

# SmartBridge Virtual Internship Program 2025



## Pattern Sense

Classifying Fabric Patterns Using Deep Learning

Submitted by

**Team ID:** LTVIP2025TMID45580

Sai Sugandha Sri Ippili

Parimi Venkatesh

Pothineni Praveen Kumar

Pasam Pavan Kalyan



Andhra Pradesh, India

June, 2025

# Contents

<b>Phase 1: Brainstorming and Ideation</b>	<b>2</b>
1.1 Problem Statement . . . . .	2
1.2 Proposed Solution . . . . .	2
1.3 Targeted Users . . . . .	2
1.4 Expected Outcomes . . . . .	2
<b>Phase 2: Requirement Analysis</b>	<b>3</b>
2.1 Technical Requirements . . . . .	3
2.2 Functional Requirements . . . . .	3
2.3 Constraints & Challenges . . . . .	3
<b>Phase 3: Project Design</b>	<b>4</b>
3.1 System Architecture . . . . .	4
3.2 User Flow . . . . .	5
3.3 UI/UX Considerations . . . . .	5
<b>Phase 4: Project Planning (Agile Methodologies)</b>	<b>6</b>
4.1 Objective . . . . .	6
4.2 Sprint Planning . . . . .	6
4.3 Task Allocation . . . . .	6
4.4 Timeline & Milestone . . . . .	7
<b>Phase 5: Project Development</b>	<b>7</b>
5.1 Introduction . . . . .	7
5.2 Technology Stack Used . . . . .	7
5.3 Development Process . . . . .	8
5.4 Challenges & Fixes . . . . .	9
<b>Phase 6: Functional and Performance Testing</b>	<b>9</b>
6.1 Test Cases Executed . . . . .	9
6.2 Bug Fixes & Improvements . . . . .	9
6.3 Final Output . . . . .	10
6.4 Deployment . . . . .	12
<b>Appendix</b>	<b>12</b>
Project Resources . . . . .	12

# Phase 1: Brainstorming and Ideation

## 1.1 Problem Statement

**Pattern Sense - Classifying Fabric Patterns using Deep Learning** is a project designed to automate the process of identifying and categorizing fabric patterns using advanced deep learning techniques. Currently, industries such as fashion, textiles, and interior design rely on manual pattern classification, which is time-consuming and inconsistent. This project addresses the need for an intelligent system that can recognize and classify fabric patterns automatically and accurately.

## 1.2 Proposed Solution

The proposed system, **Pattern Sense**, is a deep learning-based classifier that uses the ResNet50 convolutional neural network (CNN) architecture to identify and categorize fabric patterns. The trained model is integrated into a Flask-based web application, enabling users to upload images and receive instant pattern classification results. This tool can be adopted across various industries to streamline workflows and improve decision-making.

## 1.3 Targeted Users

### 1. Fashion Industry:

Designers and manufacturers work with a wide range of patterns such as stripes, polka dots, floral prints, and geometric designs. Pattern Sense automates the categorization process, saving valuable time and effort.

### 2. Textile Quality Control:

The system can be used to detect irregularities or defects in fabric patterns, ensuring that only high-quality materials are sent for production or sale.

### 3. Interior Design:

Designers can quickly identify and select appropriate fabric patterns for furniture, curtains, and upholstery. This results in improved efficiency and more informed design choices.

## 1.4 Expected Outcomes

By deploying the Pattern Sense system, the following outcomes are anticipated:

- **Automated Pattern Classification:** Users can upload fabric images and receive immediate feedback on the pattern type using the trained deep learning model.
- **Time and Cost Savings:** Automation reduces the labor-intensive process of manual inspection and classification, leading to operational efficiency.
- **Enhanced Design Workflow:** Designers can rapidly identify relevant patterns, integrate them into creative workflows, and iterate design choices faster.
- **Scalability and Adaptability:** The system is flexible enough to be integrated into digital platforms and scaled across multiple departments or clients.

