill the business needs of the banks.	

3. Key Use Cases

Use Case Title - Retrieves loyalty programmes	
Use Case ID	1
Description	Ascendas API is invoked by the vendor to retrieve the necessary loyalty programmes information
Actors	Bank
Main Flow of events	Bank's client side calls Ascenda's API to retrieve all the subscribed loyalty programmes so that it may render it.
Alternative Flow of events	No subscribers. Error 404 / blank page.
Pre-conditions	Ascenda's API is up Ports are configured to accept data packets Have a registered API key to attach to header for authorization purposes
Post-conditions	Bank's Web Page displays all associated loyal programmes.

Use Case Title - Send loyalty programmes information	
Use Case ID	2
Description	Ascendas to expose endpoint for banks to call and retrieve latest loyalty programme information
Actors	Ascendas
Main Flow of events	Bank's client side calls Ascenda's API to retrieve all the subscribed loyalty programmes so that it may render it
Alternative Flow of events	Client not authorized to call endpoint, 403 Forbidden error
Pre-conditions	Each bank should have a registered api key and attach this key to headers for authorization of access to API
Post-conditions	Bank's Web Page displays all associated loyal programmes.

Use Case Title - Convert currencies to loyalty program points	
Use Case ID	3
Description	User initiates request to convert currencies associated with a loyalty program
Actors	User Bank
Main Flow of events	User enters amount to convert to loyalty points
Alternative Flow of events	None
Pre-conditions	Bank webpage is up User should have validated their membership
Post-conditions	Users should proceed for membership validation after.

Use Case Title - Validate loyalty program membership	
Use Case ID	4
Description	Upon receiving membership details, the Bank verifies validity of the membership via Ascenda.
Actors	Bank, Ascenda
Main Flow of events	Bank invokes Ascenda's API, Ascenda's API responses with with status code (200 or 404)
Alternative Flow of events	No alternative
Pre-conditions	Ascenda's API is up Ports are configured to accept data packets
Post-conditions	Validation response which decides if the user can proceed to exchange currency

Use Case Title - Receive Accrual Information	
Use Case ID	5
Description	Bank sends accrual information to Ascenda via API endpoint
Actors	Bank, Ascenda
Main Flow of events	Bank sends requests including accrual information to Ascenda's endpoint, Ascenda replies with a unique system ID

	if accepted.
Alternative Flow of events	Bank sends requests including accrual information to Ascenda's endpoint, Asecnda rejects information due to incomplete or invalid data.
Pre-conditions	Ascenda's API is up Ports are configured to accept data packets
Post-conditions	Bank receives system ID for approved request

Use Case Title - Process Accrual Information	
Use Case ID	6
Description	Ascenda processes accrual information from the bank and converts into a format applicable to the loyalty programmes
Actors	Ascenda
Main Flow of events	Ascenda retrieves information, transforms the information by aggregation or modification and subsequently exporting it as a file format.
Alternative Flow of events	No Alternatives
Pre-conditions	Ascenda's necessary processing mechanisms should be up.
Post-conditions	An accrual file generated should be ready for export

Use Case Title - Send Accrual Information	
Use Case ID	7
Description	Ascenda sends accrual file to loyalty program
Actors	Ascenda, 3rd party loyalty program
Main Flow of events	Asecnda sends accrual file to loyalty program via SFTP
Alternative Flow of events	No Alternatives
Pre-conditions	Loyalty Program's application must be able to accept the accrual file format Loyalty Program's application must be able to accept the accrual file via HTTP
Post-conditions	Upon sending an accrual file, Ascenda should expect to

	receive a handback file in the right format

Use Case Title - Receive handback file	
Use Case ID	8
Description	Ascenda receives the handback file sent from loyalty program
Actors	Ascendas' 3rd party loyalty program
Main Flow of events	Loyalty Program sends handbackfile to Ascenda via HTTP
Alternative Flow of events	No Alternatives
Pre-conditions	Ascendas' application must be able to accept the accrual file format Ascendas' application must be able to accept the accrual file via HTTP
Post-conditions	Upon accepting the accrual information, Ascendas should check for any errors and update the DB and message broker Ascendas should update DB on outcomes of transactions from handback file

Use Case Title - Accepts banks Transaction enquiry	
Use Case ID	9
Description	Ascenda's application accepts the poll from the bank for a status on conversion status
Actors	Bank, Ascendas
Main Flow of events	Bank sends a real-time API call to Ascendas' application requesting for a status
Alternative Flow of events	No Alternative
Pre-conditions	Ascendas' application must be able to accept the banks API-call Bank has registered API key with Ascendas
Post-conditions	Ascenda's application begins processing the banks request

Use Case Title - Send transaction outcome information	
Use Case ID	10
Description	Ascenda's replies to the banks the status of the enquired

	transaction
Actors	Bank, Ascendas
Main Flow of events	Ascenda's application sends the status of the transaction via API-call to the bank's application
Alternative Flow of events	No Alternative
Pre-conditions	Bank's application must be able to accept the responses from Ascendas application via a protocol
Post-conditions	Bank's are updated on the status of the enquiry.

4. Quality Requirements

Quality Requirements	
Global Performance	
Membership Validation	< 100ms **(average from 1000 requests)**
Accrual Request & Querying	< 200ms **(average from 1000 requests)**
Scalability	
Auto Scaling Group	Auto Scaling will create a new instance when CPU utilization reaches 80% and destroys an instance when CPU utilization is less than 20%
Ease of Maintenance	
Architectural Style	Tiered (Refer to Appendix 5)
Framework Design Pattern	Django MVC
Data Security	
Personal Information	AWS Cognito, SSL Certificate, API Gateway API Key
Systems Security	
AWS Web Application Firewall	Allowing our web application or APIs to be protected against common web exploits that may affect availability or compromise security.
Amazon GuardDuty	Allowing our web application and APIs to detect threats and constantly stay vigilant by using a threat detection service that continuously monitors for malicious activities and unauthorized behaviour.

File Retention Policy	
AWS S3 & Glacier	30 days retention policy before sending to infrequent-access and retains files for 5 years.
Resilience & Disaster recovery	
AWS RDS AWS Elastic Load Balancer (ELB) with Multi-AZ deployment of EC2 instances Data resiliency	Read replica configured in separate AZ ELB performs health checks for instances in the 2 availability zones and will redirect traffic to healthy EC2 instances. If a check fails on any instance, it indicates that instance failed, and AWS will spawn another EC2 instance to meet minimum required instances to be up.. S3 buckets have 11 9's of availability by default
China	
Deployment	Able to deploy instances in Hong Kong region

5. Key Architectural Decisions

Architectural Decision - Multi-AZ Configuration	
ID	1
Issue	Single AZ configurations are not fault tolerant as a loss of the AZ would result in the entire solution being unavailable
Architectural Decision	Create 2 public and private subnets in 2 separate availability zones. Traffic is routed via a Elastic Load Balancer that performs health checks and routes to healthy instances In the event of AZ-1 failure, Read replica is promoted to standalone RDS instance
Assumptions	Assume that catastrophic failure of AWS services is limited to a single Region that houses both the AZs
Alternatives	Multi-Region configuration Deploy a CloudFormation template in another region and set up Route53 that will route traffic in the event of an entire region failure
Justification	It is highly unlikely that an entire Region or multiple AZs will go down simultaneously. The alternative is much more expensive to maintain as a solution for

	availability

Architectural Decision - Scalability	
ID	2
Issue	T2.micro instances may have maxed out on network performance and subsequent requests are bottle-necked
Architectural Decision	Place EC2 instances in an auto-scaling group with CloudWatch with NetworkOut as the metric for the policy
Assumptions	T2.micro is unable to handle all the network requests
Alternatives	Replace T2.micro with a more powerful instance type such as A1 instances
Justification	Processing power of t2.micro is sufficient and scaling horizontally is more cost-effective

Architectural Decision - Scalability	
ID	3
Issue	Multiple requests must be handled in the order that they were sent
Architectural Decision	Implement Amazon Simple Queue Service to process the requests sequentially
Assumptions	There are multiple requests coming in simultaneously
Alternatives	Use a message broker like Amazon MQ to handle the requests
Justification	Use case is simple enough and does not require a message-broker. Amazon MQ is less cost-effective since it charges per hour while SQS charges per request handled

Architectural Decision - VPC with Public and Private Subnets	
ID	4
Issue	Ascenda’s backend services and databases must not be accessible directly from the internet

Architectural Decision	Place RDS and EC2 instance containing ITSA backend API in private subnet, connected with a public subnet containing the bank's front-end page Created security groups for the bank's page to communicate with Ascenda's middleware along with an access control list that Ascenda's middleware should not be accessible from the internet directly but still transmit data to the Bank's front-end for displaying to users
Assumptions	Ascenda's middleware will be attacked if placed in the public subnet
Alternatives	Set up Ascenda's middleware on-premises and connect to the bank's front-end through AWS Direct Connect
Justification	Ascendas does not have domain expertise and is not a cost effective solution as it may require heavy investments into a IT department

Architectural Decision - Serverless Compute	
ID	5
Issue	Batch processing is only performed at the end of the day
Architectural Decision	Use AWS Lambda for the batch processing to standardise the protocol used between the services and Ascenda's middleware AWS Lambda allows Ascendas to pay for the compute power during a specified period of time
Assumptions	Loyalty programs use different protocols for adding points
Alternatives	Set up a scheduled reserved instance that handles the batch processing
Justification	AWS Lambda is cheaper to trigger minute batch job periodically

Architectural Decision - Scalability, Security, Maintainability	
ID	6
Issue	Proxy is required for optimal routing of requests, centralised point for maintenance and security checks to ensure the requests are authorised before sending the traffic to the backend

Architectural Decision	API Gateway configured to route requests to Load Balancer which then distributes the traffic across traffic across all instances in the AZs. Each bank has a unique API Token which is passed into the request headers so only authorised banks are allowed to access the gateway.
Assumptions	Configuring the API Gateway is less cumbersome than individually setting configurations on separate resources.
Alternatives	Create 2 public subnets in each of the AZs and launch NAT instances for each. Configure the inbound rules of the security group to allow traffic in from the IP addresses of the banks. Configure the outbound rules to match the inbound rules of the instances that reside in the private subnets (backend).
Justification	AWS Gateway provides a centralized approach to manage resources and reduces management overhead.

