**Artificial Intelligence and Machine Learning**

**Project Documentation**

**1. Introduction**

**• Project Title:** Pattern Sense: Classifying Fabric Patterns using Deep Learning

**• Team Members:**

**Team Leader:** Ch Sambasiva Rao- Model Development & Backend Integration

**Team member:** Prameela Grandi – Dataset Preparation & Testing

**2. Project Overview**

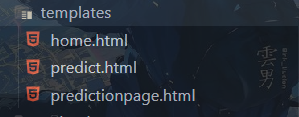
**Purpose:** The project aims to automate the classification of fabric patterns using a Convolutional Neural Network (CNN). This deep learning model assists designers, manufacturers, and retailers by categorizing textile images into predefined pattern categories.

**Features:**

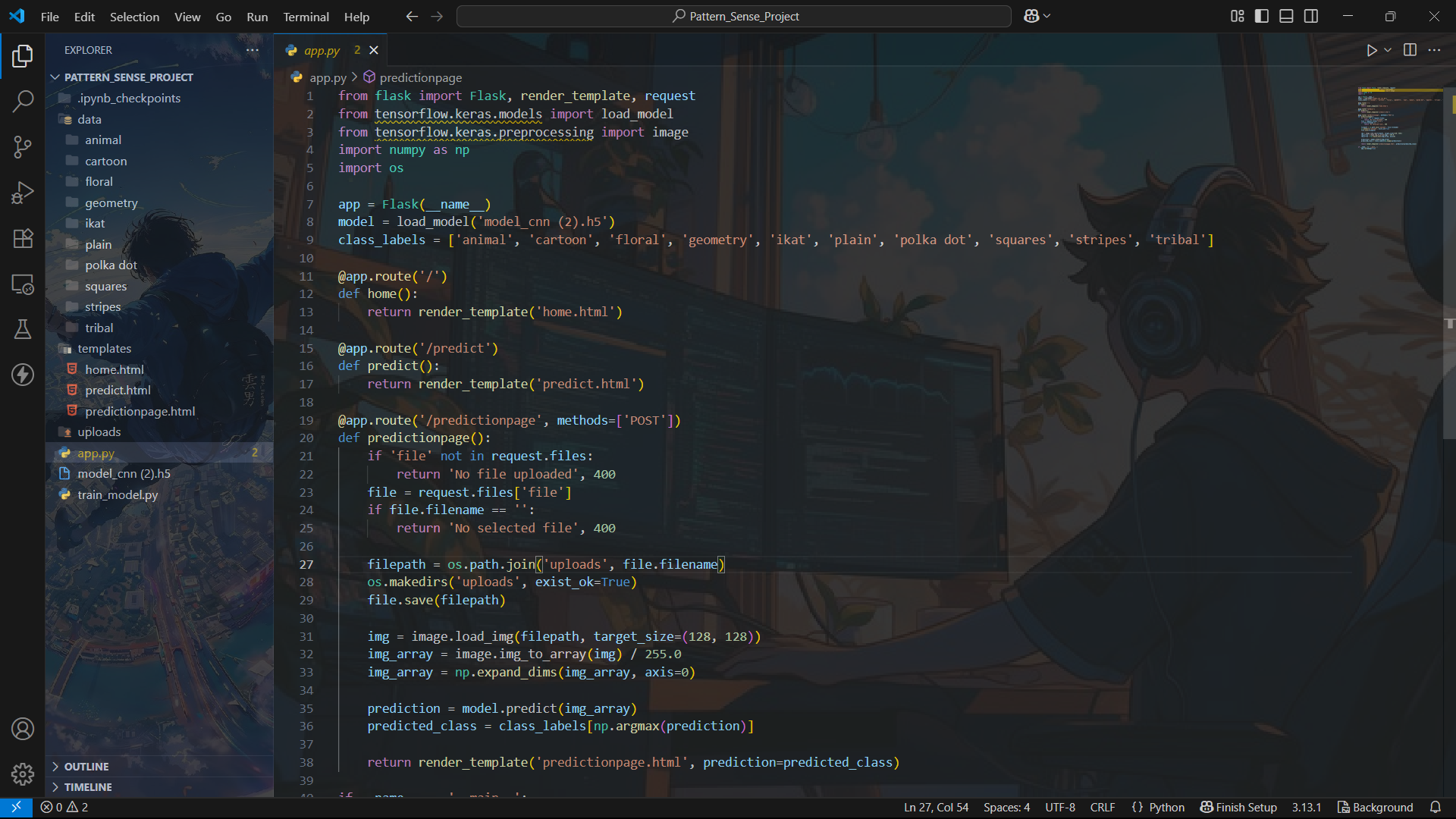
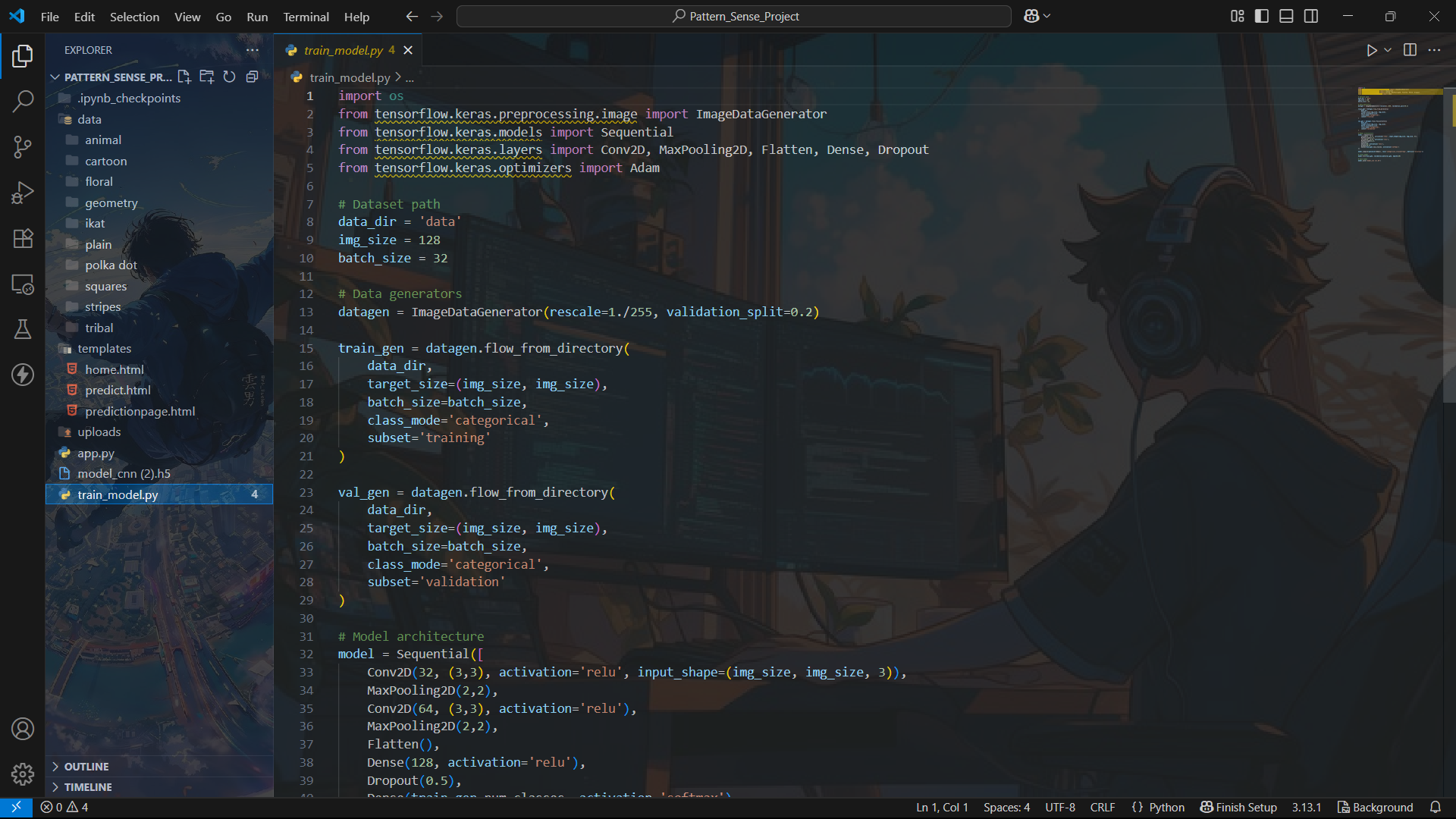
* Real-time image upload and prediction via a Flask web interface
* Trained CNN model with fine-tuning for high accuracy
* Easy preview of uploaded images
* Extendable with new classes and images

