Solution:

While designing the solution for our recommendation engine we opted for AWS services as our backbone for planning the architecture for our recommendation engine as it gives following:

Our recommendation engine’s compute solution needed to be robust enough to scale with growing data. Whether that’s in compute power, memory capacity, or both.

The compute solution needed to be cost-effective.

The compute solution needed a good monitoring tool for maintaining efficient distribution of load.

We needed to be able to store user recommendations somewhere for quick retrieval through an API

As AWS is good at transferring data between their services. It is secure, scalable and services can be easily managed.

We are processing data from RDS using Machine Learning scripting language Python and put the recommendation in Dynamo DB. It is AWS’ managed NoSQL database. It works well with AWS API Gateway, and used storage and access to the generated predictions for a particular user. That’s why, we decided to go with DynamoDB. AWS services make the execution and operations of such a recommendation system cost-effective and straightforward.

There are different set of recommendations which can be derived using Product Attributes, User Demographics and so on. We are here presenting recommendation engine produces two sets of recommendations: user-item (personalized recommendations for individual users) and item-item(Customer who Bought This). Same will be explained using Visuals analytics post architectural discussion

I’ll now walk you through a big-picture overview of how we got to a production-ready recommendation engine.

**Architechture Diagram for The recommendation Module**