Architectural Decision - Security	
ID	7
Issue	DB should only be accessible within the same VPC and not exposed to the internet. Data in-transit should be encrypted to prevent MITM attacks. Access to documents (e.g. batch files) should be limited to verified parties
Architectural Decision	Backend Django application sets up an SSL connection with RDS to transfer data securely RDS is placed in private subnet with a security group attached that allows only inbound traffic at port 3306 only within the VPC Data in RDS at rest is encrypted Pre-signed URL emailed loyalty partners for downloading of batch-files with an expiration date.
Assumptions	Data stored is sensitive and requires security (encryption etc.)
Alternatives	Encryption could be substituted with password-protected documents. Do not send sensitive information in transit but rather Codes that can be mapped to values internally on-premises.

	Mapped values to the codes are then sent across an air-gap for deciphering
Justification	Alternative is not maintainable and scalable as the number of variations for codes and mappings increases rapidly

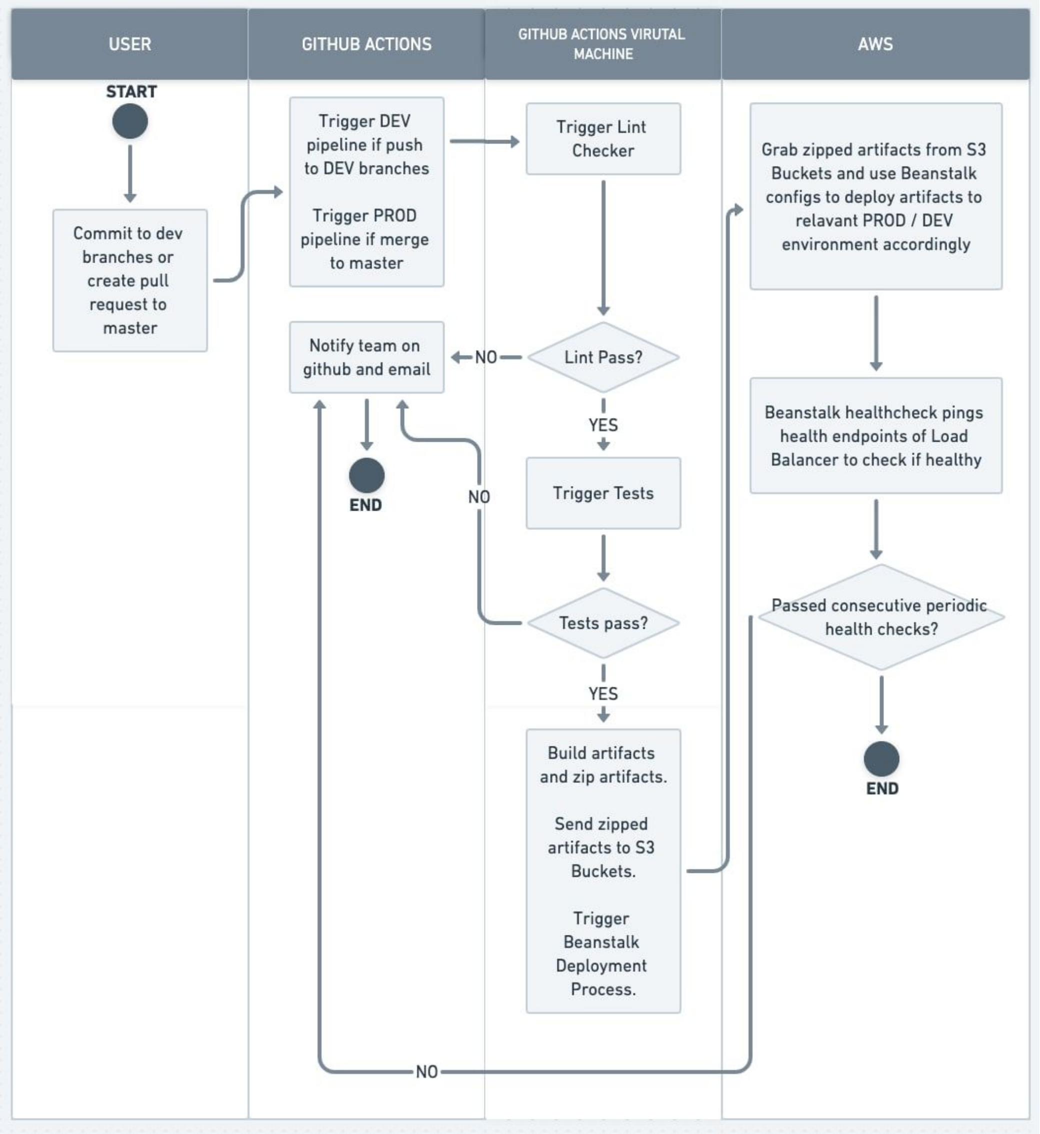
Architectural Decision - Maintainability, Availability	
ID	8
Issue	Code contributed by multiple developers should not break the existing solution. Code that breaks the solution should be be identifiable through various stages of the pipeline
Architectural Decision	Set up CICD pipeline using Github actions with a workflow that ensures a standardised development environment, and passing of test cases before deployment to production
Assumptions	There are multiple developers working on the same project at the same time
Alternatives	Only have 1 developer work on the project at a time
Justification	Not feasible as the time to market would be too long

Architectural Decision - Maintainability, Scalability	
ID	9
Issue	Limited time to create a solution requires a framework for easier configuration settings and management of the application
Architectural Decision	Django Framework deployed on Elastic Beanstalk
Assumptions	Django is more developer friend than SpringBoot Development with Django is faster Assuming that Django is highly scalable and more easily extensible
Alternatives	Use Springboot or Flask as the backend framework

Justification	Team is familiar with Python and Django framework and have a limited time to produce a solution

Architectural Decision - Maintainability, Scalability	
ID	10
Issue	Managing, provisioning and configuration of infrastructure should be in a format that is readable by all developers and easily modifiable for changing business requirements
Architectural Decision	Infrastructure-as-code using Terraform and CloudFormation
Assumptions	Terraform and CloudFormation are granular enough to configure the infrastructure without the use of the AWS Management Console, AWS CLI or SDK
Alternatives	Provision and configure the infrastructure through the AWS Management Console, AWS CLI or SDK. Document the configuration steps for other teams to replicate the environment
Justification	As the architecture becomes increasingly complex the entire solution overview would only be understandable and provisioned by the team that were in-charge of it

5. Development View


USER
GITHUB ACTIONS
GITHUB ACTIONS VIRUTAL MACHINE
AWS
START
Commit to dev branches or create pull request to master
Trigger DEV pipeline if push to DEV branches
Trigger PROD pipeline if merge to master
Trigger Lint Checker
Lint Pass?
NO
Notify team on github and email
END
YES
Trigger Tests
NO
Tests pass?
YES
Build artifacts and zip artifacts.
Send zipped artifacts to S3 Buckets.
Trigger Beanstalk Deployment Process.
Grab zipped artifacts from S3 Buckets and use Beanstalk configs to deploy artifacts to relavant PROD / DEV environment accordingly
Beanstalk healthcheck pings health endpoints of Load Balancer to check if healthy
Passed consecutive periodic health checks?
END
NO

Github Action with a workflow

1. Sets up the dependencies
2. Ensures that Python syntax does not have any errors
3. Runs unit tests defined in Django
4. Generates the Secret Configs and relevant deployment packages
5. Deploys the loyalty partner and Ascenda's API to Elastic BeanStalk after lint check and tests have passed.

