PROJECT REPORT

A NOVEL METHOD FOR HANDWRITTEN DIGIT RECOGNITION

submitted by

Team ID: PNT2022TMID07719

Akash Iyer – 19CECS003

Narayanan M S – 19CECS022

Ramalingam V – 19CECS031

Tharun Kumar P – 19CECS045

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 PROJECT OVERVIEW

Machine learning and deep learning play an important role in computer technology and artificial intelligence. With the use of deep learning and machine learning, human effort can be reduced in recognizing, learning, predictions and in many more areas.

Handwritten Digit Recognition is the ability of computer systems to recognise handwritten digits from various sources, such as images, documents, and so on. This project aims to let users take advantage of machine learning to reduce manual tasks in recognizing digits.

1.2 PURPOSE

Digit recognition systems are capable of recognizing the digits from different sources like emails, bank cheque, papers, images, etc. and in different real-world scenarios for online handwriting recognition on computer tablets or system, recognize number plates of vehicles, processing bank cheque amounts, numeric entries in forms filled up by hand (tax forms) and so on.

CHAPTER 2
LITERATURE SURVEY

2.1 EXISTING PROBLEM

The fundamental problem with handwritten digit recognition is that handwritten digits do not always have the same size, width, orientation, and margins since they vary from person to person. Additionally, there would be issues with identifying the numbers because of similarities between numerals like 1 and 7, 5 and 6, 3 and 8, 2 and 5, 2 and 7, etc. Finally, the individuality and variation of each individual's handwriting influence the structure and appearance of the digits.

2.2 REFERENCES

Handwritten Digit Recognition by Neural Networks with Single-Layer Training

S. Knerr, L. Personnaz, G. Dreyfus

This paper proposes that using neural network classifiers with single-layer training can be applied ably to complex real time classification queries like recognizing handwritten digits. This paper initiates the STEPNET algorithm, which breaks the major problem into simpler sub problems. Given suitable data representations and learning regulations are indulged, performance which are similar to those acquired by more compound networks can be achieved. It propagates the outcomes of 2 dissimilar databases; a European database consists 8000 plus separated digits, and zip code database from the US Postal Service consists 9000 plus segmented digit

Handwritten Zip Code Recognition with Multilayer Network

Y. Le Cun, O. Matan, B. Boser, J. S. Deker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jacket, H. S. Baird

Here they demonstrate a backpropagation network application for handwritten zip code recognition. The architecture of the network was very limited and made expressly for the task, so little pre-processing of the data was necessary. The network's input consists of solitary digits in size-normalized pictures. The accuracy of the zip code digits 92%. Using structured neural networks, considered to be "statistical methods with structure" that bridge the gap between purely statistical and fully structural methods.

An online cursive handwritten medical words recognition system for busy doctors in developing countries for ensuring efficient healthcare service delivery

Shaira Tabassum, Nuren Abedin, Md Mahmudur Rahman,Md Moshiur Rahman, Mostafa Taufiq Ahmed, Rafiqul Islam, Ashir Ahmed

Doctors in developing nations don't have a digitized prescription system and thus still make use of handwritten prescriptions. This makes it challenging for the patients to know what medicines have been prescribed as the handwriting is sometimes not legible enough. This paper suggests the use of a smartpen which will recognize the writings and digitize this in real life.

Reading Handwritten Digits: A ZIP Code Recognition System

Ofer Matan, Henry S. Baird, Jane Bromley, Christopher J. C. Burges, John S. Denker, Lawrence D. Jackel, Yann Le Cun, Edwin P.D. Pednault, William D. Satterfield, Charles E. Stenard, and Timothy J. Thompson

It is explained how a neural network algorithm-based system can read handwritten ZIP codes seen on actual US mail. The system employs a recognition-based segmenter, which combines vertical cuts, connected components analysis (CCA), and a recognizer from a neural network. CCA handles connected components with single digits. The vertical-cut segmenter handles CCs that are merged or divided into digits. Pre-processing, where noise is eliminated and the digits are de-italicized, CCA segmentation and recognition, vertical-cut-point estimation and segmentation, and immediately lookup are the four key processing steps. The algorithm was trained and tested using five- and nine-digit ZIP code fields obtained from actual mail in about 10,000 photos.

Handwritten digits recognition with decision tree classification: a machine learning approach

Tsehay Admassu Assegie, Pramod Sekharan Nair

This paper lists various reasons why perfect outcomes can't be achieved with current AI & ML methodologies. Along with this they have mentioned a way to recognize digits with the help of decision tree classifier. Decision Tree classifier has been implemented by using 'numpy', 'pandas' and 'sklearn'. This paper has made use of the Kaggle dataset for handwritten digits. Each image in the dataset is converted into a grey scale image of 28 x 28 pixel. Then the developed model would try to predict the class for the given input. The final accuracy of the model proposed in the system turned out to be 83.4%.

Competition on Handwritten Digit Recognition (HDRC 2013)

Markus Diem; Stefan Fiel; Angelika Garz; Manuel Keglevic; Florian Kleber; Robert Sablatnig by ICDAR2013

RGB color plays the major role in this technology, For the competition, the images are delivered in original size with a resolution of 300 dpi Single Digit dataset consists of 10 classes (0-9) with 3,578 samples per class. For the HDR competition, 7,000 digits (700 digits per class) of 67 writers have been selected as training set. It has equal size has been published with a different set of 60 writers and variations for characterizing the relation between background and foreground, a descriptor based on concavities is used. For each background pixel a 4-bit code yielded by searching for a foreground pixel in four directions is computed, if a foreground is found the corresponding bit is set to 1, in other case to 0.

A trainable feature extractor for handwritten digit recognition

Fabien Lauera, Ching Y. Suenb, Gérard Bloch

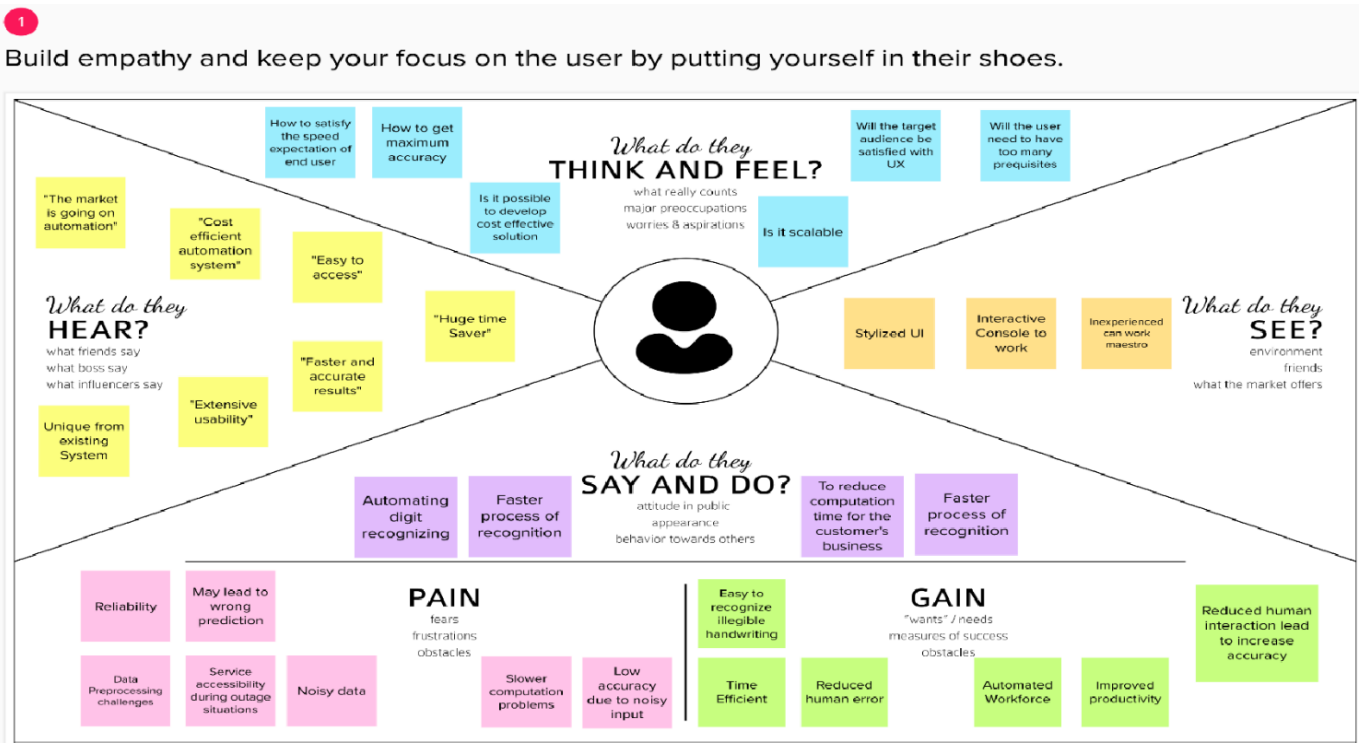
Neural networks rose to prominence among all character recognition classifiers in the 1980s, as seen by the results obtained by Lacuna's. Lent family of neural networks. which are Convolutional neural networks, as opposed to simple fully connected networks, are sensitive to the topological characteristics of the input (in this case, the image). Another significant occurrence SVMs were introduced in the region. Those SVMs based on the notion of structural risk minimization (SRM) It limits the generalization risk to a minimum, whereas the objective of neural networks is to reduce the empirical risk. In the case of handwritten recognition, SVMs gave very good results and OCR technology is very important in this field.