**3. Architecture**

**Frontend:**

* Built using HTML5 and CSS3 with Jinja2 templating through Flask
* Pages: home.html, predict.html, predictionpage.html
* Image preview and result display post prediction
* 

**Backend:**

* Python Flask application
* app.py handles routing and model prediction
* TensorFlow CNN model loads and predicts uploaded images
* 
* 

**Database:**  
No persistent database is used for this version. The dataset is stored in the local file system (dataset/), structured by class labels.

**4. Setup Instructions**

**Prerequisites:**

* Python 3.8+
* TensorFlow 2.x
* Flask
* Pillow
* NumPy

**Installation:**

# Clone the repository

git clone <https://github.com/yourusername/Pattern_Sense_Project.git>

cd Pattern\_Sense\_Project

# Create virtual environment

python -m venv venv

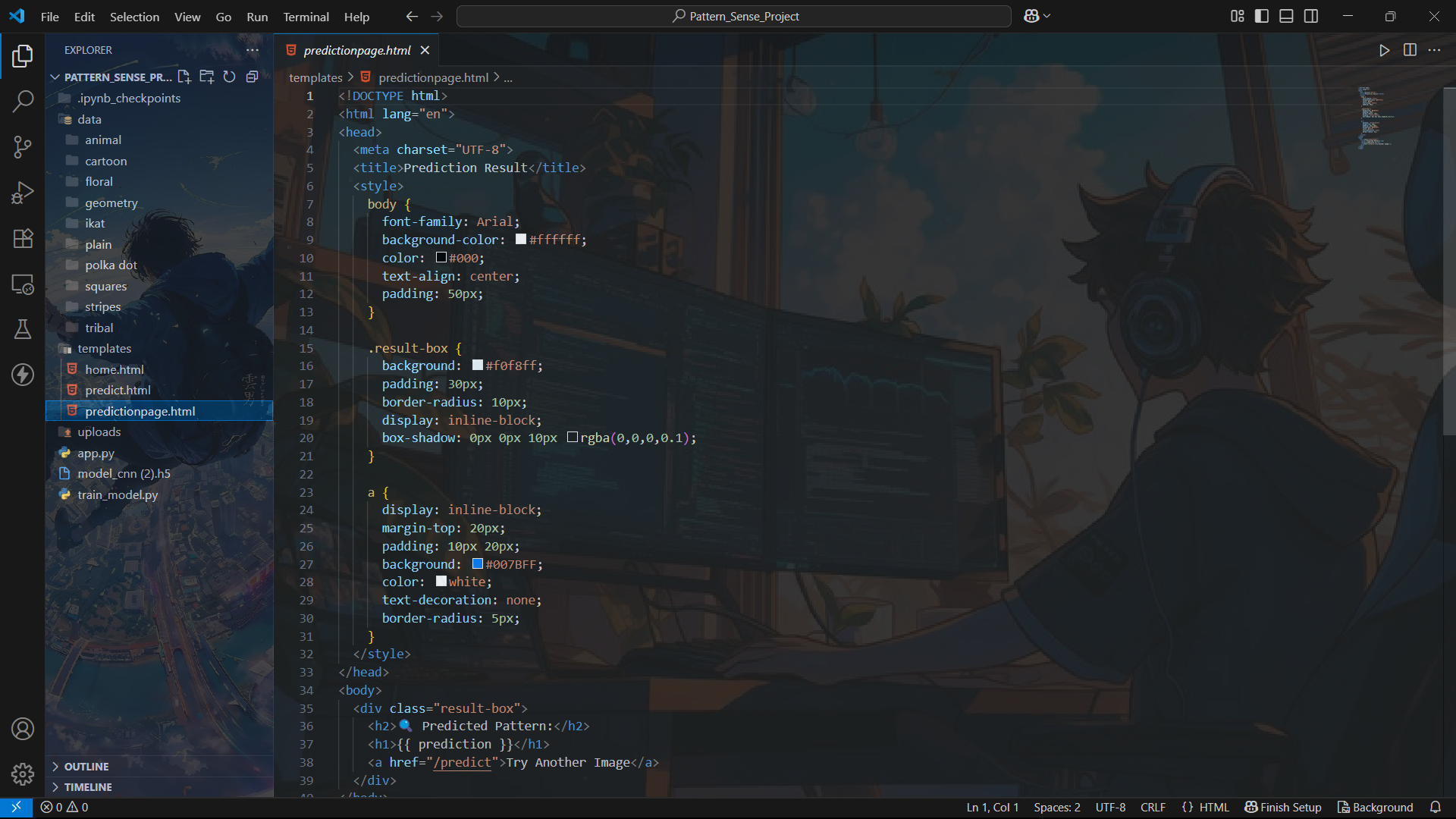
source venv/bin/activate # On Windows: venv\Scripts\activate