Notion

- Kanban board to keep track of the progress of functional and non-functional requirements of the project. Refer to **Appendix I**

6. Solution View

For network diagram, view **Appendix H.** For Deployment Diagram, view **Appendix G.**

6.1 Integration Endpoints

Source System	Destination System	Protocol	Format	Communication Mode
Bank	Ascenda	**HTTPS**	**JSON**	**Asynchronous**
Ascenda	Amazon S3	**HTTPS**	**CSV**	**Asynchronous**
Amazon SNS	Loyalty Partner	**Email (IMAP)**	**Text**	**Asynchronous**
Loyalty Partner	Ascenda	**HTTPS**	**CSV**	**Asynchronous**
Ascenda	Amazon RDS	**HTTPS**	**Text**	**Asynchronous**
Bank	Ascenda	**HTTPS**	**JSON**	**Asynchronous**

6.2 Hardware/Software/Services Required

No	Item	Quantity	License	Buy/Lease	Estimated Cost
1	Amazon S3	1	Proprietary	Lease	Using an estimate of 10000 POST requests and 1GB of storage per month, the estimated cost is USD0.25/month
2	Elastic Beanstalk	1	Proprietary	Lease	Using 3 instances of T2.micro under the EC2 instance savings plan, estimated cost is USD19.93/Month
3	Elastic Load Balancer	1	Proprietary	Lease	An ALB costs USD 18.14/Month
4	Lambda	1	Proprietary	Lease	With 60 estimated invocations per month, 200ms per request and 500mb of memory allocation, the lambda functions cost USD0.00/Month
5	API gateway	1	Proprietary	Lease	With 1 million requests per month, the API gateway costs USD 4.25/Month
6	Cognito	1	Proprietary	Lease	With estimated 30,000 Monthly

					active Users, the cost will be USD 0.00/Month
7	WAF	1	Proprietary	Lease	1 Web ACL with 5 rules USD 16/month
8	CloudWatch	1	Proprietary	Lease	Free tier for CloudWatch costs
9	RDS	1	Proprietary	Lease	1 db.t3.micro instance cost USD 17.45/month .
10	S3 Glacier	1	Proprietary	Lease	Storage is priced from **$0.004** per GB/month. Upload requests are priced from **$0.05** per 1,000 requests. Estimated (100 GB & 1000 requests) 50.40 SD/month
11	Simple Queue Service	1	Proprietary	Lease	USD 0.40/month
12	Amazon GuardDuty	1	Proprietary	Lease	With 1 million cloudtrail events, 1 million S3 data events & 50GB of VPC flow logs USD 54.61/month (First timer - 30 day free trial)
13	Domain Name	1	Proprietary	Buy	USD $12

7. Availability View

Node	Redundancy	Clustering			Replication if applicable			
		Node Config.	Failure Detection	Failover	Repl. Type	Session State Storage	DB Repl. Config	Repl. Mode
AWS RDS	Horizontal scaling Multi-AZ deployment	Active - Passive	Ping	Load-balancer	DB	NIL	Master-Slave	Asynchronous replication of

Node	Redundancy	Clustering		
		Node Config.	Failure Detection	Failover
EC2	Horizontal Scaling	Active - Passive	Auto Scaling health check	Auto Scaling replacement
	Multi-AZ deployment		Ping	Load-balancer

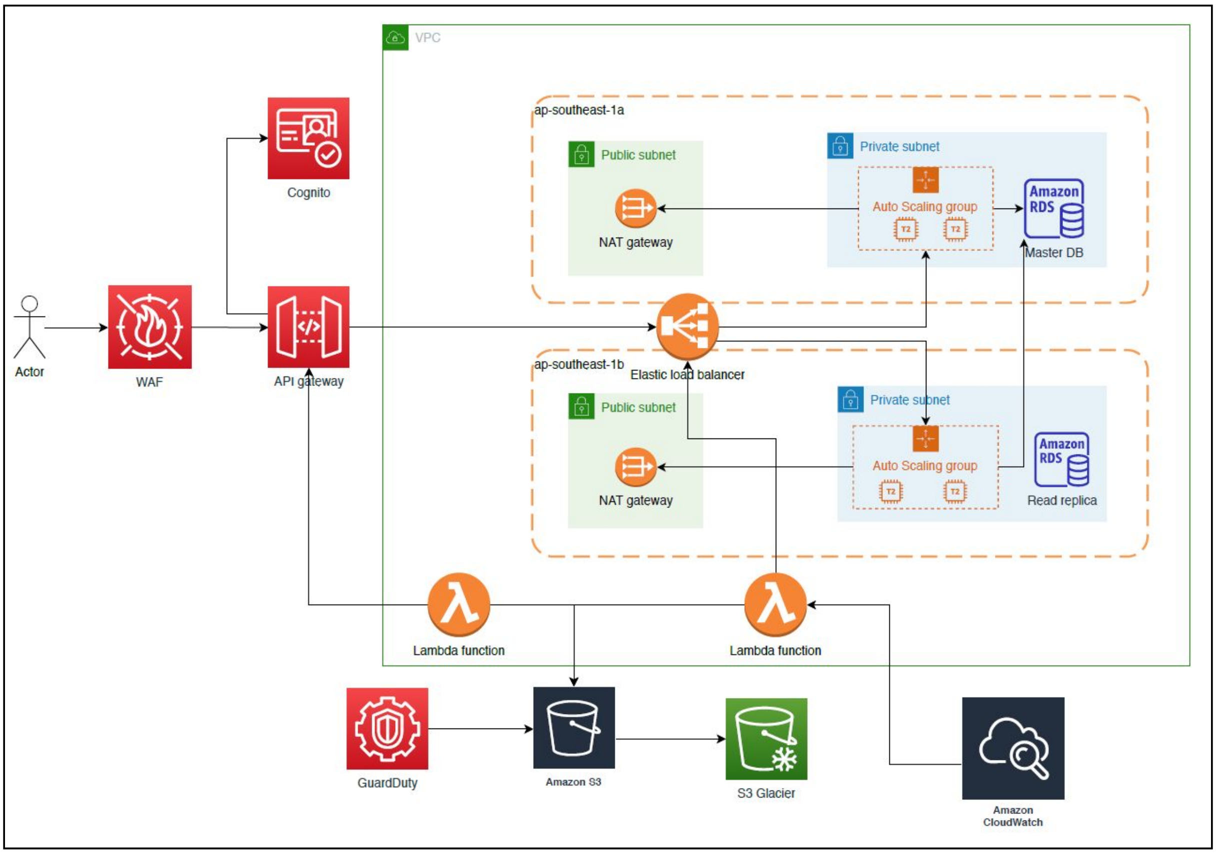
8. Security View

No	Asset/Asset	Potential Threat/ Vulnerability Pair	Potential Mitigation Controls
1	API gateway	One possible attack would be a distributed denial of service (DDos) attack that could exploit the entry API gateway, causing a single point of failure.	Implement AWS Web Application Firewall (WAF) which is a web application firewall that lets us monitor the HTTP and HTTPS requests that are forwarded to the API Gateway REST API. **[Implemented with the following rules : Amazon IP reputation list, Core Rule Set, Linux Operating System, SQL database]**
2	Users' Credentials, Membership &	One possible attack would be a malicious form of information exposure that could exploit the	Ensure RDS data and snapshots are encrypted. This ensures data at rest and snapshots are encrypted. **[Implemented through RDS configurations]**

	Transaction information	unencrypted RDS data and snapshots.	
3	Users' Credentials, Membership & Transaction information	One possible attack would be a malicious form of information exposure that could exploit the unencrypted data in transit..	Backend Django application sets up an SSL connection with RDS to transfer data securely **[Implemented]**
4	Ascenda's Database	One possible attack would be a malicious form of information exposure that could exploit the data in the database if it is too openly exposed.	We enforced a permission of least privilege where each specific bank is authorised to only poll from a specific SQS queue using IAM credentials. **[Implemented through SQS & IAM]**
5	Loyalty Partners' Credentials	Another possible attack would be server side request forgery (SSRF) which allows an intruder to make requests on behalf of a compromised loyalty partner and potentially reveal the application's network, ports and bypassing authorization.	Add multi-factor authentication (MFA) to the user pool in cognito to protect the identity of your users **[Not implemented yet]**
6	Batch File / Handback File (Stored in S3 - Standard & Glacier)	One possible attack would be a malicious form of information exposure that could exploit the unencrypted files in the s3 bucket.	Enable server side encryption for current & archival files using S3 managed keys (AES-256) or Customer master keys from KMS **[Not implemented yet]**
7	Secret Credentials to AWS resources and IAM user being exposed on current pipeline	When doing automated CI-CD, AWS secrets should not be exposed on the pipeline script itself as people could scrape the repository and gain access to the secret keys. This would allow them access to the AWS resources and misuse them.	Mask all secrets in as environment variables in Github Actions' Secrets and use these variables during the pipeline process. Never push secrets to publicly viewable repositories. **[Implemented, view Appendix A]**
8	Access to Loyalty Partner	A possible attack would be if a malicious organisation attempts to decoy themselves as a loyalty	Implement Amazon Cognito which provides authentication, authorization, and user management for your web and mobile apps This provides a token

	Portal	partner and upload corrupted files on to the system, exploiting the vulnerabilities in the server side validation of the system	to authorised users and prevents malicious sources from infiltrating. [**Implemented**]
9	Access to Ascendas API	A possible attack would be if a malicious user attempts to get unauthorised access to Ascendas API, allowing them to exploit the vulnerabilities of the system	The API Gateway provides a unique API key for each bank, which will ensure that the individual accessing Ascendas API is definitely a Bank. This prevents malicious users from accessing the endpoint. [**Implemented**]