## Phase 2: Requirement Analysis

### 2.1 Technical Requirements

The implementation of Pattern Sense requires a blend of machine learning infrastructure and web development tools. The following technical components are essential:

- **Deep Learning Framework:** TensorFlow or Keras for model training and inference.
- **Pretrained CNN Model:** ResNet50, used for extracting deep visual features from fabric images.
- **Web Framework:** Flask, to serve the trained model and enable web-based interaction.
- **Frontend:** HTML and CSS for designing a simple and responsive user interface.
- **Hardware:** A system with GPU support for training (optional during deployment), and sufficient RAM for inference and image preprocessing.
- **Dataset:** A curated image dataset of labeled fabric patterns such as stripes, polka dots, floral, geometric, etc.

### 2.2 Functional Requirements

The core functionalities that the system must support are:

- **Image Upload:** Allow users to upload fabric images through the web interface.
- **Pattern Classification:** Accurately classify the uploaded image into a predefined pattern category using the trained CNN model.
- **User Feedback:** Display the classification result in a user-friendly format.
- **Error Handling:** Provide meaningful error messages for unsupported file types or corrupted inputs.
- **Model Inference:** Ensure that model predictions are executed efficiently with minimal latency on the server side.



### 2.3 Constraints & Challenges

Despite the clarity of the objective, the project poses certain limitations and hurdles during its development and deployment phases.

## Constraints

- **Dataset Availability:** High-quality, labeled fabric pattern datasets are limited and may require manual collection or augmentation.
- **Model Size:** Pretrained models like ResNet50 are computationally intensive, making real-time deployment slightly resource-heavy.
- **Limited Categories:** The initial model may only support a few classes of patterns; expanding it requires additional data and retraining.

## Challenges

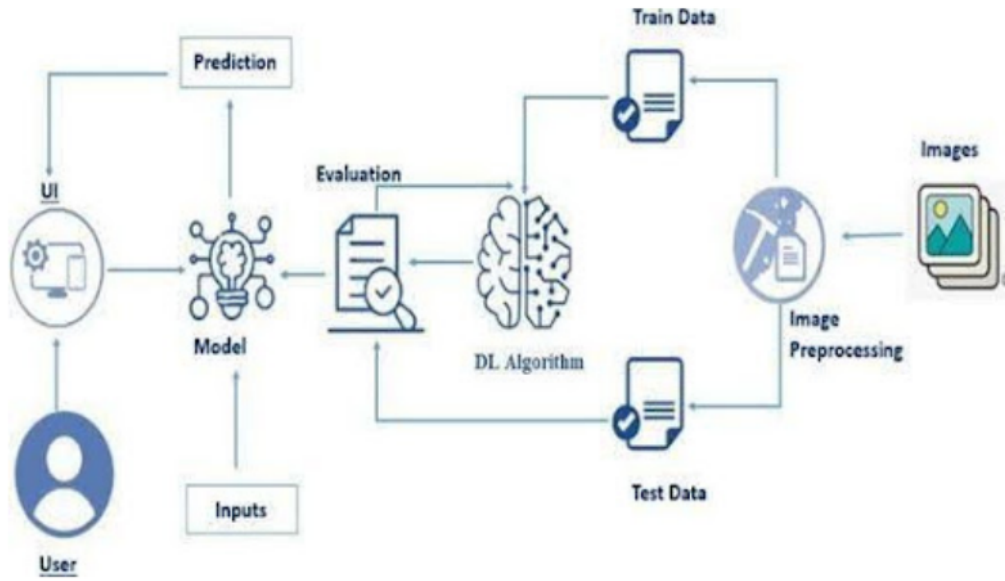
- **Generalization:** Ensuring the model performs well across varied lighting, fabric types, and image qualities is non-trivial.
- **UI Responsiveness:** Creating a clean and functional UI while maintaining backend model integration can be complex.
- **Performance Optimization:** Balancing model accuracy with inference speed is essential for a usable system.
- **User Education:** Users unfamiliar with machine learning might need guidance in interpreting results and using the platform effectively.

# Phase 3: Project Design

## 3.1 System Architecture

The system architecture for **Pattern Sense** follows a modular pipeline that enables smooth interaction between users, backend processing, and deep learning models.

