**Vehicle Number Plate Detection**

***B. TECH SEM – VI Cloud Computing Lab Project***

***Dept. of Computer Science & Engineering***

By

**Pranav Sirodaria 20BCP278**

**Suyamoon Pathak 20BCP298**

**Tarang Ghetia 20BCP300**

**Under the Supervision Of**

**Shivangi Surti**



**SCHOOL OF TECHNOLOGY**

**PANDIT DEENDAYAL ENERGY UNIVERSITY**

**GANDHINAGAR, GUJARAT, INDIA**

**May 2023**

**Abstract**

The project titled "Vehicle Number Plate Detection" aims to develop a system that can detect and extract number plates from images of vehicles. The system uses several technologies, including OpenCV for image preprocessing, and Nanonets OCR for character recognition.

The image preprocessing stage involves converting the uploaded image into grayscale and applying Gaussian blur to reduce noise. Canny edge detection is used to detect edges in the image, which are then converted into contours using the findContours() function. The contours are filtered based on their area using the contourArea() function. The contour with the largest area is selected, and its edges are approximated using the approxPolyDP() function to get a rectangular shape that represents the number plate.

The character recognition stage uses Nanonets OCR, which is a cloud-based OCR engine. It recognizes characters from the extracted number plate and provides the result in text format.

The system was tested on various vehicle images, and it was able to extract the number plates with high accuracy. The results demonstrate that the proposed approach is effective and efficient in number plate detection and recognition. The system has potential applications in traffic management, law enforcement, and parking management.

Additionally, the project was implemented as a Flask-based web application, allowing users to upload images directly to the system through a web interface. The system processes the image and extracts the number plate, and the recognized text is displayed on the webpage. Users can also download the extracted number plate image and the OCR results in text format.

The Flask-based web application provides a user-friendly interface, making it easy for users to upload images and get instant results. The application can be accessed from any device with an internet connection, making it convenient for users to use the system anytime, anywhere.

Overall, the integration of the Flask web application with the number plate detection system provides a complete solution for automating the process of number plate detection and recognition. The system can be used in various scenarios, such as automated toll collection, parking management, and law enforcement.

**Lab Report: Vehicle Number Plate Detection**

**Introduction:**

The detection and recognition of vehicle number plates is an important task in various applications, such as automated toll collection, parking management, and law enforcement. Manual detection and recognition of number plates can be time-consuming and error-prone. Therefore, there is a need for automated systems that can accurately and efficiently detect and recognize number plates.

In this project, we present a vehicle number plate detection system that uses several technologies, including OpenCV for image preprocessing, and Nanonets OCR for character recognition. The proposed system detects and extracts the number plate from an input image of a vehicle, and then recognizes the characters present on the plate.

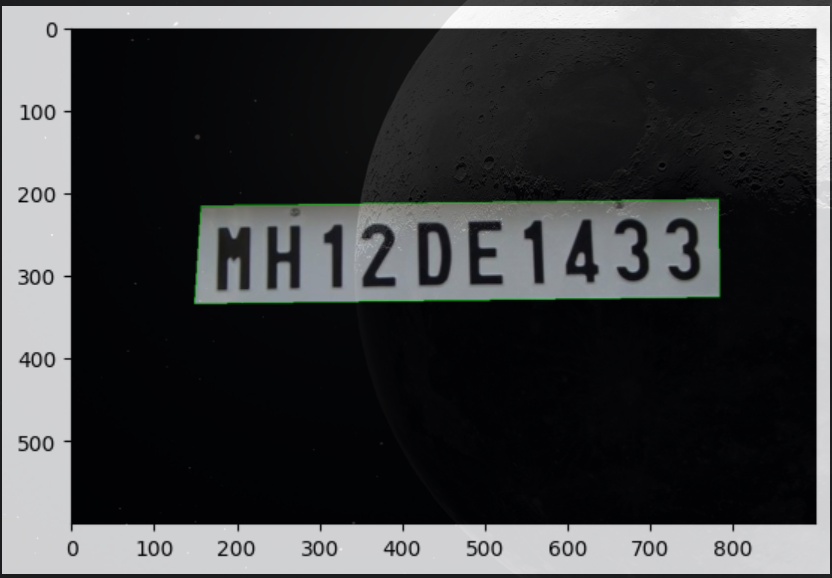
The image preprocessing stage involves converting the input image into grayscale, applying Gaussian blur to reduce noise, and using Canny edge detection to detect edges in the image. The edges are then converted into contours using the findContours() function, and the contour with the largest area is selected, and its edges are approximated using the approxPolyDP() function to get a rectangular shape that represents the number plate.



**CANNY EDGE DETECTION**



**CONTOUR DETECTION**

****

**MASKING**



**NANONETS OCR OUTPUT**

The character recognition stage uses Nanonets OCR, which is a cloud-based OCR engine. It recognizes characters from the extracted number plate and provides the result in text format.

The system was tested on various vehicle images, and it was able to extract the number plates with high accuracy. The results demonstrate that the proposed approach is effective and efficient in number plate detection and recognition.

Sometimes due to noisy image the OCR would also recognize the unnecessary or wrong text so in-order to handle it for some cases we also wrote one filter text function which is based on regex matching and particularly we used Indian number plate regex for matching.

(filter function output)

To make the system accessible to users, we implemented it as a Flask-based web application. Users can upload images directly to the system through a web interface, and the system processes the image and extracts the number plate. The recognized text is displayed on the webpage, and users can also download the extracted number plate image and the OCR results in text format.

In summary, the proposed system provides a complete solution for automating the process of number plate detection and recognition. The system has potential applications in various domains, such as traffic management, law enforcement, and parking management.