8.1 Network / Architecture Diagram



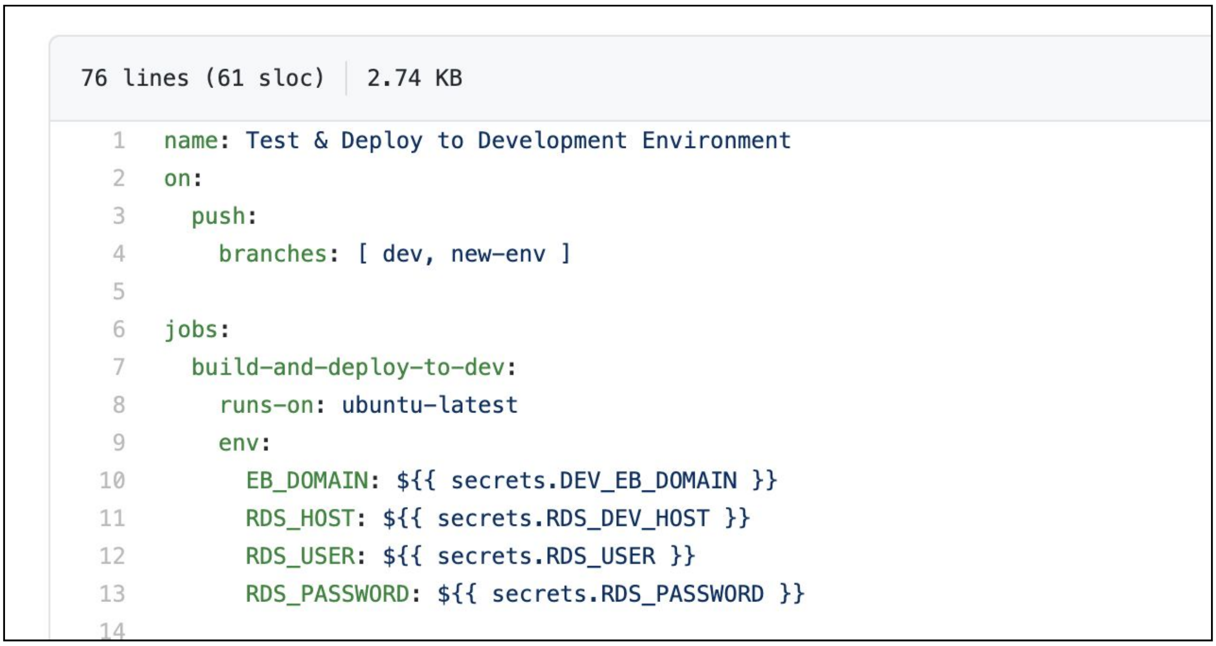
For more information regarding how we secured our applications' port access, please refer to **Appendix H**. To find out more about the vulnerabilities that our application is exposed to along with their mitigation strategies, you can refer to **Appendix D & E** to view how our application undergone penetration testing done via the OWASP ZAP tool.

9. Performance View

No	Description of Strategy	Justification	Performance Testing (Optional)
1	We used SQS for the banks to poll our queue. This allows us to decouple the application components so that they run and fail independently, increasing the overall fault tolerance of the system.	It is more secure as only authorised banks can access the queue with the given IAM credentials for polling. Furthermore, the development of the polling function can be more efficiently achieved with the functions of SQS	-
2	We used Elastic Beanstalk service (Paas) to host the loyalty partner and backend applications and concurrently leverage on the load balancers and auto scaling features provided.	Elastic Beanstalk automatically employs Auto Scaling and Elastic Load Balancing to scale and balance workloads. It automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.	Will be demonstrating
3	We enabled Multi- Availability Zone Amazon RDS Read Replicas to ease the read workload of our main RDS instance.	Amazon RDS Read Replicas provide enhanced performance and durability for the database instances. They make it easy to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.	Will be demonstrating
4	We enabled API caching in Amazon API Gateway to cache our endpoint's responses.	With API gateway caching, we can reduce the number of calls made to your endpoint and also improve the latency of requests to our API.	**See Appendix F**

Appendix

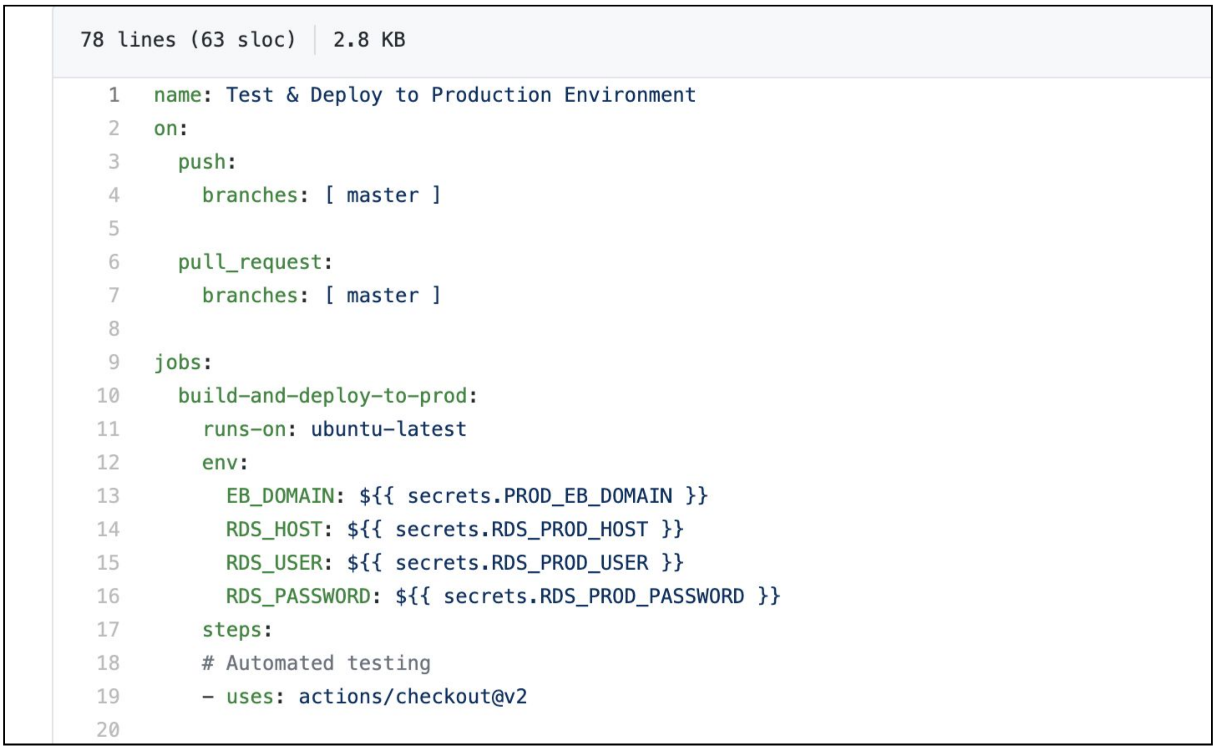
Appendix A: Github Actions Secrets



76 lines (61 sloc) | 2.74 KB

```yaml
name: Test & Deploy to Development Environment
on:
  push:
    branches: [ dev, new-env ]

jobs:
  build-and-deploy-to-dev:
    runs-on: ubuntu-latest
    env:
      EB_DOMAIN: ${{ secrets.DEV_EB_DOMAIN }}
      RDS_HOST: ${{ secrets.RDS_DEV_HOST }}
      RDS_USER: ${{ secrets.RDS_USER }}
      RDS_PASSWORD: ${{ secrets.RDS_PASSWORD }}
```

