HBO-ICT Graduation Thesis

Personalized Recommendation System

Author: Vadim Savin

Student number: 450491

Company supervisor: Adrian Garcia Ramos

University supervisor: René van den Nieuwenhoff

Table of content

Table of figures

Version History

Version number	Delivery date	Content
V0.1	22.03.2021	- Introduction chapter - Problem statement chapter - Case analysis chapter - Governance chapter
V0.2	19.04.2021	- Company logo on front page - Reading guide in introduction - Cleaned “current situation” from extra details - Reformulated the research questions - Added an intro paragraph to all chapters - Mapped the stakeholders - Added project phases chapter
V0.3	04.05.2021	- Added Moscow priorities to requirements - Project phases chapter was updated based on feedback - Project management chapter was simplified based on feedback. - Added research methods subchapter under Governance - Added research result chapter
V0.4	18.05.2021	- Split research chapter in subchapters based on research questions
V0.5	30.05.2021	- Added Design, Realisation, Validation and Conclusion chapter

		- Added Acknowledgments and Abstract chapters - Polished table of contents, table of figures, terminology

Acknowledgements

Throughout the writing of this thesis, I received a lot of support and help.

First of all, I would like to thank my university supervisor, René van den Nieuwenhoff, for guiding me throughout the whole process. Mr. René has helped me structure this thesis document and provided very constructive feedback that made this thesis improve over time. This was the first time I enjoyed the documentation process of the project, because of the clarity and support received from Mr. René.

I would also like to thank my colleagues at Fitenium for help and collaboration. Special thanks to my company supervisor, Adrian Ramos, for help, support and trust.

Abstract

The aim of this thesis is to identify and implement the best personalized recommendation system that will increase the engagement of Fitenium users. Based on the research, it was determined that the best personalized recommendation system for this use case is Amazon Personalize.

Amazon Personalize makes it easy for developers to create personalized recommendation systems. The recommendation system was designed to be integrated in the existing architecture of the Fitenium project. With the help of Lambda functions and Step Functions, an automatic workflow of retraining and deploying the machine learning model was designed.

The design was implemented by developing all the individual Lambda functions and creating the state machine that represents the workflow for retraining the model. The infrastructure was created using the Infrastructure as Code paradigm, with the help of AWS CloudFormation and Serverless Application Model.

The solution was deployed and tested in production mode. Analytical data showed that the new personalized recommendation algorithm improved the conversion of the following recommendations to 4.7%, which is almost 10 times more compared to the previous heuristic solution.

Terminology

API	Application Programming Interface
ASL	Amazon States Language
AWS	Amazon Web Services
CLI	Command Line Interface
CSV	Comma separated values
ETL	Extract Transform Load
FAANG	Facebook, Apple, Amazon, Netflix, Google
GDPR	General Data Protection Regulation
IaC	Infrastructure as Code
JSON	JavaScript Object Notation
ML	Machine Learning
RS	Recommendation System
S3	Simple Storage Solution
SAM	Serverless Application Model
SDK	Software development kit
TPS	Transactions per second
UI	User Interface
VCS	Version Control System

1. Introduction

Personalized Recommendation Systems are a very hot topic of machine learning nowadays. A lot of companies, especially FAANG companies are investing a lot of resources in this field, which led to a lot of advancements. The reason why recommendation systems are so important for companies selling or serving content online, is that it directly affects the bottom line and the success of the company.

The importance of personalized recommendation systems can be seen in the success stories of some of the most popular applications. 75% of the content people watch today on Netflix is provided by their recommendation system. Tiktok became the top downloaded app by hooking users into the app with one of the best recommendation systems. Youtube is relying mostly on the personalized recommendation system to feed content to the users, compared to a subscription approach they had a couple of years ago.

Fitenium can also benefit from a personalized recommendation system for the mobile application.

During the graduation assignment, different recommendation systems are researched and based on that, the company is advised on the best one in the context of the application requirements. After the research, the next step is to design and implement the recommendation system in the application. The system will recommend various content types, such as users to follow, videos to watch and training programs to follow.

This document will be split into the next chapters:

2. Problem statement: the problem that has to be solved is introduced and described in the context of the company. The scope of the assignment is defined here, as well as the current and desired situation.

3. Case analysis: knowing the problem, this chapter introduces the existing stakeholders, requirements (functional and non-functional) and deliverables.

4 Governance: this chapter contains information about the structures and processes used to ensure accountability, predictability and transparency during the project.

5. Research: this chapter includes the summary of the research document, with the answers to the research questions.

6. Design chapter includes the high level design of the solution and will describe the reasoning behind the choices made.

7. Realisation chapter describes the realisation process, the coding part and the low level details of the solution.

8. Validation chapter is describing the methods used to validate the solution and the results of it.

2. Problem Statement

Context

Fitenium is a Spanish based startup in the sports industry. Fitenium helps athletes to achieve their training goals using video technology in a combination with a social network aspect. People training at the gym can record their personal records and share them with their community in Fitenium mobile application, they can compete with their friends in challenges and win prizes, they can find or create training programs specifically for their needs and much more.

Fitenium targets the sports industry, which is a huge industry and it contains a lot of sub-niches. Because of that, the users in Fitenium are diverse and can be categorized in 3 big groups: powerlifting, hypertrophy and calisthenics.

- **Powerlifting** is a strength sport that consists of three attempts at maximal weight on three lifts: squat, bench press, and deadlift.
- **Hypertrophy** training focuses on the goal of increasing muscle size
- **Calisthenics** uses your *bodyweight* and involves compound exercises

These 3 categories of people are very different, and they have different interests. In some extreme cases, they can't even stand each other. A powerlifter that can lift a car, will laugh at a 60kg person training Calisthenics that is doing 10 pull ups. This was just an example showing how different some of our users are.

With a diverse community comes a great challenge for the community builders, which is Fitenium. Users should be treated individually, based on their personal preferences. They should be served content that they are interested in. They should be assisted in the process of building their own network of people with the same interest as theirs.

Current situation

Fitenium has a very primitive, heuristic recommendation system that recommends users that you might be interested in following. Here we can see the current widget:

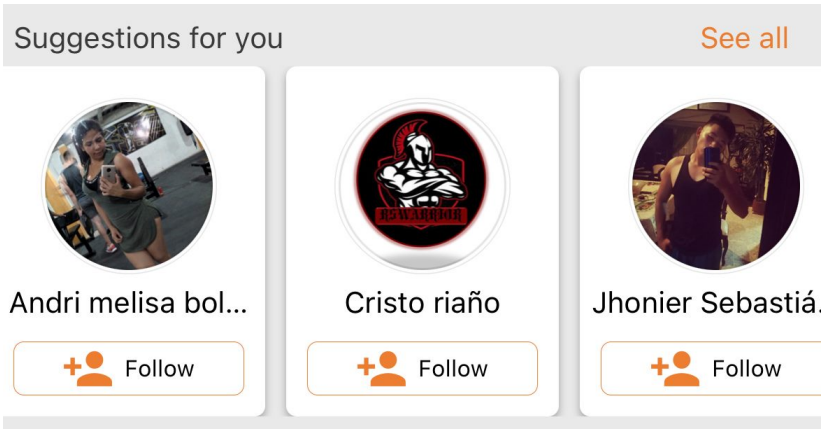


Fig. 1 Current user suggestion widget

The system is very basic and it just scores users based on general parameters (age, country, gym) and that's why it performs very poorly. In the screenshot below, we can see that for ~4000 follow suggestion widget displays, there were only 20 actions (follows), which is 0.5% conversion rate.

