Speed-up your sites with web-page pre-fetching using Machine Learning

By

Nidhi Patel V. **19CEUBS151**

A project submitted

In

partial fulfillment of the requirements
for the degree of

BACHELOR OF TECHNOLOGY
in
Computer Engineering

Internal Guide
Dr. Malay S. Bhatt,
Associate Professor
Dept. of Computer Engg.

External Guide
Mr. Prashant Kumar
Project Manager
Institute for Plasma Research

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University
April 2023

CERTIFICATE

This is to certify that the project work titled

Speed-up your sites with web-page pre-fetching using Machine Learning

is the bonafide work of

Nidhi Patel V. 19CEUBS151

carried out in the partial fulfillment of the degree of Bachelor of Technology in Computer Engineering at Dharmsinh Desai University in the academic session December 2022 to April 2023.

Dr. Malay S. Bhatt,
Associate Professor,
Dept. of Computer Engg.

Dr. C. K. Bhensdadia,
Head,
Dept. of Computer Engg.

Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University
April 2023

Company Certificate

प्लाज़्मा अनुसंधान संस्थान
Institute for Plasma Research
भाट, इन्दीरा पुल के निकट , गांधीनगर-382 428 (भारत)
Bhat, Near Indira Bridge, Gandhinagar-382 428. Gujarat (India)
Tel : +91-79-23962000 Fax : +91-7923962277 Web : www.ipr.res.in

प ऊ वि
DAE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that the following student(s) has carried out his academic project work in coordination with academic project guide(s) of Institute for Plasma Research (IPR), Gandhinagar:

Name of Student(s)	:	**Ms. Nidhi Vasantbhai Patel**
Branch/ Discipline	:	B.Tech. Computer Engineering VIII Semester
College/Institute/University	:	Dharmsinh Desai University
Project Title	:	Speed-up your sites with web-page pre-fetching using Machine Learning
Project Guide	:	Mr. Prashant Kumar (Guide)
Project Duration	:	07/12/2022 to 05/04/2023
Project Remarks	:	Satisfactorily Completed

05/04/23

Dr. Braj Kishore Shukla
Chairman
Academic Project Coordination Committee
Institute for Plasma Research
Gandhinagar – 382428 Gujarat

Date: 05-04-2023

परमाणु ऊर्जा विभाग, भारत सरकार का सहायता प्राप्त संस्थान
An Aided Institute of Department of Atomic Energy, Government of India

Acknowledgement

It is indeed a great pleasure to express my thanks and gratitude to all those who helped me during this project. This project would not have been materialized without the help from many people who asked me good questions and rescued from various red tape crisis.

Theoretical knowledge is of no importance if one doesn't know the way of its implementation. I am thankful to my University and IPR that provided me an opportunity to apply my theoretical knowledge through the project. I feel obliged in submitting this project as a part of my Internship.

I would like to take an opportunity to express my humble gratitude to my external project guide **Mr. Prashant Kumar** and internal guide **Prof. Malay Bhatt**, under whom I undertook my project. Their constant guidance and willingness to share their vast knowledge made me enhance my knowledge and helped me to complete the assigned tasks to perfection. Without their effort, support and astonishing testing ability this project may not have succeeded.

With Sincere Regards,

Nidhi Patel

Table of Content

List of Figures

Chapter I

Introduction

I. **Abstract**

Page load time is one of the most important determinants of user experience on a web site. Research shows that faster page load time directly proportional to increased page views, conversion and customer satisfaction. Using TensorFlow tooling, it is now possible to use machine learning to implement a powerful solution for your website to improve page load times. In this project, I implement an end-to-end workflow for using my site's navigation data from Google Analytics and training a custom machine learning model that can predict the user's next actions. I can use this predictions in an Angular app to pre-fetch candidate pages and dramatically improve user experience on the web site.

1. Introduction

1.1 Brief Introduction

In this project, I implement decrease page load time while accessing website with the use of TensorFlow tooling in Machine Learning. I used Tensorflow Extended(TFX) for model training. **Tensorflow Extended (TFX)** is an end to end production scale ML platform and is used to automate the process of data validation, training at scale, evaluation & validation of the generated model. After training my custom model, I want to deploy this model in my web application so it can be used to make live predictions when users visit my website.

1.2 Tools, Technology and Platform Used:

I. Programming Language: Python, HTML, SCSS, JavaScript

II. IDE:

- Google Colaboratory: Used TFX for training model.
- VS Code: Creating and Testing Angular Application.

III. Framework: Angular

IV. Library Used:

- Tensorflow.js

V. Platform: Google Analytics, Google BigQuery

1.3 Timeline for this project work:

SR. No	Task Performed	TimeLine
01	Study and revision of AI/ML Theory, TensorFlow coding	3 weeks
02	Implementation of dummy Web-Application	2 weeks
03	Analysis of data with Google Analytics & connect it with BigQuery	2 weeks
04	Model Training and Testing coding with TFX	6 weeks
05	Deployment of solution to my web-application	3 weeks

Chapter II

About the System

2. Software Requirement Specification

2.1 Description

I have created one dummy website “Apna Store” to show customer behaviour. This is a small web application where user can have many choices to buy various things. But main motto of this project is to make this website faster using TensorFlow for Web Page Prfetching in machine Learning.

The basic steps of the system are

1. Get site data from Google Analytics
2. Data preparation and pre-processing with BigQuery and Dataflow
3. Train custom model using a TFX pipeline
4. Create a client side deployable TensorFlow.js model
5. Deploy solution in your web application with Angular

2.2 Scope

This System is designed to perform faster execution of web pages with using pre-fetching mechanism in Machine Learning. Any user/company who want to make their application execution faster, can use Tensorflow tooling as it is flexible to use, scope of this system is global and open for all users.

2.3 Functional Requirements

2.3.1 Manage Products Page of the site.

This Page contains list of all Products with their descriptions and price. All Items have their images and Add-To-Cart button. Moreover its contains some of functionality which are listed below.

R1. Search Products Functionality

Description: User can search for particular product available on the site using this function.

Input: Type Name of the product or item in the search box.

Output: Display all the available items related to that entered term.

R2. Selection Icons of different section of products

Description: Using this user can get categorized products i.e, Electronics, Clothes, jewellery etc.

Input: User need to click on particular icon of product category.

Output: List of available items of selected category.

R3. Selection of Add-To-Cart Icon

Description: This button is used to shorlist the items that user want to buy.

Input: User needs to click on this button.

Output: It will redirect user to the cart page and display selected products list.

2.3.2 Manage Add-To-Cart Page of the site

This page displays list of products which are being add-to-cart by the user for shopping. It also displays Grand Total of all selected products. It contains some additional functionality which are listed below.

R1. Delete the product from cart

Description: This functionality will be used to remove particular product from the cart.

Input: User need to click on the delete button.

Output: selected Product will be removed from the cart.

R2. Manage Shop More

Description: User can use this option to add more products to the cart.

Input: Click on Shop More Icon.

Output: This will redirect user to the Products page from where user can choose more items.

R3. Manage Empty Cart

Description: Users can use this function when they want to clear the cart and start from the fresh shopping. After doing this empty cart will give “Shop More” option to the users.

Input: Click on Empty Cart Icon.

Output: This will remove all the products from the cart and display “Your cart is empty”.

2.4 Non-Functional Requirements

2.4.1 Performance:

The Project should run very efficiently. It must be user friendly in nature and the prediction should be as fast as possible.

2.4.2 Reliability:

The prediction returned by the program should be as accurate as possible.