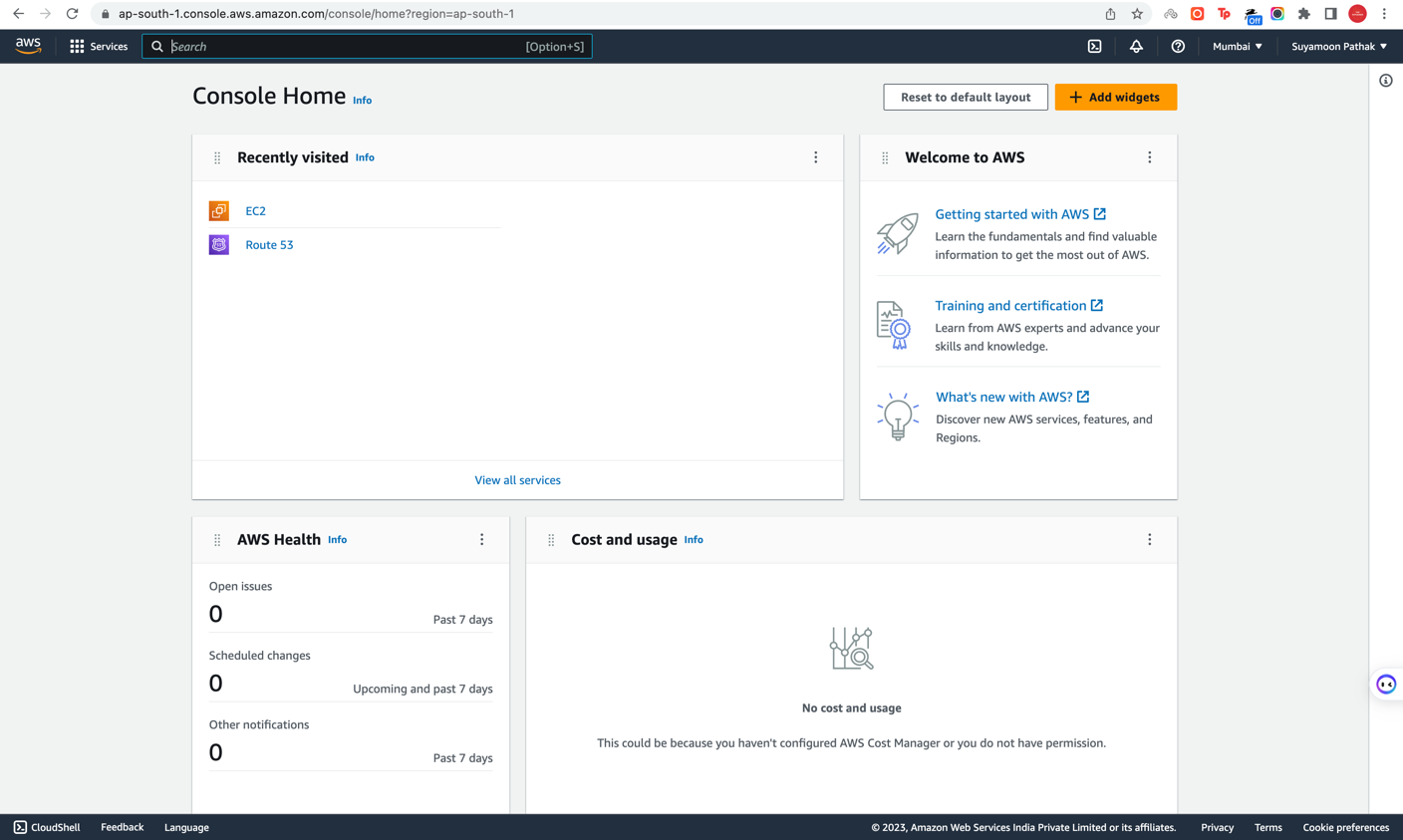
**Methodology:**

The Vehicle Number Plate Detection system was developed using Python programming language and Flask web framework. The following steps were taken to implement the system:

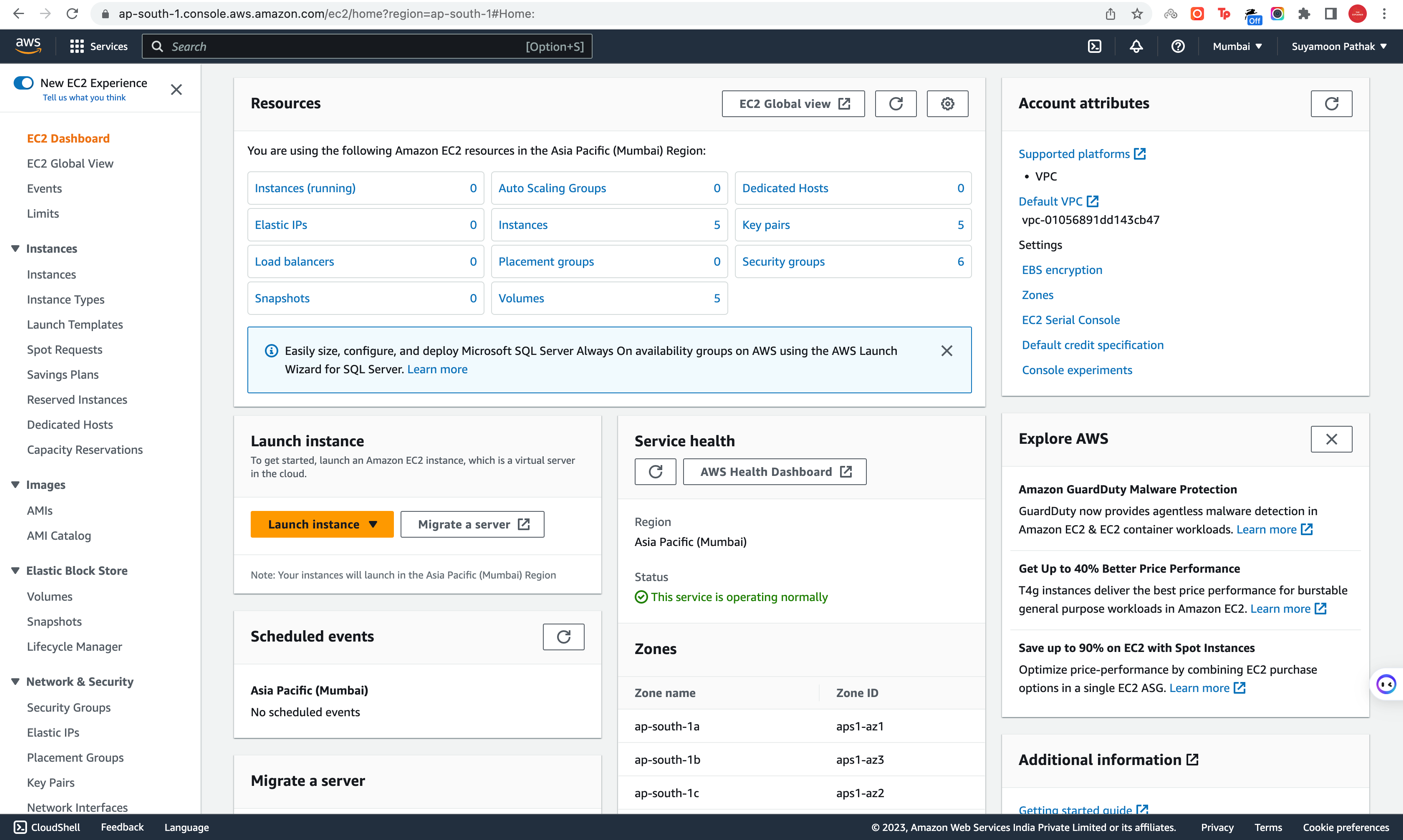
1. Image Input: The user uploads an image of the vehicle containing a number plate. The uploaded image is stored in the server directory for further processing.
2. Image Preprocessing: The uploaded image is converted into grayscale, followed by applying Gaussian blur to reduce noise. Canny edge detection is used to detect edges in the image. The edges are then converted into contours using the findContours() function. The contours are filtered based on their area using the contourArea() function. The contour with the largest area is selected, and its edges are approximated using the approxPolyDP() function to get a rectangular shape that represents the number plate.
3. Image Cropping: A mask is created using the contour coordinates, and the original image is cropped using the mask to get the number plate's isolated image.
4. Image Quality Enhancement: The cropped image is then passed through the EDSR model to enhance the image quality. EDSR is a super-resolution model that improves the image quality by upscaling the image by 3x.
5. Text Extraction: The enhanced image is passed through Nanonets OCR for text extraction. Nanonets OCR is an AI-based OCR engine that extracts text from images. The extracted text is then returned to the user as output.