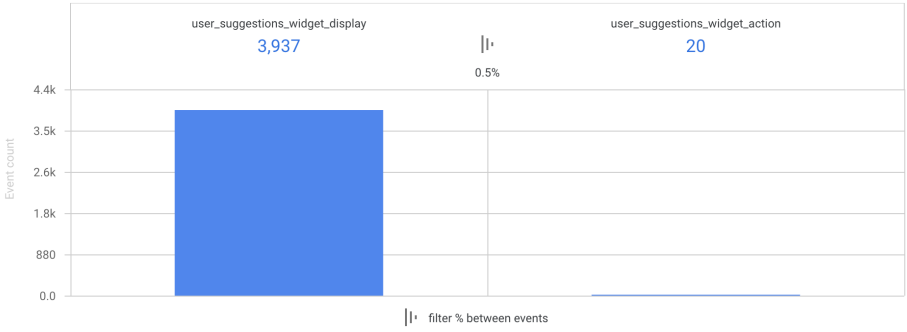


Fig. 2 Bar diagram of nr. of widget displays vs widget actions

For content recommendation or training program recommendation, there is currently nothing implemented.

Desired situation

Fitenium wants to implement a personalized recommendation system that will recommend users to follow, content to consume and training programs to train.

The goal of the user recommendation system will be to increase the current “follow suggestions” widget performance to at least 3% conversion rate. This will increase the amount of connections in the whole network, and will grow the number of people that a user follows on average.

For the content recommendation system, fitenium wants to increase the engagement rate and the amount of time a user spends in the app by serving the user the best content for him.

For the training program recommendation system, fitenium wants to maximize the revenue. Training programs are the monetization strategy of Fitenium, and by recommending more personalized training programs to the users, we increase the chance that a user will buy the PRO version, which will directly impact the revenue for Fitenium.

Scope

The assignment will cover the research and implementation of the 3 personalized recommendation systems, as described above: users, content and training programs. The assignment does not include manual data gathering in case some data is missing. The company will provide the necessary data, and will provide assistance in data gathering if needed.

Research questions

The research will focus on answering the main question: What is the best personalized recommendation system to increase the user engagement of Fitenium?

To answer the main question, first of all, the next sub questions about recommendations system should be answered.:

- What are Recommendation Systems?
- What types of recommendation systems do exist?

With this information about recommendation systems, it will be possible to research and answer some specific questions to the assignment context. The answers to the next questions will help design and model the recommendation system, specifically for Fitenium use-case.

- How to design a scalable solution for recommendation systems that will fit in Fitenium's architecture?
- What data do we need to implement a specific recommendation system in the context of Fitenium use-case?

3. Case analysis

Having a clear understanding of the problem, this chapter will focus on what has to be built. It will introduce the stakeholders, which are the important parties to be taken into consideration when defining the requirements and carrying out the project. Also, it will define the requirements, both functional and non-functional. Lastly, it will describe the deliverables in terms of documentation, code and other artifacts.

Stakeholders

Speaking with the founders about all the parties involved, their influence and their power, the next stakeholders have been defined:

- **Customers** are the most important stakeholders. They are directly impacted by the quality and the value of the application.
- **Founders** have invested their time and resources to start this project.
- **Employees** are the next stakeholders. They believed in the vision of the company, and chose to invest their time in this project.
- **Sport communities** are another stakeholder. They have a huge influence in the life of Fitenium's customers because they organize tournaments, gatherings and other sport related events.
- **Government** is also an important stakeholder, as it has the power to regulate the sport industry, data protection laws (ex: GDPR).

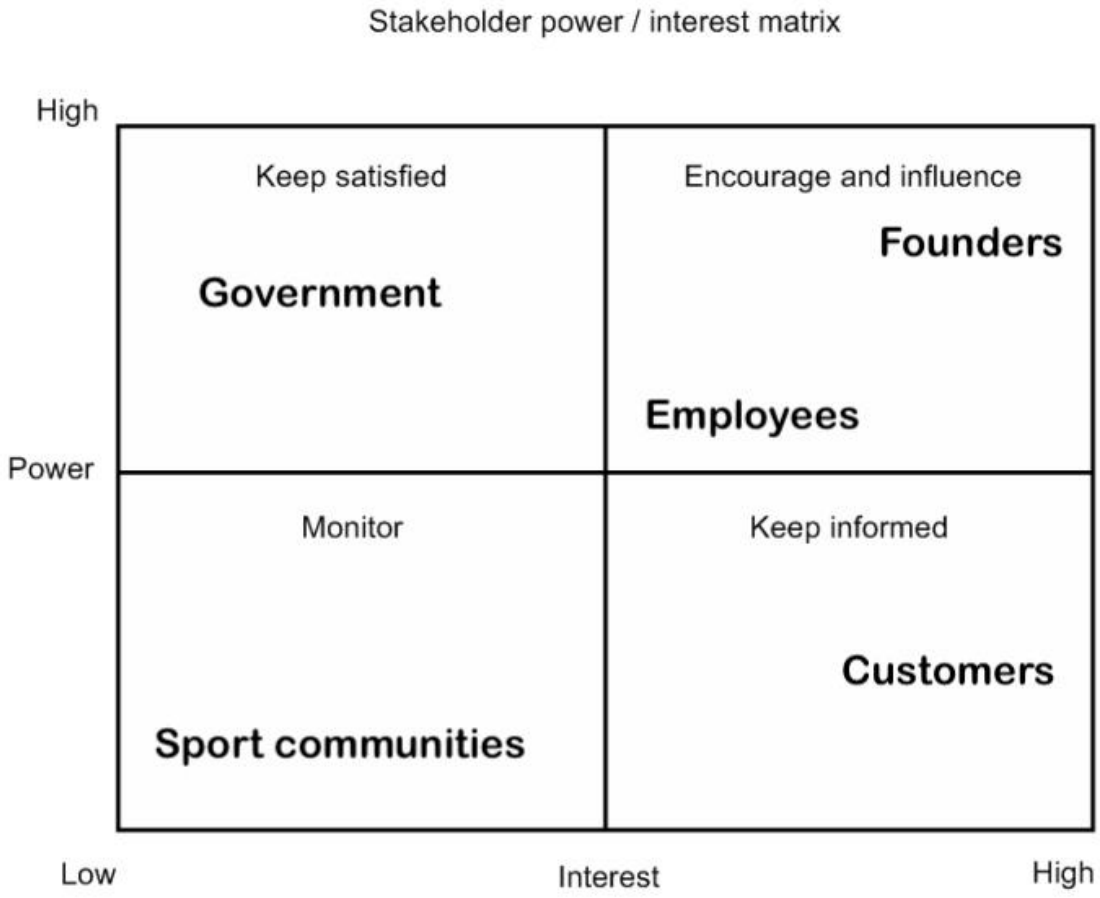


Fig. 3 Stakeholder Matrix (Power / Interest)

Requirements

The next requirements are the result of multiple interviews and meetings with the founders and other Fitenium stakeholders. All the information and ideas were gathered from the founders, as well as technical details about the system, architecture and existing data from other employees. With this information, it was decided on what is feasible to be done in the timeframe of the project, and what are the priorities of each requirement.

Functional requirements

1. [M] Users must see personalized recommendations of other users to follow, while browsing the app
2. [M] When browsing the training program directory, users must see the training programs sorted based on how well it matches the user's needs.
3. [M] Once a week, the user must receive an in-app notification, and a push notification, of the best training program that the system found for him
4. [C] Once a week, all the user’s that do not currently have an active training program, could receive an email with 3 recommended training programs to join.
5. [M] In the posts discovery screen, users must see a feed with recommended posts to see.
6. [S] The app should track events about user interactions with the recommended content, so it will be possible to track and improve the system.