2.5 Hardware Requirements

Operating System: Windows 8 or Higher

Processor: Intel i3 or Higher

Memory: 4GB or More

Chapter III

Analysis

3. Design

3.1 Use Case Diagram

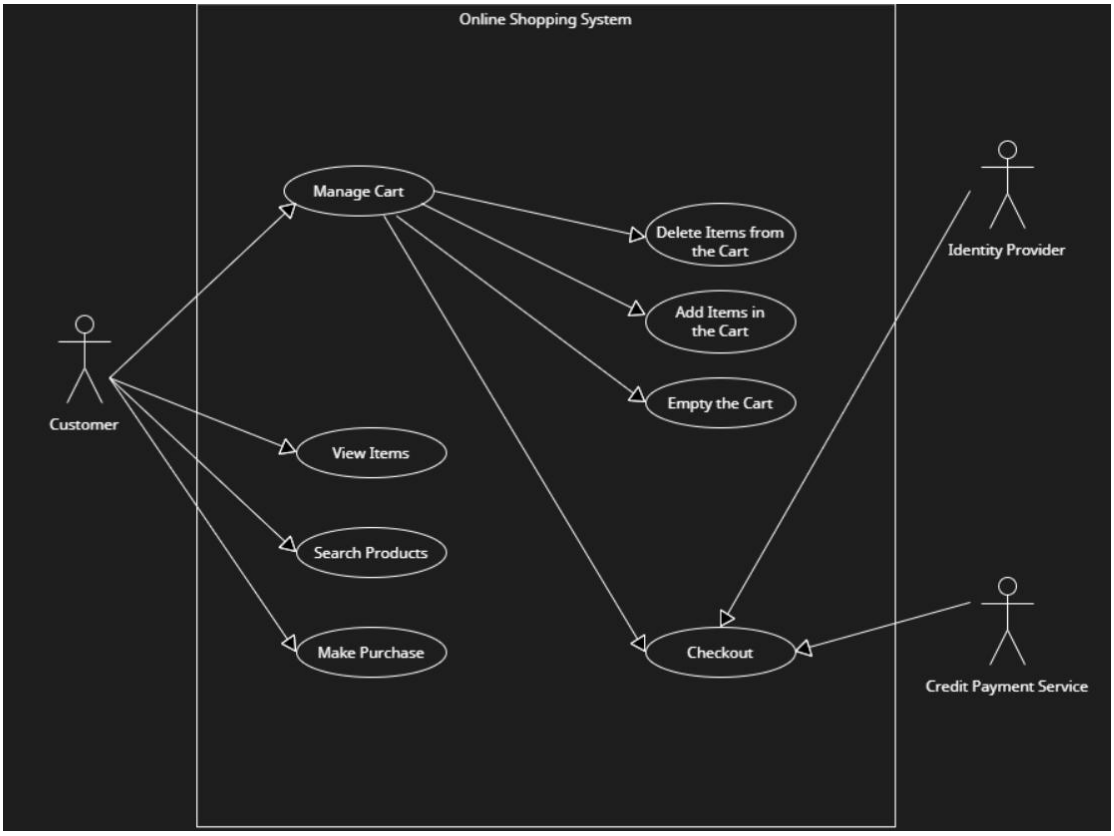


Fig. 3.1 Use Case Diagram

3.2 Activity Diagram

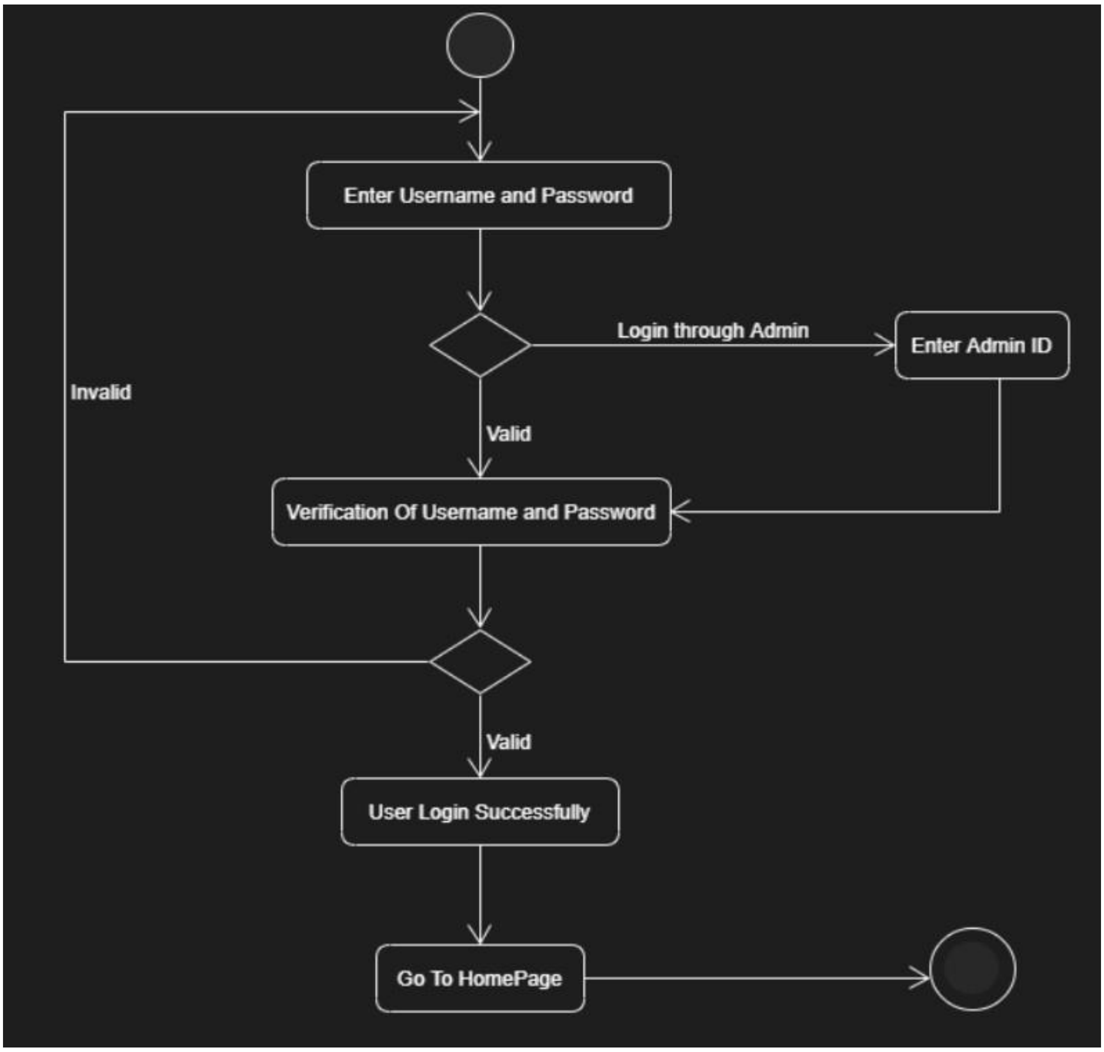


Fig. 3.2 Activity Diagram 1

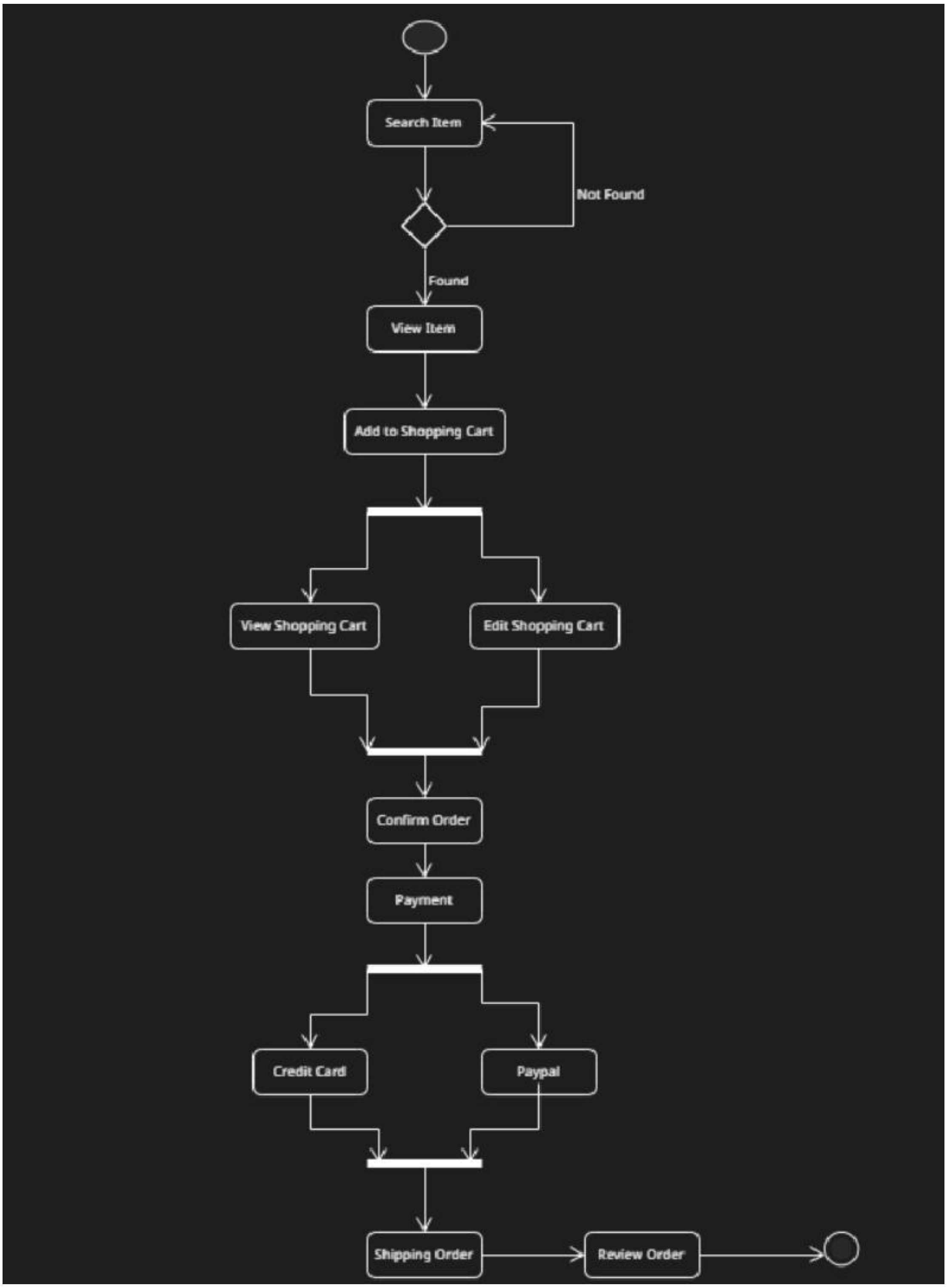


Fig. 3.3 Activity Diagram 2

3.3 Sequence Diagram

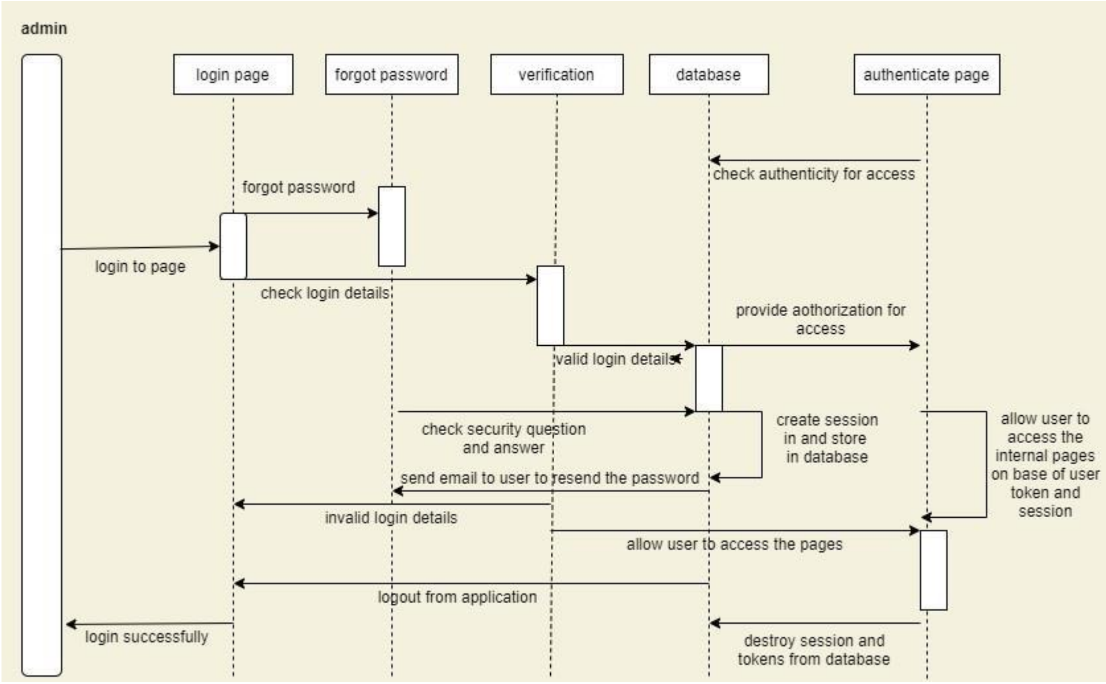


Fig. 3.4 Sequence Diagram 1

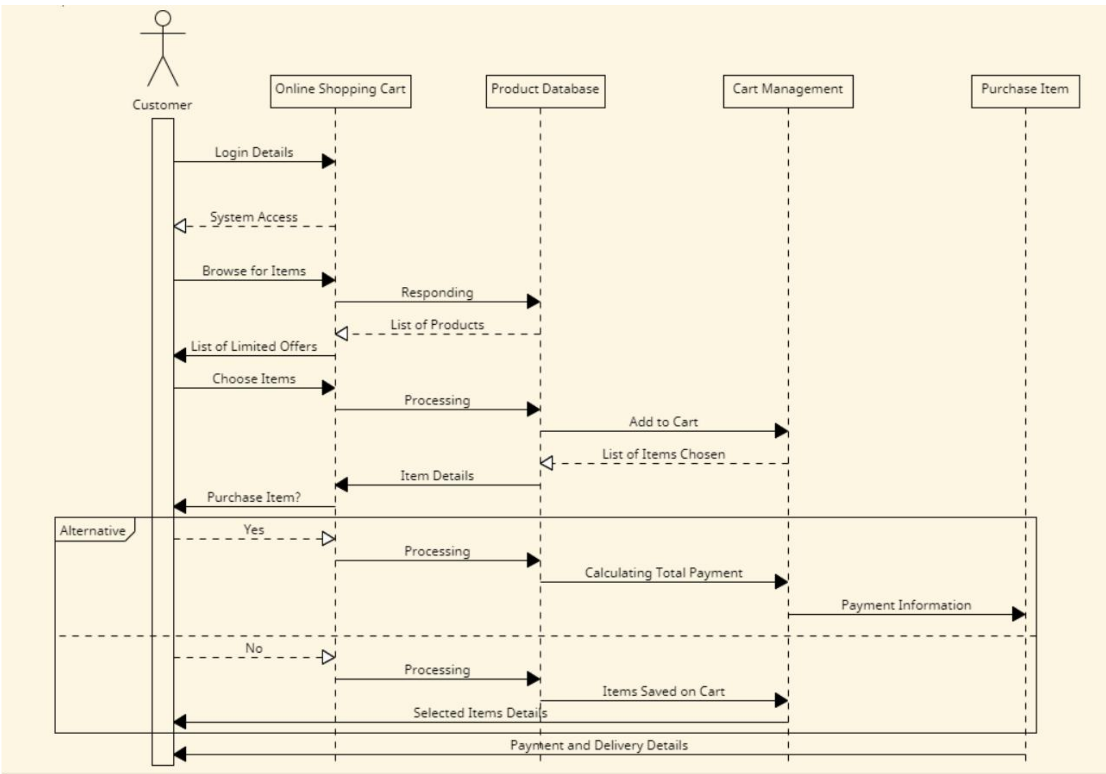


Fig. 3.5 Sequence Diagram 2

Chapter IV

Implementation

4. Implementation Details

In this, I have created one dummy web application named "Apna Store" using Angular framework to show user's behaviour. I build an end-to-end workflow for using site's data from Google Analytics and training a custom machine learning model that can predict the user's next actions. I can use these predictions in an Angular application to pre-fetch candidate pages and dramatically improve user experience on my web-site.