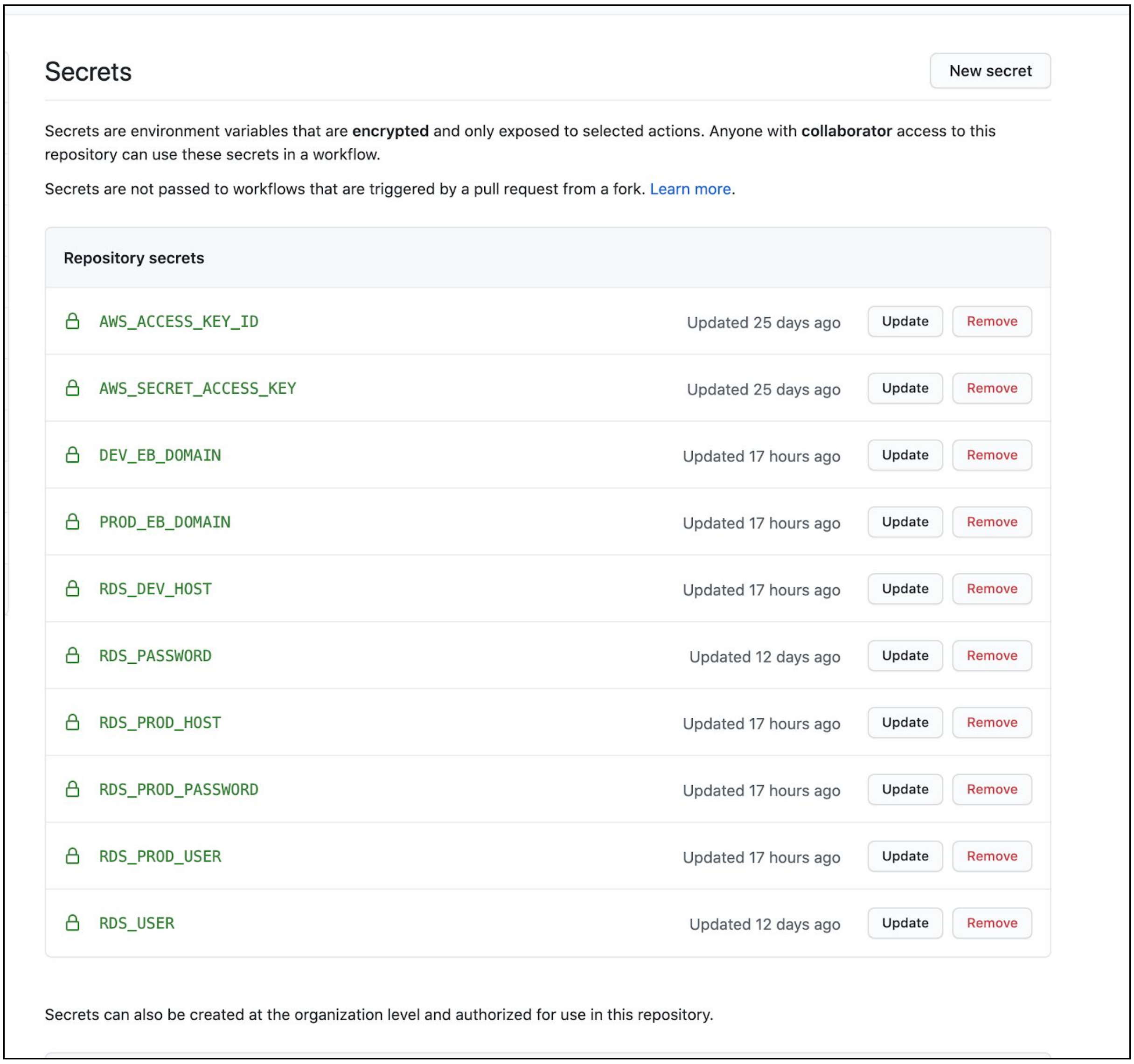


78 lines (63 sloc) | 2.8 KB

```yaml
name: Test & Deploy to Production Environment
on:
  push:
    branches: [ master ]

  pull_request:
    branches: [ master ]

jobs:
  build-and-deploy-to-prod:
    runs-on: ubuntu-latest
    env:
      EB_DOMAIN: ${{ secrets.PROD_EB_DOMAIN }}
      RDS_HOST: ${{ secrets.RDS_PROD_HOST }}
      RDS_USER: ${{ secrets.RDS_PROD_USER }}
      RDS_PASSWORD: ${{ secrets.RDS_PROD_PASSWORD }}
    steps:
    # Automated testing
    - uses: actions/checkout@v2
```



Secrets

Secrets are environment variables that are **encrypted** and only exposed to selected actions. Anyone with **collaborator** access to this repository can use these secrets in a workflow.

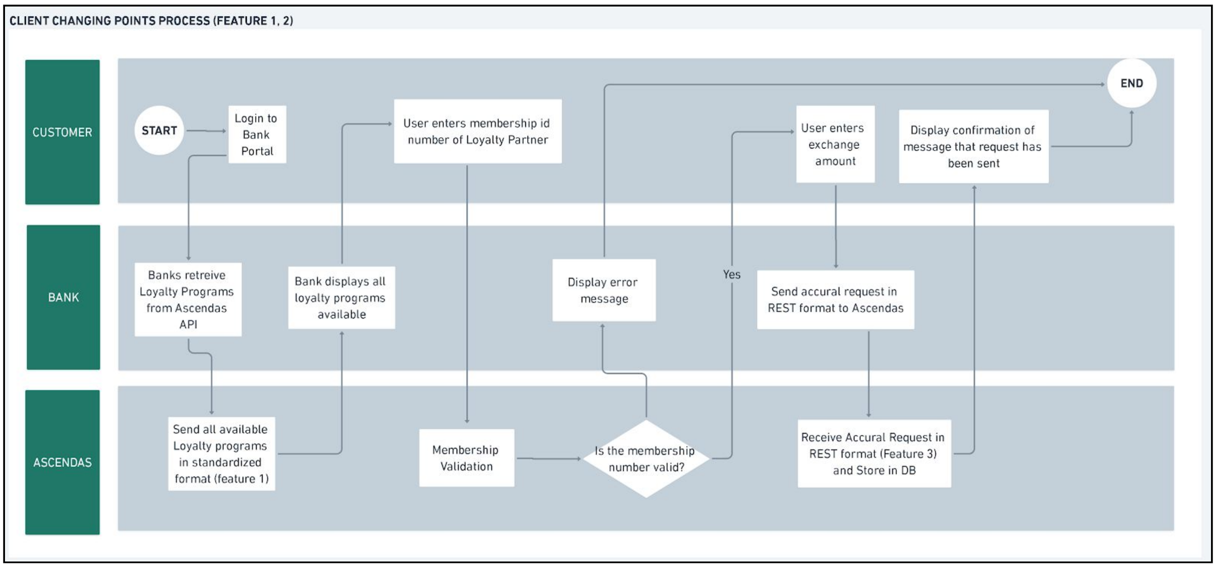
Secrets are not passed to workflows that are triggered by a pull request from a fork. Learn more.

Repository secrets

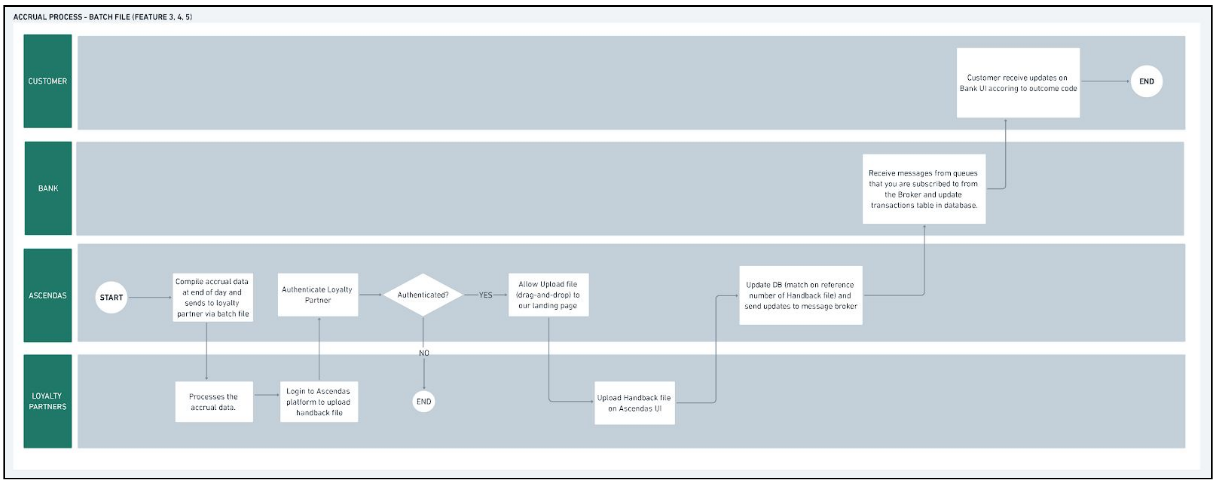
AWS_ACCESS_KEY_ID	Updated 25 days ago
AWS_SECRET_ACCESS_KEY	Updated 25 days ago
DEV_EB_DOMAIN	Updated 17 hours ago
PROD_EB_DOMAIN	Updated 17 hours ago
RDS_DEV_HOST	Updated 17 hours ago
RDS_PASSWORD	Updated 12 days ago
RDS_PROD_HOST	Updated 17 hours ago
RDS_PROD_PASSWORD	Updated 17 hours ago
RDS_PROD_USER	Updated 17 hours ago
RDS_USER	Updated 12 days ago

Secrets can also be created at the organization level and authorized for use in this repository.