2.3 PROBLEM STATEMENT DEFINITION

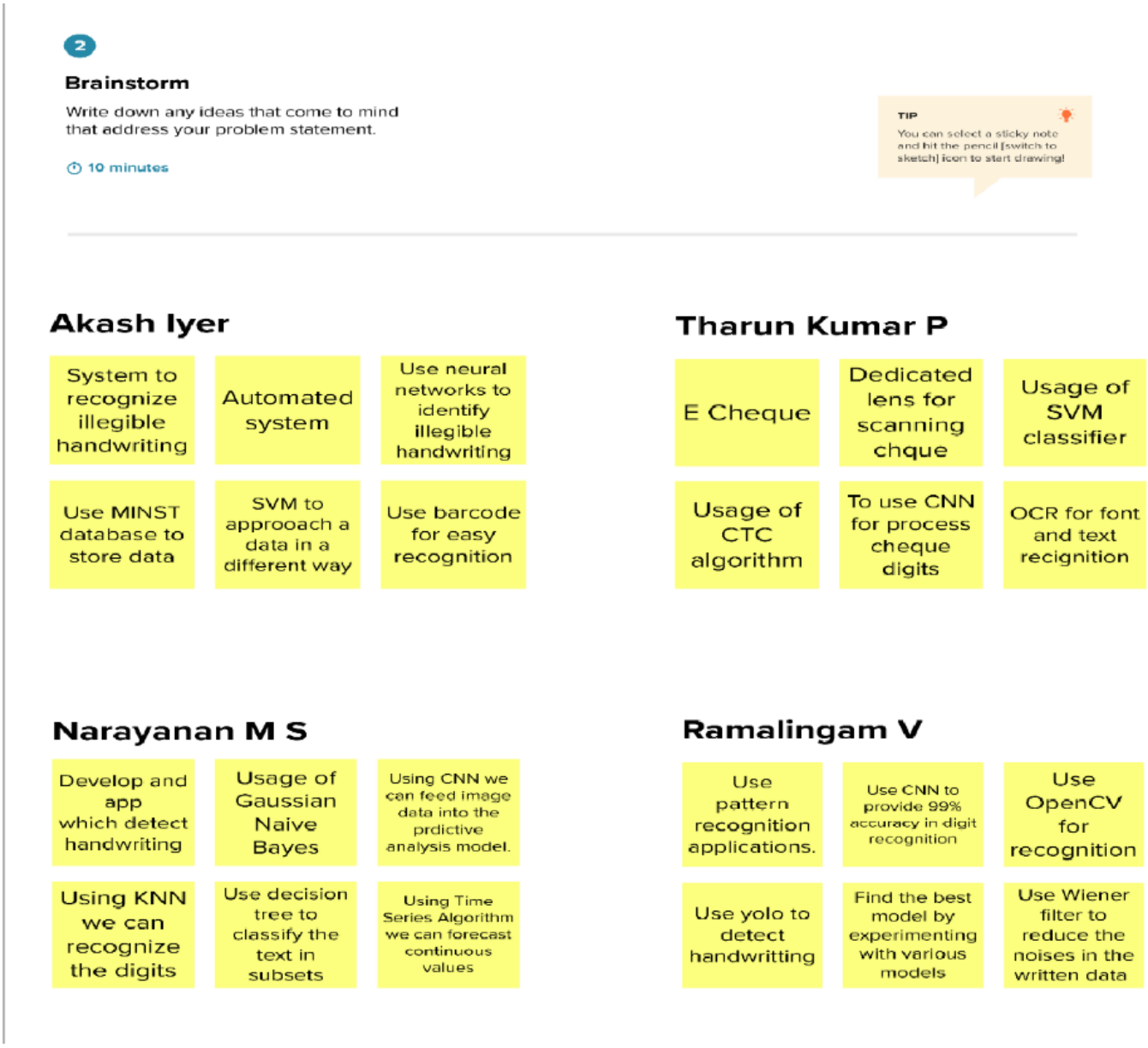
For years, the cheque processing has traditionally been done by the bank staff. They have to manually process the cheques one by one. This leads to delays and sometimes errors might creep in. So, our project aims to simplify this process by scanning the cheque using OCR and CNN. The scanned details are then uploaded to the database for future reference. Therefore, the goal of this project is to help reduce the time taken in cheque processing and automate the process to an extent.

CHAPTER 3
IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING



4

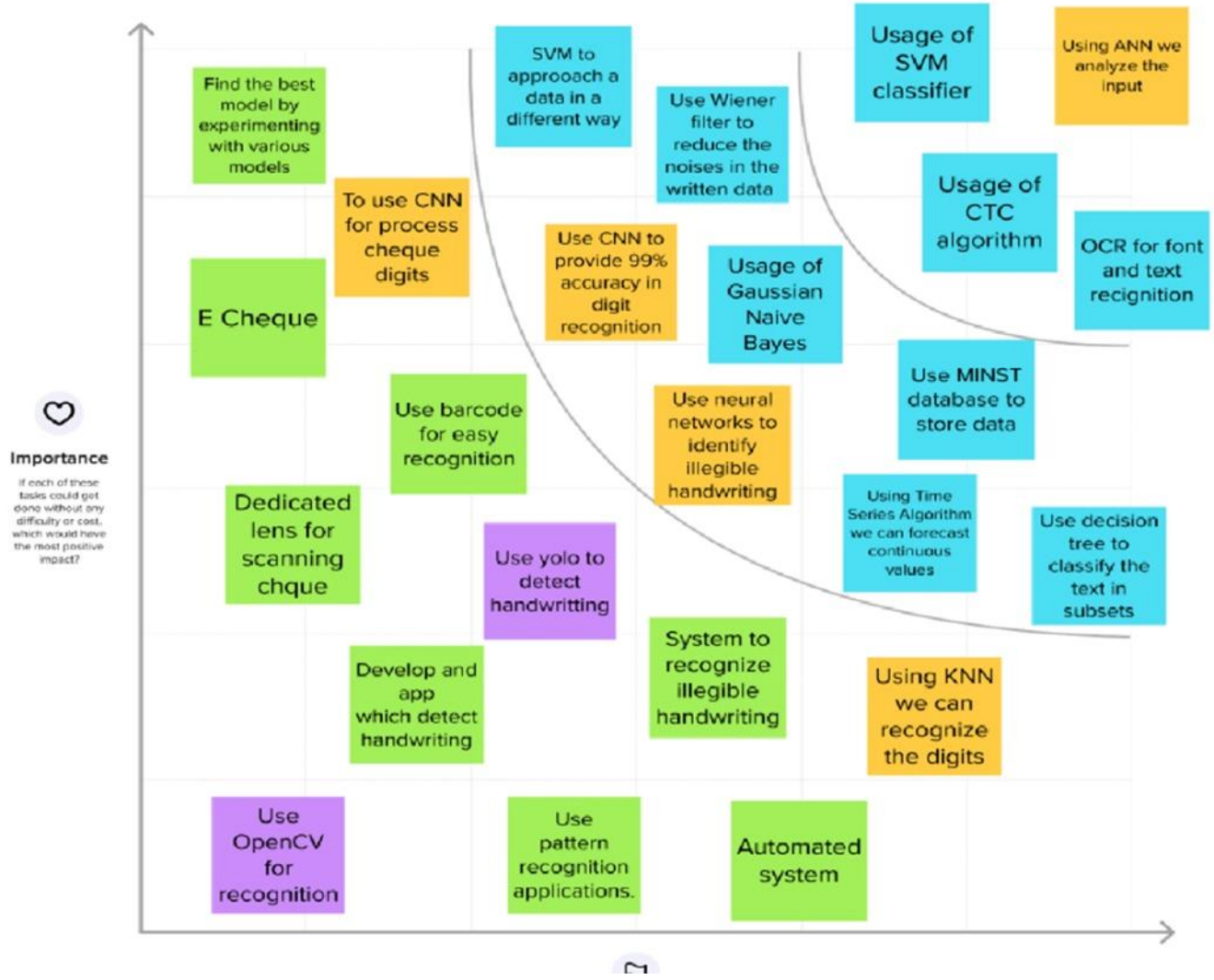
Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