I use Google BigQuery to store and preprocess the site's Google Analytics data, then train the custom model using TensorFlow Extended (TFX) to run my model training pipeline, produce a site-specific model, and then convert into a web-deployable TensorFlow.js format. This Client side model will be loaded in a sampler Anglular web app for an e-store to demonstrate how to deploy the model in a web application.

Below model illustrates overall flow of this project and detailed explanation of each steps are further described.

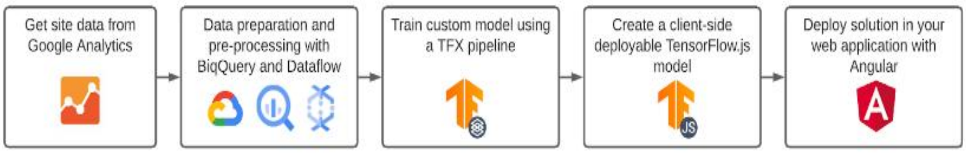


Fig. 4.1 Process Overview Model

1. Get site data from Google Analytics:

Google Analytics stores each page visit as an event, providing key aspects such as the page name, visit time, and load time. This data contains everything I need to train my model.

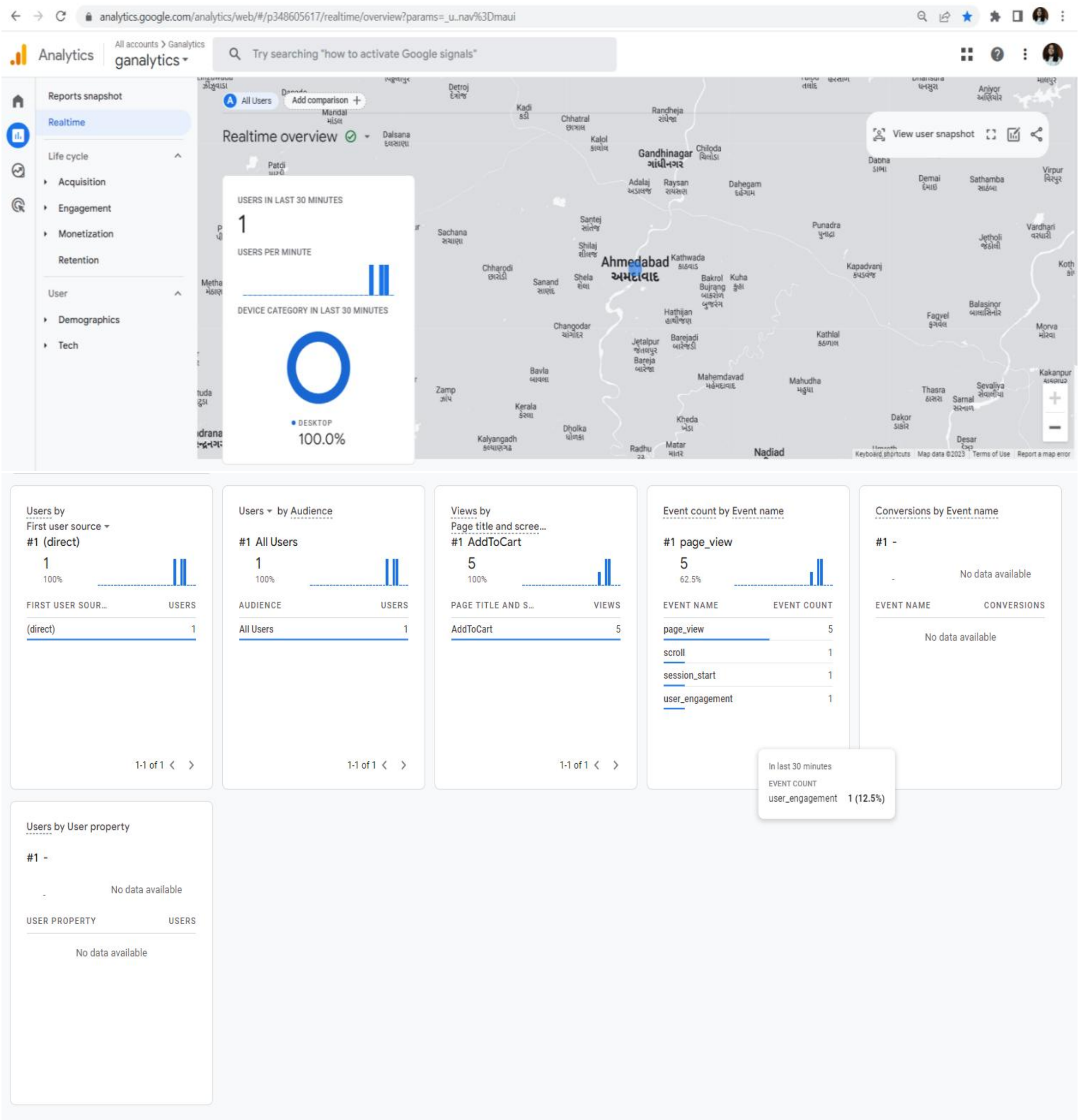
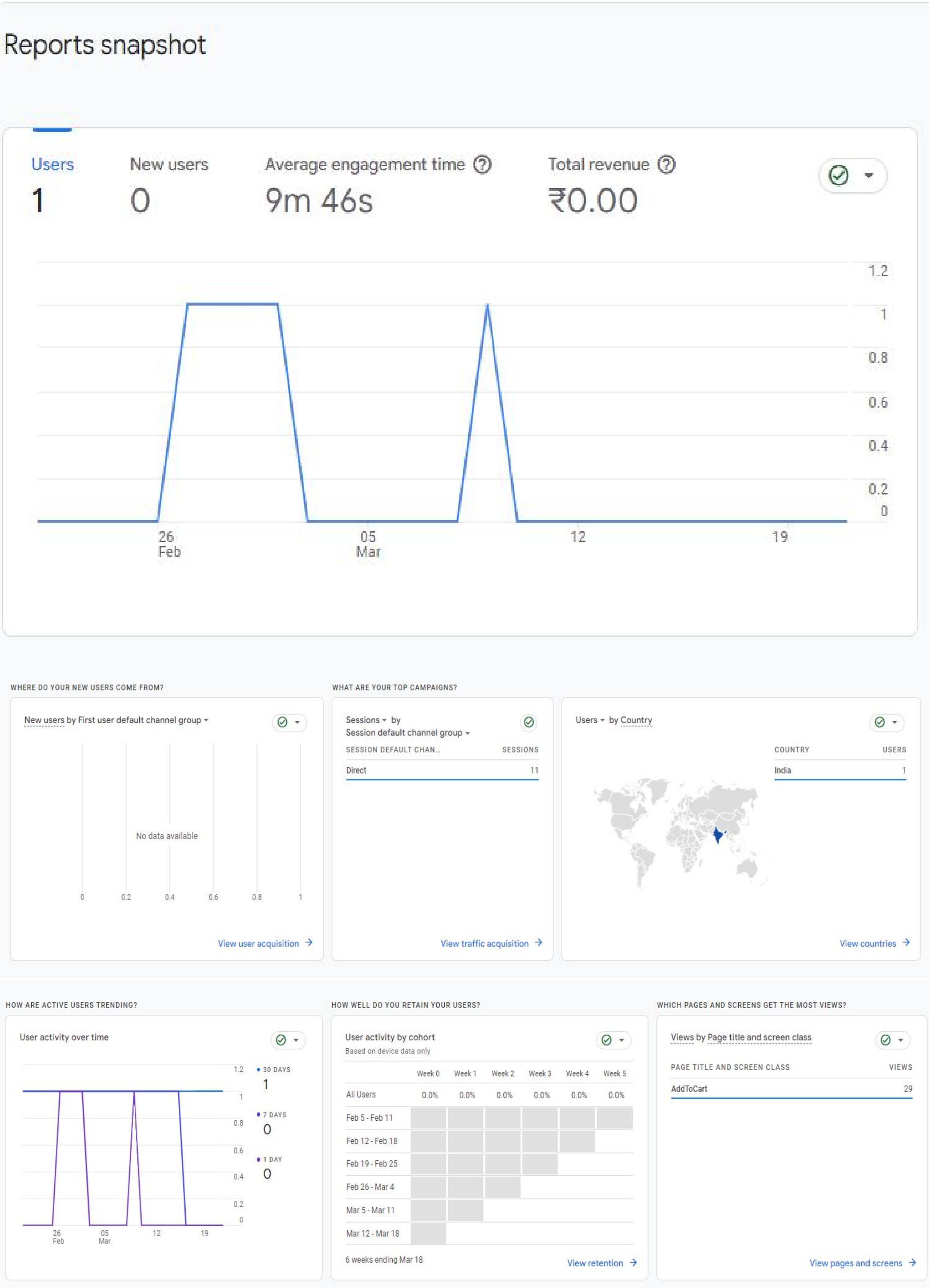