Non-functional requirements

1. [S] Scalability - The tool should support a user load of at least 100 concurrent users
2. [S] Latency - for any user request to the recommendation system should be less than 1 second
3. [M] Fault Tolerance – Tool must be available 24/7
4. [S] Monitoring and Alarm – Dashboard for health metrics and alarms should be set up so any problems with scalability, latency or fault tolerance are quickly addressed

Deliverables

By the end of the assignment, the following deliverables will be provided:

1. Research report about recommendation algorithms, including the suggested algorithm for Fitenium use-case
2. Code for the recommendation system and for the mobile app integration

3. Design document for all new system
4. Running environments
5. Dashboards and alarms configured

4. Governance

The assignment will be carried out in a professional environment and for that reason, clear processes and structures must be defined from the beginning.

This chapter will describe the project phases that will be carried out. This will make the project more predictable and trackable for all the parties involved.

It will also describe the project management methodologies and tools that will be used throughout the realization of the project.

Project Phases

The project is divided into 5 phases in order to be more trackable. Every phase has its own deliverables and at the end of the phase, during a sprint retrospective, the deliverables are presented and discussed with the rest of the team, including the company supervisor.

Introduction

The **introduction phase** will start with getting to know the company, the team, the codebase and the tools that are used in the project. It will follow with the problem investigation. The requirements, stakeholders and other important aspects about the project will be gathered through different methods such as meetings with the managers and interviews with the employees. During this phase, the priority will be to gather enough information in order to be able to draw a plan of approach for the project. The plan of approach will be deliverable of the first phase of the project, and it will be shared and approved with both company and university supervisors.

Research

The second phase will be the **research.** It will start with an in-depth research about the recommendations systems, about different types of recommendation systems, their pros and cons and the use-case they are performing best. The

deliverables of this phase will be the research report and advisory report on the best suited recommendation system to be implemented for each type of content (users, training programs, posts).

Proof of concept

During this phase, the data that will be needed in order to train and test the system will be gathered. With this data, a proof of concept system will be developed. At this stage, the focus will be on creating and training the model. We will not focus on non-functional requirements such as scalability, latency, availability etc. A python notebook will suffice. The deliverables of this phase will be the proof of concept itself, which will be presented to the whole team during a demo day.

Implementation

After the model and the logic of the recommendation system is finished, it will be integrated with the application backend. At this stage, the focus will be on meeting the non-functional requirements, making the system scalable, fast and fault-tolerant. The next step will be to integrate the mobile application with the backend of the recommendation system, and visualize the recommendations in the UI of the app. The deliverables of this phase will be the code, infrastructure and code documentation for the recommendation system.

Deployment and Monitoring

After everything is ready and has been tested thoroughly, the system will be deployed to production, to be used by the real users. Then, the system will be monitored in order to measure the performance of the model. The parameters of the model will be fine tuned while analyzing their impact on the users. If it is needed, A/B testing will be integrated to compare different parameters or algorithms, and decide which one performs better.

Research Methods

During the research phase, different research methodologies will be applied. The most important methodology will be literature research. There are a lot of research papers and literature on the general topics of recommendation systems and their different types. This will help answer the first 3 research questions:

- What are Recommendation Systems?
- What types of recommendation systems do exist?
- What data do we need to implement a specific recommendation system in the context of Fitenium use-case?

For the last research question “How to design a scalable solution for recommendation systems that will fit in Fitenium’s architecture”, literature research will not be enough. To properly answer this question, more contextual information will be needed regarding the existing Fitenium architecture. To gather this information, interviews will be carried out with the employees that know more about this.

Another important research method is building a proof of concept. This will answer a lot of technical specific questions and will clear any doubts. By building a proof of concept, any roadblocks or limitations will be found early, which will allow the team to adjust the plan accordingly.

Project Management

The project will be carried out using the Scrum agile methodology. All the main Scrum ceremonies will be practiced. There will be daily standups at the beginning of each working day with the development team and the supervisor.

The sprints are 2 weeks long and start with a planning meeting. The sprint ends with a retrospective meeting, where the tasks that have been finished during the previous sprint are analyzed and the most interesting features are presented during a demo. It is also discussed what went well during the previous sprint and what can be improved during the next ones.

5. Research

The research phase was focused on answering the main research question and the subquestions, which were introduced in previous chapters. This chapter will present a summary with the findings and answers to all the research questions. The full process and information can be found in the research document.

What are Recommendation Systems?

This is a very broad and general question, but a very important one. To design and implement a good recommendation system, first of all we should understand what recommendation systems are in general, how they are implemented and what are the success stories from companies that are using recommendation systems. To answer this question, a lot of literature research was conducted.

It was found that a recommendation system is an application of machine learning that provides recommendations to users on what they might like based on their historical preferences. In other words, recommendation systems are algorithms responsible for suggesting “the best” items to users, based on what is already known about the user. The items that are recommended vary from use-case to use-case. Some common examples are movies or videos to watch (Netflix, Youtube), user-generated content to consume (Instagram, TikTok), products that you might be interested to buy (Amazon), ads relevant to you (Facebook, Google).

Recommendation systems are really critical in some industries and they can generate a significant amount of income if implemented efficiently. In some cases, such as TikTok, it also serves as a differentiating factor from competitors, which contributed to the huge success of the platform. This is the reason why the field of recommendation systems is so hot at the moment and also why the big companies are investing so much in it. Netflix organised a challenge (the “Netflix prize”) where the goal was to produce a recommender system that performs better than its own algorithm with a prize of 1

million dollars to win [1]. Also, 75% of the content people watch today on Netflix is provided by their recommendation system.

The huge investments in the research and development of recommendation systems made this field rich in solutions and techniques for different use-cases. The next chapters will focus on different types of recommendation systems, the use-cases that they are designed for, their advantages and disadvantages.

What types of recommendation systems do exist?

There are six main types of recommendation systems (RS): Collaborative RS, Content-based RS, Demographic based RS, Utility based RS, Knowledge based RS and Hybrid RS.

Collaborative Filtering (CF) is the process of filtering or evaluating items using the opinions of other people. The underlying assumption of the collaborative filtering approach is that if person A has the same opinion as person B on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person. The greatest strength of collaborative techniques is that they are completely independent of any machine-readable representation of the objects being recommended and work well for complex objects where variations in taste are responsible for much of the variation in preferences.

Content-based (CBRS) filtering uses item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback. This system creates a user profile, based on the features of the items the user liked. Having the user profile, it matches the items, represented by their features, with the user profile represented by the same features. The challenge of creating a well performing CBRS is feature engineering. What are the important features of an item, and how important are they when it comes to recommending it. The advantage of CBRS is that it does not require a lot of users. A CBRS can be built having a single user, because the only thing it takes into consideration is the user’s actions. Content based RS is adaptive, meaning its quality improves over time.

Demographic Recommender system aims to focus on the users' demographic profile such as age, sex, education, occupation, locality, etc. It generally uses clustering techniques to categorise target users according to demographic information. The benefit of a demographic approach is that it does not require a history of user ratings like that in collaborative and content based recommender systems. However, if the demographic attributes remain unchanged, the user will receive the same recommendations for the same set of items. Therefore, they might miss some new and worthwhile recommendations.

Knowledge based RS attempts to recommend items on the basis of a predefined set of constraints (rules) and/or similarity metrics. The major difference compared to the approaches of collaborative filtering and content based filtering is that in knowledge based recommender deep knowledge about the product catalogue is needed in order to be able to determine recommendations. Though KBRS is capable of providing the required information that cannot be achieved through the conventional approaches, the knowledge modelling and handling techniques in KBRSs are comparatively the most expensive.

Utility based recommender systems make suggestions based on computation of the utility of each object for the user. Of course, the central problem for this type of system is how to create a utility for individual users. The main advantage of using a utility based recommender system is that it can factor non-product attributes, such as vendor reliability and product availability, into the utility computation. This makes it possible to check real time inventory of the object and display it to the user.

Combining any of the two systems in a manner that suits a particular industry is known as **Hybrid Recommender system**. This is the most complex recommender system, as it combines the strengths of more than two recommender systems and also eliminates any weakness which exists when only one recommender system is used.