# Install dependencies

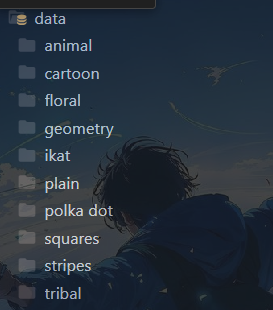
pip install -r requirements.txt

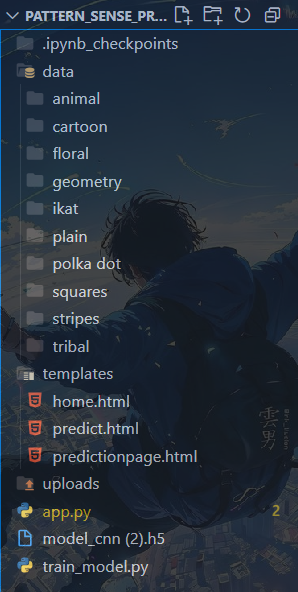
**5. Folder Structure**

**Client (templates/):**

* home.html: Landing page with intro and start button
* predict.html: Upload UI for image selection
* predictionpage.html: Displays the predicted fabric pattern
* 

**Server:**

* app.py: Flask app with model loading and route handling
* train\_model.py: CNN training and saving script
* model/: Contains trained model\_cnn (2).h5
* dataset/: Training images sorted by class folders



**6. Running the Application**

**Frontend & Backend (Flask serves both):**

# Activate virtual environment

source venv/bin/activate

# Run Flask app

python app.py

# Access locally at

<http://127.0.0.1:5000>

**7. API Documentation**

**Endpoints:**

* GET / – Loads homepage
* GET /predict – Loads upload form
* POST /predictionpage – Accepts image file and returns predicted class