Fig. 4.2 Google Analytics Real-Time Overview


Reports snapshot
Users
1
New users
0
Average engagement time
9m 46s
Total revenue
₹0.00
1.2
1
0.8
0.6
0.4
0.2
0
26
Feb
05
Mar
12
19
WHERE DO YOUR NEW USERS COME FROM?
New users by First user default channel group
No data available
0
0.2
0.4
0.6
0.8
1
View user acquisition
WHAT ARE YOUR TOP CAMPAIGNS?
Sessions by
Session default channel group
SESSION DEFAULT CHAN...
SESSIONS
Direct
11
View traffic acquisition
Users by Country
COUNTRY
USERS
India
1
View countries
HOW ARE ACTIVE USERS TRENDING?
User activity over time
30 DAYS
1
7 DAYS
0
1 DAY
0
26
Feb
05
Mar
12
19
HOW WELL DO YOU RETAIN YOUR USERS?
User activity by cohort
Based on device data only
Week 0
Week 1
Week 2
Week 3
Week 4
Week 5
All Users
0.0%
0.0%
0.0%
0.0%
0.0%
0.0%
Feb 5 - Feb 11
Feb 12 - Feb 18
Feb 19 - Feb 25
Feb 26 - Mar 4
Mar 5 - Mar 11
Mar 12 - Mar 18
6 weeks ending Mar 18
View retention
WHICH PAGES AND SCREENS GET THE MOST VIEWS?
Views by Page title and screen class
PAGE TITLE AND SCREEN CLASS
VIEWS
AddToCart
29
View pages and screens

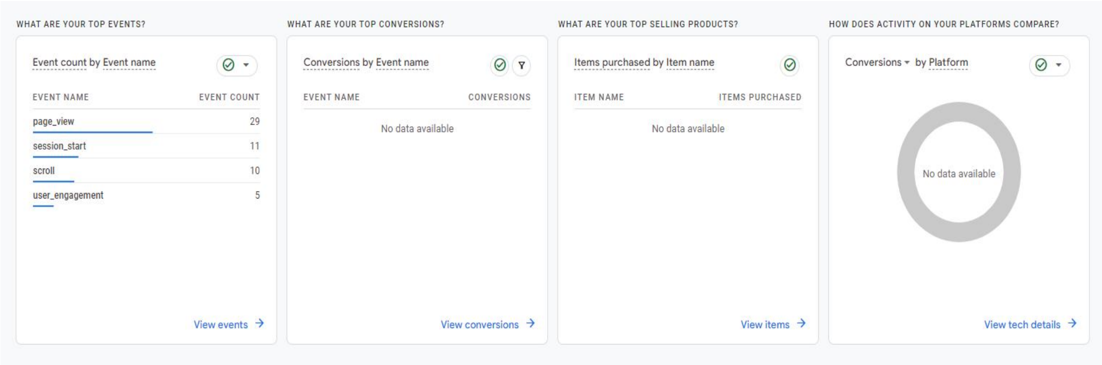


Fig. 4.3 Google Analytics Report Snapshots

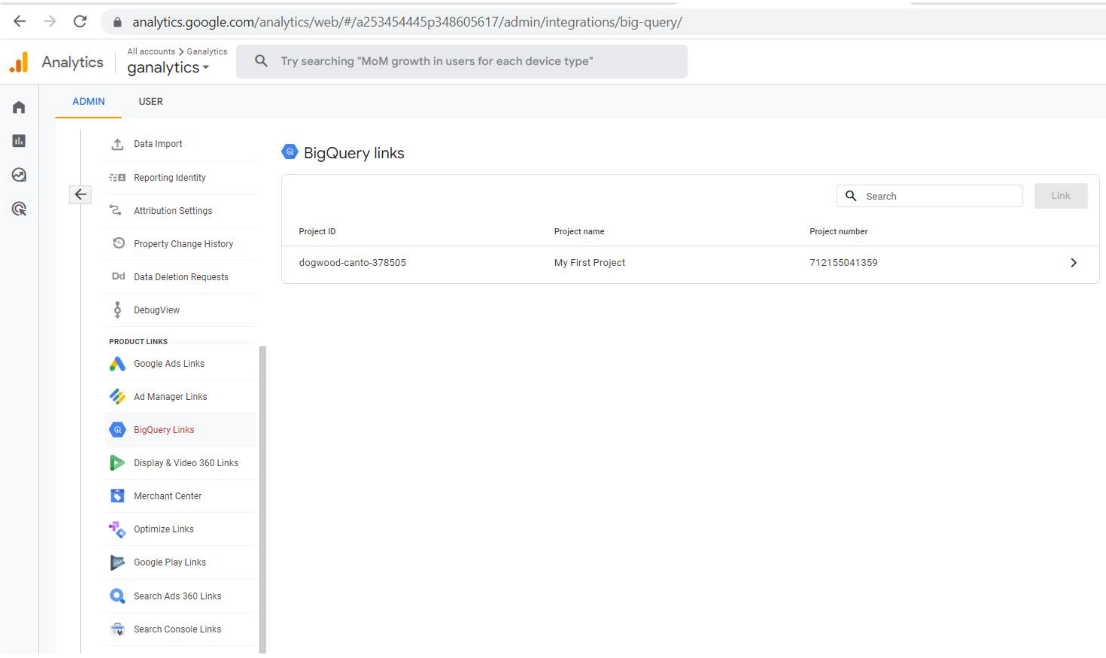


Fig. 4.4 Google Analytics link to BigQuery

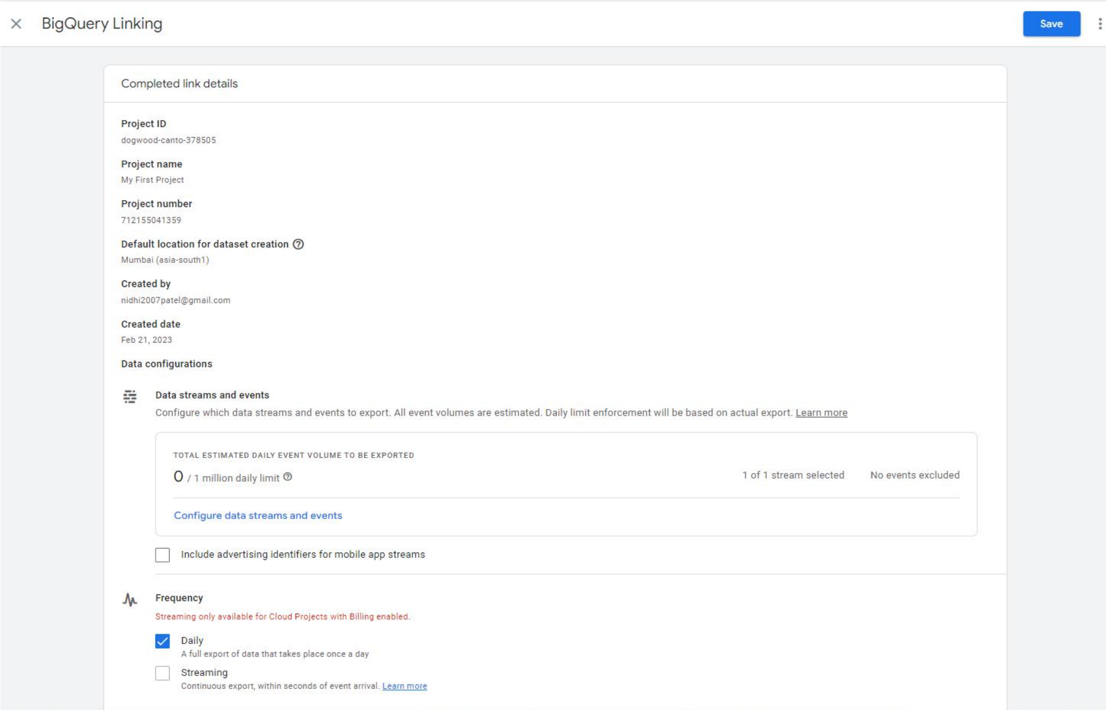


Fig. 4.5 BigQuery Linking Project Details

2. Data preparation and pre-processing with BigQuery and Dataflow:

By leveraging existing support for exporting the Google Analytics data to a large scale cloud data store called BigQuery.