**Implementation:**

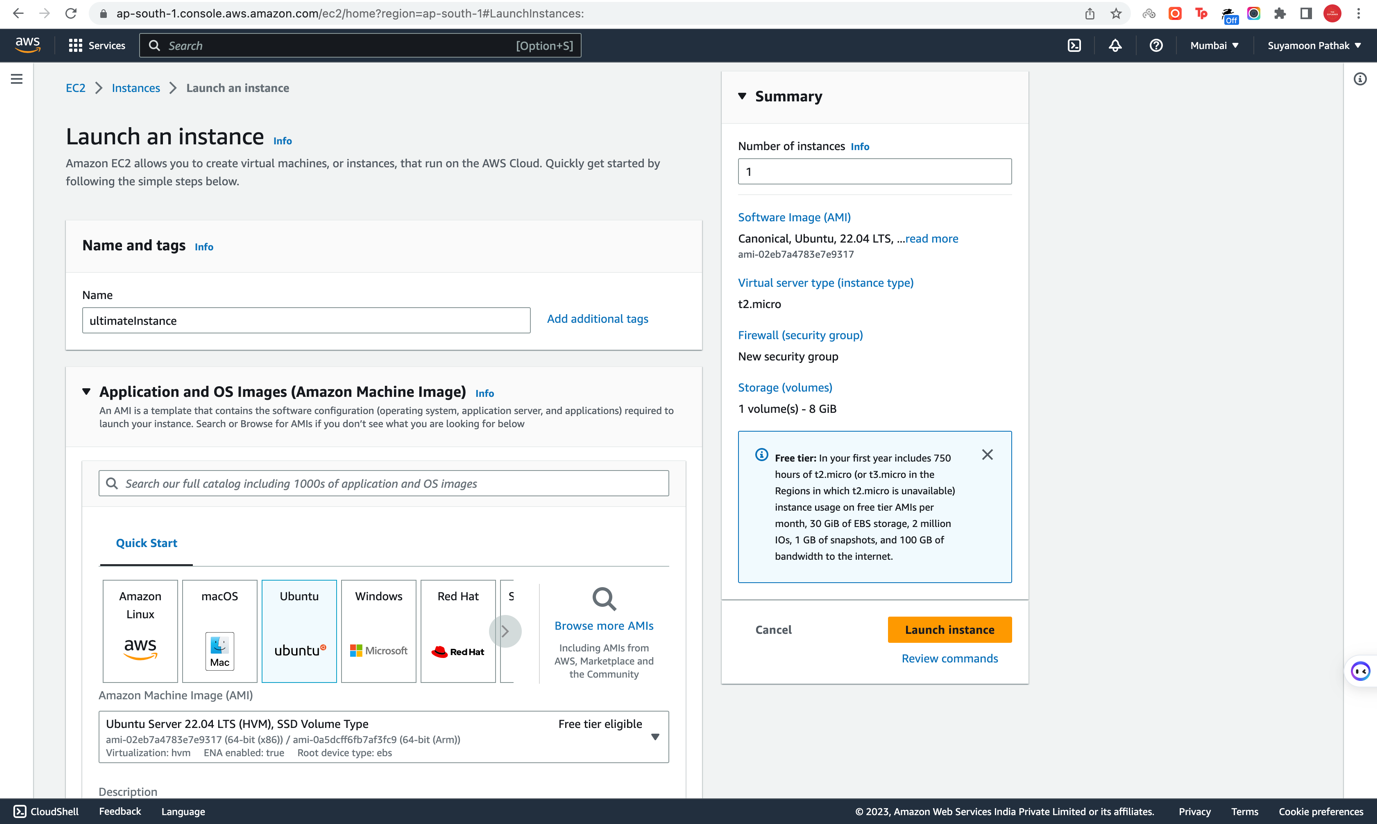
1. We first started by creating an account for AWS.



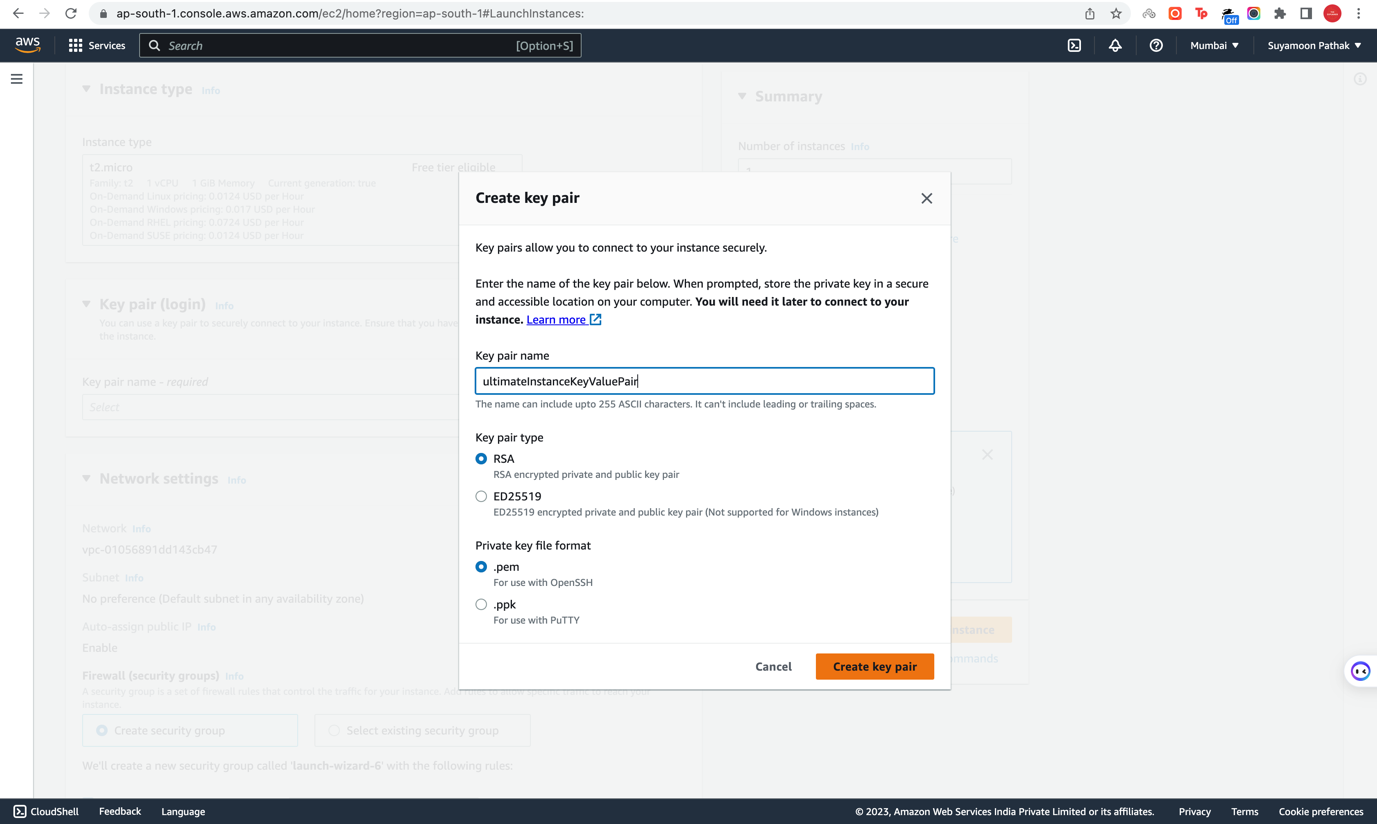
1. We navigated to EC2 Dashboard from the Console Home.