- **Frontend:** A simple HTML/CSS interface allows users to upload fabric images and view the classification result.
- **Flask Backend:** Handles image routing, preprocessing, and calling the trained CNN model (ResNet50) for predictions.
- **Model:** A pre-trained ResNet50 deep learning model, fine-tuned on fabric pattern data, performs the classification.
- **Server Response:** The backend returns the predicted class (e.g., stripes, floral, etc.) which is rendered on the frontend.

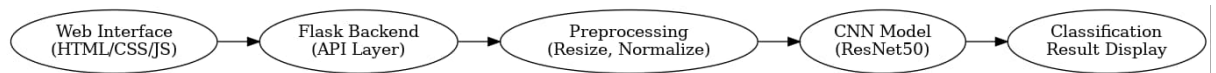


This architecture ensures low latency, easy deployment, and maintainability. The components are loosely coupled for modular updates and improvements.

### 3.2 User Flow

The user flow outlines how a typical end-user interacts with the system from start to finish.

1. User accesses the web interface through a browser.
2. User uploads an image of a fabric pattern.
3. The image is sent to the Flask backend server.
4. The backend preprocesses the image and forwards it to the ResNet50 model.
5. The model predicts the pattern class and sends the result back to the frontend.
6. The predicted pattern type is displayed to the user in a readable format.



This flow ensures an intuitive and efficient experience for both technical and non-technical users.

### 3.3 UI/UX Considerations

The user interface of Pattern Sense is designed with simplicity and clarity in mind. Key considerations include:

- **Minimal Design:** Clean layout with limited distractions for straightforward user interactions.

- **Responsive Upload Interface:** A clearly labeled file upload box that supports common image formats (JPG, PNG).
- **Visual Feedback:** Progress indicators or animations to assure the user that the image is being processed.
- **Error Handling:** Alerts or warnings in case of unsupported file types or prediction issues.
- **Accessibility:** The interface ensures color contrast and readable fonts for usability across diverse user groups.

These design principles ensure that the system is usable, accessible, and adaptable for a wide range of users across industries.

## Phase 4: Project Planning (Agile Methodologies)

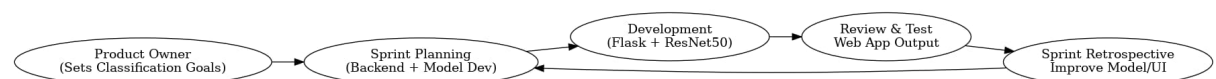
### 4.1 Objective

The objective of this planning phase is to apply Agile principles for structured and iterative development of the **Pattern Sense** project. Agile enables faster development cycles, frequent testing, and quick adaptation to changes.

### 4.2 Sprint Planning

The project was divided into multiple sprints, each targeting a specific module of the system. Each sprint spanned one week and followed the cycle of planning, development, testing, and review.

- **Sprint 1:** Data collection and preprocessing pipeline.
- **Sprint 2:** Model training using ResNet50 and validation.
- **Sprint 3:** Flask backend development and API integration with the model.
- **Sprint 4:** Frontend (HTML/CSS) design and integration.
- **Sprint 5:** Testing, bug fixing, and performance improvements.
- **Sprint 6:** Final deployment and documentation.



Each sprint included daily stand-ups and a retrospective to evaluate progress and adapt the next plan accordingly.

### 4.3 Task Allocation

Tasks were distributed among active contributors based on availability and strengths. Primary focus areas included model development, backend integration, and frontend interface design. Despite challenges in collaboration, key milestones were achieved within planned timelines.