Google BigQuery provides processed data after 24 hours of user visiting the web application.

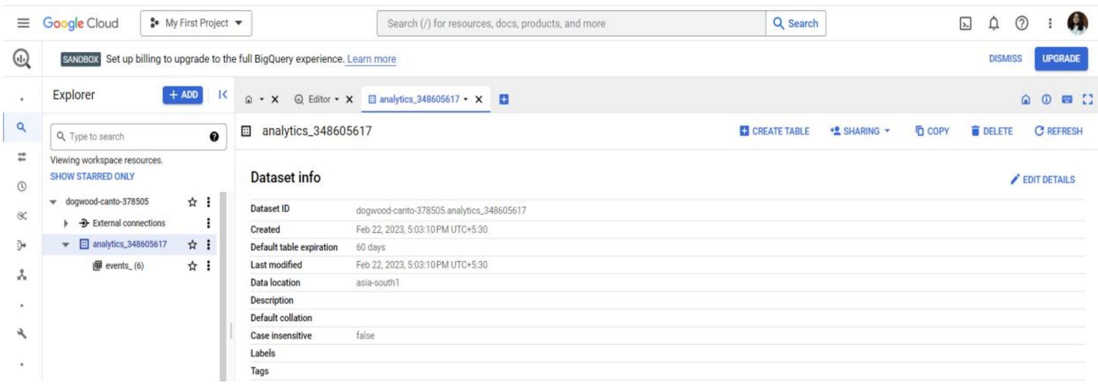


Fig. 4.6 BigQuery Project Dataset info

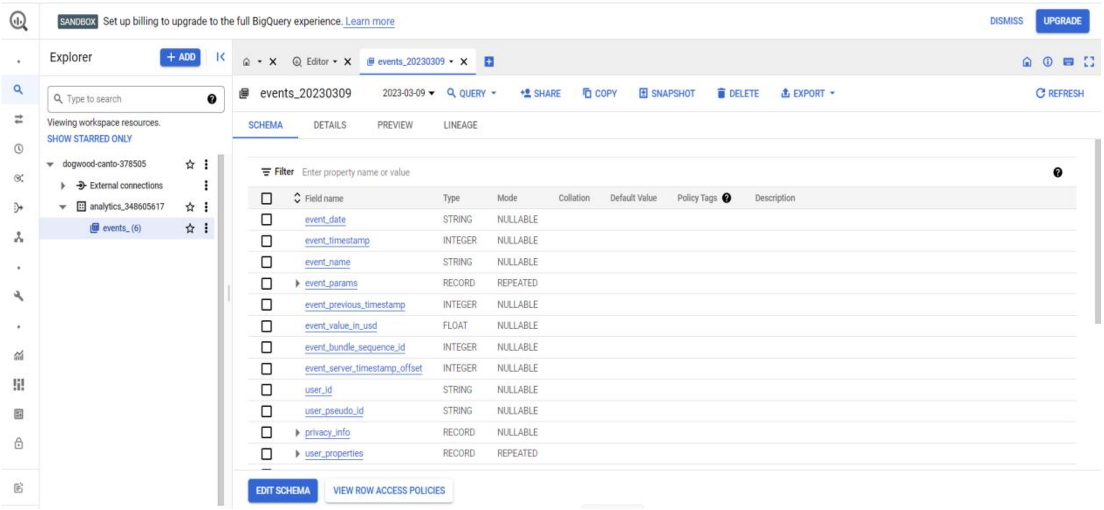


Fig. 4.7 BigQuery Project Dataset schema

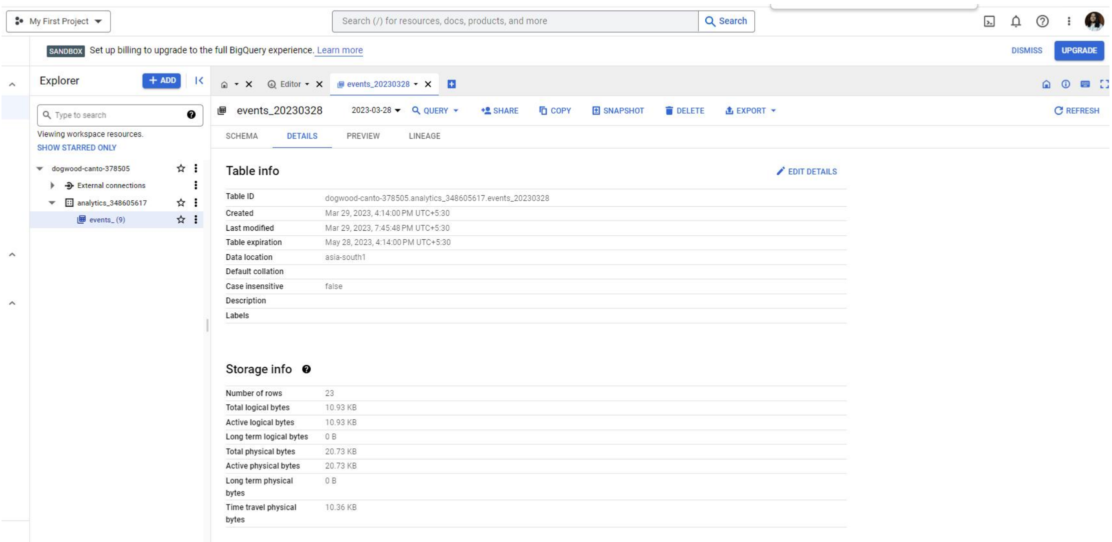


Fig. 4.8 BigQuery Project Dataset Table info

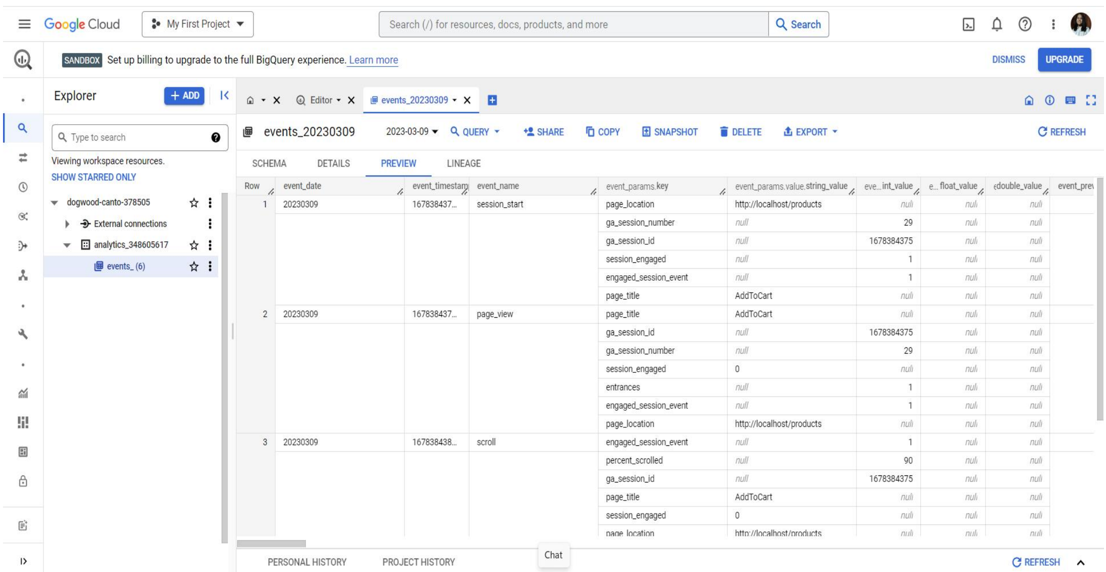


Fig. 4.9 BigQuery Project Dataset Preview

3. Train custom model using TFX pipeline:

4.3.1 Technological overview of TensorFlow Extended:

- TFX is a Google-production-scale machine learning toolkit based on TensorFlow.
- TensorFlow extended is an end-to-end platform for deploying production Machine Learning pipeline.
- A TFX pipeline is a sequence of components that implement Machine Learning Pipeline which is specifically designed for scalable, high-performance machine learning tasks.
- Components are built using TFX libraries which can also be used individually.
- It also provides a configuration framework to integrate common components needed to define, launch, and monitor the machine learning system.
- TFX provides a powerful platform for every phase of a machine learning project, from research, experimentation, and development on your local machine, through deployment.
- In order to avoid code duplication and eliminate the potential for training/serving skew it is strongly recommended to implement your TFX pipeline for both model training and deployment of trained models, and use Transform components which leverage the TensorFlow Transform library for both training and inference.
- By doing so you will use the same pre-processing and analysis code consistently, and avoid differences between data used for training and data fed to your trained models in production.

In my project, I have used Apache beam pipeline in order to get effective and scalable model training of user's data and deployment of trained model to my web application.

Apache beam pipeline-

1. Reads data from BigQuery
2. Sorts and filters the events in a session
3. Walks through each session, creating examples that take properties of the current event as features and the page visit in the page visit in the next event as the label

4. Stores these generated examples in Google Cloud Storage so that they can be used by TFX for training.

I run our Beam pipeline in Dataflow.