After analysing the strengths and weaknesses of all six types of Recommendation systems: collaborative, content-based, demographic based, utility based, knowledge based and hybrid; it was decided that the best fit for the Fitenium use-case is a hybrid system using collaborative filtering and content based filtering.

The collaborative filtering method depends on minimal data. It depends on the data of the interactions between users and items. Providing this interaction information, the model will optimize itself to maximize these interactions (following, liking, and training), which is exactly what the goal is.

Content based recommendations system can benefit the use case by taking into consideration specific details about items and users. Metadata about users, such as their age, gender, and most importantly, the strength category (powerlifting, calisthenics, hypertrophy) will be gathered and used for the model training. The same goes about items to be recommended. The training programs are also categorized in complexity levels, strength category, equipment needed and more. With this data, the model can learn to recommend powerlifting training programs to powerlifting users.

The Hybrid approach using these 2 methods will give the best result. The system will not be biased, and the pros and cons of these 2 systems will balance out.

How to design a scalable solution for recommendation systems that will fit in Fitenium's architecture?

To answer this question, a good understanding of the existing architecture was needed. For that, multiple interviews were conducted with employees who are responsible for the high level architecture.

Deploying a scalable and performant Recommendation System is quite challenging, and can be very expensive. When designing the architecture of the recommendation system, all the steps in the pipeline should be taken into consideration, starting with data gathering, data storage, model training, model deployment and operation. Designing, developing and maintaining a DIY infrastructure for a scalable and performant recommendation system will require more time and resources than developing the model for the system.

For the above mentioned reasons, and taking into consideration the scalability, performance, complexity and expenses, it was decided that a 3rd party recommendation engine will be used - Amazon Personalize. This Amazon Web Service, which behind the scenes uses a mix of collaboration and content based algorithms, is a fully managed machine learning service that trains, tunes, and deploys ML models. Developing a customizable recommendation system using Amazon Personalize will ensure highly personalized and secure recommendations for Fitenium users.

Fitenium is currently using a variety of AWS services for the backend architecture. From servers, databases, storage for videos, media processing and more. This means that using another AWS service for the recommendation system fits very well in the current architecture.

What data do we need in order to integrate Amazon Personalize?

The main resource used to answer this question was the official documentation of Amazon Personalize. However, that was not enough to clear all doubts about the data requirements. Specifically, it was still not clear how the optional datasets (items

dataset and user metadata) impact the performance of the model. That is why the proof of concept was also needed. After the research, it was found that, in order to train an optimal model, we will need to gather and provide the next data [2]:

- **User metadata** - this dataset stores metadata about users. This might include information such as age, gender, or loyalty membership, which can be important signals in personalization systems.
- **Items** – this dataset stores metadata about items. This might include information such as price, SKU type, or availability.
- **Interactions** – This dataset stores historical and real-time data from interactions between users and items. This data can include impressions data and contextual metadata on your user's browsing context, such as their location or device (mobile, tablet, desktop, and so on). We must at minimum create an Interactions dataset.

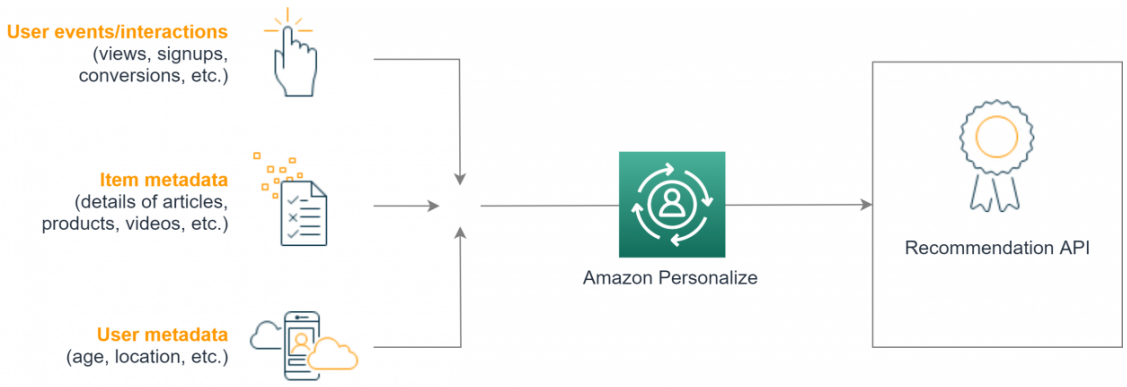


Fig. 4: Amazon Personalize Datasets

6. Design

After the problem is well defined, the requirements gathered and the research done, it is time to design the solution. During the research phase, it was decided that Amazon Personalize will be used for the recommendation engine. While Amazon Personalize manages a lot of things under the hood such as training, optimizing and hosting the machine learning models, the rest of the steps should be designed and developed customly. Here we talk about data collection, data preparation, choosing a model, querying and managing predictions. In the below diagram, we can see the 7 steps of a machine learning process [3], which ones are managed by Amazon Personalize and which ones should be designed and deployed separately.

Fig. 5 7 steps of machine learning

The next chapters will introduce all the services and technologies that are going to be used in the final architecture. We are going to find out what these technologies are, how they work and why they are used in the final architecture.

Microservices

Microservices - also known as the micro-service architecture [5] - is an architectural style that structures an application as a collection of services that are:

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities

The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack.

Choosing the micro-services architecture was a natural choice, because most of the software built at Fitenium is designed as microservices. Moreover, Amazon Personalize is a managed micro-service as well, and it will be easier to integrate it in a microservice architecture than into a monolith one.

Serverless

Serverless computing is a method of providing backend services on an as-used basis [5]. A serverless architecture allows users to write and deploy code without the hassle of worrying about the underlying infrastructure. A company that gets backend services from a serverless vendor is charged based on their computation and does not have to reserve and pay for a fixed amount of bandwidth or number of servers, as the service is auto-scaling. Note that although called serverless, physical servers are still used but developers do not need to be aware of them.

These benefits are valuable to the Fitenium use case. As Fitenium is a small startup, the pay as you go model is the best option, because there are no upfront costs for servers. The underlying services will autoscale to meet the demand of the application. Moreover, the time that will be saved on not doing server maintenance, is the time that developers will be able to invest into business critical features, which is a major priority of the startup.

Amazon Personalize

The central piece of the architecture will be the Amazon Personalize Service. This is a fully managed machine learning service that trains, tunes, and deploys custom ML models. When using Amazon Personalize, the user/developer should only provide the input dataset in a S3 file, set up the model settings and hyperparameters, and after

Amazon Personalize trains the model, the user can get recommendations using the Customized personalization API [6].

More about the reasons why Amazon Personalize was chosen can be found in the research document.

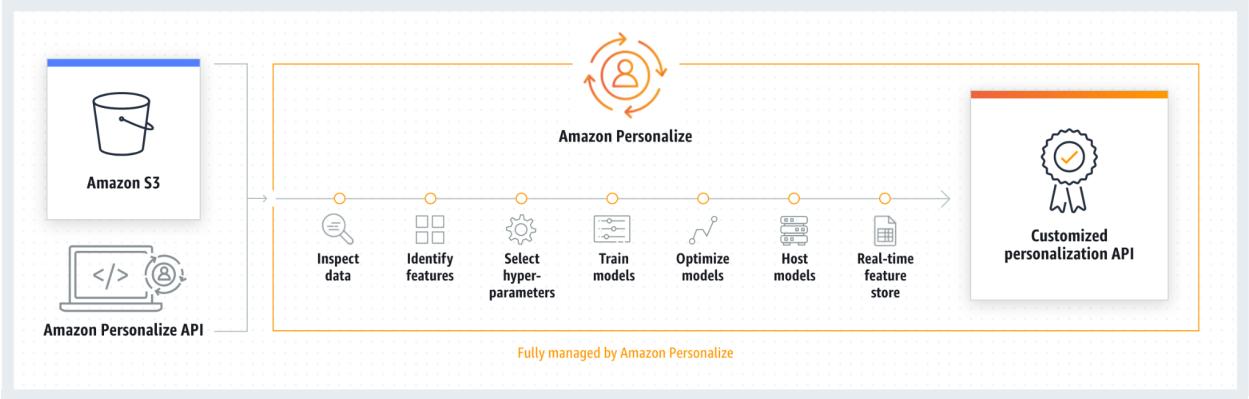


Fig. 6 How Amazon Personalize works

AWS Lambda functions

AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers [7]. AWS Lambda executes the code only when needed and scales automatically, from a few requests per day to thousands per second. It charges only for the compute time that is consumed. AWS Lambda runs the code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging. All the developers need to do is supply the code and the rest is taken care of by AWS.