## 4.4 Timeline & Milestone

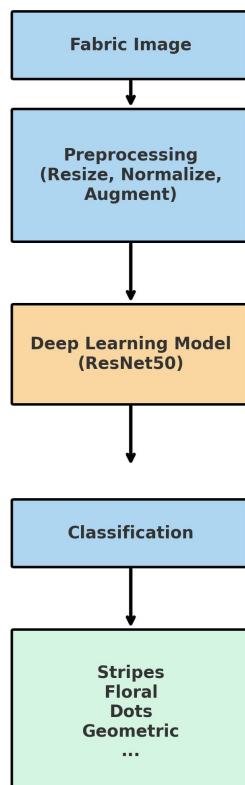
- Gathered project requirements and planned the overall development flow and objectives.
- Developed the image classification model using ResNet50 to recognize fabric patterns.
- Built the backend using Flask to handle image uploads and serve model predictions.
- Designed the frontend interface using HTML and CSS for user interaction.
- Integrated frontend with backend and tested the complete system for functionality.
- Deployed the final application and reviewed it to ensure it met project goals.

## Phase 5: Project Development

### 5.1 Introduction

Fabric images are uploaded and preprocessed before being passed to the deep learning model based on ResNet50. The model classifies the patterns into output categories such as stripes, polka dots, floral, and geometric.

#### Pattern Sense - Fabric Pattern Classification



### 5.2 Technology Stack Used

The development of **Pattern Sense** leveraged a set of modern tools and technologies to build an efficient, scalable, and user-friendly solution:

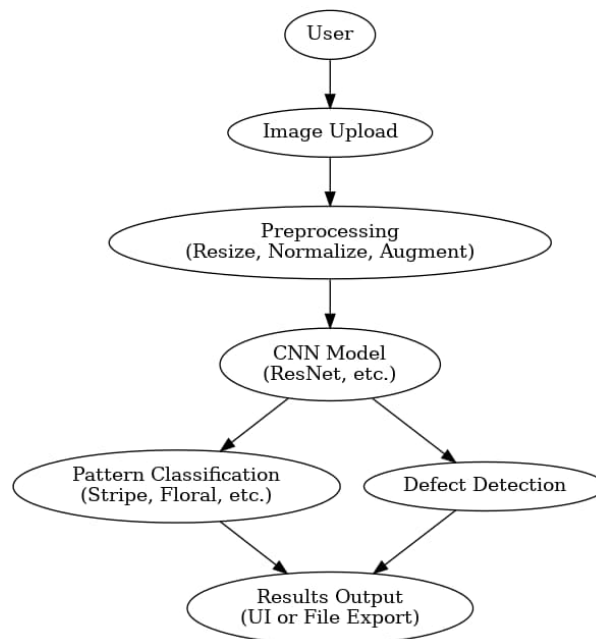


- **Programming Language:** Python
- **Deep Learning Framework:** TensorFlow and Keras
- **Model Architecture:** ResNet50 (pretrained on ImageNet, fine-tuned on fabric datasets)
- **Web Framework:** Flask (for backend and API integration)
- **Frontend:** HTML, CSS (for user interface)
- **Development Environment:** Google Colab, VS Code

### 5.3 Development Process

The project followed an iterative development cycle, with progressive stages from model training to final deployment. Major steps in the process included:

1. **Data Collection and Preprocessing:** Fabric pattern images were collected from open datasets and augmented using flipping, rotation, and scaling.
2. **Model Building:** A ResNet50 model was fine-tuned to classify multiple pattern types such as floral, striped, polka dot, and geometric.
3. **Backend API Development:** A Flask backend was developed to handle image uploads, invoke the model, and return predictions.
4. **Frontend Development:** A simple HTML/CSS interface was created to allow users to upload images and receive predictions.
5. **Integration:** The backend and frontend were integrated to ensure seamless user experience and accurate output rendering.



## 5.4 Challenges & Fixes

Several challenges were encountered during the development of Pattern Sense:

- **Dataset Imbalance:** Some fabric pattern categories had fewer examples. This was mitigated using data augmentation techniques.
- **Model Overfitting:** Initial training showed signs of overfitting. Regularization, dropout, and early stopping were applied to improve generalization.
- **Deployment Integration:** Connecting the deep learning model with Flask and ensuring fast response times required optimization. The model was saved in HDF5 format and loaded efficiently using Keras.
- **UI Compatibility:** Ensuring the frontend worked across browsers was a minor challenge resolved through basic responsive design principles.

Despite these hurdles, the final system was successfully developed and deployed with satisfactory performance and usability.

