



**DELHI PUBLIC SCHOOL, BANGALORE NORTH**  
**Computer Science Project Report**  
**(2025-2026)**

**Project Name:FlowText**

**Amaan Akhtar**

**Hrishikesh Neettiyath**

# INDEX

1	Certificate
2	Acknowledgement
3	Introduction
4	Development Platform
5	Project Design
6	Source Code
7	Output Snapshot
8	Further Enhancements
9	Bibliography



# **CERTIFICATE**

This is to certify that the following students of class XII

1) Amaan Akhtar

BOARD ROLL NUMBER \_\_\_\_\_

2) Hrishikesh Neettiyath

BOARD ROLL NUMBER \_\_\_\_\_

have successfully completed the Project Work titled

\_\_\_\_\_

under the guidance of

Mrs. Uzma F

during the academic year 2025-2026 in partial fulfillment  
of practical examination for the subject

COMPUTER SCIENCE (083)

conducted by SSCE, CBSE

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

**PRINCIPAL**

--

## **ACKNOWLEDGEMENT**

We would like to express our gratitude to our principal Mrs. Manju Balasubramanyam for providing the opportunity for us to work on this project.

We would like to express our sincere gratitude to our computer teachers, Mrs. Uzma Fatima and Mrs. Manjula S, for their constructive review and guidance at every stage of the project.

We also render our gratitude to CBSE for including this meaningful project as a part of the Computer Science syllabus.

# Introduction

FlowText is a tool that bridges the gap between handwritten notes and digital analysis. By capturing a simple photo of your notes, it converts handwriting into plain, searchable text. Once digitised, you can quickly locate keywords, perform frequency analysis, and gain insights into your writing. With more features planned, FlowText aims to transform messy stacks of notes into a structured, searchable, and intelligent knowledge base.

## PURPOSE/OBJECTIVE :

Develop a GUI based physical notes management system to streamline digital and physical note sharing and analysis.

## EXISTING SYSTEM :

Currently there is no major app addressing the specific issue of physical notes conversion, however there exist a plethora of apps that focus mainly on text digitization.

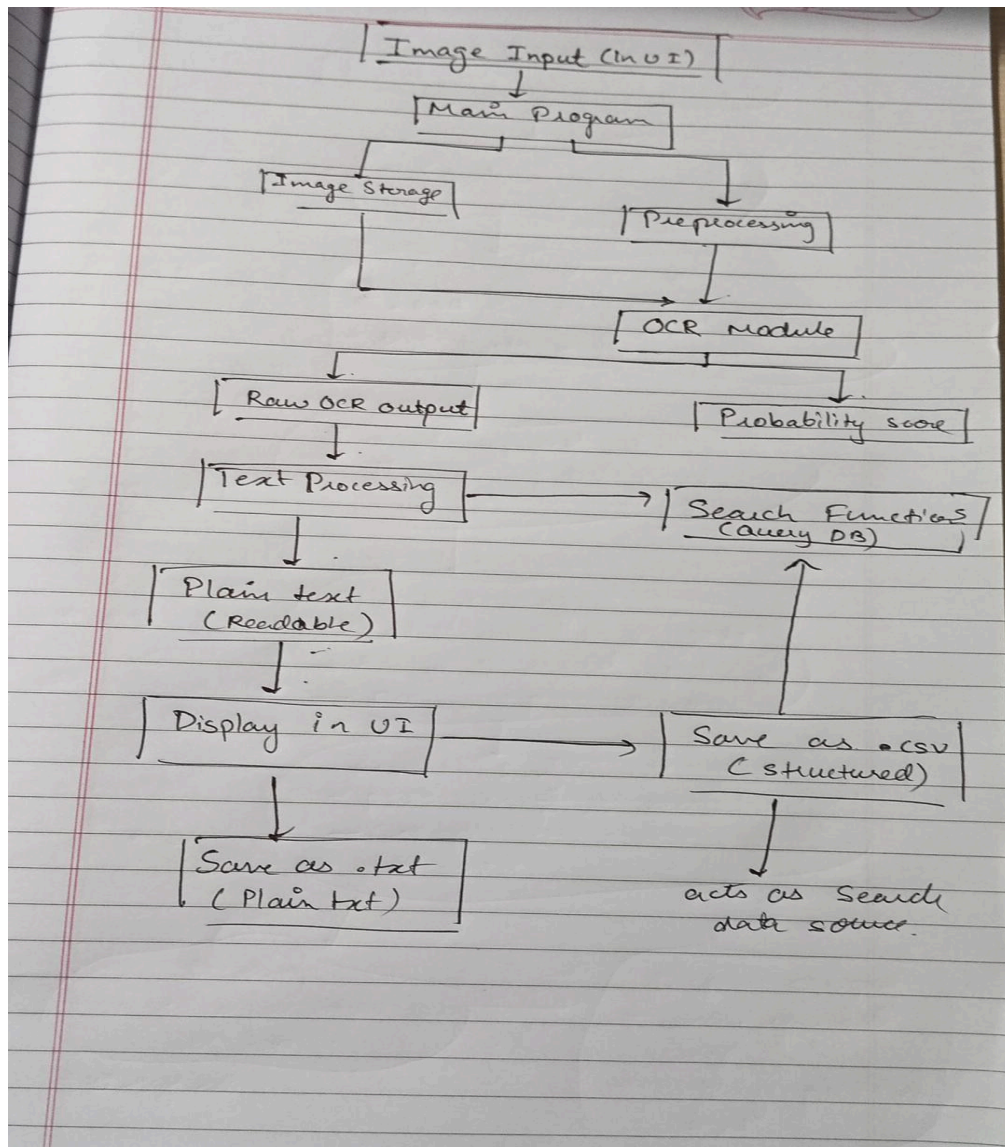
## PROPOSED SYSTEM :

Our app aims at providing a much more focused service on notes management that other OCR based converter apps lack. Typically they lack organization and accessibility when it comes to viewing the outputted digitized texts. Currently the system focuses mainly on plain text conversion but with time we plan to add LaTeX capabilities too.