For the above mentioned reasons, AWS Lambda functions will be used for all the compute tasks in our final solution.

AWS Step Functions

AWS Step Functions is a serverless function orchestrator that makes it easy to sequence AWS Lambda functions and multiple AWS services into business-critical

applications [8]. The output of one step acts as an input to the next. Each step in the application executes in order, as defined by the business logic.

Orchestrating a series of individual serverless applications, managing retries, and debugging failures can be challenging. Because there are multiple steps in the machine learning process, each handled by individual functions, the complexity of managing them also grows. With its built-in operational controls, Step Functions manages sequencing, error handling, retry logic, and state, removing a significant operational burden from the development team.

AWS S3

Amazon Simple Storage Service, widely known as Amazon S3, is a highly scalable, fast, and durable solution for object-level storage of any data type [9]. Unlike the operating systems we are all used to, Amazon S3 does not store files in a file system, instead it stores files as objects. Object Storage allows users to upload files, videos, and documents like you were to upload files, videos, and documents to popular cloud storage products like Dropbox and Google Drive. This makes Amazon S3 very flexible and platform agnostic.

AWS S3 will be used as a storage solution for the datasets that Amazon personalize will use to train the model. The datasets will be CSV and JSON files containing information about users, items and interactions.

High Level Design

In order to have an always up-to-date machine learning model, it should be retrained periodically with the newest user data. Users are humans, and their preferences are in constant change. That is why it is important to retrain the model periodically.

The process of retraining the model contains multiple steps. From a high-level, these are the main steps of retraining the model:

- Collect data from original data sources (ex: database)

- Transform the data
- Import the dataset to Amazon Personalize
- Trigger Amazon Personalize to retrain the model
- Create a batch recommendation job and get recommendation for all users
- Parse user recommendation and write it to database

These steps are executed by individual AWS Lambda functions. The lambda functions are orchestrated by AWS Step Function which helps define the data flow, the order of execution and parallelization.

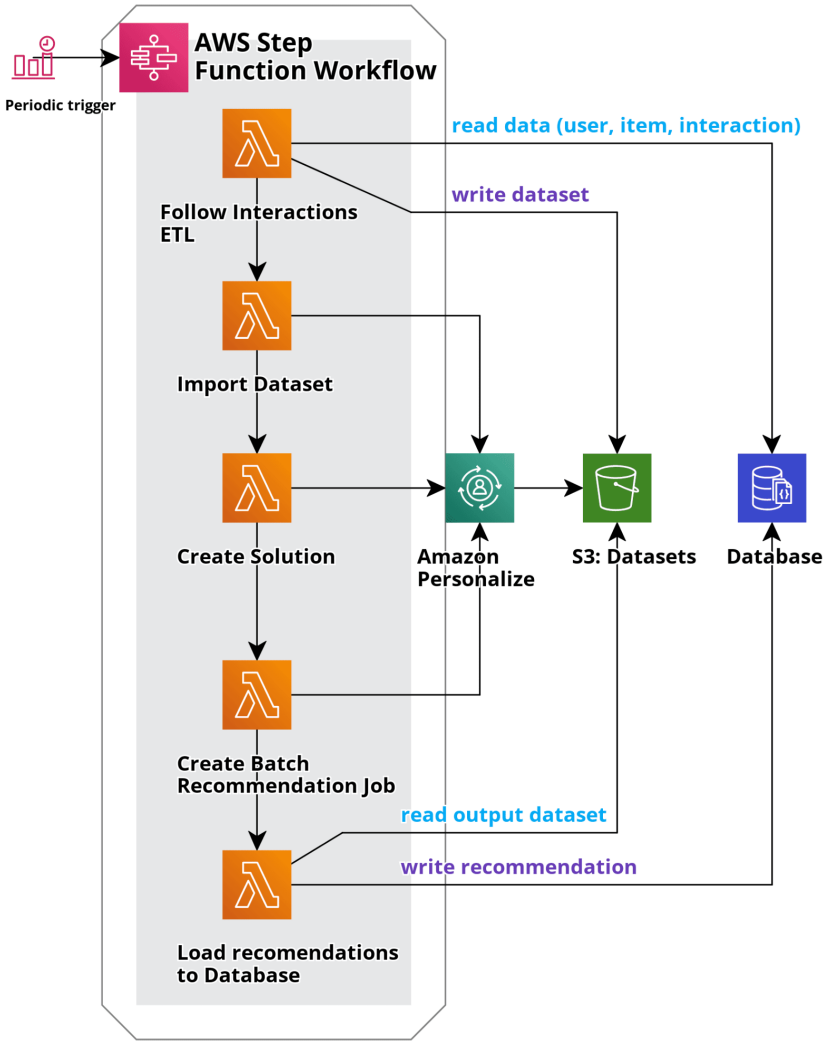


Fig. 7 High Level Architecture Diagram

The Step function workflow starts with the customly coded ETL Lambda Functions. The goal of these ETL Lambda function will be:

- (E - Extract) gather all the data about users, items and interactions that will be needed to train the model (the data will be read from the application database)
- (T - Transform) transform the data from original form, to the form that Amazon Personalize expects, based on the service documentation.
- (L - Load) write the data to an S3 file, that Amazon Personalize can use

The next Lambda functions in the workflow will invoke Amazon Personalize through the API. First of all it will trigger Amazon Personalize to re-import the datasets from Amazon S3 files generated in the previous step. After that, it will invoke Amazon Personalize to re-create the solution, which means to retrain the machine learning model with the fresh data.

The next step is to create a Batch Inference Job in Amazon Personalize. The batch job expects an input, which will be a list of users that we want to generate recommendations for. The output of the job is a list of users with their recommendations. The output is automatically written to an S3 file.

The last Lambda function from the workflow, reads the S3 file generated by the Batch Inference Job, parses it and then writes the results back to the database so that it can be served to the users that are browsing the app.

The whole Step Function workflow will be automatically triggered periodically. Taking in consideration the cost - performance balance, the recommended interval for Fitenium use-case is 24 hours. This will allow the model to be kept up to date with users' preferences in order to recommend accurate items.

Cost analysis

Cost analysis is a very important and sensitive step for a startup, and sometimes can be a breaking point for some decisions. The cost will be estimated for the current moment, when Fitenium has 13.000 users.

In the designed architecture, the cost will depend on the next services: Amazon Personalize, Lambda functions and Step Functions.

Lambda functions and Step functions pricing

For Lambda functions and Step function, the oficial AWS calculator was used to estimate the costs. Because the lambda functions will be executed only when retraining the model, the cost for it does not depend on the number of users, but rather on the number of times the model is retained per month. These 2 services have a free tier and start charging only after the free tier is consumed. Based on the estimated calculations, the usage of both services will be way lower than the free tier, which means that the cost will be 0. Below you can find the detailed calculations.