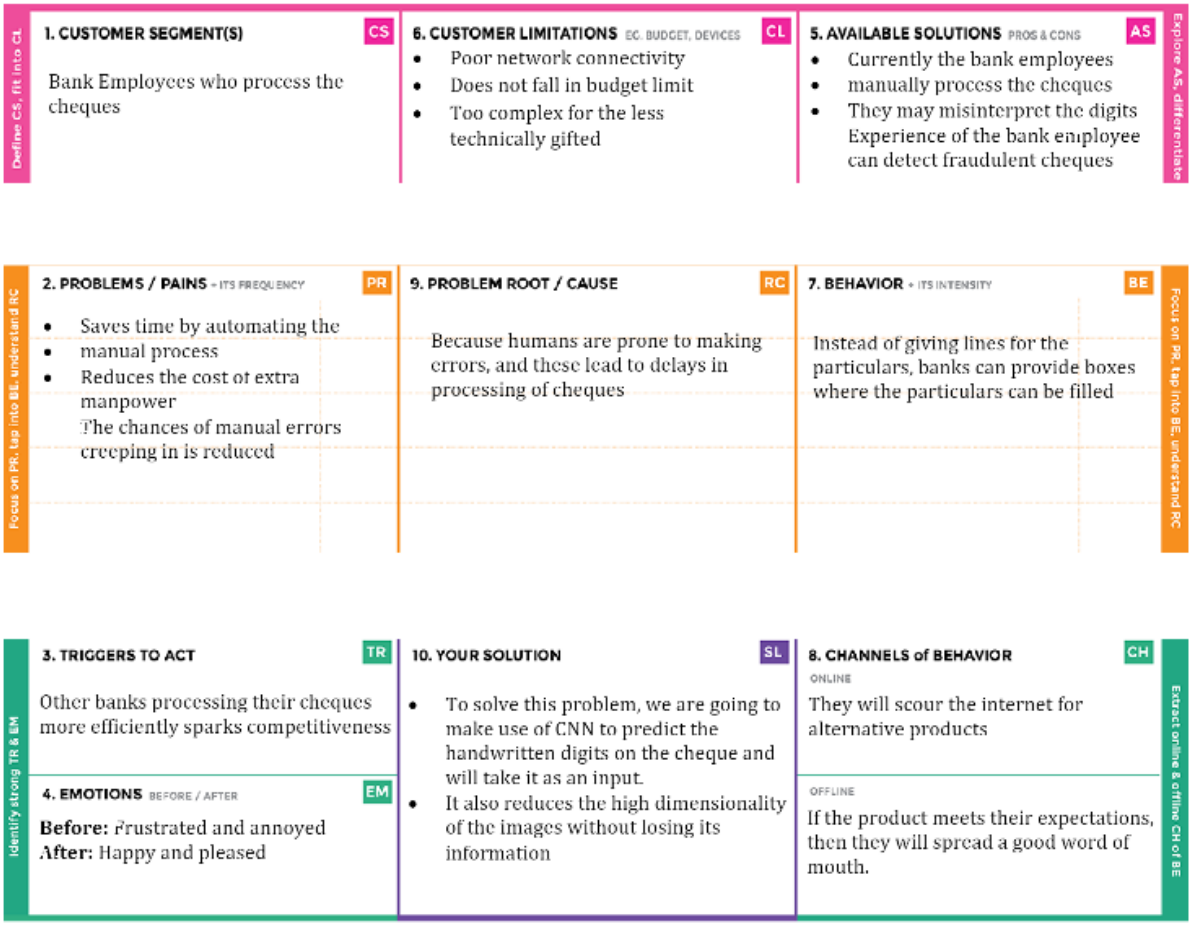


3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The processing of bank cheques takes a lot of time due to the manual nature of the work. Also there is a chance for human errors to take place.
2.	Idea / Solution description	To solve this problem we are going to make use of CNN to predict the handwritten digits on the cheque and will take it as an input.
3.	Novelty / Uniqueness	• Can be used offline • Will automatically classify all the inputs.
4.	Social Impact / Customer Satisfaction	The primary societal benefit of this effort is to ensure effective and trustworthy methods for handwritten digit recognition and facilitate error-free financial transactions. Customers will feel comfortable using it because it is simple and convenient. This has a wide range of applications and the accuracy rate will also be good.
5.	Business Model (Revenue Model)	Since this solution primarily targets financial institutions for the purpose of processing cheques **Key Partners:** Financial Institutions **Key Activities:** Classify the cheque details, digitizing the details and allowing it to be copied. **Key Resources:** Webcam **Customer Relationships:** We can communicate with the customers in the form of feedbacks and review meetings. **Cost Structure:** Hardware Cost, Advertisement. **Revenue Stream:** We intend to charge on a cost per cheque basis where there will be a fixed processing fee for each cheque. And

		the payment will be made on a monthly basis.
6.	Scalability of the Solution	Financial Institutions such as banks are facing issues in Recognizing written digits such as in cheques etc. This can be handled by our handwritten digit recognition project as they expand into different business domains without impacting performance. Our proposed solution is scalable as it is dynamic and also trained using AI and deep learning Models

3.4 PROBLEM SOLUTION FIT



CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

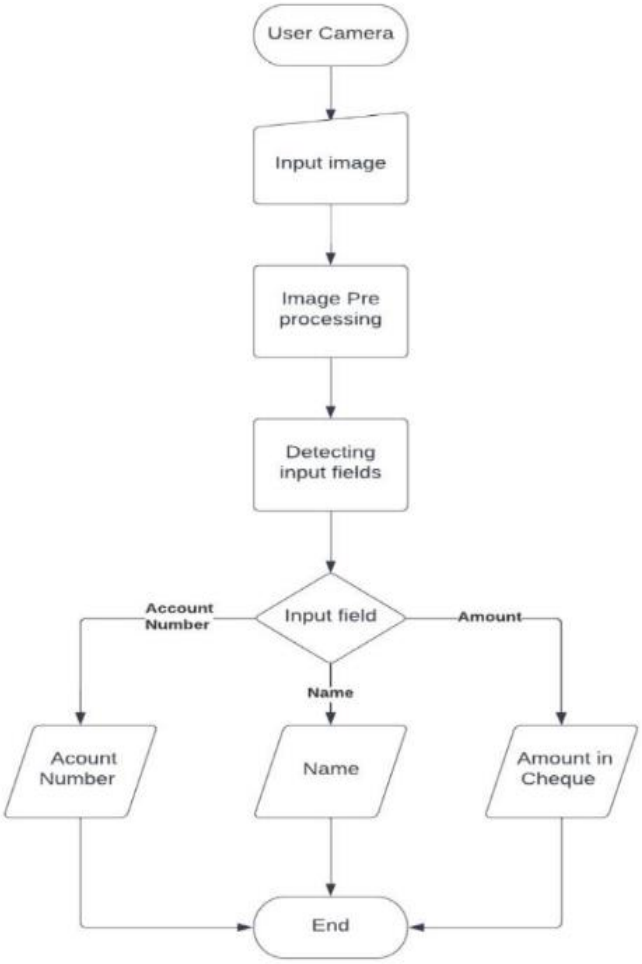
FR No.	Functional Requirement	Description
FR-1	**Website**	• A website having a login feature, where each user will have to register and then he/she will be able to login using his username and password.
FR-2	**Upload Image**	• Must be able to take the handwritten inputs in the form of the images. (JPG or PNG)
FR-	**Input correlation**	• Image Correlation is a technique used to recognize characters from images. Collecting data and prepare it for training.
FR-4	**Feature extraction**	• Feature extraction is analyzing the images and deriving some characteristics from these images that identify each specific element.
FR-5	**Output**	• System should retrieve characters present in the image and display them to the user • Must be able to display the accurate output in text format