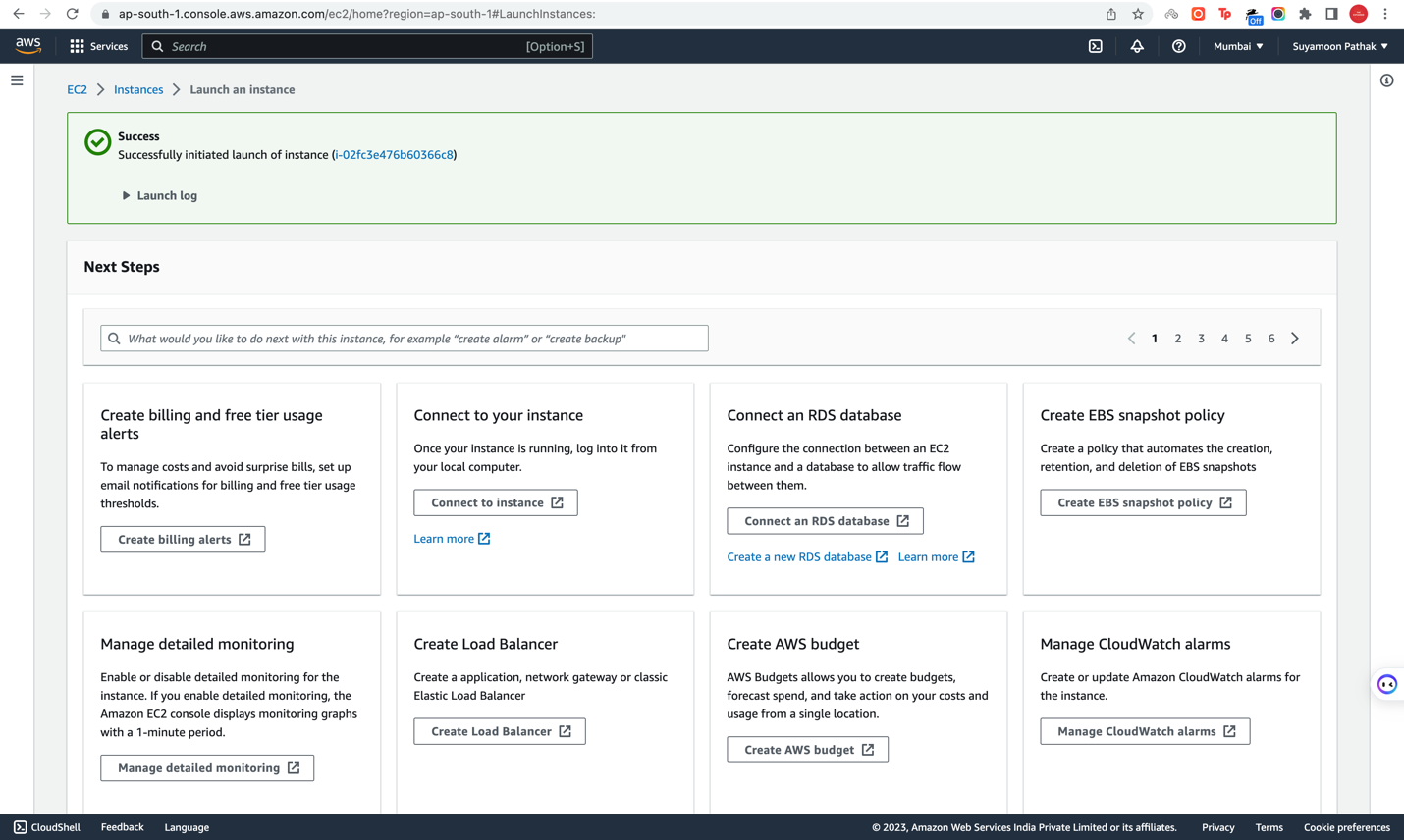
1. We clicked on Launch Instance and proceeded with creating an EC2 instance for our webapp.