Appendix B. Process flow

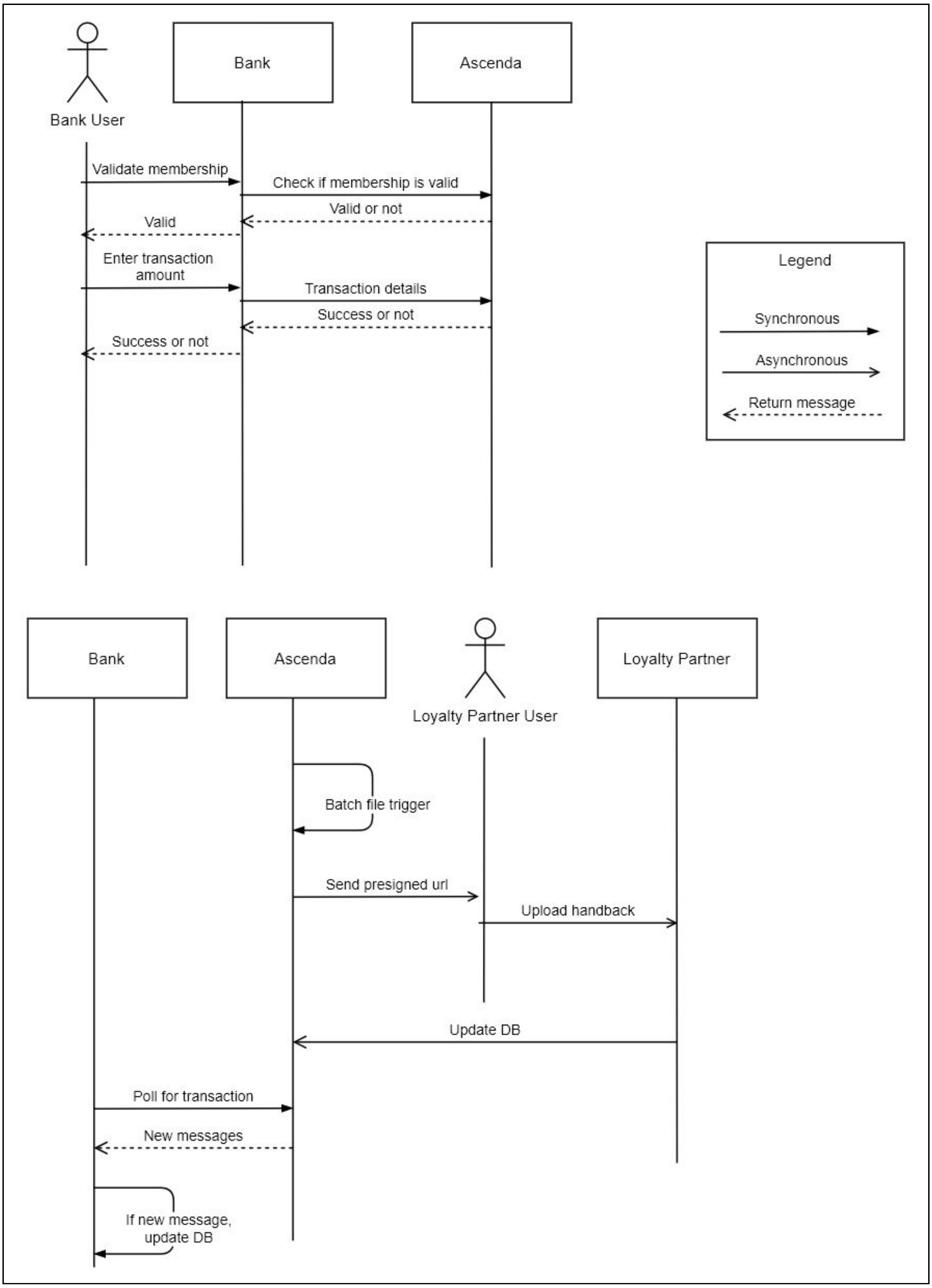

CLIENT CHANGING POINTS PROCESS (FEATURE 1, 2)

CUSTOMER
BANK
ASCENDAS
START
Login to Bank Portal
Banks retreive Loyalty Programs from Ascendas API
Send all available Loyalty programs in standardized format (feature 1)
Bank displays all loyalty programs available
User enters membership id number of Loyalty Partner
Membership Validation
Is the membership number valid?
Display error message
Yes
User enters exchange amount
Send accural request in REST format to Ascendas
Receive Accural Request in REST format (Feature 3) and Store in DB
Display confirmation of message that request has been sent
END

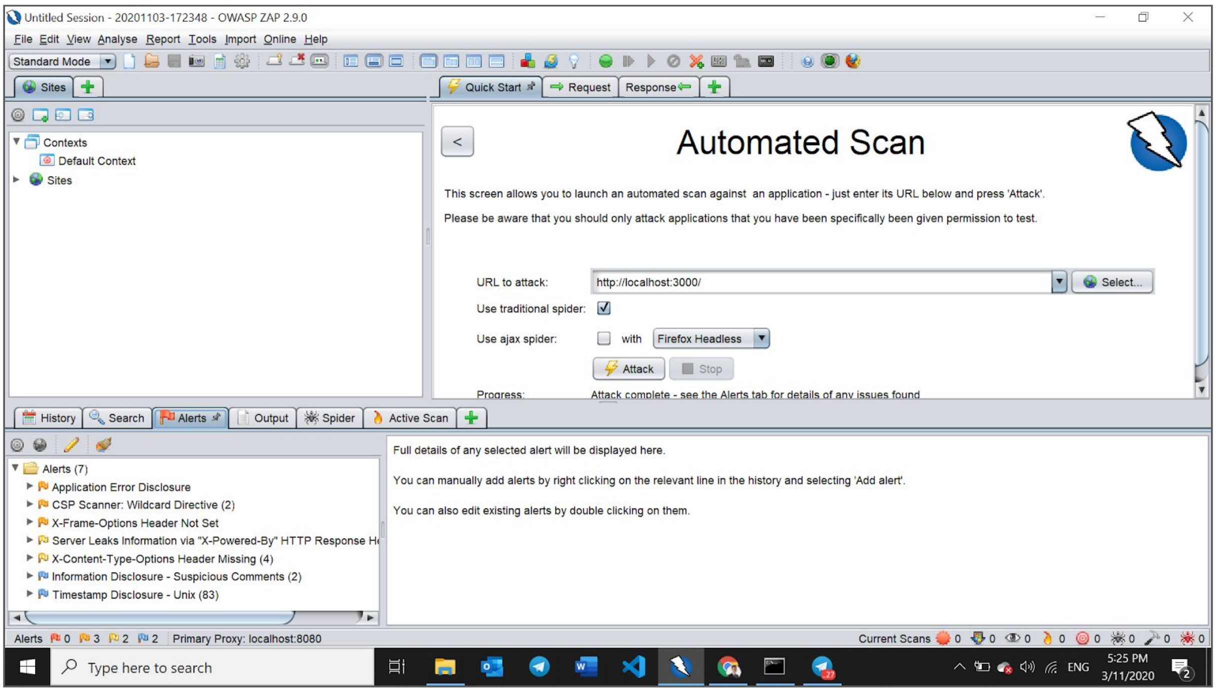

ACCRUAL PROCESS - BATCH FILE (FEATURE 3, 4, 5)

CUSTOMER
BANK
ASCENDAS
LOYALTY PARTNERS
START
Compile accrual data at end of day and sends to loyalty partner via batch file
Processes the accrual data.
Login to Ascendas platform to upload handback file
Authenticate Loyalty Partner
Authenticated?
NO
END
YES
Allow Upload file (drag-and-drop) to our landing page
Upload Handback file on Ascendas UI
Update DB (match on reference number of Handback file) and send updates to message broker
Receive messages from queues that you are subscribed to from the Broker and update transactions table in database.
Customer receive updates on Bank UI accoring to outcome code
END

Appendix C: Sequence Diagram



Appendix D. Penetration testing on localhost & vulnerabilities



Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://localhost:3000/

Use traditional spider: ☑

Use ajax spider: □ with Firefox Headless

Attack Stop

Progress: Attack complete - see the Alerts tab for details of any issues found

Alerts (7)
- Application Error Disclosure
- CSP Scanner: Wildcard Directive (2)
- X-Frame-Options Header Not Set
- Server Leaks Information via "X-Powered-By" HTTP Response He
- X-Content-Type-Options Header Missing (4)
- Information Disclosure - Suspicious Comments (2)
- Timestamp Disclosure - Unix (83)

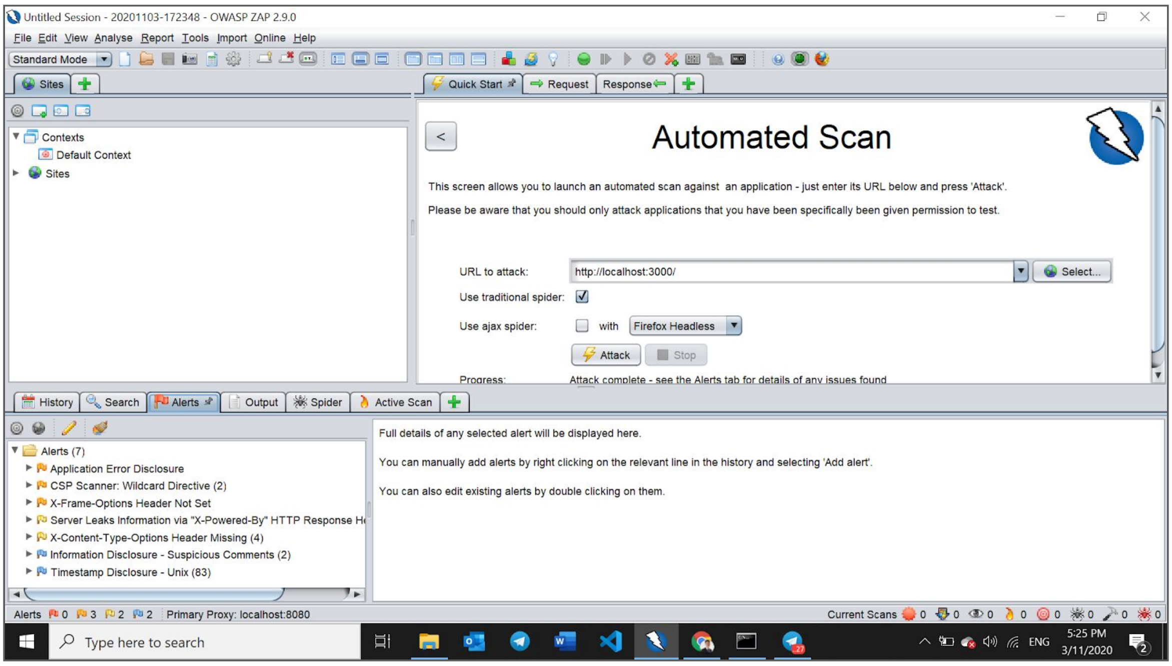
Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

Application Error Disclosure - Covered by beanstalk
CSP Scanner: Wildcard Directive (2) - Covered by beanstalk
Server leaks Information via "X-Powered-By" HTTP Response Header Fields - Covered by beanstalk
X-Frame-Options Header Not Set
X-Content-Type-Options Header Missing
Information Disclosure - Suspicious Comments
Timestamp Disclosure - Unix

Appendix E: Penetration Testing on AWS Elastic Beanstalk & Future work to overcome vulnerabilities



Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: http://localhost:3000/ Select...

