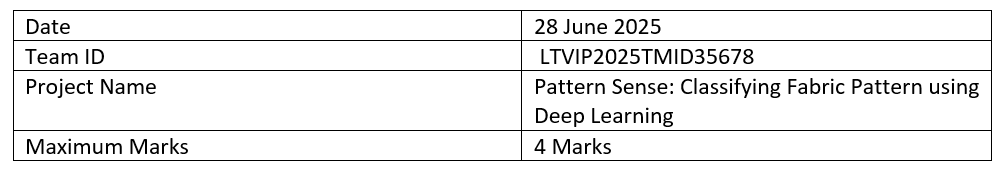
**4 Project Design Phase**

**4.3 Solution Architecture**

****

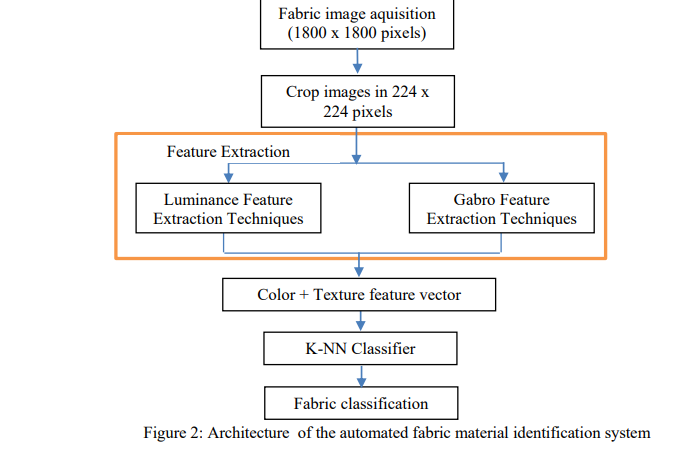
**Solution Architecture:**

In the Pattern Sense project, the architecture bridges the gap between the challenge of manual fabric pattern identification and a scalable, automated AI solution.

**Goals of the Solution Architecture:**

* **✅** Identify and implement the most suitable AI model (CNN) for classifying fabric patterns.
* ✅ Structure the system into components: data processing, model training, prediction service, and UI interface.
* ✅ Define features such as real-time image classification, confidence display, and extensibility for more pattern types.
* ✅ Ensure the solution is deployable locally and scalable to cloud infrastructure for broader adoption.

**Architecture Diagram:**

****

1. Frontend (User Interface)

* Technology: HTML, CSS, JavaScript
* Role: Provides a user-friendly interface where users can upload fabric images.
* Functionality:
  + Image upload form
  + Displays prediction results (e.g., "Polka Dotted", "Plain")
  + Optional pages: About, Get Started, Results page

2. Backend (Flask Application)

* Technology: Flask (Python)
* Role: Acts as a bridge between the frontend, image processing logic, and the CNN model.
* Key tasks:
  + Receives image upload from frontend via POST request
  + Processes the image (resizes to model input size, normalizes)
  + Loads and runs inference using the pre-trained CNN model (model\_cnn.h5)
  + Returns prediction result to frontend

3. CNN Model (Deep Learning Component)

* Technology: TensorFlow / Keras
* Model File: model\_cnn.h5
* Role: Classifies fabric pattern into categories such as:
  + Plain
  + Striped
  + Polka-Dotted
  + Checked
* Training Phase:
  + Model is trained on a labeled dataset of fabric images.
  + Trained on Colab or VS Code with Google Drive integration
* Inference Phase:
  + Used within the Flask backend to predict pattern class from uploaded images

4. Image Preprocessing

* Libraries: Keras load\_img, img\_to\_array
* Steps:
  + Resize image to model input shape (e.g., 150x150)
  + Normalize pixel values (e.g., divide by 255.0)
  + Convert image to array format suitable for model prediction

5. Data Storage

* Training Dataset: Stored locally or in Google Drive during training
* Uploaded Images: Temporarily stored in /static/uploads or similar path on the Flask server