

Software Design

Specification

Traffic Intensity Prediction

**Table of Contents**

**Table of Contents ii**

**1.** **Introduction 1**

1.1 Problem Statement 1

1.2 Purpose 1

1.3 List of abbreviations 1

1.3 Technologies 1

**2.** **Design and Architecture 1**

2.1 Process Flow 1

2.2 User Interface 1

2.3 Validations

**3.** **Database Schema 1**

3.1 Database 1

3.2 Tables 1

3.3 ER Diagram

3.4 Data-Flow Diagram 1

# Introduction

## Problem Statement

Web application design and implementation of the Traffic Intensity Prediction (TIP), an Artificial Intelligence (AI) project was taken up and carried out by a group of two graduate students from Dr. Vishwanath Karad MIT World Peace University, Pune, in <mention date (not finalized)> 2023, based on the assignments and requirements mentioned and proposed by authorities at Centre for Development of Advanced Computing.

The (TIP) web application is still undergoing development. The system has its roots in various technological fields. The TIP is designed and implemented for prediction purposes of the intensity of traffic at a particular junction, given an input of the junction and the amount of time it needs the prediction of. The application is based on a to be trained Machine Learning model on which the provided data is predicted against.

Several algorithms are going to be reviewed and studied for the future implementation and development of the application

## Purpose

Traffic prediction plays an essential role in intelligent transportation systems. Accurate traffic prediction can assist route planning, guide vehicle dispatching, and mitigate traffic congestion.

The modern city is gradually developing into a smart city. The acceleration of urbanization and the rapid growth of urban population bring great pressure to urban traffic management. The Intelligent Transportation System (ITS) is an indispensable part of smart city, and traffic prediction is an important component of ITS. Accurate traffic prediction is essential to many real-world applications. For example, traffic flow prediction can help cities alleviate congestion; car-hailing demand prediction can prompt car-sharing companies to preallocate cars to high demand regions. The growing available traffic related datasets provide us potential new perspectives to explore this problem.

Creation of a web application of a desktop/android application makes it very faster and easier to operate and look at the ground truth reality without having the burden of doing manual setup, installations and configurations for an inexperienced user.

## Technologies

**1.3.1 Client Software (Front End)**

Front-end development is programming done on the front end of a website (Client Software) – the part that the user views and interacts with. The technologies employed for developing this software are Node.Js, Angular/CLI, HTML, CSS, JavaScript and TypeScript.

1. Node –version: 18.12.0
2. Angular CLI –version: 15.0.5
3. Package Manager: npm 8.19.2
4. Typescript –version: 4.8.4
5. HTML
6. CSS
7. JavaScript

**1.3.2 Server Software (Back End)**

If the front end of a website is the user interface and navigation, the back end is the nuts and bolts. It makes no difference how well a website looks or functions if it lacks content. Java, Spring Boot, Python are the technologies used for developing the backend for this system.

1. Java –version: 19.0.1
2. Python –version: (3.9.13)
3. Python Libraries:
   1. datetime
   2. flask
   3. flask-cors
   4. numpy
   5. pandas
   6. matplotlib
   7. seaborn
   8. scipy
   9. scikit-learn
   10. statsmodels
4. Maven –version: 4.0.0
5. Spring –version: 3.0.1

## List of Abbreviations

| **Sr. No** | **Acronym** | **Naming Convention** |
| --- | --- | --- |
| 1. | ER | Entity Relationship |
| 2. | TIP | Traffic Intensity Prediction |
| 3. | AI | Artificial Intelligence |
| 4. | MIT | Maharashtra Institute of Technology |
| 5. | ITS | Intelligent Transportation System |
| 6. | CLI | Command Line Interface |
| 7. | HTML | Hyper Text Markup Language |
| 8. | CSS | Cascading Style Sheet |

Table 1: Abbreviations

# Design and Architecture

## Process Flow

In the application, the user needs to upload the preferred dataset, choose and enter the training to testing to validation ratio that the data has to be divided into and just start the process by clicking the button. Then, the system checks whether the ratios add up to 1. If not then it reports back to the user with a warning message. At the start of the process, the application screen shows the visualization of the data uploaded. Then, the training process of the model starts. The user has to input his/her conditions and constraints for prediction. The application shows the results in two formats, graph and text.