4.2 NON-FUNCTIONAL REQUIREMENTS

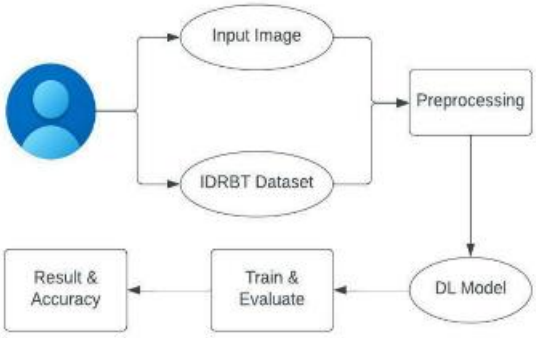
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	• Application for digit recognition include filling out forms, processing bank checks, and sorting mail. • Should be easy to use even by the less technically gifted
NFR-2	**Security**	• As it will be used in the banking sector, it should be able to store the cheque details securely. • This will be done by authenticating the users using their username and password.
NFR-3	**Reliability**	• This software should work reliably for low resolution image and should not run into any errors
NFR-4	**Performance**	• The software should be responsive and provide output quickly even for complex handwriting
NFR-5	**Accuracy**	• Optical Character Recognition (OCR) technology provides higher than 99% accuracy with typed characters in high- quality images. However, the diversity in human writing types, spacing differences, and inequalities of handwriting causes less accurate character recognition.
NFR-6	**Scalability**	• Large numbers of users can recognize the digits at a time without any restriction.

CHAPTER 5
PROJECT DESIGN

5.1 DATA FLOW DIAGRAM



5.2 SOLUTION & TECHNICAL ARCHITECTURE



5.3 USER STORIES

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web User)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password	User must enter valid email-id and mobile number.	Medium	Sprint-1
	Login	USN-2	As a user, I can log into the application by entering email & password	Must use valid user id and password	Medium	Sprint-1
	Image Scanning	USN-3	As a user, I'm allowed to capture images using the webcam.	Only the clear images shall be uploaded for processing.	Medium	Sprint-4
	Image Uploading	USN-4	As a user, I am allowed to upload the images from the local file storage	The image should be of the specified format and size.	High	Sprint-4
	Copy Processed Details	USN-5	As a user, I can copy the processed details of the uploaded cheque	-	Medium	Sprint-3
	Previous Cheque Details	USN-6	As a user, I can access the cheques that have been previously processed for reference	-	Low	Sprint-3
Developer	Dataset Collection, Pre-processing, Training & saving the model	USN-7	In order to train the model, we have to collect and pre-process the data set. After pre-processing, we can train and save the model.	Model with higher accuracy is saved and used for prediction	High	Sprint-2
	Integrating the model with Flask API, database creation	USN-8	The image given by the user is processed and given to the model for prediction and the results are sent back to the user	-	Medium	Sprint-3

CHAPTER 6
PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

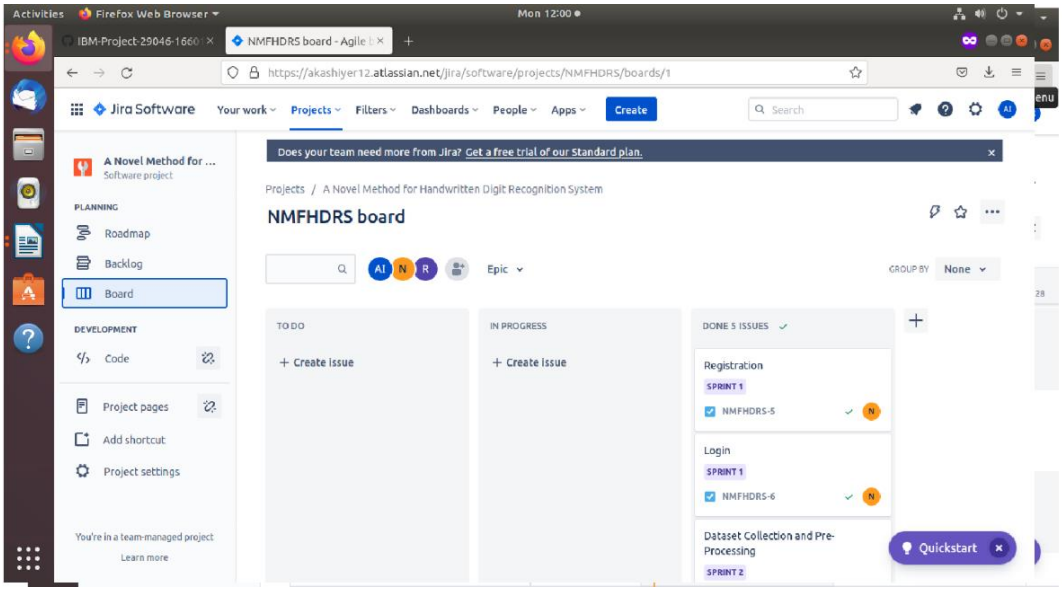
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	12	Medium	Ramalingam V Narayanan MS
Sprint-1	Login	USN-2	As a user, I can log into the application by entering email & password.	8	Medium	Ramalingam V Narayanan MS
Sprint-2	Dataset Collection and Pre-Processing	USN-7	To train the model, the dataset is collected and pre-processed.	10	High	Akash Iyer Tharun Kumar, Narayanan MS
Sprint-2	Training & saving the model	USN-7	With the help of the pre-processed data the model is trained and the model is tunned for producing higher accuracy.	10	High	Akash Iyer Tharun Kumar
Sprint-3	Integrating the model with Flask API	USN-8	The image given by the user is processed and given to the model for prediction and the results are sent back to the user.	14	Medium	Ramalingam V Akash Iyer
Sprint-3	Creating the database.	USN-8	The database is used to store the user details and the previously processed cheques	6	Medium	Tharun Kumar
Sprint-4	Image scanning and uploading	USN-3,4	The user uploads the image of the cheque to be processed either by scanning or uploading it from the local file structure.	12	Medium	Narayanan MS Ramalingam V
Sprint-4	Displaying the results	USN-5,6	We get the results from the backend and display it to the user.	8	Low	Akash Iyer Tharun Kumar

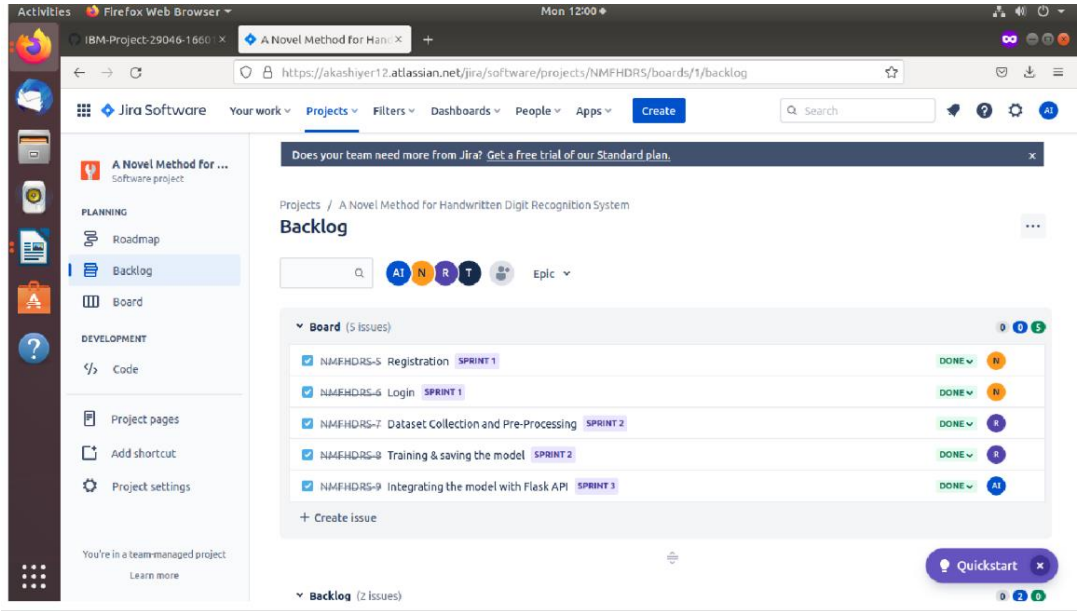
6.2 SPRINT DELIVERY SCHEDULE

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 REPORTS FROM JIRA





CHAPTER 7
CODING & SOLUTIONING

7.1 REACT CODE

index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <title>Cheque Processing System</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

app.js

```jsx
import React from "react";
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Home from "./Components/Home";
import Login from "./Components/Login";
import "./App.css";
import "../node_modules/bootstrap/dist/css/bootstrap.min.css";
import Registration from "./Components/Registeration";
import Scan from "./Components/Scan";
import Upload from "./Components/Upload";

function App() {
  return (
    <div className="App">
      <div className="outer">
        <div className="inner">
          <BrowserRouter>
            <Routes>
              <Route path="/register" element={<Registration />} />
              <Route exact path="/" element={<Login />} />
              <Route path="/Home" element={<Home />} />
              <Route path="/Scan" element={<Scan />} />
              <Route path="/Upload" element={<Upload />} />
            </Routes>
          </BrowserRouter>
        </div>
      </div>
    </div>
  );
}

export default App;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Home.js