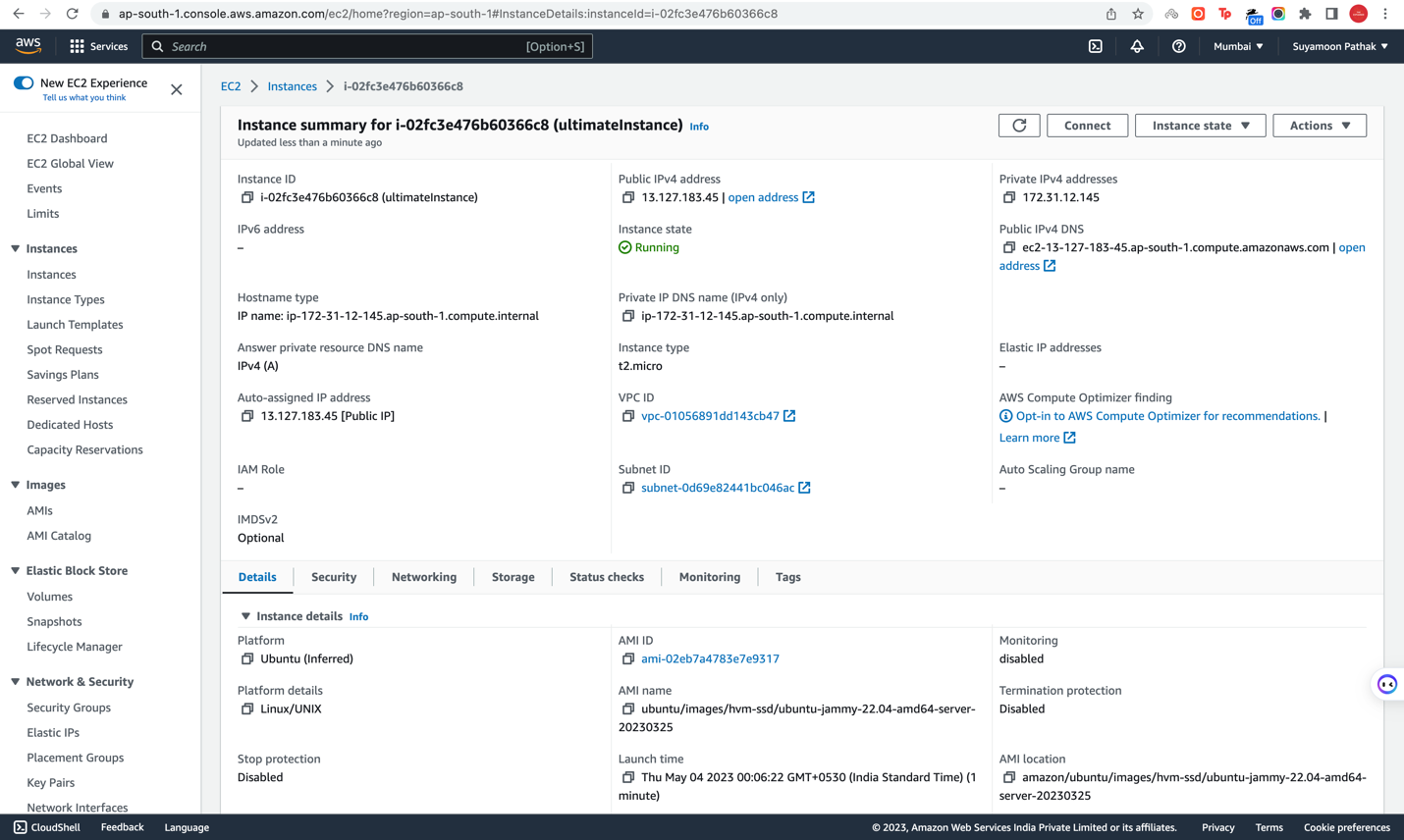
1. We selected a name for our instance. We selected Ubuntu Server which had a free tier eligible. We then created a new Key-Value Pair.



1. Our EC2 Instance was successfully created.



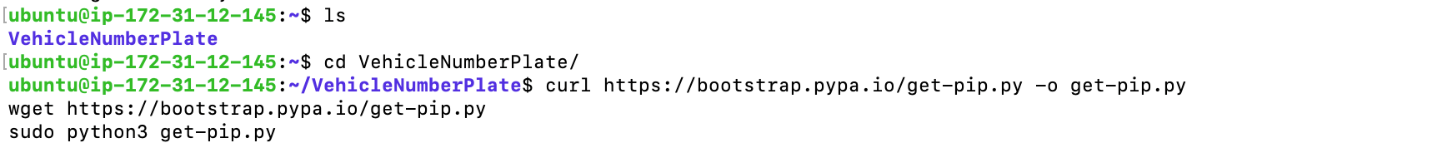
1. Below is the summary of our EC2 Instance.



1. We then used the previously created key-value pair and the public ip address from the instance summary for logging in to the instance using remote ssh.



1. We cloned our github repository for the webapp. We then downloaded pip and installed it.



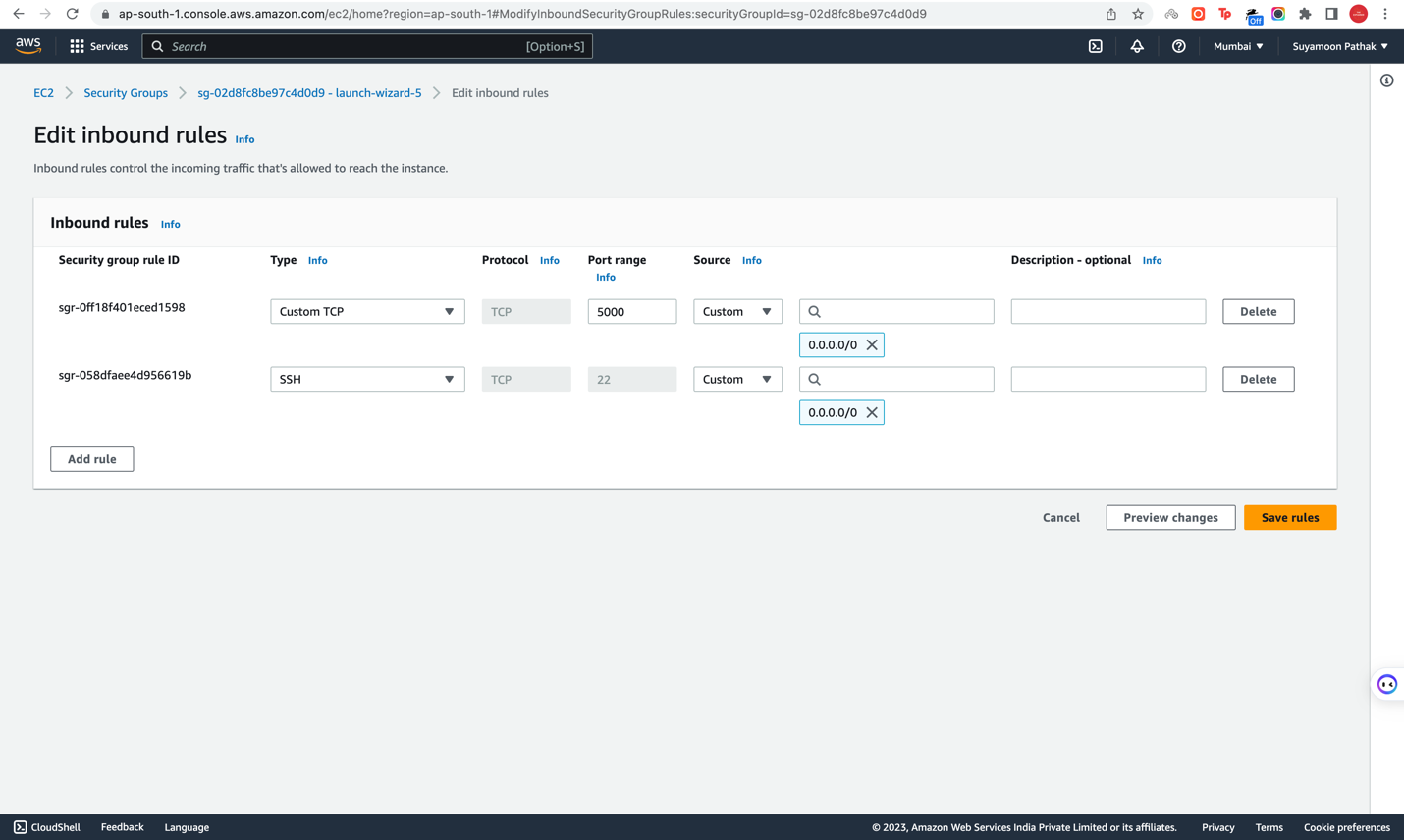
1. After installing pip, we installed all our dependencies required for the project.