Pricing calculations for Lambda functions [10]:

- Each workflow execution invokes around 10 lambda functions
- The workflow will be executed daily (30 times a months)
- Lambda functions will use 128 MB RAM memory
- The average execution time, based on the tests performed, is 2 minutes
- 300 requests x 120,000 ms x 0.001 ms to sec conversion factor = 36,000.00 total compute (seconds)
- 0.125 GB x 36,000.00 seconds = 4,500.00 total compute (GB-s)
- 4,500.00 GB-s - 400000 free tier GB-s = -395,500.00 GB-s
- Max (-395500.00 GB-s, 0) = 0.00 total billable GB-s
- 300 requests - 1000000 free tier requests = -999,700 monthly billable requests
- Max (-999700 monthly billable requests, 0) = 0.00 total monthly billable requests
- Lambda costs - With Free Tier (monthly): 0.00 USD

Pricing calculations for Step functions [11]:

- 50 state transitions per workflow x 30 workflow requests = 1,500.00 state transitions per month

- 1,500.00 state transitions per month - 4000 Free Tier state transitions = -2,500.00 billable state transitions
- Max (-2500.00 billable state transitions, 0) = 0.00 billable state transitions per month
- 0.00 billable state transitions x 0.000025 USD per state transition = 0.00 USD per month
- Standard Workflows pricing (monthly): 0.00 USD

Amazon Personalize Pricing

Amazon Personalize charges only for resources that are being used, and there is no upfront fee. The charges are split in 3 categories: data ingestions, training, recommendations (inference) [12].

For **data ingestion** we are charged for per GB of data uploaded to Amazon Personalize. The price, at the time of writing this report, is **$0.05 per GB.**

For **training** we are charged for training hours consumed to train the machine learning model. The price for a training hour is **$0.24**.

Recommendations (inference) are the actual personalized recommendations that we are requesting from Amazon personalize for each user. There are 2 types of inferences, and they are both priced differently.

First type is **real-time recommendations**, where amazon personalize will provide an API endpoint that developers can query on demand to get recommendations for individual users and requests. Because these are real time recommendations, behind the scenes the backend infrastructure of Amazon Personalize is always running and always ready to give these real time recommendations. The real time recommendations are measured in transactions per seconds (TPS), and the minimum possible value is 1 TPS. The price for 1 TPS hour is **$0.2.**

The second type of recommendations are **batch recommendations**. This is an asynchronous method of getting recommendations for multiple users at once. With batch recommendation, AWS charges for the number of users processed, and the price is **$0.067 for 1000** recommendations.

Conclusion

Below we can see the detailed calculations for using Amazon Personalize. As we can see, the real time recommendations option has a very steep cost, because of the minimum 1TPS requirement. Even though Fitenium is still a small startup, with not a lot of users, it would need to use this minimum 1TPS. This adds $165 to the total costs. On the other hand, the Batch recommendations have a more linear cost, and as of today, Fitenium would have to pay an extra $47.

Because the price is a sensitive topic for the company, it is advised to start with batch recommendations at the moment and switch to real time recommendations when there will be more 100k users.

		Now (13k users)		Estimaated for 100k users		Estimaated for 1M users	
	Price	Usage	Cost (USD)	Usage	Cost (USD)	Usage	Cost (USD)
Data ingestion	$0.05 per GB	0.005 GB	0.0075	0.1 GB	0.15	2 GB	3
Training	$0.24 per training hour	3 hours	21.6	10 hours	72	20 hours	144
Real time recommendations	$0.2 per TPS-hour	1 TPS	144	1 TPS	144	5 TPS	720
Batch recommendation	$0.067 per 1000 users	13000 users	26.13	100000 users	201	100000000 user	2010
	Total for Real Time recommendations		$165.61		$216.15		$867.00
	Total for batch recommendations		$47.74		$273.15		$2,157.00

Fig. 8 Price calculation

7. Realisation

Having a well defined high level design, the next step is the realisation phase. This phase includes coding, testing and deploying the solution to the testing environment, and later to the production environment.

Coding

The first step in the realisation part, was to code all the individual Lambda functions that our solution is built from. Because it was decided that the solution will use a micro-service architecture, we are going to have multiple individual micro-services with very specific roles and responsibilities. The problem was split into multiple smaller problems, and each microservice will solve one and only one atomic problem.

Some of the micro-services are as little as a couple of lines of code, but this ensures the single responsibility principle. For example, below we can see the Lambda function's code that is responsible for retraining the model, or also called create a new solution version. It initializes the Amazon Personalize SDK client, and when the function is invoked, it calls the method `createSolutionVersion` and passes the unique id (solutionArn) of the solution we want to retrain.

```
const AWS = require('aws-sdk');

const personalizeClient = new AWS.Personalize();

const solutionArn = process.env.SOLUTION_ARN;

Add Debug Configuration | Edit Debug Configuration (Beta)
module.exports.handler = async () => {
  const response = await personalizeClient.createSolutionVersion({ solutionArn }).promise();

  return response.solutionVersionArn;
};
```

Fig. 9 Lambda Function Retraining the model

The ETL functions are more complex, because they have to read the files from S3 bucket, do some custom transformations on the data by cleaning and preprocessing it, and then writing it back to the database.

Step functions

During the design process, it was decided that the Lambda functions will be orchestrated by the Step function workflow. Using Step Functions, we define the workflow as a state machine, which transforms complex code into easy to understand statements and diagrams. The state machine is defined by the Amazon States Language (ASL). Amazon States Language is a JSON-based, structured language used to define the state machine.

Below we can see the detailed, low-level state machine that was created. The execution starts on the top, and the first function that is called is the `Prepare Dataset Infos`. This function fetches and returns an array of datasets that has to be re-computed. Each individual element from the array (each dataset) will invoke a parallel workflow, seen with a greed box background in the diagram. This design allows multiple individual steps to run in parallel. For each dataset, it will first run the ETL job, then it will invoke amazon personalization to import the newly generated dataset. Because this is an asynchronous job, we need to implement a logic of checking when the job is finished. For that, we have implemented a loop of steps: wait X seconds, check the import job status, if it is in progress, go back to waiting X seconds, but if it has completed, move to the next step. It will also handle the situation when the job failed, and that will stop the execution of the whole state machine and will notify about failure.

After all the datasets have been imported to Amazon personalize, the workflow will invoke the lambda function that will trigger the model re-training. This step is also asynchronous, and it can take up to a couple of hours. For that reason, the same mechanism described above for waiting and checking the status has been implemented here.

After the solution has been re-trained, we generate the batch job input, and then we create the batch job. When the batch job finishes execution, the last lambda function is invoked that is responsible for reading, transforming and writing the data from S3 back to the database.