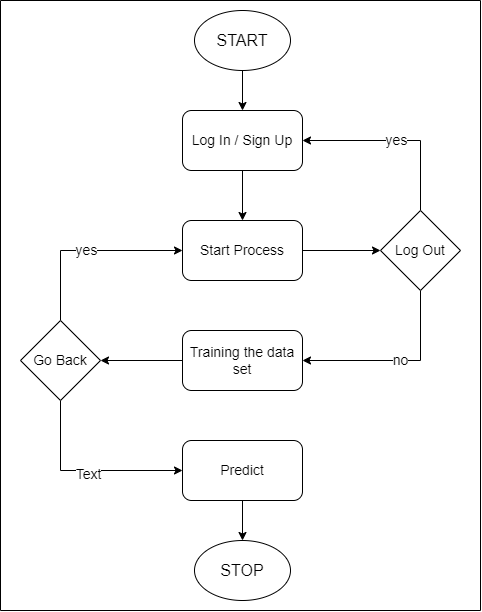


Fig 1. Process Flow Diagram

## User Interface

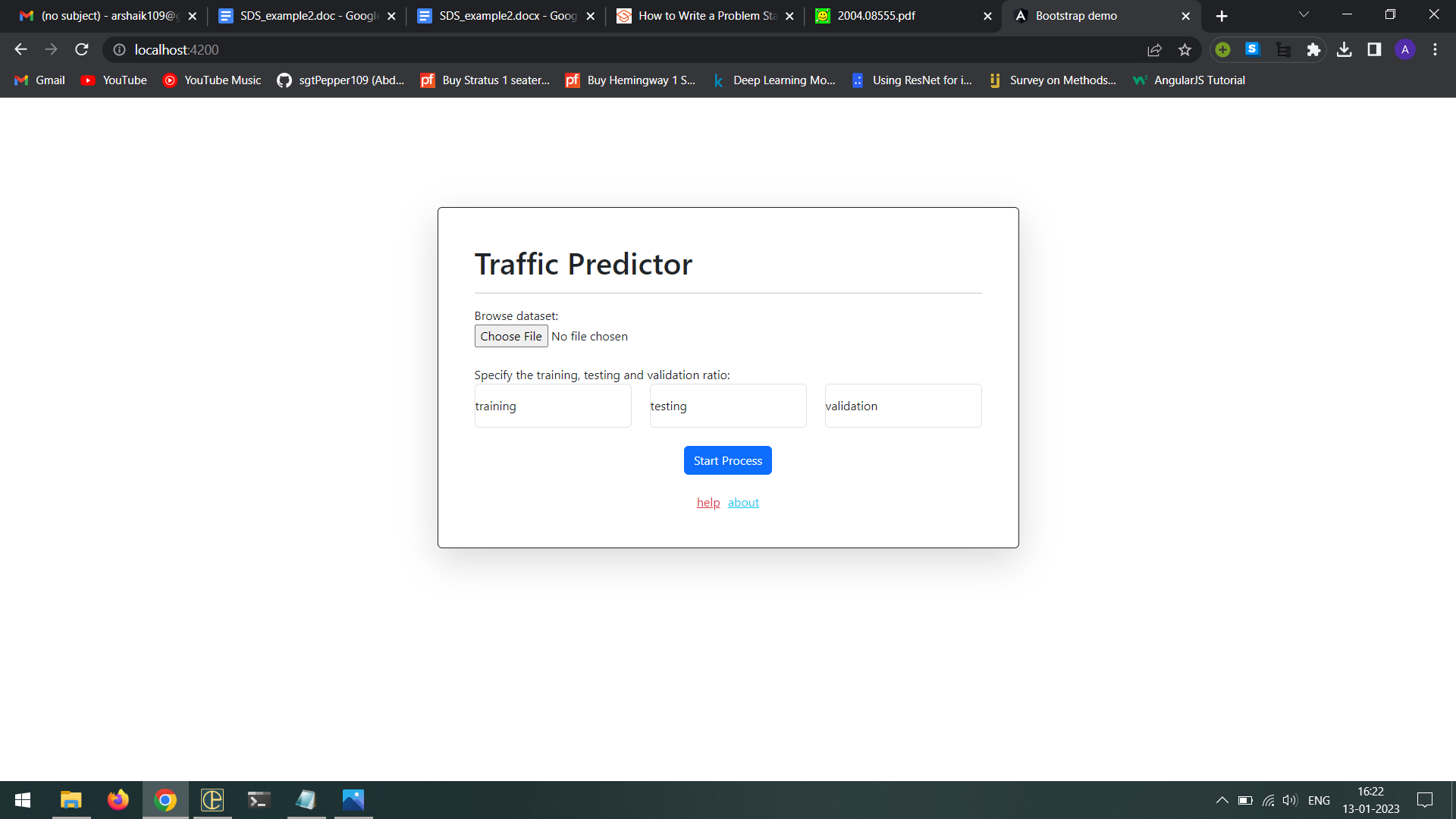


Fig 2. Screen 1 - Home Page

The home page asks for the data set input and the input for training, testing and validation ratios respectively to start the further process of training the same and prediction.