In the following table, each row represents a training examples:

cur_page	session_index	label
page2	0	page3
page3	8	page1

Fig. 4.10 Training Features

While my training example only contains two training features (cur_page and session_index), additional features from Google Analytics can be easily added to create a richer dataset and used for training to create a more powerful model. To do so, extend the following code:

```
def ga_session_to_tensorflow_examples(session: List[Any]):
  """Converts a Google Analytics Session to Tensorflow Examples."""
  examples = []
  for i in range(len(session) - 1):
    features = {
        # Add any additional desired training features here.
        'cur_page': [_sanitize_page_path(session[i]['page']['pagePath'])],
        'label': [_sanitize_page_path(session[i + 1]['page']['pagePath'])],
        'session_index': [i],
    }
    examples.append(create_tensorflow_example(features))
  return examples
```

Fig. 4.11 Function for Training Data

```python
def run_beam_pipeline():
  """Run the apache beam pipeline with the specified flags."""

  # Params used for running the Beam pipeline. Update these based on your
  # requirements.
  params = {}
  # Specify the projectid for BigQuery
  params['projectId'] = 'my_project_id'
  # Specify the datasetid for BigQuery
  params['datasetId'] = 'my_dataset_id'
  # Specify the table for BigQuery
  params['tableId'] = 'my_table_id'
  # Specify the list of flags for the Beam pipeline
  params['flags'] = ['--temp_location=my_temp_location']
  # Specify the desination for the generated examples.
  params['destination'] = 'my_destination'

  table_spec = bigquery.TableReference(
      projectId=params['projectId'],
      datasetId=params['datasetId'],
      tableId=params['tableId'])
  with beam.Pipeline(
      options=beam.options.pipeline_options.PipelineOptions(
          flags=params['flags'])) as p:
    _ = (
        p
        | 'ReadTable' >> beam.io.ReadFromBigQuery(table=table_spec)
        | 'ConvertToTensorFlowExamples' >> beam.ParDo(ExampleGeneratingDoFn())
        | 'Write' >> beam.io.tfrecordio.WriteToTFRecord(
            'gs://tfxdata/data/output',
            coder=beam.coders.ProtoCoder(tf.train.Example),
            file_name_suffix='.tfrecord.gz'))
```

Fig. 4.12 Function to run Apache Beam Pipeline with Specified flags

TFX is an end to end production scale ML platform and is used to automate the process of data validation, training at scale, evaluation & validation of the generated model.

To create a model within TFX, I must provide the preprocessing function and the run function. The preprocessing function defines the operations that should be performed on the data before it is passed to the main model. These include operations that involve a full pass over the data, such as vocab creation. The run function defines the main model and how it is to be trained.

Models to predicting next page:

```
# TFX Transform will call this function.
def preprocessing_fn(inputs):
  """Callback function for preprocessing inputs.

  Args:
    inputs: map from feature keys to raw not-yet-transformed features.

  Returns:
    Map from string feature key to transformed feature operations.
  """
  outputs = inputs.copy()

  # Compute a vocabulary based on the TOP-K current pages and labels seen in
  # the dataset.
  vocab = tft.vocabulary(
      tf.concat([inputs[_CUR_PAGE_FEATURE_KEY], inputs[_LABEL_KEY]], axis=0),
      top_k=_TOP_K,
      vocab_filename=_VOCAB_FILENAME)

  # Apply the vocabulary to both the current page feature and the label,
  # converting the strings into integers.
  for k in [_CUR_PAGE_FEATURE_KEY, _LABEL_KEY]:
    # Out-of-vocab strings will be assigned the _TOP_K value.
    outputs[k] = tft.apply_vocabulary(inputs[k], vocab, default_value=_TOP_K)
  return outputs
```

Fig. 4.13 PreProcessing Function

```
def _input_fn(file_pattern: List[str],
              data_accessor: tfx.components.DataAccessor,
              tf_transform_output: tft.TFTransformOutput,
              batch_size: int = 200) -> tf.data.Dataset:
  """Generates features and label for tuning/training.

  Args:
    file_pattern: List of paths or patterns of input tfrecord files.
    data_accessor: DataAccessor for converting input to RecordBatch.
    tf_transform_output: A TFTransformOutput.
    batch_size: representing the number of consecutive elements of returned
      dataset to combine in a single batch.

  Returns:
    A dataset that contains (features, indices) tuple where features is a
      dictionary of Tensors, and indices is a single Tensor of label indices.
  """
  dataset = data_accessor.tf_dataset_factory(
      file_pattern,
      tfxio.TensorFlowDatasetOptions(
          batch_size=batch_size, label_key=_LABEL_KEY),
      tf_transform_output.transformed_metadata.schema)

  return dataset.repeat()
```

Fig. 4.14 Input Function

```
# TFX Trainer will call this function.
def run_fn(fn_args: tfx.components.FnArgs):
  """Train the model based on given args.

  Args:
    fn_args: Holds args used to train the model as name/value pairs.
  """
  tf_transform_output = tft.TFTransformOutput(fn_args.transform_output)

  train_dataset = _input_fn(
      fn_args.train_files,
      fn_args.data_accessor,
      tf_transform_output,
      batch_size=_TRAIN_BATCH_SIZE)

  eval_dataset = _input_fn(
      fn_args.eval_files,
      fn_args.data_accessor,
      tf_transform_output,
      batch_size=_EVAL_BATCH_SIZE)

  mirrored_strategy = tf.distribute.MirroredStrategy()
  with mirrored_strategy.scope():
    model = _build_keras_model()

  model.fit(
      train_dataset,
      steps_per_epoch=fn_args.train_steps,
      validation_data=eval_dataset,
      validation_steps=fn_args.eval_steps,
      verbose=2)
```

Fig 4.15 Run Function

4. Create a client-side deployable Tensorflow.js:

After training my custom model, I want to deploy this model in my web application so it can be used to make live predictions when users visit my website. For this, I use TensorFlow.js, which is TensorFlow's framework for running machine learning models directly in the browser client-side. By running this code in the browser client-side, I can reduce latency associated with server-side roundtrip traffic, reduce server-side costs, and also keep user's data private by not having to send any session data to the server.

TFX employs the Model Rewriting Library to automate conversion between trained TensorFlow models and the TensorFlow.js format. As part of this library, I have implemented a TensorFlow.js rewriter. I simply invoke this rewriter within the run_fn to perform the desired conversion.

5. Deploy solution in your web application with Angular

Once I have the model, I can use it within an Angular application. On each navigation, I will query the model and prefetch the resources associated with the pages that are likely to be visited in the future.

A service worker is a script that my browser runs in the background in a new thread, separate from a web page. It also allows me to plug into a request cycle and provides me with cache control.

When the user navigates across the application, I will post messages to the service worker with the pages they have visited. Based on the navigation history, the service worker will make predictions for future navigation and prefetch relevant product assets.

Below process model illustrates how live predictions were made when user hit different functionalities of the web application.

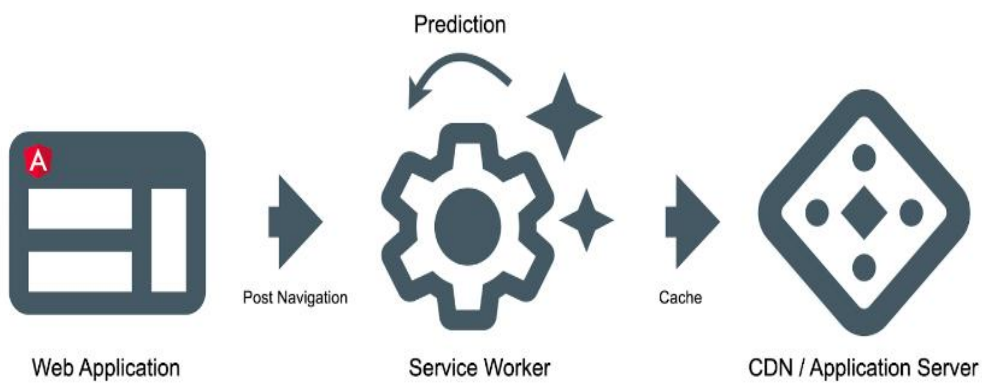


Fig 4.16 Model of the page prediction Process in Web-application

From within the main file of my Angular application, I can load the service worker:

This snippet will download the prefetch.worker.js script and run it in the background. As the next step, I want to forward navigation events to it:

main.ts U ×

src > main.ts > ...

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));

if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/prefetch.worker.js', { scope: '/' });
}
```

Fig 4.17 main.ts file of the application where we load Service Worker

In the snippet below, I watch for changes of the parameters of the URL. On change, I forward the category of the page to the service worker.

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'add-to-cart';
}
this.route.params.subscribe((routeParams) => {
  if (this._serviceWorker) {
    this._serviceWorker.postMessage({ page: routeParams.category });
  }
});
```

Fig 4.18 app.components.ts file

In the implementation of the service worker I need to handle messages from the main thread, make predictions based on them, and prefetch the relevant information. On a high-level this looks as follows:

```javascript
// prefetch.worker.js

addEventListener('message', ({ data }) => prefetch(data.page));

const prefetch = async (path) => {
  const predictions = await predict(path);
  const cache = await caches.open(ImageCache);

  predictions.forEach(async ([probability, category]) => {
    const products = (await getProductList(category)).map(getUrl);
    [...new Set(products)].forEach(url => {
      const request = new Request(url, {
        mode: 'no-cors',
      });
      fetch(request).then(response => cache.put(request, response));
    });
  });
};
```

Fig 4.19 prefetch.worker.js

Within the service worker, I listen for messages from the main thread. When I receive a message, I trigger the logic responsible for making predictions and prefetching data.

In the prefetch function I first predict, which are the pages the user could visit next. After that, I iterate over all the predictions and fetch the corresponding resources to improve the user experience in subsequent navigation.

Chapter V

Testing

5. Testing

Below screen-shots illustrate improved page load times with machine learning based predictive pre-fetching implemented.

These screen-shots were taken from console page of my web application. They determines total time taken for each page to load when user visit the web application.

Here, I also attached Screen-shot of detailed information related to products page load time.