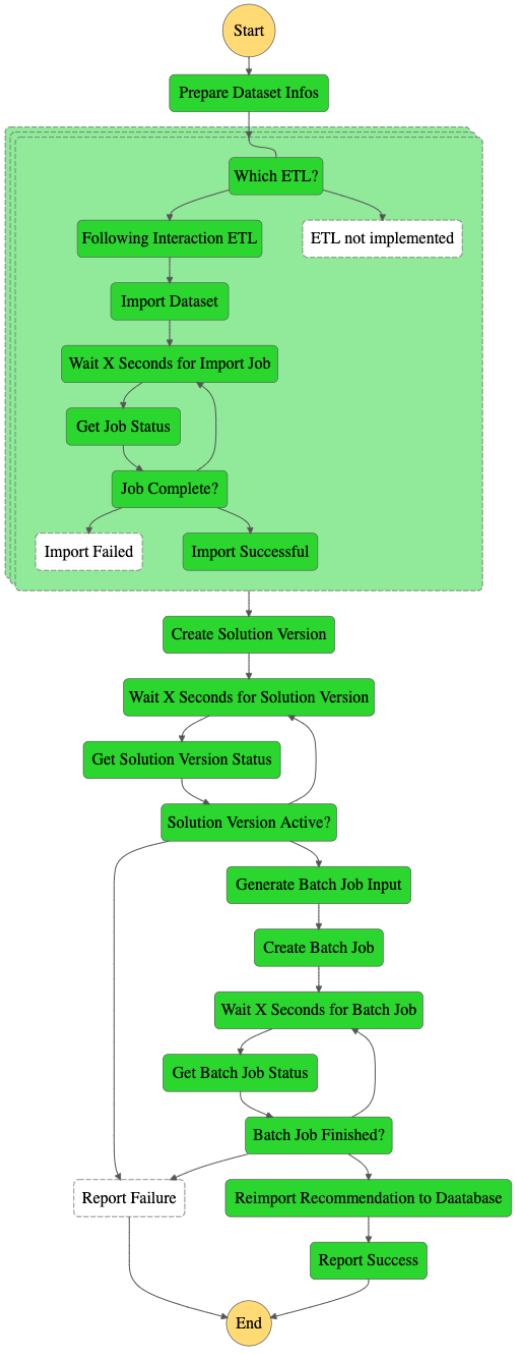


Fig. 10 Step Function State Machine

Infrastructure

While the proof of concept was created using the AWS visual console, the final solution should be created in a more repeatable and documentable way. The problem with setting up the services on AWS using the visual console, by clicking around and following the guides on the platform, is that the end solution is not reproducible. As an example, if later on the company wants to build the same solution but in different regions, or with slightly different datasets, they would have to do everything again from scratch. Moreover, this way is very hard to document, so that the next person working on the solution would know how it was created.

For the above mentioned reasons, it was decided that the infrastructure of the final solution would be created using code [13]. This paradigm is called Infrastructure as Code (IaC) which is the practice of describing all software runtime environment and networking settings and parameters in simple textual format, that can be stored in your Version Control System (VCS), along with the source code. This makes the infrastructure easily reproducible, well documented and versioned, the same way we version the source code.

The AWS service that makes IaC possible is called AWS CloudFormation [14]. A CloudFormation template describes the desired resources and their dependencies so it can be launched and configured together as a stack.

Another useful tool used for creating, testing and deploying the infrastructure was AWS Serverless Application Model (SAM). The AWS Serverless Application Model (SAM) is an open-source framework for building serverless applications [15]. It provides shorthand syntax to express functions, APIs, databases, and event source mappings. With just a few lines per resource, you can define the application you want and model it using YAML. During deployment, SAM transforms and expands the SAM syntax into AWS CloudFormation syntax, enabling developers to build serverless applications

faster. SAM CLI provides a Lambda-like execution environment that lets developers locally build, test, and debug applications defined by SAM templates.

Below we can see how a Lambda function is defined in the final SAM template. Firstly, it defines that the type of the resource we define is an `AWS::Serverless::Function`, in other words a Lambda function. After that, the properties contains the function name, the code uri and the name of the file and function it references as the handler. It also defines that the function should timeout if it has not finished the execution after 900 seconds, and that it should use 512 mb of memory. Lastly, it references the role that it should use, and the role was defined above where it was specified what resources should this function have access to.

The same way, all the required Lambda functions have been defined, together with their roles.

```
S3ToDatabase:
  Type: AWS::Serverless::Function
  Properties:
    FunctionName: !Sub '${AWS::StackName}-S3ToDatabase'
    CodeUri: S3ToDatabase/
    Handler: lambda.handler
    Timeout: 900
    MemorySize: 512
    Role:
      Fn::GetAtt:
        - "ETLRole"
        - "Arn"
```

Fig. 11 Lambda function definition in SAM template

Testing

Before the application was packaged and uploaded to the AWS Cloud, it was thoroughly tested. AWS SAM command line interface gives the possibility to test the application in a local environment.

Each Lambda Function was tested as an individual unit. This way, it was possible to test each step individually, while it was being developed. For each Lambda function, a couple of events, which serve as function input, were created. The events were for both good weather and bad weather scenarios.

When the whole application was finished, and unit tested locally, it was uploaded to AWS Cloud to the development environment. This environment is identical to the production environment. That is why additional end to end integration tests were performed on the development environment directly in AWS Cloud. This process was very helpful in detecting issues before they reached the production version. The outcome of the end-to-end integration tests can be seen in the appendix 2.

8. Validation

When it comes to machine learning, and specifically to the recommendation system, the result is very subjective and it is more challenging to validate it. There is no easy way to say that a model is valid, or is better than another one.

One way to validate a model is to see the results it leads to and to compare it with the heuristic version, or to compare it with other models. This method is feasible for this use case, because Fitenium previously had a primitive heuristic recommendation system and there is also analytical data. The new model was deployed to production, and data was gathered about user behaviour. The result of this validation method showed that users follow more recommended users. The conversion rate of the recommendation widget increased to 4.7%. This means that almost 5 out of 100 times the recommendation widget is displayed in the app, it will lead to a new follower. This might not seem a lot, but it is 10 times better than the conversion rate of the previous heuristic algorithm , which was 0.5%.

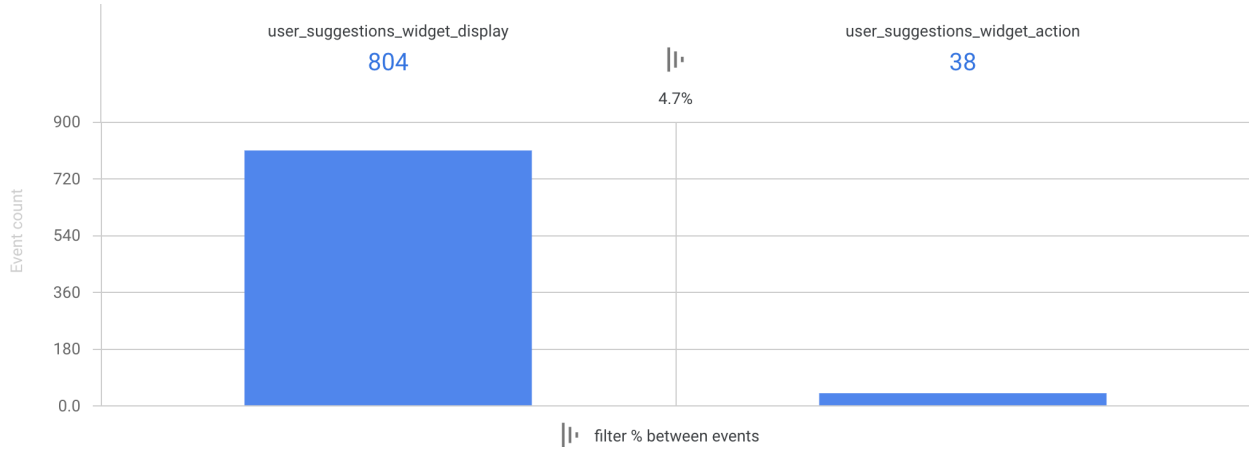


Fig. 12 Conversion rate of the new recommendation system

Conclusion

The aim of this project was to identify and implement the best personalized recommendation system that will increase the engagement of Fitenium users. Based on the research, it was determined that the best personalized recommendation system for this use case is Amazon Personalize.

While Amazon Personalize makes it easier for developers to create personalized recommendation systems, without much machine learning knowledge, the challenging part was to design a scalable and automatic system that would fit well in the existing application infrastructure. The system was designed to automatically retrain the machine learning model with new data in order to keep up to date with the always changing user behaviours.

With the new personalized recommendation system deployed to production, Fitenium users started following more users from the recommendation widget. To be more precise, the conversion rate of the recommendation widget increased to 4.7%. This is almost 10 times more than the conversion rate of the previous heuristic solution, which was only 0.5%. This is a good performance indicator of the system, but it can also be improved.