## Phase 6: Functional and Performance Testing

### 6.1 Test Cases Executed

To ensure the reliability of the system, several test cases were executed on both the model and the web interface. These included:

- **Input Validity:** Tested various image formats (.jpg, .png) to ensure compatibility.
- **Model Prediction:** Checked the accuracy of predictions across multiple pattern types.
- **Backend-Frontend Integration:** Verified that the Flask API returned predictions correctly when triggered by the frontend.
- **Performance Under Load:** Tested response time when processing large images or multiple requests.
- **User Experience:** Evaluated the UI for accessibility and ease of use.

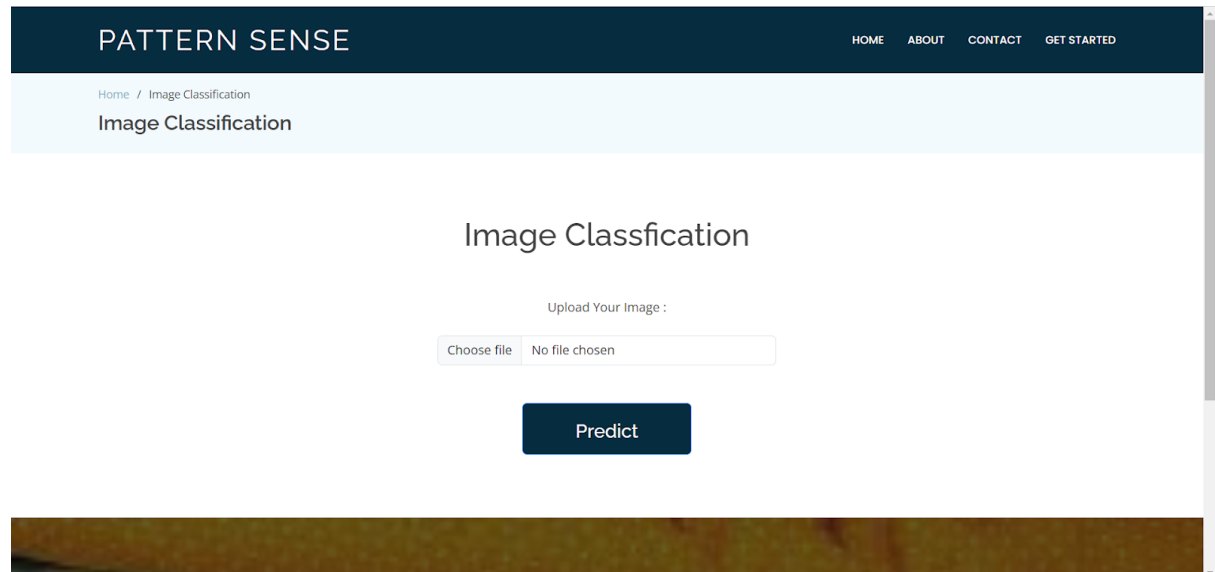
### 6.2 Bug Fixes & Improvements

Throughout testing, the following issues were encountered and resolved:

- **Issue:** API occasionally returned null results when large images were uploaded.  
**Fix:** Implemented image resizing before model prediction.
- **Issue:** Classification labels were not properly mapped.  
**Fix:** Updated label decoding logic in the model wrapper function.
- **Issue:** UI failed to update predictions dynamically.  
**Fix:** Refined JavaScript function to refresh the prediction box on file upload.