```
import React, { useState, useEffect } from "react";
import { Link } from "react-router-dom";

function fileScan() {
  <Link to="/Scan" />;
}

function Home() {
  const [data, setData] = useState([]);

  useEffect(() => {
    const options = {
      headers: { "Content-type": "application/x-www-form-urlencoded" },
    };

    fetch(
      `http://localhost:5000/getDetails?username=${localStorage.getItem(
        "name"
      )}`,
      options
    )
      .then((response) => response.json())
      .then((data) => setData(data));
  }, []);

```

```
26    return (
27      <div>
28        <nav className="side">
29          <div className="logo-name">
30            <div className="logo-image"></div>
31            <span className="logo_name">Welcome</span>
32          </div>
33
34          <div className="menu-items">
35            <ul className="nav-links">
36              <li>
37                <a href="/Home">
38                  <i className="uil uil-estate"></i>
39                  <span className="link-name">Dashboard</span>
40                </a>
41              </li>
42              <li>
43                <a href="/Scan">
44                  <i className="uil uil-files-landscapes"></i>
45                  <span onClick={fileScan()} className="link-name">
46                    Scan a cheque
47                  </span>
48                </a>
49              </li>
50            </ul>
51          </div>
52        </nav>
53        <div className="dashboard">
54          <div className="top">
55            <nav className=" navbar navbar-expand-lg navbar-dark bg-dark">
56              <i className=" icon fa-brands fa-neos fa-xl"></i>
```

```
55            <nav className=" navbar navbar-expand-lg navbar-dark bg-dark">
56              <i className=" icon fa-brands fa-neos fa-xl"></i>
57              <button
58                className="navbar-toggler"
59                type="button"
60                data-bs-toggle="collapse"
61                data-bs-target="#navbarSupportedContent"
62                aria-controls="navbarSupportedContent"
63                aria-expanded="false"
64                aria-label="Toggle navigation"
65              >
66                <span className="navbar-toggler-icon"></span>
67              </button>
68              <div
69                className="collapse navbar-collapse "
70                id="navbarSupportedContent"
71              >
72                <ul className="navbar-nav ml-auto mb-1 mt-1 px-5">
73                  <li className="nav-item ms-auto ">
74                    <a className="nav-link logout" href="/">
75                      Logout
76                    </a>
77                  </li>
78                </ul>
79              </div>
80            </nav>
81          </div>
82        </div>
83        <h1>Previously Scanned Cheques</h1>
84        {data.length === 0 ? (
85          <h3>No Cheques scanned so far...</h3>
```

```
          <h3>No Cheques scanned so far...</h3>
        ) : (
          <table className="table table-bordered table1">
            <thead>
              <tr>
                <th scope="col">S.No</th>
                <th scope="col">Pay To</th>
                <th scope="col">Account Number</th>
                <th scope="col">Date</th>
                <th scope="col">Amount</th>
                <th scope="col">Amount in words</th>
              </tr>
            </thead>
            <tbody>
              {data.map((item) => {
                return (
                  <tr key={item.sNo}>
                    <th scope="row">{item.sNo + 1}</th>
                    <td>{item.name}</td>
                    <td>{item.accNo}</td>
                    <td>{item.date}</td>
                    <td>{item.amt}</td>
                    <td>{item.amtWord}</td>
                  </tr>
                );
              })}
```

```
            </tbody>
          </table>
        )}
      </div>
    );
  }

  export default Home;
```

Login.js

```
import React, { useState } from "react";
import { Alert } from "react-bootstrap";
import { useNavigate } from "react-router-dom";
import Home from "./Home";

function Login() {
  const [emaillog, setEmaillog] = useState(" ");
  const [passwordlog, setPasswordlog] = useState(" ");
  const navigate = useNavigate();

  const [flag, setFlag] = useState(false);

  const [home, setHome] = useState(true);

  function handleRegister(e) {
    navigate("/Register");
  }

  function handleLogin(e) {
    e.preventDefault();

    if (emaillog === "" || passwordlog === "") {
      console.log("Enter the values");
    } else {
      var xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function () {
        if (this.readyState === 4 && this.status === 200) {
          if (xhttp.responseText === "Successful") {
            localStorage.setItem("name", emaillog);
            navigate("/home");
          } else {
            console.log(xhttp.responseText);
          }
        }
      };
      var url = "http://localhost:5000/login";
      xhttp.open("POST", url);
      xhttp.setRequestHeader(
        "Content-type",
        "application/x-www-form-urlencoded"
      );
```

```
      xhttp.send("mail=" + emaillog + "&pass=" + passwordlog);
    }
  }

  return (
    <div>
      {home ? (
        <form onSubmit={handleLogin}>
          <h2>LogIn</h2>
          <div className="form-group">
            <label>Email</label>
            <input
              type="email"
              className="form-control"
              placeholder="Enter email"
              onChange={(event) => setEmaillog(event.target.value)}
            />
          </div>

          <div className="form-group">
            <label>Password</label>
            <input
              type="password"
              className="form-control"
              placeholder="Enter password"
              onChange={(event) => setPasswordlog(event.target.value)}
            />
          </div>

          <button
            type="submit"
            className="btn btn-dark btn-lg btn-block button1"
          >
            Login
          </button>

          {flag && (
            <Alert color="primary" variant="warning">
              Fill correct Info else keep trying.
            </Alert>
```

```
          )}
          <p onClick={handleRegister} className="forgot-password text-right">
            Don't have an Account?
          </p>
        </form>
      ) : (
        <Home />
      )}
    </div>
  );
}

export default Login;
```

Registeration.js

```
import React, { useState } from "react";
import { Alert } from "react-bootstrap";
import { useNavigate } from "react-router-dom";
import Login from "./Login";

function Registration() {
  const [name, setName] = useState("");
  const [username, setUsername] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");

  const [flag, setFlag] = useState(false);
  const [login, setLogin] = useState(true);
  const navigate = useNavigate();

  function handleFormSubmit(e) {
    e.preventDefault();
    if (!name || !email || !password || !username) {
      setFlag(true);
    } else {
      var xhttp = new XMLHttpRequest();
      xhttp.onreadystatechange = function () {
        if (this.readyState === 4 && this.status === 200) {
          if (xhttp.responseText === "Successful") {
            navigate("/");
          } else {
            console.log(xhttp.responseText);
          }
        }
      };
      var url = "http://localhost:5000/register";
      xhttp.open("POST", url);
      xhttp.setRequestHeader(
        "Content-type",
        "application/x-www-form-urlencoded"
      );
      xhttp.send(
        "name=" +
          name +
          "&username=" +
          username +
          "&mail=" +
```

```
          "&username=" +
          username +
          "&mail=" +
          email +
          "&pass=" +
          password
      );
      // setLogin(!login);
    }
  }

  function handleClick() {
    setLogin(!login);
  }

  return (
    <>
      <div>
        {" "}
        {login ? (
          <form onSubmit={handleFormSubmit}>
            <h2>Register</h2>

            <div className="form-group">
              <label>Name</label>
              <input
                type="text"
                className="form-control"
                placeholder="Enter Full Name"
                name="name"
                onChange={(event) => setName(event.target.value)}
              />
            </div>

            <div className="form-group">
              <label>Username</label>
              <input
                type="text"
                className="form-control"
                placeholder="Enter your Username"
```

```
            <div className="form-group">
              <label>Email</label>
              <input
                type="email"
                className="form-control"
                placeholder="Enter your email"
                onChange={(event) => setEmail(event.target.value)}
              />
            </div>

            <div className="form-group">
              <label>Password</label>
              <input
                type="password"
                className="form-control"
                placeholder="Enter your password"
                onChange={(event) => setPassword(event.target.value)}
              />
            </div>