The team is advised to further experiment tuning the machine learning model performance using A/B testing, where the delivered solution would be tested against the new experimented solution. The tests can include variations in datasets or different hyperparameters. If in future Fitenium starts collecting more user data, the new data can improve the performance of the recommendation system.

The team is also advised to implement the real time recommendation method provided by Amazon personalized, together with real time data streaming to Amazon personalized so that the model will be kept up to date in real time. This method is expected to improve the performance of the system, as it will adjust in real time to the user’s behaviour and interest.

Bibliography

[1] Netflix about “Netflix Prize”. Winners of the Grand Prize of $1M were team “BellKor’s Pragmatic Chaos”. September 21, 2009.
https://www.netflixprize.com/

[2] Alexander Spivak: “Automating Recommendation Engine Training with Amazon Personalize and AWS Glue”. AWS Architecture Blog published on January 13, 2021.
https://aws.amazon.com/blogs/architecture/automating-recommendation-engine-training-with-amazon-personalize-and-aws-glue/

[3] Yufeng G: “The 7 Steps of Machine Learning”. Towards Data Science blog published on August 31, 2017.
https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e

[4] Chris Richardson: “Microservice Architecture”, 2020.
https://microservices.io/

[5] Cloudflare: “What is serverless computing? | Serverless definition”, 2021.
https://www.cloudflare.com/en-gb/learning/serverless/what-is-serverless/

[6] Amazon Personalize official documentation
https://aws.amazon.com/personalize/

[7] AWS Lambda official documentation
https://aws.amazon.com/lambda/

[8] AWS Step Functions official documentation
https://aws.amazon.com/step-functions/

[9] Amazon Simple Storage Service (Amazon S3) official documentation
https://aws.amazon.com/s3/

[10] AWS Lambda Pricing
https://aws.amazon.com/lambda/pricing/

[11] AWS Step Functions Pricing Standard Workflows pricing details
https://aws.amazon.com/step-functions/pricing/

[12] Amazon Personalize Pricing details
https://aws.amazon.com/personalize/pricing/

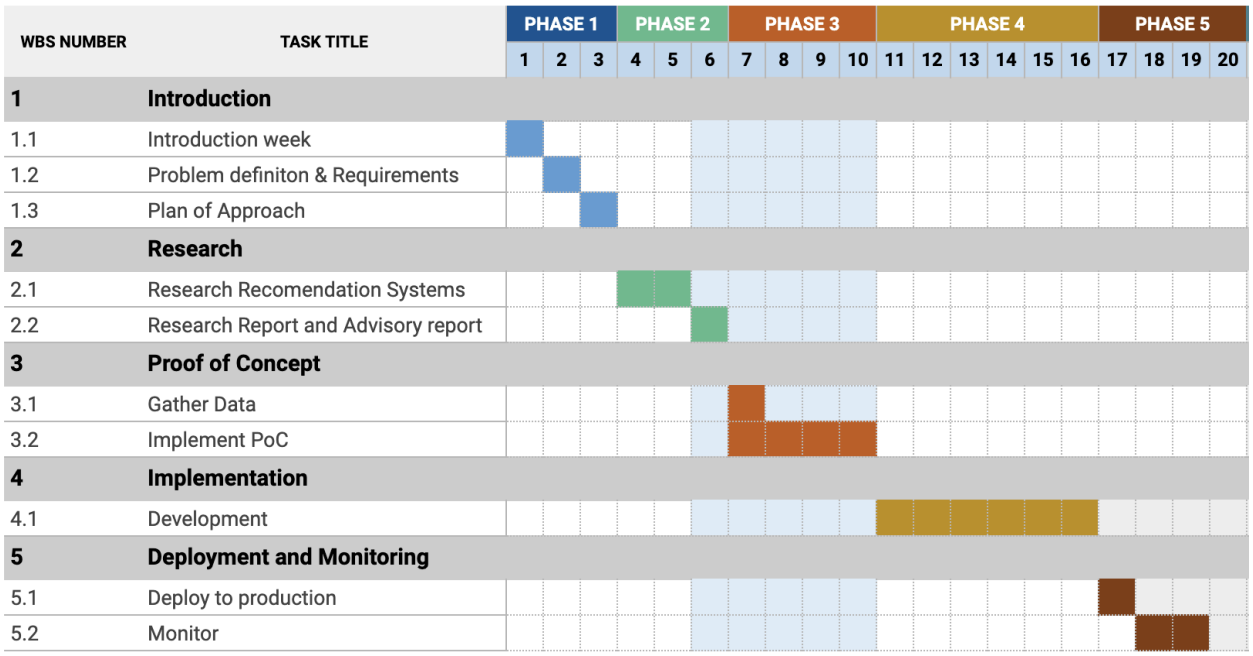
[13] Digital Guide Ionos: “Infrastructure as Code (IaC): using code for IT infrastructure management”
https://www.ionos.com/digitalguide/server/know-how/infrastructure-as-code/

[14] AWS CloudFormation - Speed up cloud provisioning with infrastructure as code
https://aws.amazon.com/cloudformation/

[15] AWS Serverless Application Model official documentation
https://aws.amazon.com/serverless/sam/

Appendices

Appendix 1: Project planning (Gantt chart)



WBS NUMBER	TASK TITLE	PHASE 1			PHASE 2			PHASE 3				PHASE 4						PHASE 5			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	**Introduction**																				
1.1	Introduction week																				
1.2	Problem definiton & Requirements																				
1.3	Plan of Approach																				
2	**Research**																				
2.1	Research Recomendation Systems																				
2.2	Research Report and Advisory report																				
3	**Proof of Concept**																				
3.1	Gather Data																				
3.2	Implement PoC																				
4	**Implementation**																				
4.1	Development																				
5	**Deployment and Monitoring**																				
5.1	Deploy to production																				
5.2	Monitor																				

Appendix 2: End-to-end integration tests

○	fa19b51d-f9c5-a194-2197-c784ad7f7b73	✓ Succeeded	May 30, 2021 11:34:23.092 AM
○	cb7aff99-7d6a-f6bf-b57f-a79244901f2a	✗ Failed	May 26, 2021 07:53:52.414 PM
○	a5362305-4a63-9a93-1942-10008065e79c	✗ Failed	May 26, 2021 05:46:37.218 PM
○	28fd8fef-51f3-8c2c-f44d-5a19cc4f6d6c	✗ Failed	May 26, 2021 04:48:58.837 PM
○	6c2ca69e-2cae-c0b3-251a-c0228c1cec37	✗ Failed	May 26, 2021 03:03:24.186 PM
○	acb3baa7-35e7-181b-1d12-14401d6602c9	✗ Failed	May 26, 2021 01:34:35.855 PM
○	371e9ec7-d849-71ce-9076-8c021aefcfa5	✗ Failed	May 26, 2021 11:54:06.164 AM
○	7be25590-8f02-95b6-45a5-e8d2c1f63cb3	✓ Succeeded	May 24, 2021 05:21:08.167 PM
○	133db8e2-a01c-6cea-6a2b-3d1ae79aefe7	✗ Failed	May 24, 2021 05:05:52.599 PM
○	f12f9f04-462c-8497-644c-ea8c960583e3	✗ Failed	May 24, 2021 05:03:21.341 PM
○	6fed0161-d0c9-c4c2-1bb5-11b8099d77f5	✗ Failed	May 24, 2021 04:55:56.118 PM
○	6b61f6b9-1664-290f-e118-6fdcad629b5b	✗ Failed	May 24, 2021 04:14:58.292 PM
○	c8cb8eb9-5297-202c-07f6-4307242848e2	✗ Failed	May 20, 2021 09:18:11.174 AM
○	a462109d-0357-cd22-870f-3c5fab16ba87	✗ Failed	May 20, 2021 08:00:58.826 AM
○	675fb453-19e1-075c-686f-a2db4f2df3e6	✗ Failed	May 19, 2021 09:34:40.640 PM