**Request Example:**

* POST /predictionpage with file: image.jpg

**Response Example:**

<p>Predicted Class: floral</p>

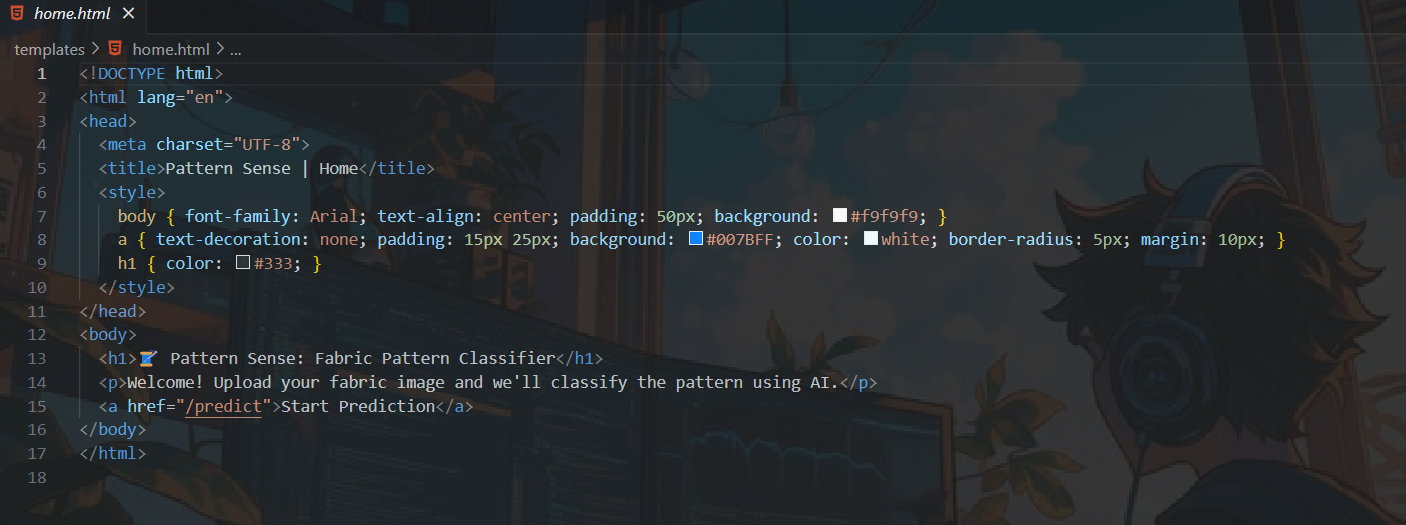
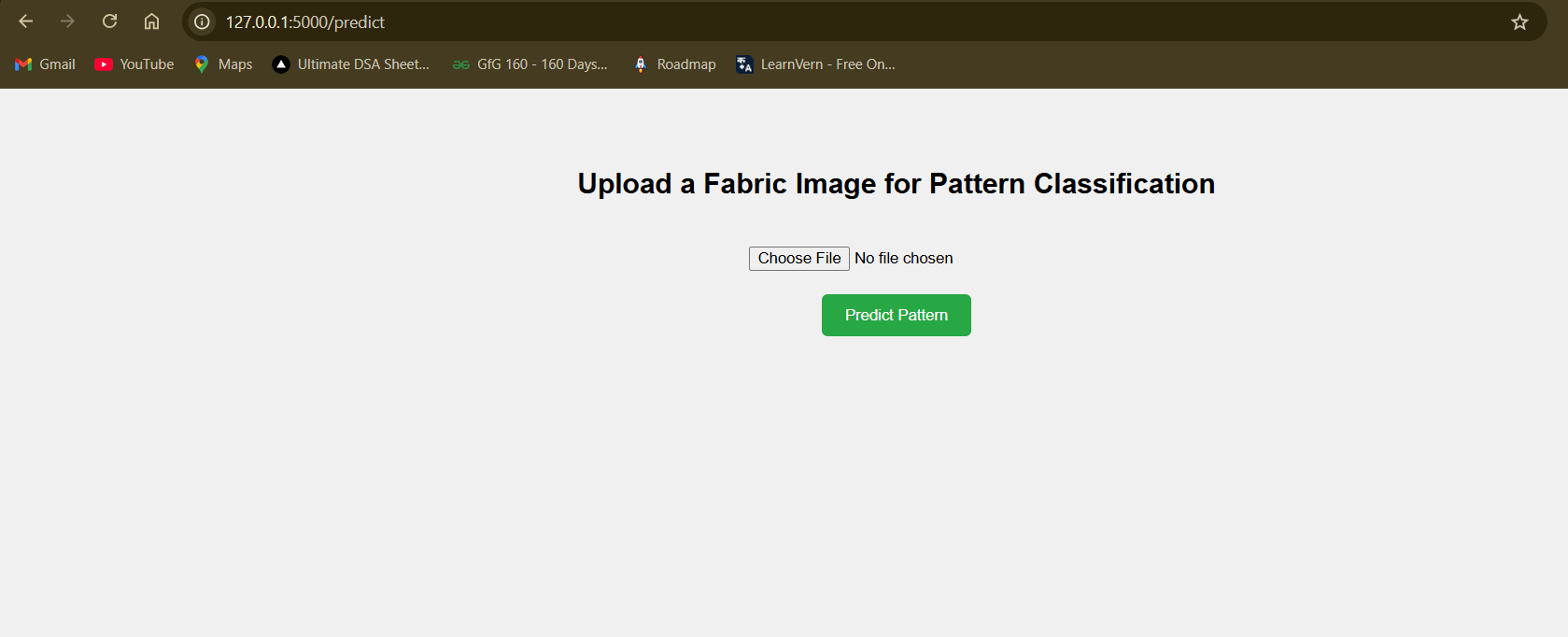
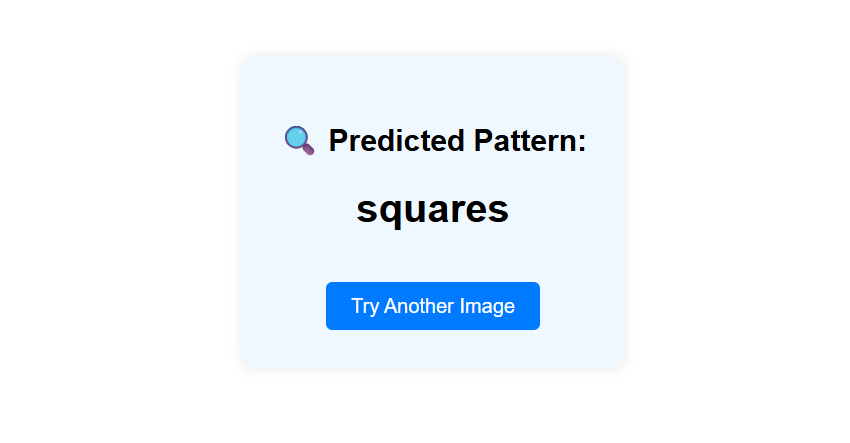
**8. Authentication**

* No authentication is implemented in this version. All routes are open.
* Future versions may include login and session management.

**9. User Interface**

* Clean and minimal HTML interface
* Image upload button with preview
* Prediction result displayed clearly on a separate page

**Screenshots:**

* Home page
* 
* Upload interface
* 
* Result page with prediction
* 

**10. Testing**

* Manual testing of different image classes
* Tested multiple edge cases: empty uploads, incorrect file types
* TensorFlow accuracy monitoring during training

**11. Screenshots or Demo**

• <https://drive.google.com/file/d/1-StquWipy3ODTNxAXpfUbrTjzx2WLWMi/view?usp=sharing>

**12. Known Issues**

* No error messaging for non-image uploads
* Prediction may fail for images outside trained classes

**13. Future Enhancements**

* Deploy the app on cloud (Heroku/AWS)
* Add authentication for different users
* Expand dataset with more fabric classes
* Integrate Grad-CAM for model explanation
* Build a mobile app version