Intro

Apple has given developers a method to receive more revenue from customers using InApps purchases. InApps allow customers very conveniently to purchase additional content for the game or in-game inventory.

From the moment InApps were introduced, a few hackers found a way how to spoof iOS device to think that any attempt to buy InApps product was a success. Apart from the lost revenue to the developer this behaviour has had a few more negative facets.

a) Any statistical analysis based on events which were fired right after successful InApps purchase are wrong. That is because pirated purchases tend to be done in large amounts and they skew the reported revenue by large margin.
b) Games that hold user balance in server side need to be able to verify the purchase before modifying any balances in database.

Apple has also given developers ways to verify whether a certain purchase is valid or not. It is done by posting a request to a certain Apple's web-server and reading a response indicating receipt validity.

Your task is to write a server that does this kind of a validation.

Architecture

The server component should read its input from a Amazon SQS service. The message in SQS will be in a following JSON format:

```
{ "receipt" : String,
  "user_id" : Number,
  "post_queue: String }
```

Where "receipt" is a base64 encoded Apple receipt and "user_ id" is a number representing a single user in our system. "post_queue" is a name of a SQS queue where result of the verification should be posted.

After retrieving a receipt from queue, the server must validate it using the Apple service.

After validation the server must verify if the receipt has been validated before. In order to do that server should call DynamoDB providing transaction_id as a query parameter. The database returns nothing if receipt has not been verified before and returns and error if validation process for this transaction has been performed already.

As a last step the component must post a JSON response to a SQS queue defined by "post_queue" value. Response to "post_queue" should be in JSON format:

```
{ "user_id" : Number,
  "transaction_id" : String,
  "status" : String }
```

Values of status can be: OK or INVALID.

Hints

* Erlcloud (https://github.com/erlcloud/erlcloud) is a good open source library that exposes the AWS api (SQS, Dynamo, etc)

Deliverables

* Server code written in Erlang using rebar3 (http://www.rebar3.org/) as build tool.
* Cloudformation template of the necessary AWS infrastructure (https://aws.amazon.com/cloudformation/)
* Time sheet with major subtasks and the time it took to complete them.

Discussion points

* What strategies would you employ should the number of validations increase hundred fold?
* What mechanisms would you put in place in order to improve fault tolerance of the service?
* Live failures/bugs will always occur, what measures would you take in order to decrease the time it takes to find issues?

Attachments

More info how about Amazon SQS API can be found at:

https://aws.amazon.com/documentation/sqs/

Information how to verify store receipts:

http://developer.apple.com/library/ios/#documentation/NetworkingInternet/Conceptual/StoreKitGuide/VerifyingStoreReceipts/VerifyingStoreReceipts.html

Sample receipt of an InApp made in sandbox environment:


```
ewoJInNpZ25hdHVyZSIgPSAiQWh0UkxxU0dlWHF3SEVNUmZkTk5hSWZBS2NSOHpSZ2tsd2ZpWkhLZWEvbkVJQVR5d
E8xbWhDOWkyVVlQdmQ3WnVQR1B2Y3JMdnd2aVl0NkwrU1FLeTJqcmRtVFhUU2txUnZGOXFla1VzenRoemlBQ0EyZn
RjQUhQUGQ4QTI4MFBmdzEydEdUenNzaHFPZUNqYkRIejlJbERydU5yWGRldVZZUXVVMU9HNDdFc0FBQURWekNDQTF
Nd2dnSTdvQU1DQVFJQ0NHVVVrVTNaV0FTMU1BMEdDU3FHU0liM0RRRUJCUVVBTUg4eEN6QUpCZ05WQkFZVEFsVlRN
Uk13RVFZRFZRUUtEQXBCY0hCc1pTQkpibU11TVNZd0pBWURWUVFMREIxQmNIQnNaU0JEWlhKMGFXWnBZMkYwYVc5d
UlFRjFkR2h2Y21sMGVURXpNREVHQTFVRUF3d3FRWEJ3YkdVZ2FWUjFibVZ6SUZOMGIzSmxJRU5sY25ScFptbGpZWF
JwYjI0Z1FYVjBhRzl5YVhSNU1CNFhEVEE1TURZeE5USXlNRFUxTmxvWERURTBNRFl4TkRJeU1EVTFObG93WkRFak1
DRUdBMVVFQXd3YVVIVnlZMmhoYzJWU1pXTmxhWEIwUTJWeWRHbG1hV05oZEdVeEd6QVpCZ05WQkFzTUVrRndjR3hs
SUdsVWRXNWxjeUJUZEc5eVpURVRNQkVHQTFVRUNnd0tRWEJ3YkdVZ1NXNWpMakVMTUFrR0ExVUVCaE1DVlZNd2daO
HdEUVlKS29aSWh2Y05BUUVCQlFBRGdZMEFNSUdKQW9HQkFNclJqRjJjdDRJclNkaVRDaGFJMGc4cHd2L2NtSHM4cC
9Sd1YvcnQvOTFYS1ZoTmw0WElCaW1LalFRTmZnSHNEczZ5anUrK0RyS0pFN3VLc3BoTWRkS1lmRkU1ckdYc0FkQkV
```