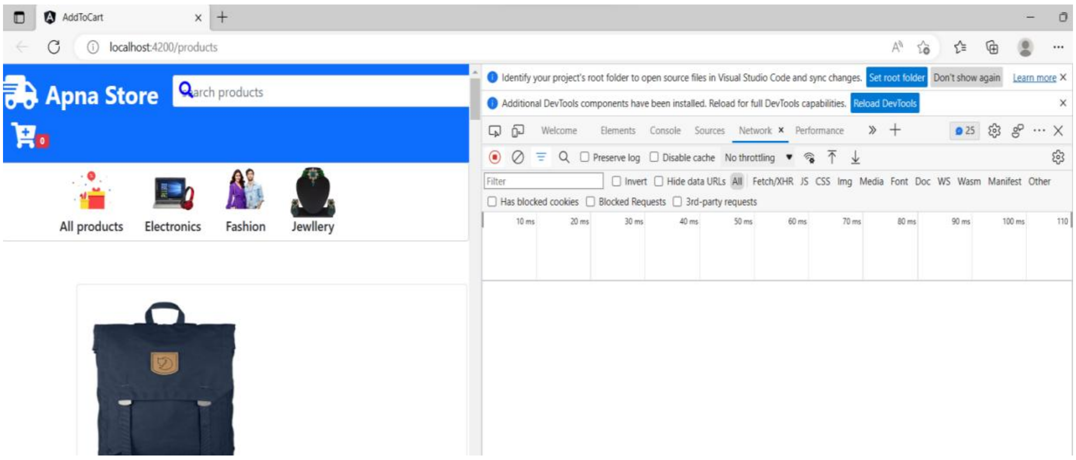


Fig. 5.1 Inspect->Network Page of the web application(Before refresh)

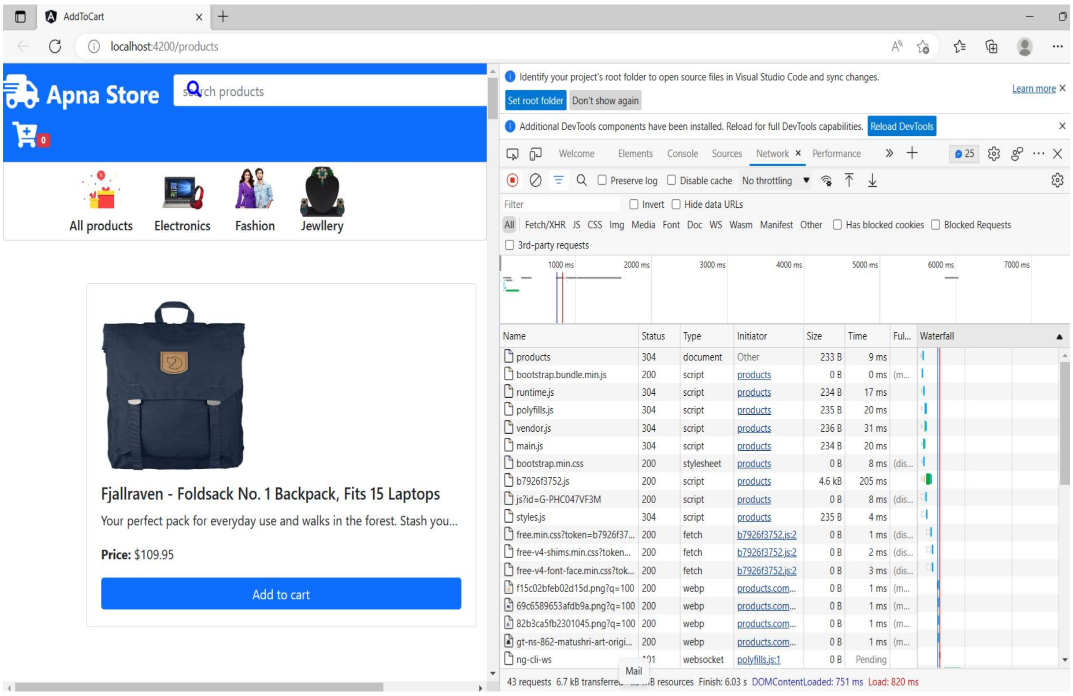


Fig. 5.2 Inspect->Network Page of the web application(After refreshed)