            <button
              type="submit"
              className="btn btn-dark btn-lg btn-block button1"
            >
              Register
            </button>
            <p onClick={handleClick} className="forgot-password text-right">
              Already registered log in?
            </p>
            {flag && (
              <Alert color="primary" variant="danger">
                I've got it that you are in hurry! But every Field is
                important...
              </Alert>
            )}
          </form>
        ) : (
          <Login />
        )}
      </div>
    </>
```

Scan.js

```
import { useNavigate } from "react-router-dom";
import { React, useState } from "react";
import axios from "axios";

function Scan() {
  const navigate = useNavigate();
  const [selectedFile, setSelectedFile] = useState(null);

  function handleChange(event) {
    setSelectedFile(event.target.files[0]);
  }

  const handleSubmit = async (event) => {
    event.preventDefault();
    const formData = new FormData();
    formData.append("selectedFile", selectedFile);
    formData.append("filename", selectedFile.name);
    try {
      const response = await axios({
        method: "post",
        url: "http://localhost:5000/upload",
        data: formData,
        headers: { "Content-Type": "multipart/form-data" },
      });
      var query = `?imgUrl=${response.data.imgUrl}&name=${response.data.name}&date=${response.data.date}&amt=${response.data.amt}&accNo=${response.data.accNo}`;
      navigate("/Upload" + query);
    } catch (error) {
      console.log(error);
    }
  };

  return (
    <div>
      <form onSubmit={handleSubmit}>
        <h1>Scan the cheque</h1>
        <h3>Please select the correct file</h3>
        <input type="file" className="upload-input" onChange={handleChange} />
        <button type="submit">Upload</button>
      </form>
    </div>
  );
}
```

Upload.jsx

```jsx
import React, { useState } from "react";
import { useNavigate, useSearchParams } from "react-router-dom";
import { numberToWords } from "amount-to-words";

function Upload() {
  function inWords(num) {
    var str = "";
    str = numberToWords(num) + " only";
    return str;
  }

  const navigate = useNavigate();
  const [searchParams] = useSearchParams();
  const [name, setName] = useState(searchParams.get("name"));
  const [amt, setAmt] = useState(searchParams.get("amt"));
  const [date, setDate] = useState(searchParams.get("date"));
  const [acc, setAcc] = useState(searchParams.get("accNo"));
  const [amtWord, setAmtWord] = useState(inWords(searchParams.get("amt")));

  function saveData() {
    const xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
      navigate("/Home");
    };
    var fname = searchParams.get("imgUrl");
    var username = localStorage.getItem("name");
    xhttp.open("POST", "http://localhost:5000/saveDetails");
    xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhttp.send(
      `username=${username}&name=${name}&amt=${amt}&date=${date}&accNo=${acc}&amtWord=${amtWord}&file=${fname}`
    );
  }

  return (
    <div>
      <img
        src={require("../../../../chequeProcessor/fileUpload/procFile/" +
          searchParams.get("imgUrl"))}
        alt="cheque"
        width="600px"
        height="270px"
      />
```

```
43        <form>
44          <div className="form-group">
45            <label>Name</label>
46            <input
47              type="text"
48              className="form-control"
49              placeholder="Enter Full Name"
50              value={name}
51              onChange={(event) => setName(event.target.value)}
52              name="name"
53            />
54          </div>
55
56          <div className="form-group">
57            <label>Amount</label>
58            <input
59              type="text"
60              className="form-control"
61              placeholder="Enter Amount"
62              value={amt}
63              onChange={(event) => {
64                setAmt(event.target.value);
65                setAmtWord(inWords(event.target.value));
66              }}
67            />
68          </div>
69
70          <div className="form-group">
71            <label>Amount in words</label>
72            <input
73              disabled
74              type="text"
75              value={amtWord}
76              className="form-control"
77            />
78          </div>
79
80          <div className="form-group">
81            <label>Date</label>
82            <input
83              type="text"
84              className="form-control"
```

```
79
80          <div className="form-group">
81            <label>Date</label>
82            <input
83              type="text"
84              className="form-control"
85              placeholder="Enter Date"
86              value={date}
87              onChange={(event) => setDate(event.target.value)}
88            />
89          </div>
90
91          <div className="form-group">
92            <label>Account Number</label>
93            <input
94              type="Phone"
95              className="form-control"
96              placeholder="Enter Accoune Number"
97              value={acc}
98              onChange={(event) => setAcc(event.target.value)}
99            />
100         </div>
101
102         <button
103           type="submit"
104           className="btn btn-dark btn-lg btn-block button1 "
105           onClick={saveData}
106         >
107           Submit
108         </button>
109       </form>
110     </div>
111   );
112 }
113
114 export default Upload;
```

7.2 FLASK CODE

Main.py

```python
import os
import test1
import mysql.connector
import bcrypt
import cv2
from flask import Flask, render_template, request
from flask_cors import CORS
from werkzeug.utils import secure_filename

app = Flask(__name__)
CORS(app)

app.config['UPLOAD_FOLDER'] = "fileUpload"


@app.route('/')
def home():
    return render_template('upload.html')


@app.route('/register', methods=["POST"])
def register():
    # Getting the arguments from the URL
    name = request.form.get('name')
    username = request.form.get('username')
    mail = request.form.get('mail')
    password = request.form.get('pass')

    # Hashing the password
    byte = password.encode('utf-8')
    salt = bcrypt.gensalt()
    password = bcrypt.hashpw(byte, salt)

    # Checking if user already exists
    conn = mysql.connector.connect(host='localhost', user='root', password='', database='chequesystem')
    cursor = conn.cursor()

    cursor.execute(f"SELECT * FROM users WHERE username = '{username}' OR mail = '{mail}'")
    res = cursor.fetchall()
    if len(res) != 0:
        return 'user exists'
```

```
    # Inserting user data
    query = 'INSERT INTO users (name, username, mail, password) VALUES(%s, %s, %s, %s)'
    val = (name, username, mail, password)
    cursor.execute(query, val)
    conn.commit()
    return 'Successful'


@app.route('/login', methods=["POST"])
def login():
    # Getting the arguments from the URL
    mail = request.form.get('mail')
    password = request.form.get('pass')

    # Hashing the password
    password = password.encode('utf-8')

    # Checking if user already exists
    conn = mysql.connector.connect(host='localhost', user='root', password='', database='chequesystem')
    cursor = conn.cursor()

    cursor.execute(f"SELECT password FROM users WHERE mail = '{mail}'")
    res = cursor.fetchall()
    if len(res) == 0:
        return 'Invalid Username'
    if bcrypt.checkpw(password, res[0][0].encode('utf-8')):
        return 'Successful'
    return 'Invalid Password'


@app.route('/upload', methods=["POST"])
def upload():
    f = request.files['selectedFile']
    validExt = ['png', 'jpg', 'jpeg', 'tif', 'eps', 'gif']
    extension = f.filename.split('.')[1]
    if extension not in validExt:
        return "Not a valid extension"
```

```python
@app.route('/upload', methods=["POST"])
def upload():
    f = request.files['selectedFile']
    validExt = ['png', 'jpg', 'jpeg', 'tif', 'eps', 'gif']
    extension = f.filename.split('.')[1]
    if extension not in validExt:
        return "Not a valid extension"

    fname = secure_filename(str(len(os.listdir('fileUpload')))) + "." + extension
    f.save(os.path.join(app.config['UPLOAD_FOLDER'], fname))
    img = cv2.imread("fileUpload\\" + fname)
    res = test1.crop(img, fname)
    res["imgUrl"] = fname
    res["date"] = res["date"][:2] + "/" + res["date"][2:4] + "/" + res["date"][4:]
    return res


@app.route('/get_image', methods=["POST"])
def get_image():
    fname = request.form.get("name")
    imgPath = os.path.abspath("fileUpload/procFile/" + fname)
    return imgPath


@app.route('/saveDetails', methods=["POST"])
def saveDetails():
    # Get Parameters from the request
    username = request.form.get('username')
    name = request.form.get('name')
    amt = request.form.get('amt')
    date = request.form.get('date')
    accNo = request.form.get('accNo')
    amtWord = request.form.get('amtWord')
    fname = request.form.get('file')

    # Creating a database cursor
    conn = mysql.connector.connect(host='localhost', user='root', password='', database='chequesystem')
    cursor = conn.cursor()
```