qQndSSX
hleFRldngzSExFRkdBdDFtb0t4NTA5ZGh4dGlJZERnSnYyWWFWczQ5QjB1SnZOZHk2U01xTk5MSHNETHpEUzlvWkh
BZ01CQUFHamNqQndNQXdHQTFVZEV3RUIvd1FDTUFBd0h3WURWUjBqQkJnd0ZvQVVOaDNvNHAyQzBnRVl0VEpyRHRk
REM1RllRem93RGdZRFZSMFBBUUgvQkFRREFnZUFNQjBHQTFVZERnUVdCQlNwZzRQeUdVakZQaEpYQ0JUTXphTittV
jhrOVRBUUJnb3Foa2lHOTJOa0JnVUJCQUlGQURBTkJna3Foa2lHOXcwQkFRVUZBQU9DQVFFQUVhU2JQanRtTjRDL0
lCM1FFcEszMlJ4YWNDRFhkVlhBZVZSZVM1RmFaeGMrdDg4cFFQOTNCaUF4dmRXLzNlVFNNR1k1RmJlQVlMM2V0cVA
1Z204d3JGb2pYMGlreVZSU3RRKy9BUTBLRWp0cUIwN2tMczlRVWU4Y3pSOFVHZmRNMUV1bVYvVWd2RGQ0TndOWXhM
UU1nNFdUUWZna1FRVnk4R1had1ZIZ2JFL1VDNlk3MDUzcEdYQms1MU5QTTN3b3hoZDNnU1JMdlhqK2xvSHNTdGNUR
XFlOXBCRHBtRzUrc2s0dHcrR0szR01lRU41LytlMVFUOW5wL0tsMW5qK2FCdzdDMHhzeTBiRm5hQWQxY1NTNnhkb3
J5L0NVdk02Z3RLc21uT09kcVRlc2JwMGJzOHNuNldxczBDOWRnY3hSSHVPTVoydG04bnBMVW03YXJnT1N6UT09Ijs
KCSJwdXJjaGFzZS1pbmZvIiA9ICJld29KSW05eWFXZHBibUZzTFhCMWNtTm9ZWE5sTFdSaGRHVXRjSE4wSWlBOUlD
SXlNREV5TFRBMExURTRJREExT2pBM09qRXpJRUZ0WlhKcFkyRXZURzl6WDBGdVoyVnNaWE1pT3dvSktaHZjM1JsW
kMxcFlYQXRkbVZ5YzJsd
mJpSWdQU0FpTVM0d0xqRWlPd29KSW05eWFXZHBibUZzTFhSeVlXNXpZV04wYVc5dUxXbGtJaUE5SUNJeE1EQXdNRE
F3TURReU9USTROVFkzSWpzS0NTSmlkbkp6SWlBOUlDSXhMakV1TVNJN0Nna2lkSEpoYm5OaFkzUnBiMjR0YVdRaUl
EMGdJakV3TURBd01EQXdOREk1TWpnMU5qY2lPd29KSW5GMVlXNTBhWFI1SWlBOUlDSXhJanNLQ1NKdmNtbG5hVzVo
YkMxd2RYSmphR0Z6WlMxa1lUmxMVzF6SWlBOUlDSXhNek0wTnpVd09ETXpNREF3SWpzS0NTSndjbTlrZFdOMExXb
GtJaUE5SUNKamIyMHViV2x1YVdOc2FYQXVZVzVwYldGc2MyaGxiSFJsY2k1amRYSnlaVzVqZVZCaFkyc3hJanNLQ1
NKcGRHVnRMV2xrSWlBOUlDSTBOekF5TVRneE9EZ2lPd29KSW1KcFpDSWdQU0FpWTI5dExtMXBibWxqYkdsd0xtRnV
hVzFoYkhOb1pXeDBaWElpT3dvSkluQjFjbU5vWVhObExXUmhkR1V0YlhNaUlEMGdJakV6TXpRM05UQTRNek13TURB
aU93b0pJbkIxY21Ob1lYTmxMV1JoZEdVaUlEMGdJakl3TVRJdE1EUXRNVGdnTVRJNk1EYzZNVE1nUlhSakwwZE5WQ
0k3Q2draWNIVnlZMmhoYzJVdFpHRjBaUzF3YzNRaUlEMGdJakl3TVRJdE1EUXRNVGdnTURVNk1EYzZNVE1nUVcxbG
NtbGpZUzlNYjNOZlFXNW5aV3hsY3lJN0Nna2liM0pwWjJsdVlXd3RjSFZ5WTJoaGMyVXRaR0YwWlNJZ1BTQWlNakF
4TWkwd05DMHhPQ0F4TWpvd056b3hNeUJGZEdNdlIwMVVJanNLZlE9PSI7CgkiZW52aXJvbm1lbnQiID0gIlNhbmRi
b3giOwoJInBvZCIgPSAi MTAwIjsKCSJzaWduaW5nLXN0YXR1cyIgPSAiMCI7Cn0=