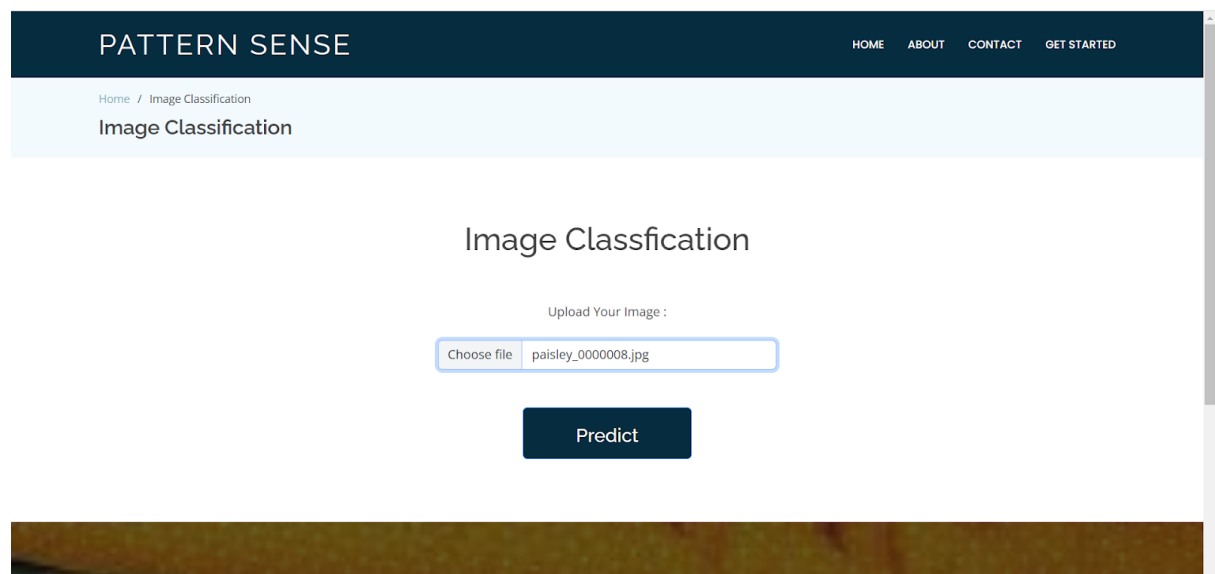
## 6.3 Final Output

Click on Get Started at the top right corner to go to the inner page as below:



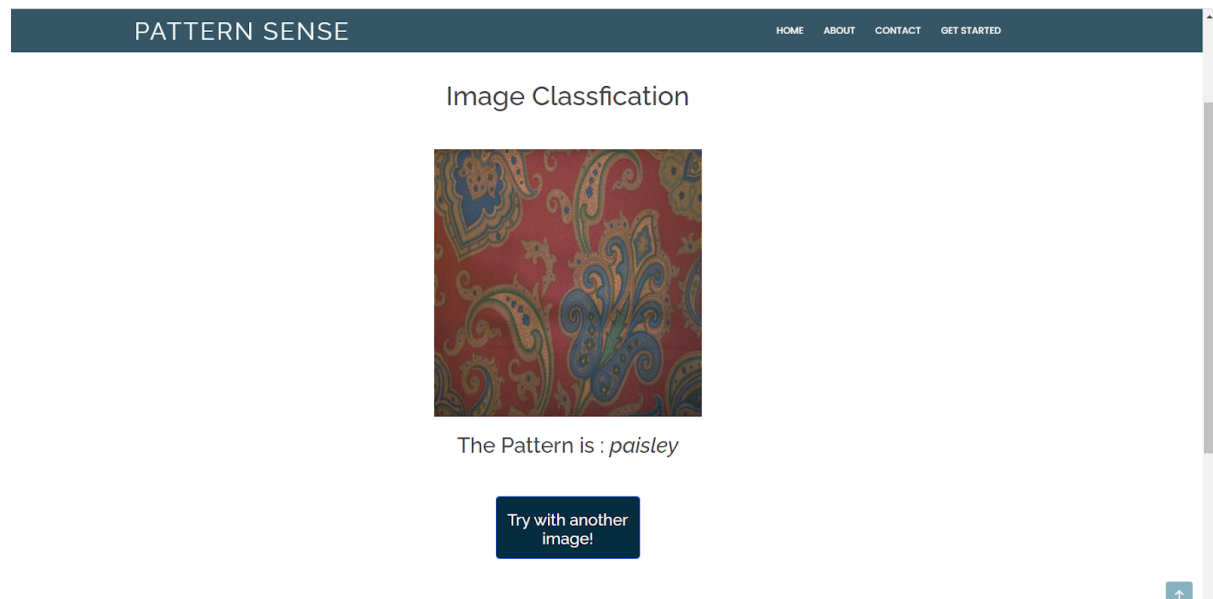
Upload the image & Click on Predict button