```
    # Inserting user data
    query = 'INSERT INTO cheque(username, payName, amt, accNo, date, amtWord, fname) VALUES(%s, %s, %s, %s, %s, %s, %s)'
    val = (username, name, amt, accNo, date, amtWord, fname)
    cursor.execute(query, val)
    conn.commit()
    return "Successful"


@app.route('/getDetails', methods=["GET"])
def getDetails():
    username = request.args.get("username")
    conn = mysql.connector.connect(host='localhost', user='root', password='', database='chequesystem')
    cursor = conn.cursor()

    cursor.execute(f"SELECT * FROM cheque WHERE username = '{username}'")
    data = cursor.fetchall()
    res = []
    for i, d in enumerate(data):
        temp = {'sNo': i, 'name': d[2], 'amt': d[3], 'accNo': d[4], 'date': d[5], 'amtWord': d[6]}
        res.append(temp)
    return res


if __name__ == '__main__':
    app.run()
```

Number Finder.py

```python
import cv2
import numpy as np
from tensorflow import keras
import tensorflow as tf

model = keras.models.load_model('models/number.h5')


# Function to sort the contours
def sort_contours(cnts, method="left-to-right"):
    # Initialize the reverse flag and sort index
    reverse = False
    i = 0

    # Handle if we need to sort in reverse
    if method == "right-to-left" or method == "bottom-to-top":
        reverse = True

    # Handle if we are sorting against the y-coordinate rather than the x-coordinate of the bounding box
    if method == "top-to-bottom" or method == "bottom-to-top":
        i = 1

    # Construct the list of bounding boxes and sort them from top to bottom
    boundingBoxes = [cv2.boundingRect(c) for c in cnts]
    (cnts, boundingBoxes) = zip(*sorted(zip(cnts, boundingBoxes), key=lambda b: b[1][i], reverse=reverse))

    # Return the list of sorted contours and bounding boxes
    return cnts


# Function to predict the numbers in the image
def predict_number(img0, name):
    res = ""

    # Converting the image to grayscale
    gray = cv2.cvtColor(img0, cv2.COLOR_BGR2GRAY)

    # Binarizing and removing noise from the image
    th, threshed = cv2.threshold(gray, 160, 255, cv2.THRESH_OTSU | cv2.THRESH_BINARY_INV)
    morphed = cv2.morphologyEx(threshed, cv2.MORPH_OPEN, np.ones((2, 2)))

```

```
    # Finding all possible contours in the image and sorting them according to their order of occurances from left to right
    cnts = cv2.findContours(morphed, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
    cnts = sort_contours(cnts)

    # Working with each of the contours
    for cnt in cnts:

        # Storing the values of bounding box of the current contour
        x, y, w, h = cv2.boundingRect(cnt)

        # Setting the minimum area a contour must satisfy and starting next iteration if condition is not met
        minArea = 100
        if (h * w < minArea) or (w > h):
            continue

        try:
            # Cropping the image with the bounding box info
            cropped = threshed[y - 5:y + h + 5, x - 5:w + x + 5]
            # Preprocessing the cropped image as per our model need
            resized = cv2.resize(cropped, (28, 28), interpolation=cv2.INTER_AREA)
        except:
            # Cropping the image with the bounding box info
            cropped = threshed[y:y + h, x:w + x]
            # Preprocessing the cropped image as per our model need
            resized = cv2.resize(cropped, (28, 28), interpolation=cv2.INTER_AREA)

        # Drawing contours for visual purpose
        cv2.rectangle(imgO, (x, y), (x + w, y + h), (255, 0, 255), 1, cv2.LINE_AA)

        newImg = tf.keras.utils.normalize(resized, axis=1)
        newImg = np.array(newImg).reshape(-1, 28, 28, 1)

        # Predicting the output
        prediction = model.predict(newImg)

        # Storing the result
        op = np.argmax(prediction)
        res += str(op)

    return res
```

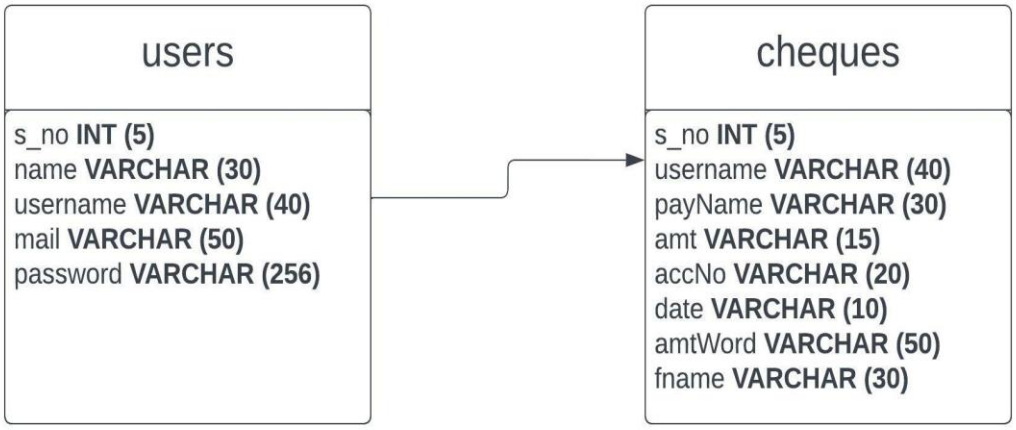
Test1.py

```python
import numberFinder
import cv2
import numpy as np
import pytesseract

pytesseract.pytesseract.tesseract_cmd = 'D:\\Softwares\\Tesseract\\tesseract.exe'

#############################################################

roiPnts = [[(1218, 70), (1540, 126), 'number', 'date'],
               [(1200, 332), (1540, 416), 'number', 'amt'],
               [(212, 448), (520, 498), 'number', 'accNo'],
               [(4, 760), (1590, 888), 'micr', 'micr'],
               [(116, 170), (1240, 250), 'text', 'name']]


#############################################################


def stack_images(scale, imgArray):
    rows = len(imgArray)
    cols = len(imgArray[0])
    rowsAvailable = isinstance(imgArray[0], list)
    width = imgArray[0][0].shape[1]
    height = imgArray[0][0].shape[0]
    if rowsAvailable:
        for x in range(0, rows):
            for y in range(0, cols):
                if imgArray[x][y].shape[:2] == imgArray[0][0].shape[:2]:
                    imgArray[x][y] = cv2.resize(imgArray[x][y], (0, 0), None, scale, scale)
                else:
                    imgArray[x][y] = cv2.resize(imgArray[x][y], (imgArray[0][0].shape[1], imgArray[0][0].shape[0]),
                                                None, scale, scale)
                if len(imgArray[x][y].shape) == 2: imgArray[x][y] = cv2.cvtColor(imgArray[x][y], cv2.COLOR_GRAY2BGR)
        imageBlank = np.zeros((height, width, 3), np.uint8)
        hor = [imageBlank] * rows
        hor_con = [imageBlank] * rows
        for x in range(0, rows):
            hor[x] = np.hstack(imgArray[x])
        ver = np.vstack(hor)
```

```
    else:
        for x in range(0, rows):
            if imgArray[x].shape[:2] == imgArray[0].shape[:2]:
                imgArray[x] = cv2.resize(imgArray[x], (0, 0), None, scale, scale)
            else:
                imgArray[x] = cv2.resize(imgArray[x], (imgArray[0].shape[1], imgArray[0].shape[0]), None, scale, scale)
            if len(imgArray[x].shape) == 2: imgArray[x] = cv2.cvtColor(imgArray[x], cv2.COLOR_GRAY2BGR)
        hor = np.hstack(imgArray)
        ver = hor
    return ver


def crop(img, fname):
    data = {'name': fname}
    img = cv2.resize(img, (1600, 900))
    cv2.imwrite('fileUpload/procFile/' + fname, img)

    imgShow = img.copy()
    imgMask = np.zeros_like(imgShow)

    for x, r in enumerate(roiPnts):
        cv2.rectangle(imgMask, ((r[0][0]), r[0][1]), ((r[1][0]), r[1][1]), (0, 255, 0), cv2.FILLED)
        imgShow = cv2.addWeighted(imgShow, 0.99, imgMask, 0.1, 0)
        imgCrop = img[r[0][1]:r[1][1], r[0][0]:r[1][0]]

        if r[2] == 'number':
            if r[3] == 'date':
                num = numberFinder.predict_number(imgCrop, r[3])
                data[r[3]] = num

            elif r[3] == 'accNo':
                gray = cv2.cvtColor(imgCrop, cv2.COLOR_BGR2GRAY)
                th, threshed = cv2.threshold(gray, 150, 255, cv2.THRESH_OTSU | cv2.THRESH_BINARY_INV)
                morphed = cv2.morphologyEx(threshed, cv2.MORPH_OPEN, np.ones((2, 2)))
                num = pytesseract.image_to_string(morphed)
                num.strip()
                num = ''.join(num.splitlines())
                data[r[3]] = num
            else:
                num = numberFinder.predict_number(imgCrop, r[3])
                data[r[3]] = num

```