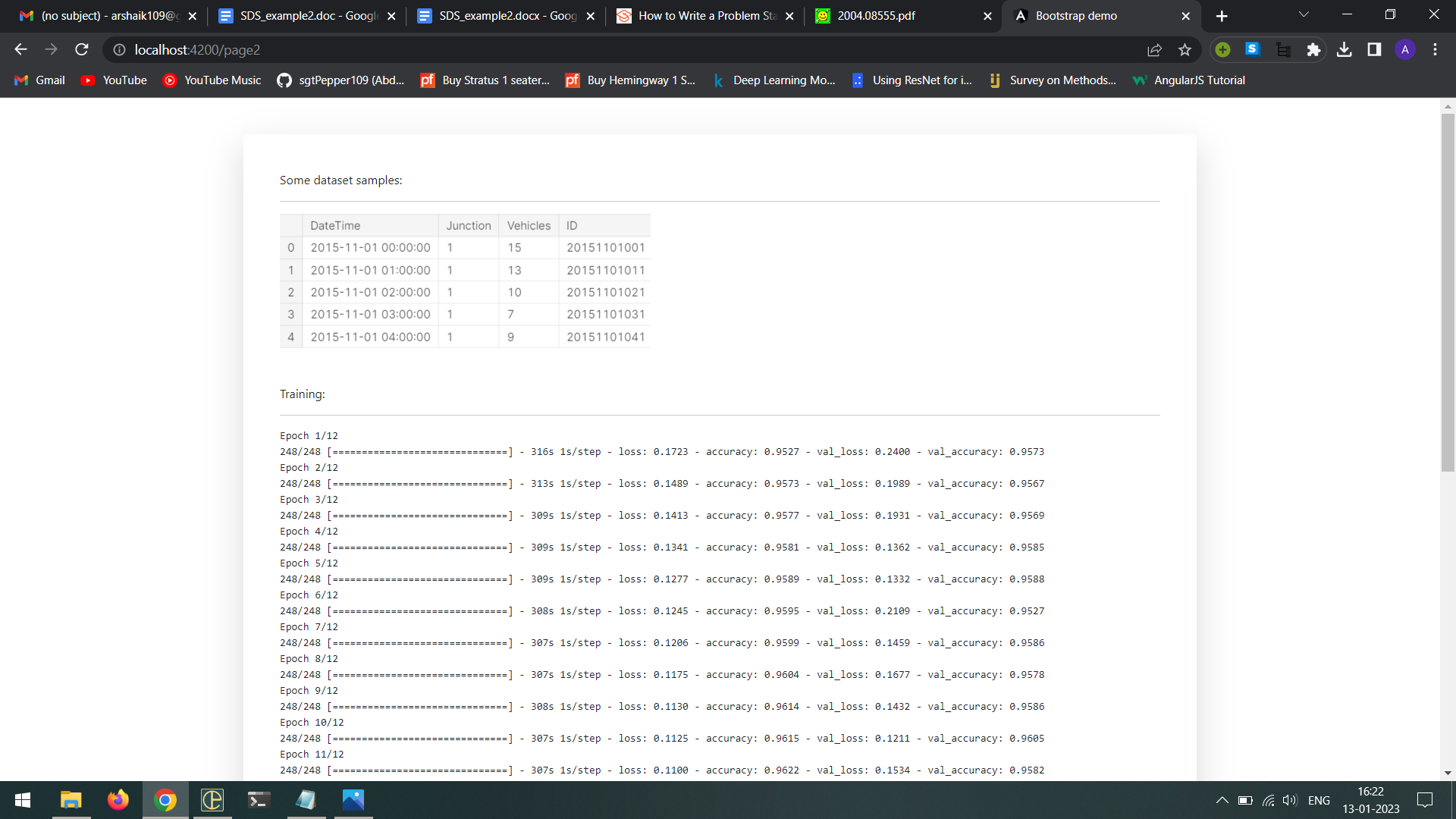


Fig 3. Screen 2

The training and visualization of data is done in the main page of the application with the help of python.