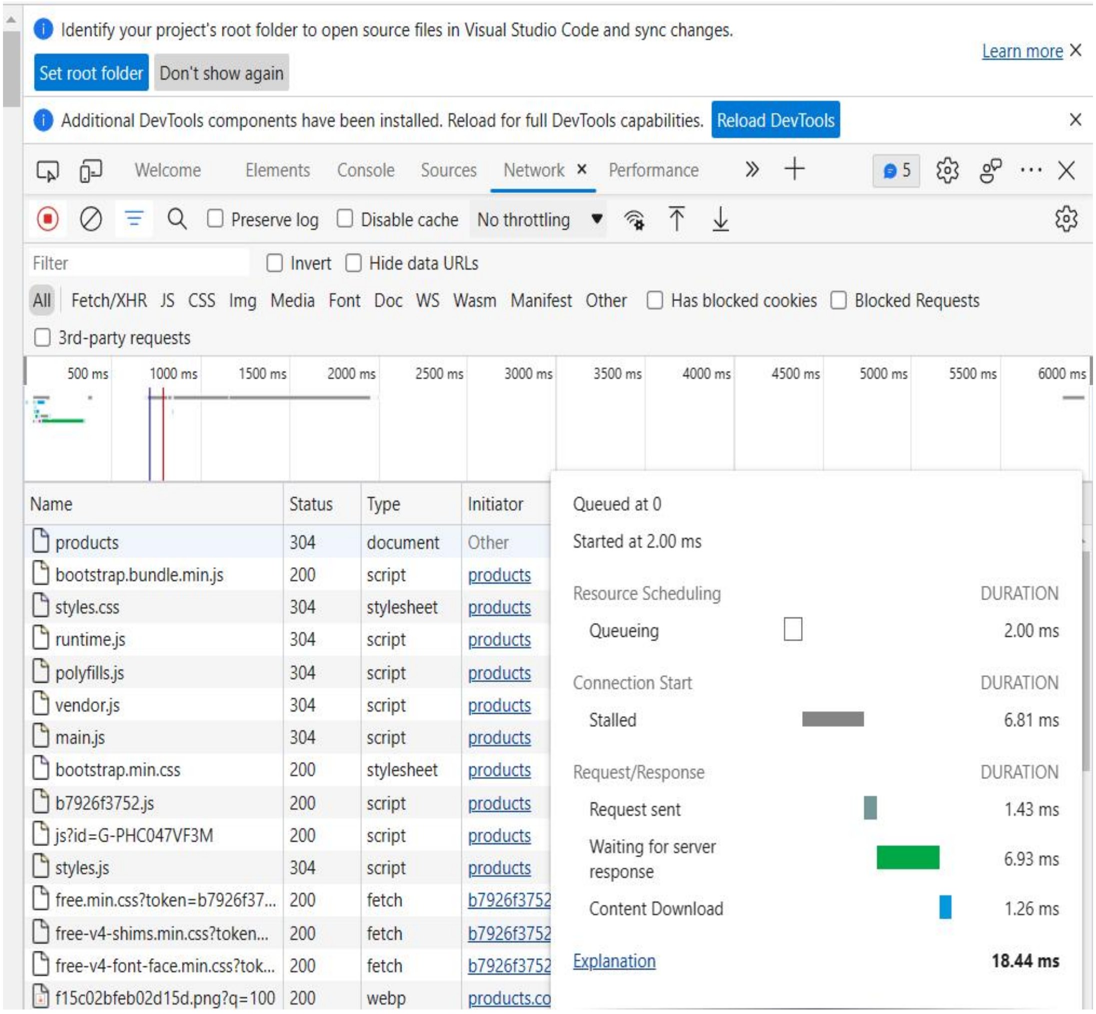


Fig. 5.3 Detailed Information of particular data

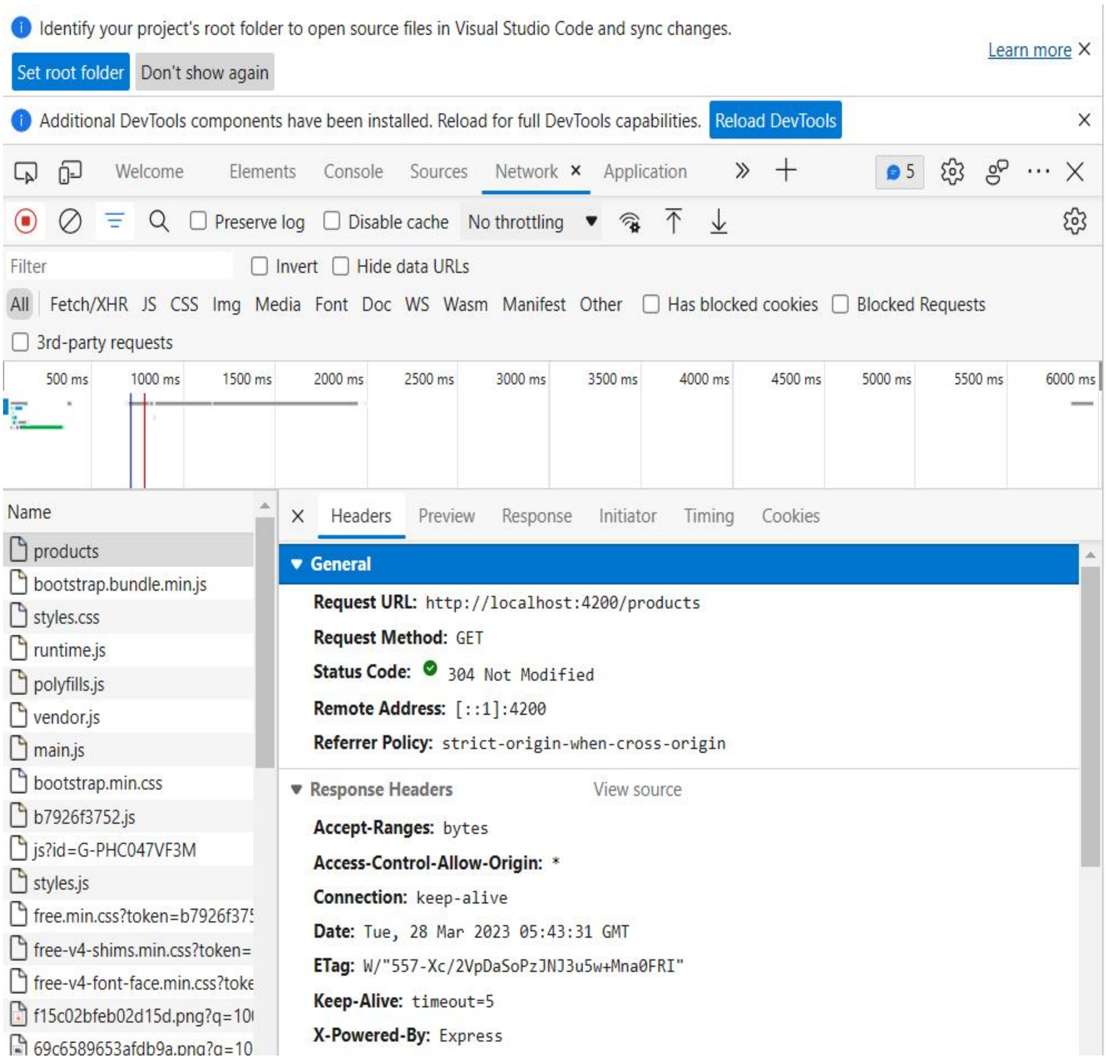


Fig. 5.4 Information of Headers for data

6. Screen shots

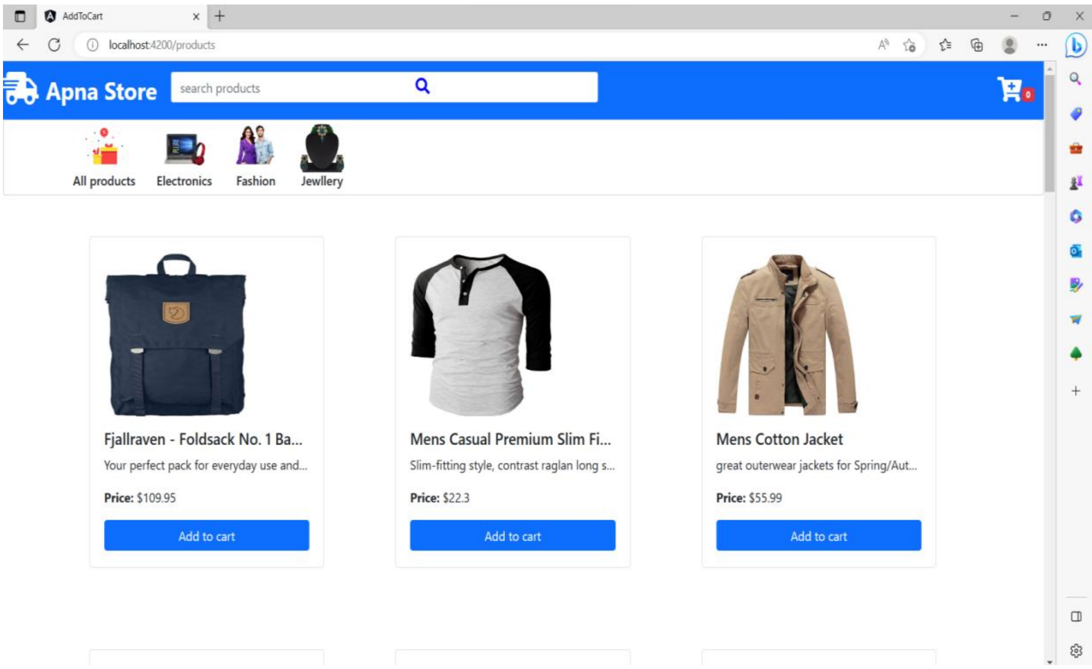


Fig. 6.1 Products page

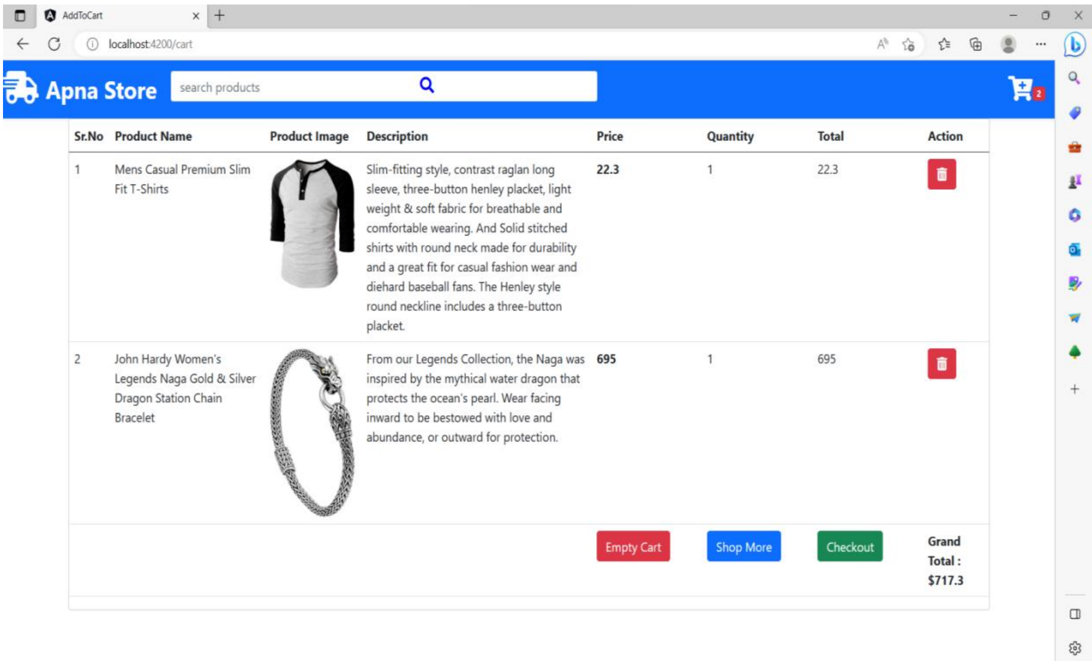


Fig. 6.2 Add-To-Cart Page

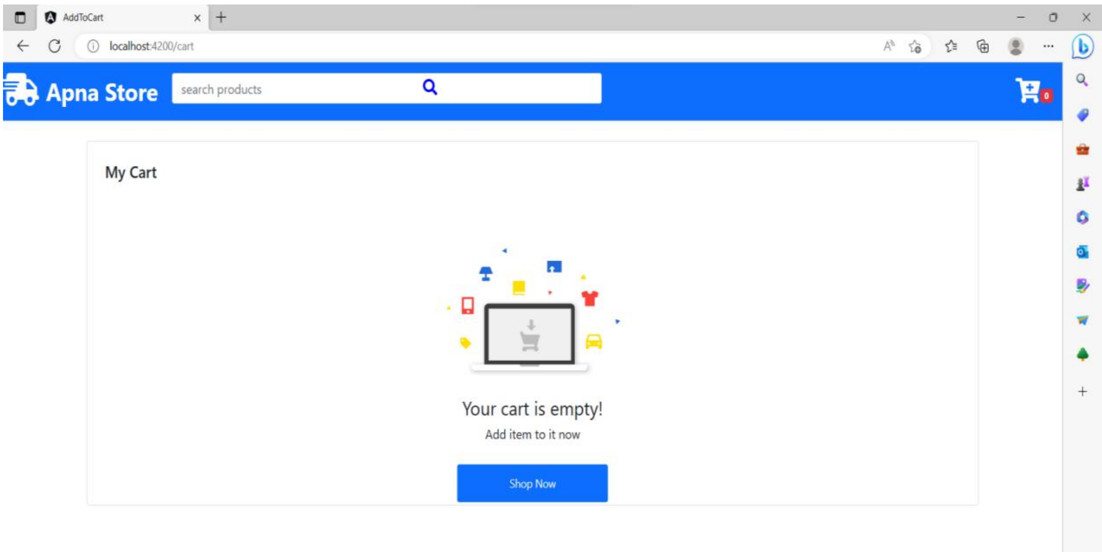


Fig. 6.3 Empty Cart Page

Chapter VI

Conclusion

7. Conclusion

- Page load time is one of the most important determinants of user experience on a web site.

- In this project, I show an end-to-end workflow for using my site’s navigation data from Google Analytics and training a custom machine learning model that can predict the user's next actions.

- I can use these predictions in an Angular app to pre-fetch candidate pages and dramatically improve user experience on my web site.

- I used Google Cloud services to store and preprocess the site’s Google Analytics data, then train a custom model using TensorFlow Extended to run my model training pipeline, produce a site-specific model, and then convert it into a web-deployable TensorFlow.js format.

8. Limitation and Future Extension

8.1 Limitations

- When a user is on a constrained network connection or a limited data-plan, every last byte counts. While prefetching a number of pages ahead of time can make sense for users on a great WiFi connection.

- To prefetch the resources associated with all the possible future navigation paths would have much higher bandwidth consumption. Using machine learning, I can predict only the pages, which are likely to be used next and reduce the number of false positives.

- Without subscription of Google BigQuery, it provides site's data of user after 24 hours of visiting the site, for analysis purpose.

8.2 Future Extensions

- I can improve this project to prefetch more accurate pages from web application with the use of Google BigQuery subscription because it will provide me more managed data and many other advance functionalities, that too in real time.

- I have implemented this project for very small dummy web application which I have created, can extend this to improve page load times for richer data sets using large scale web-sites which people use in real life. I can provide large data storage for model training examples as currently I don't have it.

9. Bibliography

https://www.tensorflow.org/tutorials

https://towardsdatascience.com/how-to-deploy-tensorflow-models-to-the-web-81da150f87f7

https://developers.google.com/analytics

https://cloud.google.com/bigquery

https://stackoverflow.com/

https://docs.python.org/3/

https://www.google.com/

https://angular.io/

https://angular.io/guide/service-worker-intro

https://blog.bitsrc.io/prefetching-links-using-service-workers-d9f6babfd0b

https://developer.chrome.com/docs/devtools/network/