Use traditional spider: ☑

Use ajax spider: □ with Firefox Headless

Attack Stop

Progress: Attack complete - see the Alerts tab for details of any issues found

Alerts (7)
- Application Error Disclosure
- CSP Scanner: Wildcard Directive (2)
- X-Frame-Options Header Not Set
- Server Leaks Information via "X-Powered-By" HTTP Response H
- X-Content-Type-Options Header Missing (4)
- Information Disclosure - Suspicious Comments (2)
- Timestamp Disclosure - Unix (83)

Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

Alerts 0 3 2 2 Primary Proxy: localhost:8080

Current Scans 0 0 0 0 0 0 0 0

X-Frame-Options Header Not Set

Description: X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.

Recommendation: Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers).

Incomplete or No Cache-control and Pragma HTTP Header Set

Description: The cache-control and pragma HTTP header have not been set properly or are missing allowing the browser and proxies to cache content.

Recommendation: Whenever possible ensure the cache-control HTTP header is set with no-cache, no-store, must-revalidate; and that the pragma HTTP header is set with no-cache.

X-Content-Type-Options Header Missing

Description: The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing.

Recommendation: Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.

Information Disclosure - Suspicious Comments

Description: The response appears to contain suspicious comments which may help an attacker. Note: Matches made within script blocks or files are against the entire content not only comments.

Recommendation: Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.

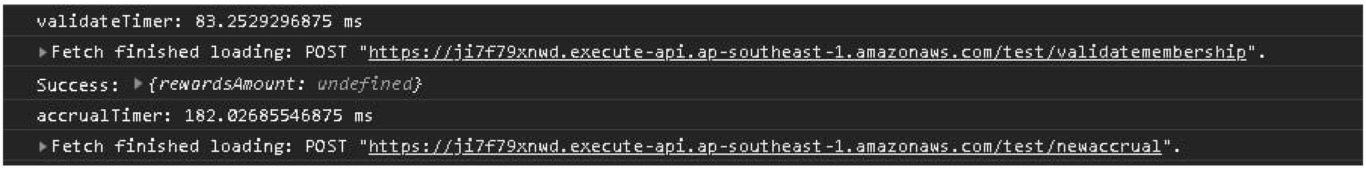
Timestamp Disclosure - Unix

Description: A timestamp was disclosed by the application/web server - Unix

Recommendation: Manually confirm that the timestamp data is not sensitive, and that the data cannot be aggregated to disclose exploitable patterns.

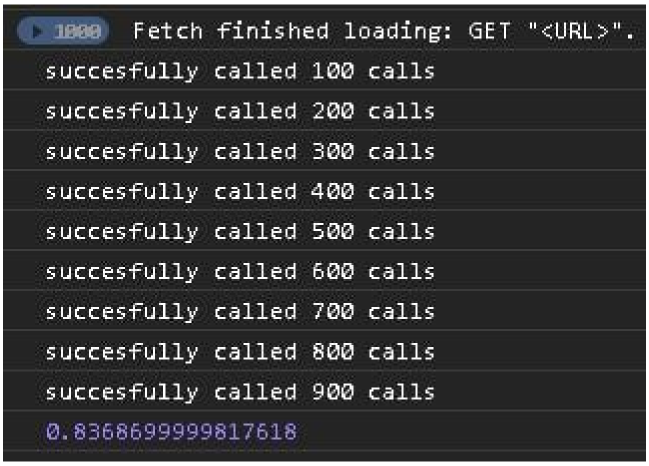
Appendix F. Quality Requirements

Global Performance



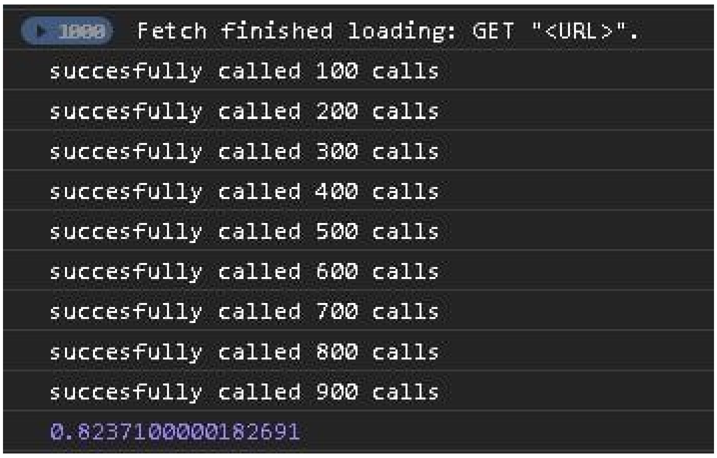
```
validateTimer: 83.2529296875 ms
▸Fetch finished loading: POST "https://ji7f79xnwd.execute-api.ap-southeast-1.amazonaws.com/test/validatemembership".
Success: ▸{rewardsAmount: undefined}
accrualTimer: 182.02685546875 ms
▸Fetch finished loading: POST "https://ji7f79xnwd.execute-api.ap-southeast-1.amazonaws.com/test/newaccrual".
```

Without Cache



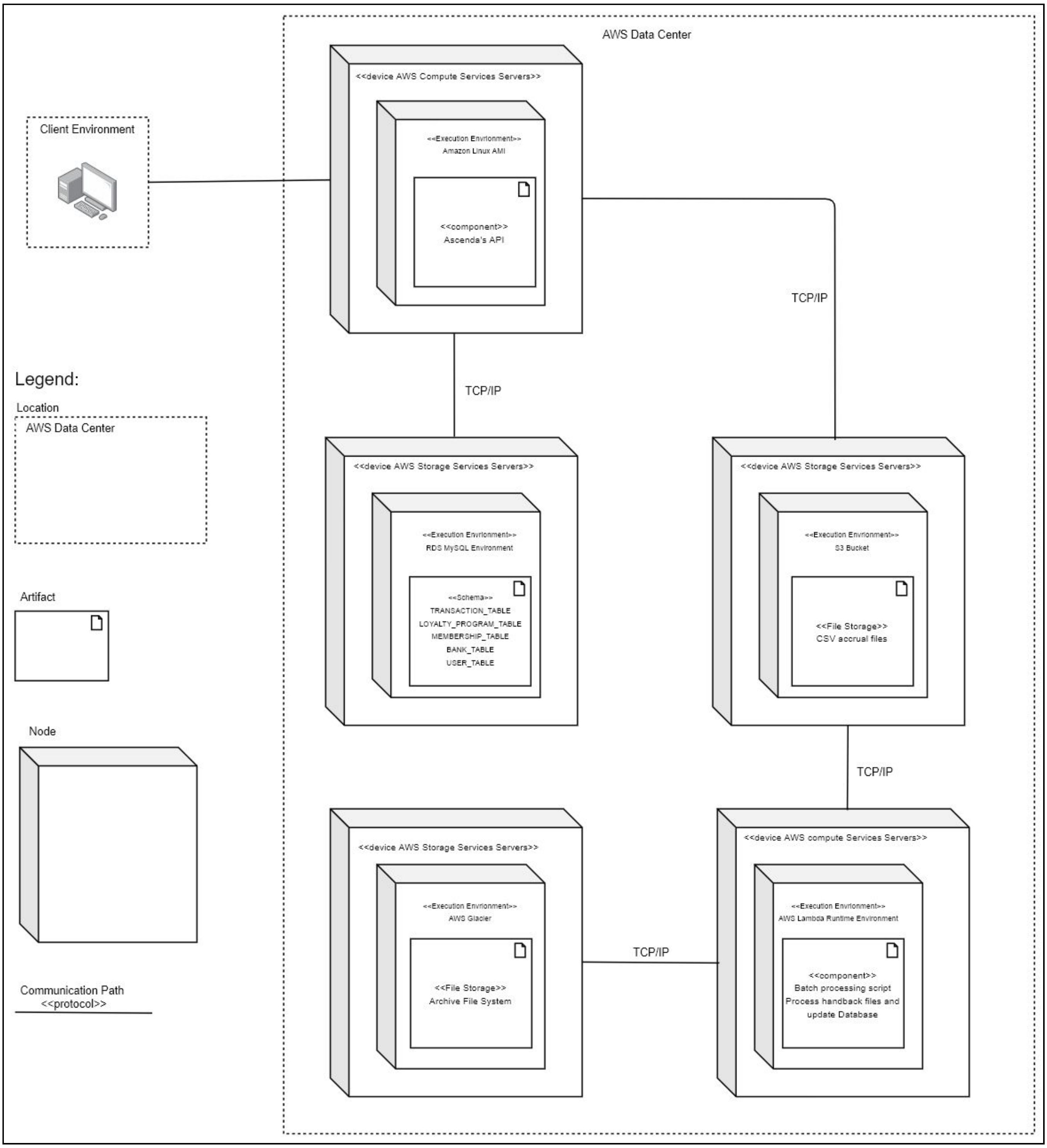
```
▸ 1000  Fetch finished loading: GET "<URL>".
 succesfully called 100 calls
 succesfully called 200 calls
 succesfully called 300 calls
 succesfully called 400 calls
 succesfully called 500 calls
 succesfully called 600 calls
 succesfully called 700 calls
 succesfully called 800 calls
 succesfully called 900 calls
 0.8368699999817618
```