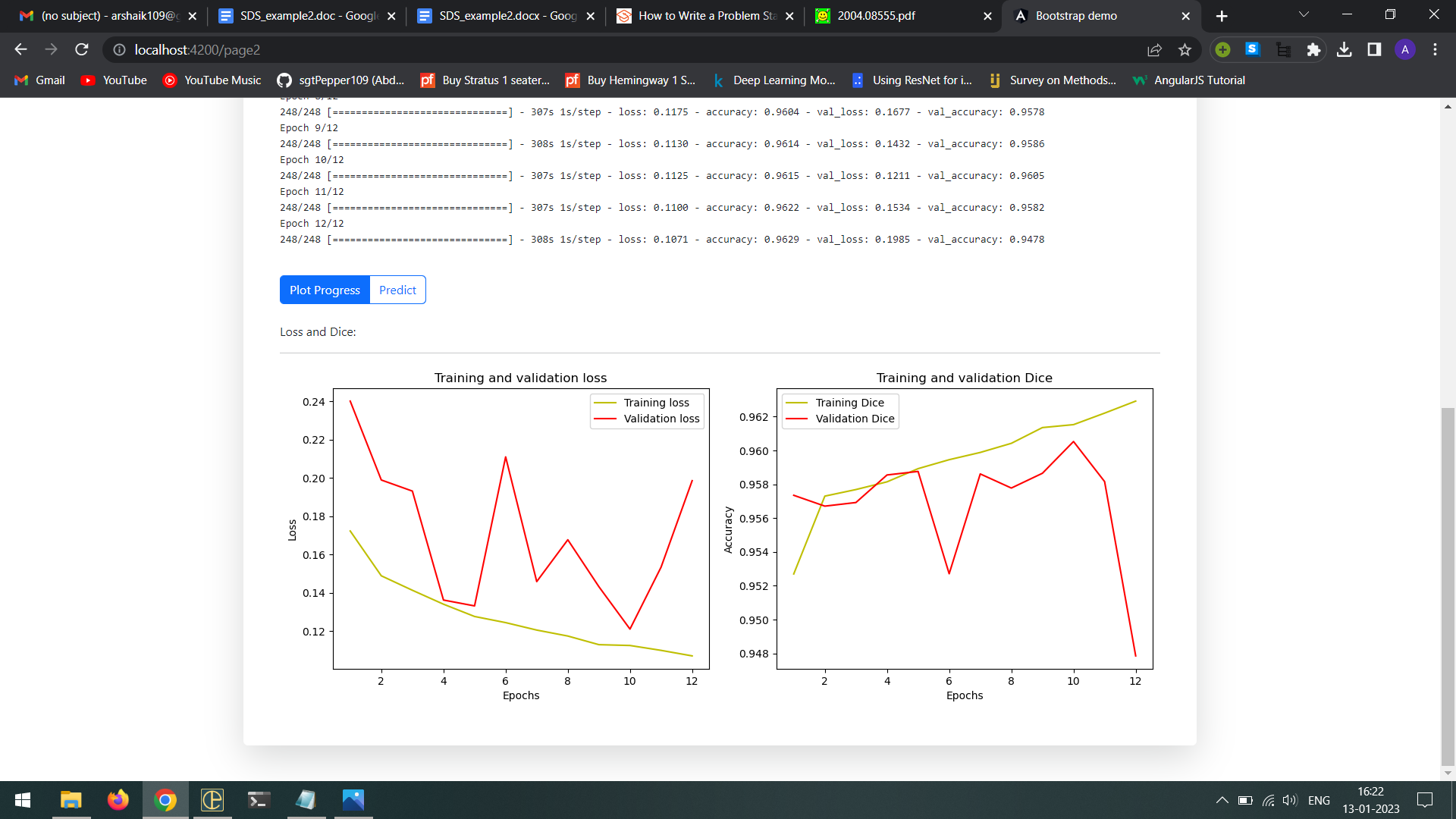


Fig 4. Screen 2 Graphs

* 1. **Validations**

| **Sr. No** | **Field / Control** | **Rule** |
| --- | --- | --- |
| 1. | dataset\_path | The dataset\_path provided must not be empty. The user has to provide a dataset so that the application works |
| 2. | training\_ratio | The training ratio must not be empty |
| 3. | test\_ratio | The testing ratio must not be empty |
| 4. | val\_ratio | The validation ratio must not be empty |
| 5. | train\_ratio, test\_ratio, val\_ratio | All the ratios must add up to 1 |
| 6. | addOperation() | The function tries to add the data to the database and checks whether the query was successful on which the further actions proceed |
| 7. | dataset\_type | The dataset\_type is checked if it has .csv or a .xlsx or a .data extension |
| 8. | getPlot() | This function tries to get the line plot of the probability of number of vehicles at the given junction and checks if the request is completed upon which display is done |
| 9. | inputJunction | The inputJunction variable has only 5 values i.e. [‘Choose Junction’, 1, 2, 3, 4] from which only the last 4 numeric digits are allowed. |
| 10. | inputMonths | The inputMonths variable must not be empty. It is needed for prediction |
| 11. | predict() | The predict method tries to get the predicted plots and data from the backend and only when there is a response, the plots are shown |
| 12. | toggleError | If the validations on page 1 are incorrect then this variable shows the error message on the screen |
| 13. | plotReadyFor1 | shows probability plot of junction 1 when true |
| 14. | plotReadyFor2 | shows probability plot of junction 2 when true |
| 15. | plotReadyFor3 | shows probability plot of junction 3 when true |
| 16. | plotReadyFor4 | shows probability plot of junction 4 when true |
| 17. | predictionImageReady | shows the predicted plot if true |