**Input:1**

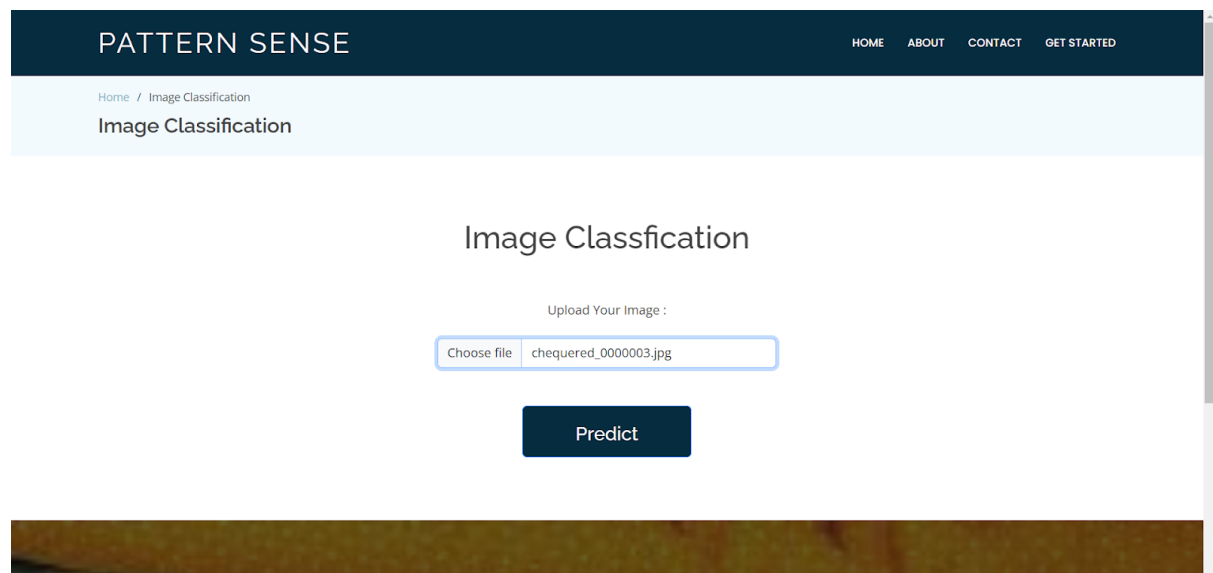


Once you upload the image and click on the submit button, the output will be displayed on the below page

**Output:1**



## Input:2



Once you upload the image and click on the submit button, the output will be displayed on the below page

## Output:2

## Image Classification



The Pattern is : *chequered*

Try with another  
image!

## 6.4 Deployment

The final application was deployed locally and tested in a simulated production environment. The Flask backend served the deep learning model via API endpoints, while the frontend interface facilitated smooth and intuitive user interaction.

The deployment process involved the following key steps:

1. The trained ResNet50-based model was saved in HDF5 (.h5) format.
2. The Flask web application was hosted using a lightweight WSGI server (e.g., Gunicorn).
3. The HTML/CSS frontend was connected to the backend using AJAX for dynamic interaction.

## Appendix

### Project Resources

The following resources are available to better understand, reproduce, or extend the work done in the Pattern Sense project:

- **GitHub Repository Link:**  
<https://github.com/Saisugandhasri/Pattern-Sense-Classifying-Fabric-Patterns-using-Deep-Learning>
- **Demo Video:**  
<https://drive.google.com/file/d/1xdOaJVlnUI7YJw4DH5zJob2vEpbmCwei/view?usp=drivesdk>
- **Project Source Code:**  
<https://drive.google.com/drive/folders/1GBbcyzwTvbqlOtLdM09lmRmD0FXFn.JTW>
- **Dataset Samples:**  
<https://drive.google.com/drive/folders/1G4bf2qSb0qduTIZwN153yvk1EzGuzacv>