With Cache

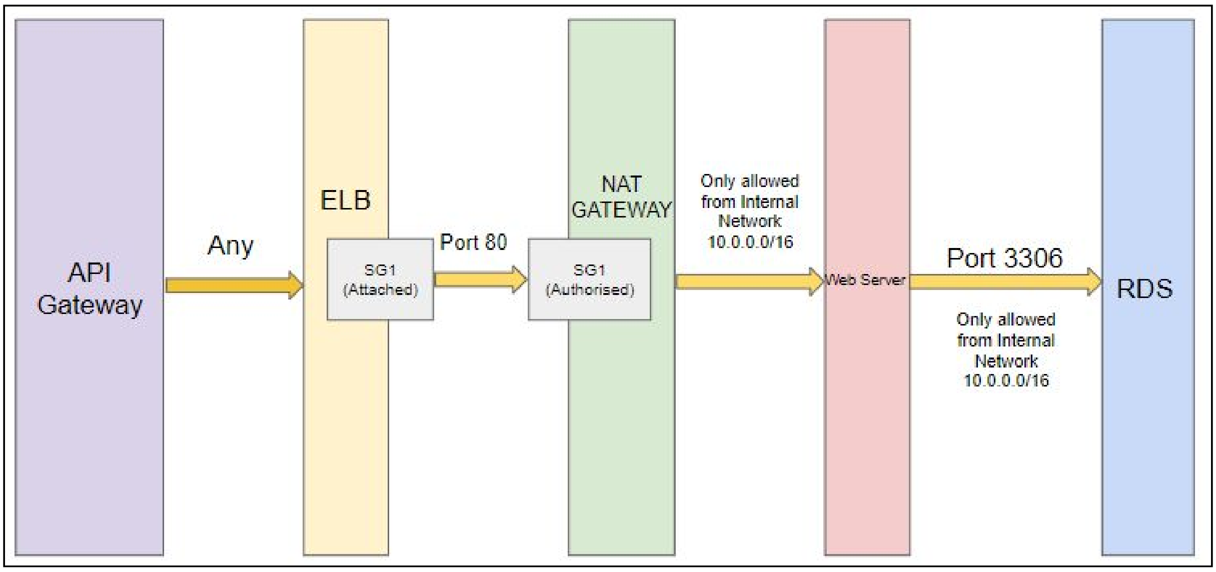


```
1000  Fetch finished loading: GET "<URL>".
  succesfully called 100 calls
  succesfully called 200 calls
  succesfully called 300 calls
  succesfully called 400 calls
  succesfully called 500 calls
  succesfully called 600 calls
  succesfully called 700 calls
  succesfully called 800 calls
  succesfully called 900 calls
  0.82371000000182691
```

Appendix G: Deployment Diagram

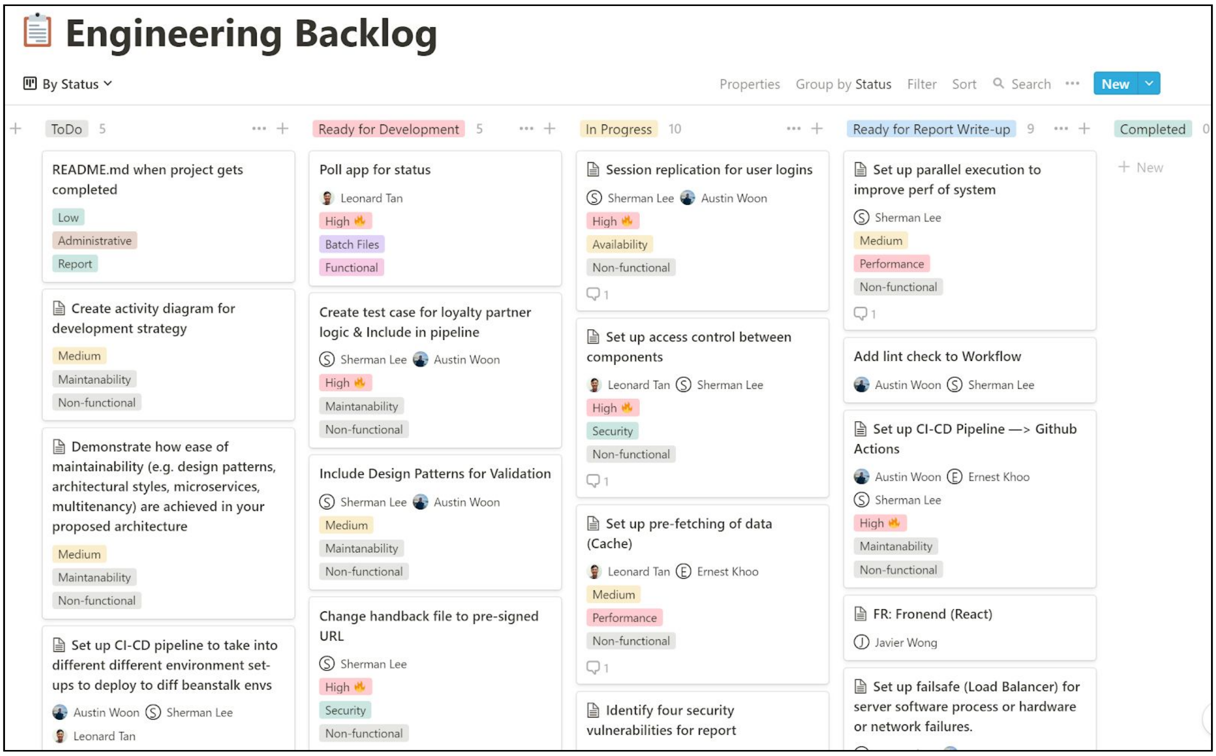


Appendix H: Secure Port Access Diagram


API Gateway
Any
ELB
SG1 (Attached)
Port 80
NAT GATEWAY
SG1 (Authorised)
Only allowed from Internal Network 10.0.0.0/16
Web Server
Port 3306
Only allowed from Internal Network 10.0.0.0/16
RDS

As part of our measures to keep unwanted traffic out, we kept our port access as restrictive as possible. For example, the NAT gateway can only be accessed from traffic originating from the elastic load balancer. Similarly, the web server and the RDS can only be accessed by components in the private network.

Appendix I: Kanban Board



Engineering Backlog

By Status

ToDo 5

README.md when project gets completed
Low
Administrative
Report

Create activity diagram for development strategy
Medium
Maintanability
Non-functional

Demonstrate how ease of maintainability (e.g. design patterns, architectural styles, microservices, multitenancy) are achieved in your proposed architecture
Medium
Maintanability
Non-functional

Set up CI-CD pipeline to take into different different environment set-ups to deploy to diff beanstalk envs
Austin Woon Sherman Lee
Leonard Tan

Ready for Development 5

Poll app for status
Leonard Tan
High 🔥
Batch Files
Functional

Create test case for loyalty partner logic & Include in pipeline
Sherman Lee Austin Woon
High 🔥
Maintanability
Non-functional

Include Design Patterns for Validation
Sherman Lee Austin Woon
Medium
Maintanability
Non-functional

Change handback file to pre-signed URL
Sherman Lee
High 🔥
Security
Non-functional

In Progress 10

Session replication for user logins
Sherman Lee Austin Woon
High 🔥
Availability
Non-functional
1

Set up access control between components
Leonard Tan Sherman Lee
High 🔥
Security
Non-functional
1

Set up pre-fetching of data (Cache)
Leonard Tan Ernest Khoo
Medium
Performance
Non-functional
1

Identify four security vulnerabilities for report

Ready for Report Write-up 9

Set up parallel execution to improve perf of system
Sherman Lee
Medium
Performance
Non-functional
1

Add lint check to Workflow
Austin Woon Sherman Lee

Set up CI-CD Pipeline —> Github Actions
Austin Woon Ernest Khoo
Sherman Lee
High 🔥
Maintanability
Non-functional

FR: Frontend (React)
Javier Wong

Set up failsafe (Load Balancer) for server software process or hardware or network failures.

Completed 0

+ New