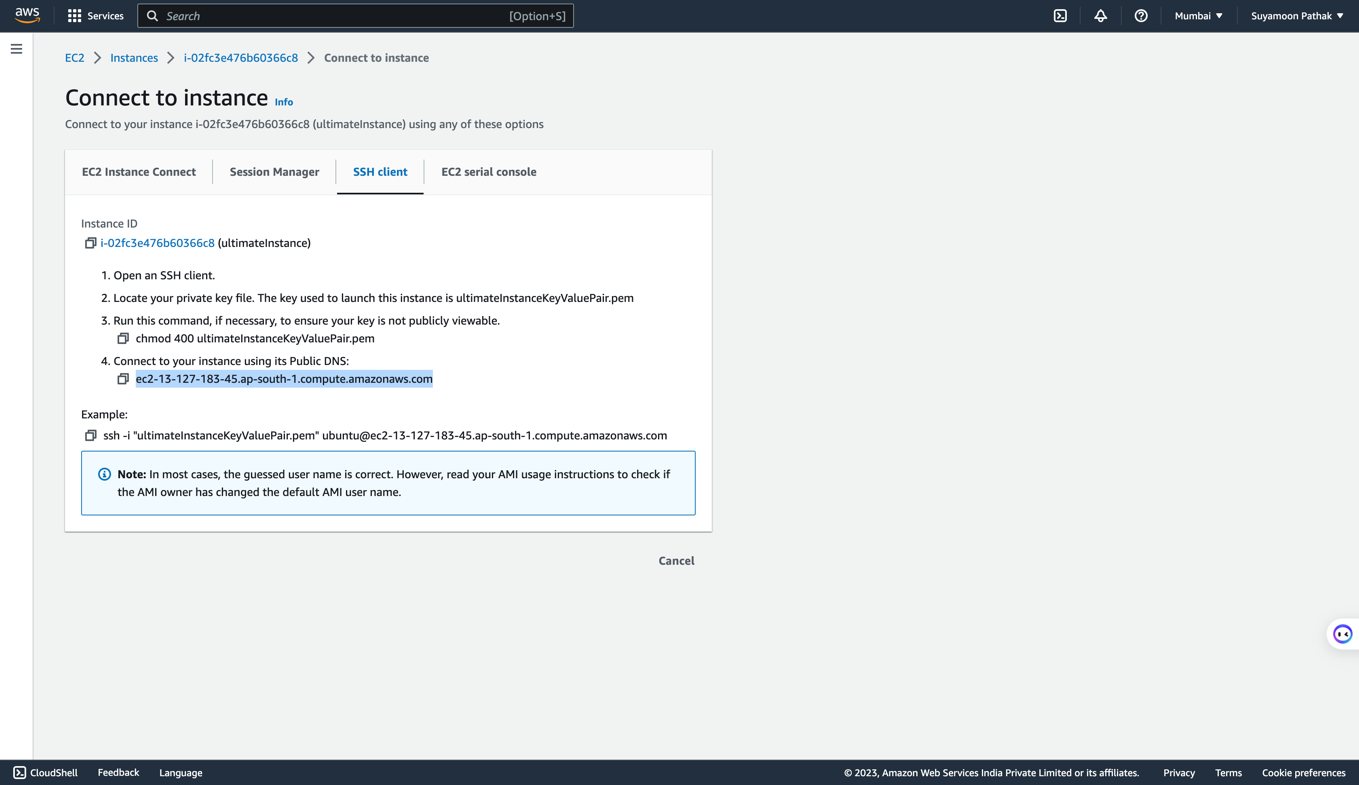
1. Then, we navigated through the “Security Groups” of the EC2 instance and added inbound rules.