Table 2. Validations

# Database Schema

## Database

A single database needs to be created which comprises of two tables i.e. users, which is used to store the login and signup information of the user and operation\_info which is used to store the dataset location on the test system and the training, testing, validation ratio for the continued process.

## Tables

Two tables are linked, stored and represented well in the dataset with the help of key constraints. The operation\_id table is linked to the users table with user\_id as a foreign key in operation\_id table to associate the choice of dataset the user prefers.

| Field | Type | Null | Key | Extra | Remarks |
| --- | --- | --- | --- | --- | --- |
| operation\_id | int | NO | PRI | auto\_increment | Store the id of the current operation, a primary key |
| dataset\_path | varchar(255) | NO |  |  | stores the data set's path chosen |
| train\_ratio | int | NO |  |  | stores the training ratio provided by the user |
| test\_ratio | int | YES |  |  | stores the testing ratio provided by the user |
| val\_ratio | int | YES |  |  | stores the validation ratio provided by the user |
| user\_id | int | NO | MUL |  | Refers to **users(id)**, a foreign key |

(Table 3: operation\_info)

| Field | Type | Null | Key | Extra | Remarks |
| --- | --- | --- | --- | --- | --- |
| id | int | NO | PRI | auto\_increment | stores the user id, a primary key, linked to operation\_info(user\_id) |
| email | varchar(255) | NO |  |  | stores the mail id of the user |
| password | varchar(255) | NO |  |  | stores the hash of the password provided by the user |

(Table 4: users)

## Entity-Relationship diagram

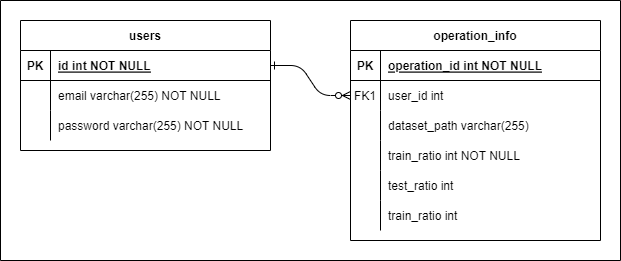


Fig 5. Entity-Relationship Diagram

## Data-Flow diagram

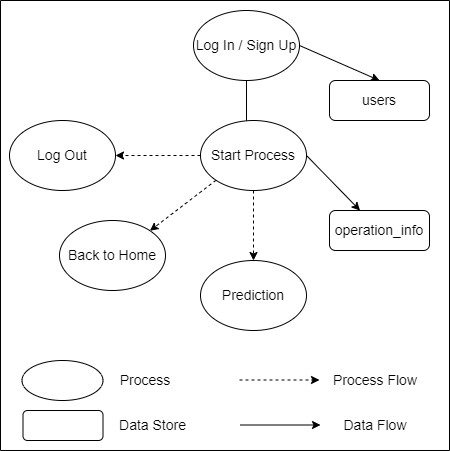


Fig 6. Data Flow Diagram