# Development Platform

- Hardware Specifications:
  - Processor: Intel Core i5
  - RAM: 8GB
  - OS: Windows
  - Software: Python 3.12.6
- Visual Studio Code : Visual Studio Code (VS Code) is a free, lightweight, and powerful source code editor developed by Microsoft for Windows, macOS, and Linux
- Pycharm : PyCharm is an integrated development environment used for programming in Python, it also supports web development with Django
- Django : Django is a free and open-source, Python-based web framework that follows the model–template–views architectural pattern
- Required Python libraries:
  - OS
  - CSV
  - Numpy
  - Pytorch
  - Streamlit(optional)

# Project Design



# Source Code

```
import torch
from torch import nn, optim
from torchvision import datasets, transforms
from torch.utils.data import DataLoader

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")

transform = transforms.ToTensor()

train_dataset = datasets.MNIST(root="data", train=True, download=True, transform=transform)
test_dataset = datasets.MNIST(root="data", train=False, download=True, transform=transform)

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=1000, shuffle=False)

class DigitModel(nn.Module):
    def __init__(self):
        super(DigitModel, self).__init__()
        self.net = nn.Sequential(
            nn.Flatten(),
            nn.Linear(784, 128),
            nn.ReLU(),
            nn.Linear(128, 10)
        )

    def forward(self, x):
        return self.net(x)

model = DigitModel().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

epochs = 15
for epoch in range(epochs):
    model.train()
    running_loss = 0.0

    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
```

```
running_loss += loss.item()

model.eval()
correct = 0
total = 0
with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        predictions = torch.argmax(outputs, dim=1)
        correct += (predictions == labels).sum().item()
        total += labels.size(0)

accuracy = 100 * correct / total
print(f"Epoch {epoch+1}/{epochs} - Loss: {running_loss:.4f} - Accuracy: {accuracy:.2f}%")

torch.save(model.state_dict(), "mnist_model.pth")
print("Model saved as mnist_model.pth")
```

# Output Snapshot

Using device: cpu

Epoch 1/15 - Loss: 326.1110 - Accuracy: 94.61%  
Epoch 1/15 - Loss: 326.1110 - Accuracy: 94.61%  
Epoch 1/15 - Loss: 326.1110 - Accuracy: 94.61%  
Epoch 1/15 - Loss: 326.1110 - Accuracy: 94.61%  
Epoch 1/15 - Loss: 326.1110 - Accuracy: 94.61%  
Epoch 1/15 - Loss: 326.1110 - Accuracy: 94.61%  
Epoch 1/15 - Loss: 326.1110 - Accuracy: 94.61%  
Epoch 1/15 - Loss: 326.1110 - Accuracy: 94.61%  
Epoch 1/15 - Loss: 326.1110 - Accuracy: 94.61%  
Epoch 2/15 - Loss: 148.6501 - Accuracy: 96.38%  
Epoch 3/15 - Loss: 102.0461 - Accuracy: 96.92%  
Epoch 4/15 - Loss: 76.7468 - Accuracy: 97.19%  
Epoch 5/15 - Loss: 59.9525 - Accuracy: 97.28%  
Epoch 6/15 - Loss: 48.4229 - Accuracy: 97.67%  
Epoch 6/15 - Loss: 48.4229 - Accuracy: 97.67%  
Epoch 6/15 - Loss: 48.4229 - Accuracy: 97.67%  
Epoch 6/15 - Loss: 48.4229 - Accuracy: 97.67%  
Epoch 7/15 - Loss: 39.0970 - Accuracy: 97.67%  
Epoch 8/15 - Loss: 31.7749 - Accuracy: 97.81%  
Epoch 9/15 - Loss: 26.2569 - Accuracy: 97.42%  
Epoch 10/15 - Loss: 21.4429 - Accuracy: 97.88%  
Epoch 11/15 - Loss: 17.9184 - Accuracy: 97.92%  
Epoch 12/15 - Loss: 15.2036 - Accuracy: 97.80%  
Epoch 13/15 - Loss: 13.0666 - Accuracy: 98.02%  
Epoch 14/15 - Loss: 10.7514 - Accuracy: 97.89%  
Epoch 15/15 - Loss: 8.4853 - Accuracy: 97.94%

Model saved as mnist\_model.pth

# Further Enhancements

- Implementing support for more complex notes, i.e. expanding to digitising equations and other miscellaneous text and image artifacts, via a proprietary OCR solution or existing advanced OCR software.
- Including further support of editing and making notes directly in the app interface, instead of it being a static notes collection.
- Add in-built LaTeX support to facilitate an aesthetic and modern look, while also providing an industry and academic standard mark up language support.

# Bibliography