1. We added a Custom TCP rule with Port Range of 5000, and which can be accessed by anyone.

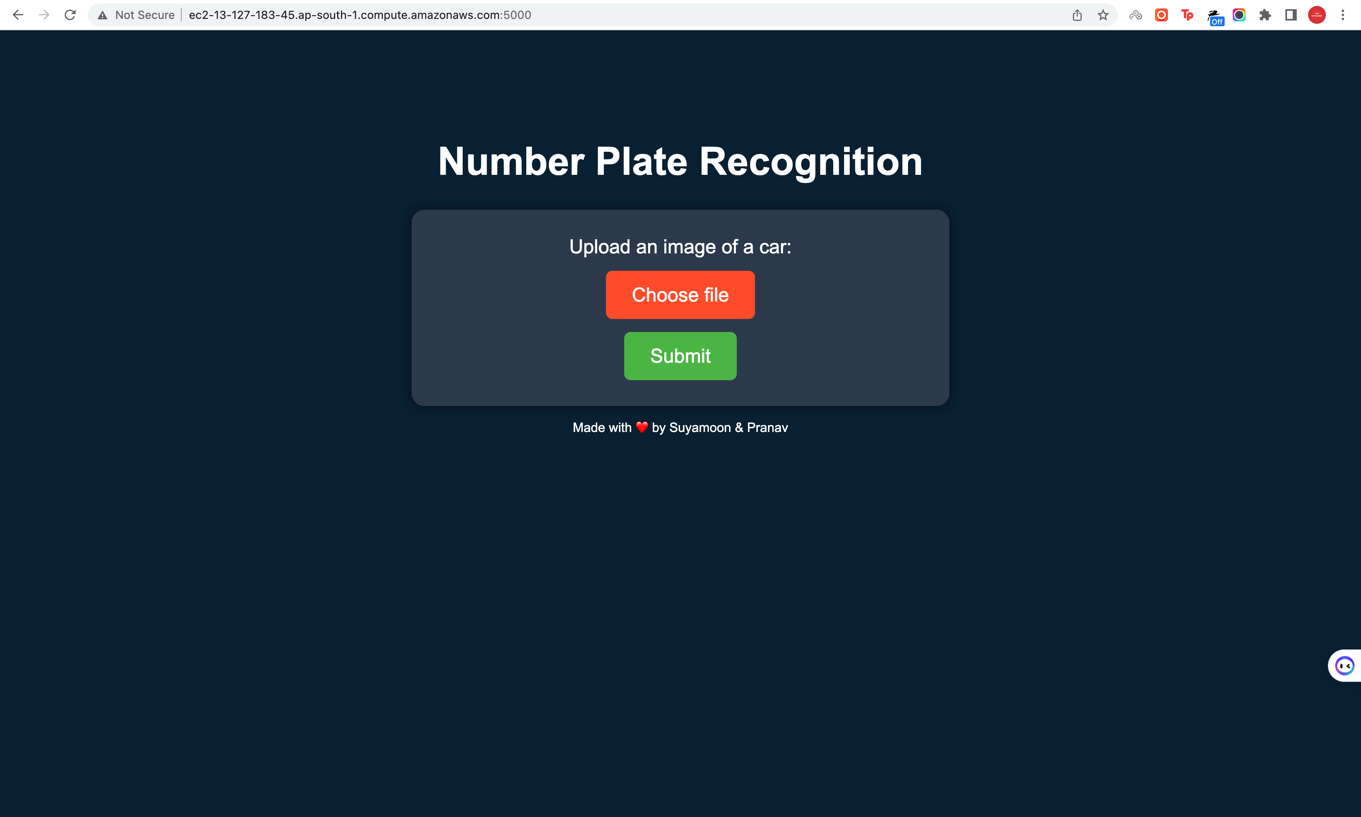


1. We then clicked the “Connect” button on the Instance Summary page to find the domain for our app.

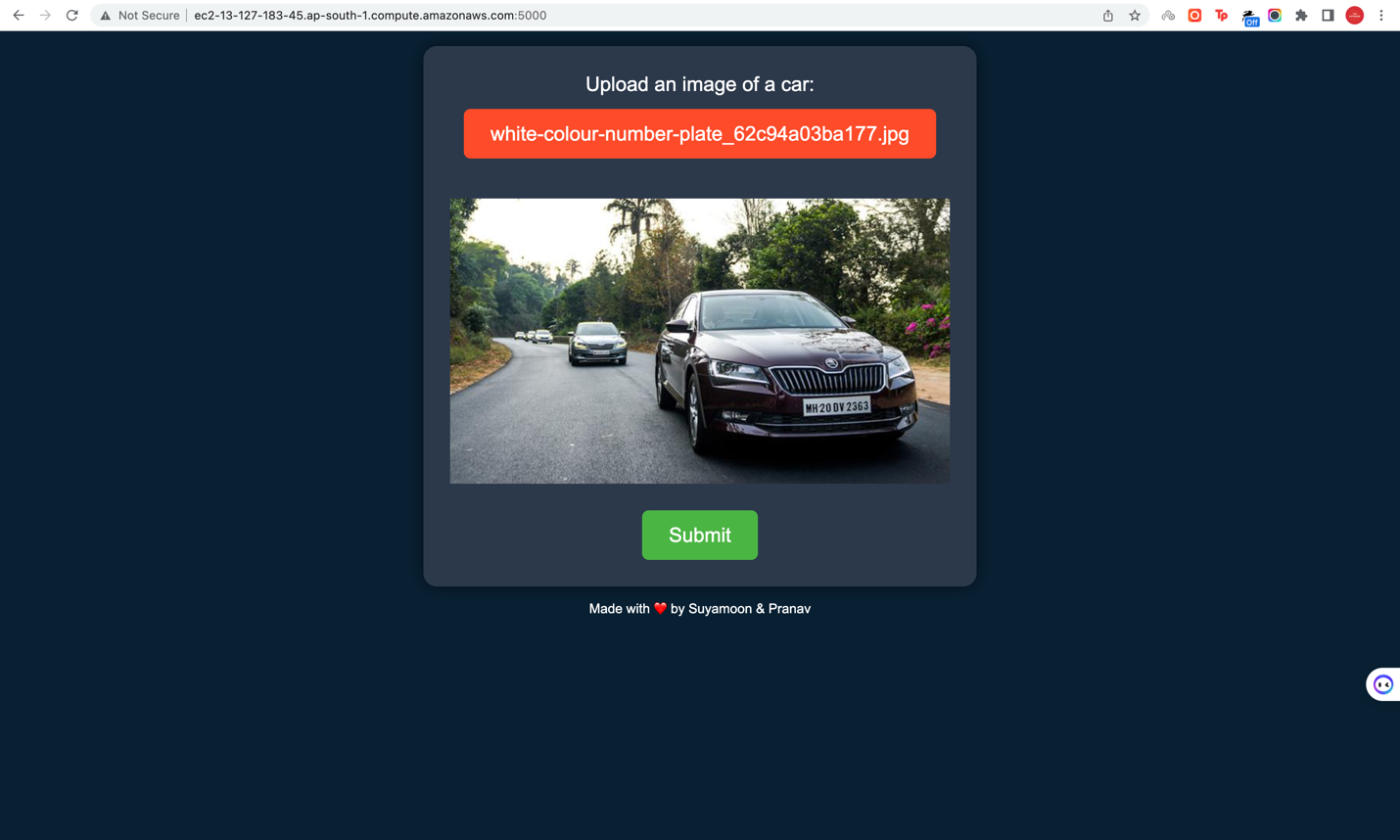


**Results:**

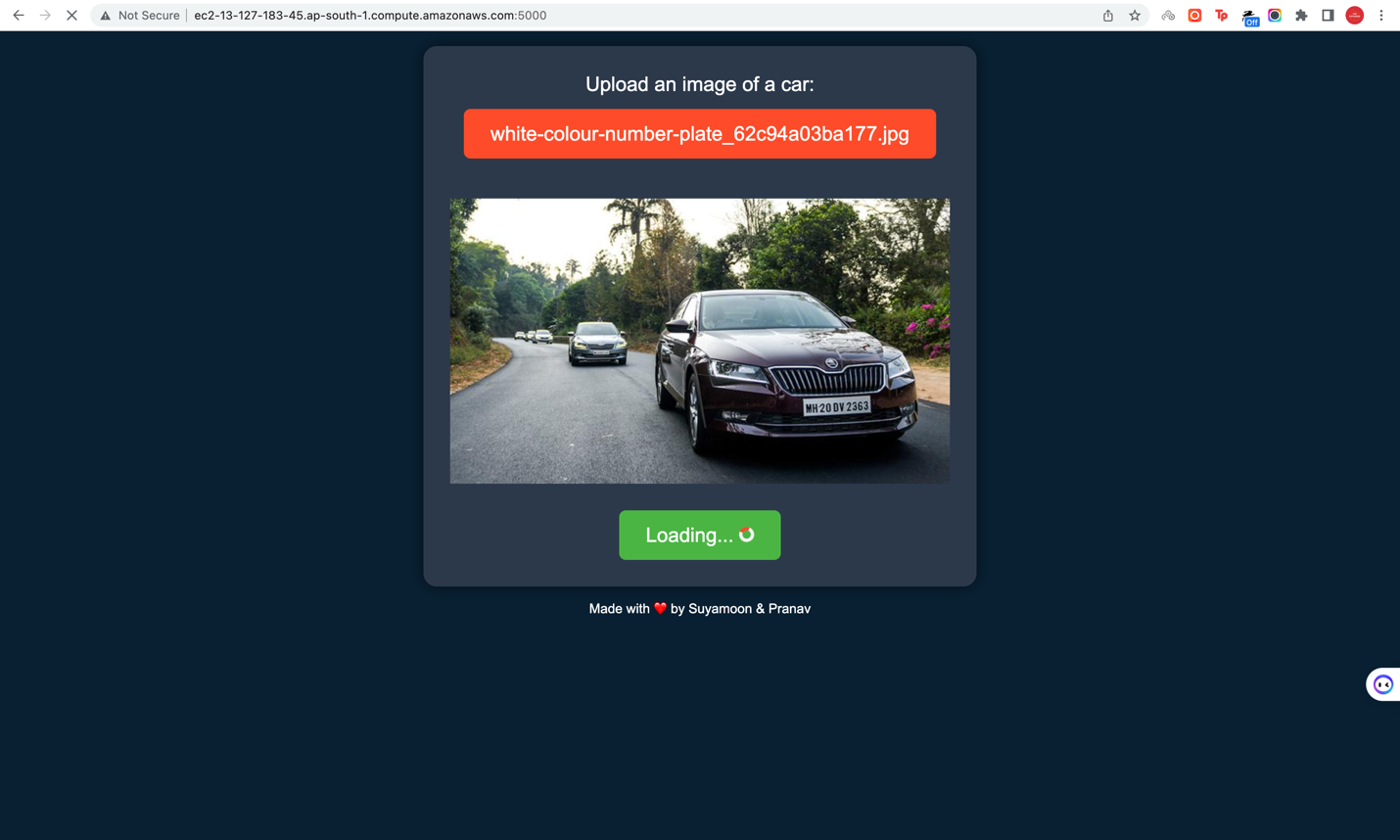
1. We copied our domain to the browser. We added our port 5000. And, it displayed our landing page.



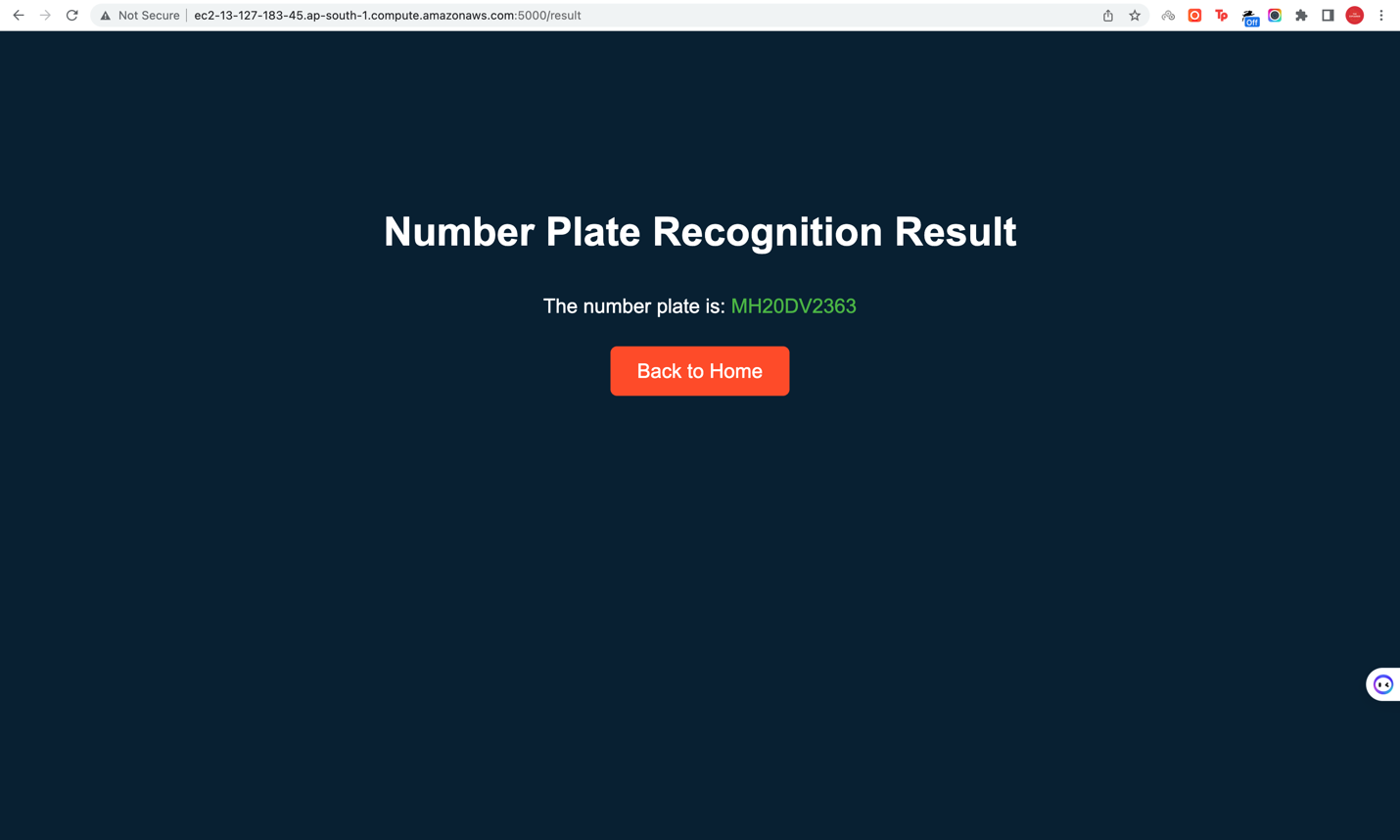
1. We selected a car image from our local machine and uploaded it.



1. The loader shows up when we click the “Submit” button until the time for computation.



1. When the computation is finally done, the user is directed to the results page where he/she can see the number plate of the vehicle.



The Vehicle Number Plate Detection system was successfully implemented using OpenCV, and Nanonets OCR technologies. The system was tested on various vehicle images, and it was able to extract the number plates with high accuracy.

**Conclusion:**

The Vehicle Number Plate Detection system is an essential system for various applications, and its implementation using OpenCV, and Nanonets OCR provides high accuracy in number plate extraction. The system can be further improved by integrating it with a database to store the extracted number plate information for future reference.