```
        elif r[2] == 'text':
            text = pytesseract.image_to_string(imgCrop)
            text.strip()
            text = ''.join(text.splitlines())
            data[r[3]] = text
    return data
```

7.3 DATABASE SCHEMA



CHAPTER 8
TESTING

8.1 TEST CASES

Test caseID	Feature type	Component	Test Scenario	Excepted Result	Actual Result	Status	Executed By
Homepage_TC_001	Functional	Home Page	Verify user is able to see the Homepage when clicked on the link	Home page should be displayed	Working as expected	PASS	Naryanan M S Thaun Kumar P
Homepage_TC_002	UI	Home Page	Verify the UI elements in Homepage	The homepage must be displayed properly	Working as expected	PASS	Naryanan M S Thaun Kumar P
Homepage_TC_003	Functional	Image Uploading	Check if the user can upload image from local file structure	The uploaded file must be displayed on the website	Working as expected	PASS	Akash Iyer Ramalingam V
Homepage_TC_004	Functional	Dashboard	Check if user can see the previously processed cheques	The previously processed cheques details must be visible on the dashboard	Working as expected	PASS	Akash Iyer Ramalingam V
Homepage_TC_005	Functional	Home Page	Check if user cannot upload unsupported files	The application Should not allow user to select a non-image file	User is able to upload only image file	PASS	Thaun Kumar P Akash Iyer
Homepage_TC_006	Functional	Home Page	Check if the page redirects to result page once the input is given	The page should redirect to the result page	Working as expected	PASS	Naryanan M S Ramalingam V

BE_TC_001	Functional	Backend	Check if the connection is correctly established	The localhost connection must be correctly established	Working as expected	**PASS**	Naryanan M S Ramalingam V
M_TC_001	Functional	Model	Check if the model can handle various sizes	The model should rescale the image and predict the results	Working as expected	**PASS**	Ramalingam V Akash Iyer
M_TC_002	Functional	Model	Check if the model predicts the digits	The model should predict the number	Working as expected	**PASS**	Narayanan M S Tharun Kumar P
M_TC_003	Functional	Model	Check if the model handles complex input image	The model should predict the number in the complex image	The model fails to identify the digit since the model is not built to handle such data	**PASS**	Akash Iyer Narayanan M S
Predict_TC_OO5	Functional	Result page	Verify user is able to navigate to the predict to and view the predicted result	User must be navigated to the predict page and must view the predicted result	Working as expected	**PASS**	Ramalingam V Narayanan M S
R_TC_001	UI	Result Page	Verify UI elements in the Result page	The result page must be displayed properly	The result displayed as expected	**PASS**	Akash Iyer Tharun Kumar P
R_TC_002	UI	Result Page	Check if the input image is displayed properly	The input page must be displayed properly	Working as expected	**PASS**	Akash Iyer Tharun kumar P
R_TC_003	UI	Result Page	Check if the result image is displayed properly	The result should be displayed properly	Working as expected	**PASS**	Narayanan M S Ramalingam V

8.2 USER ACCEPTANCE TESTING

8.2.1 DEFECT ANALYSIS

Resolution	Severity level 1	Severity level 2	Severity level 3	Severity level 4	Total
By Design	0	0	1	0	**1**
Duplicate	0	0	0	0	**0**
External	0	0	1	0	**1**
Fixed	3	1	0	1	**5**
Not Reproduced	0	0	0	1	**1**
Skipped	0	0	0	0	**0**
Won't Fix	0	0	0	0	**0**
Total	3	1	2	2	**8**

8.2.2 TEST CASE ANALYSIS

Section	Total Cases	Not Tested	Fail	Pass
Client Application	10	0	1	79
Security	2	0	1	1
Performance	3	0	1	2
Exception Reporting	2	0	0	2

CHAPTER 9
RESULTS

9.1 PERFORMANCE METRICS

Locust Test Report

During: 11/12/2022, 7:05:40 AM - 11/12/2022, 7:14:47 AM

Target Host: http://127.0.0.1:5000/

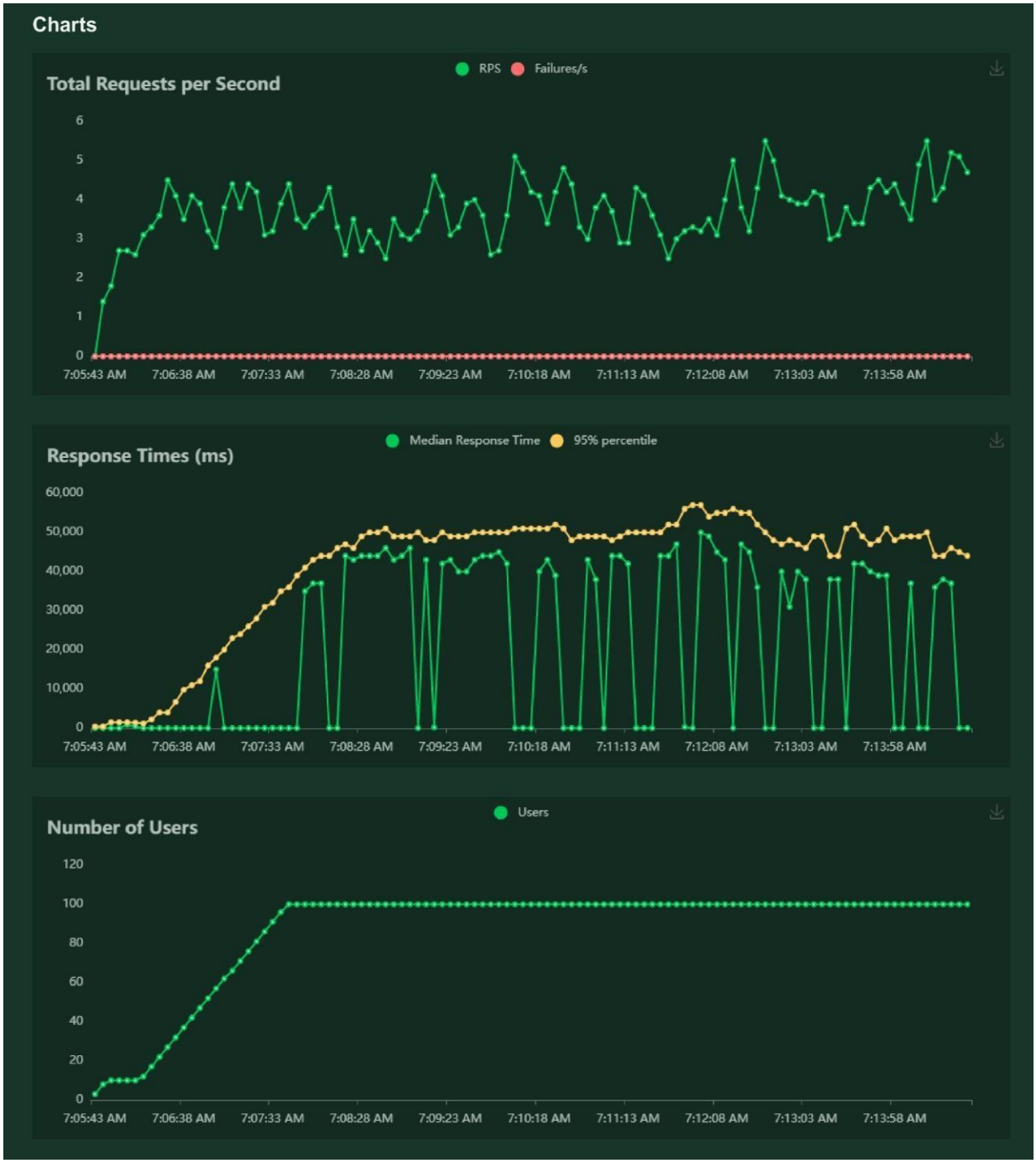
Script: locust.py

Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	//	1043	0	13	4	290	1079	1.9	0.0
GET	//predict	1005	0	39648	385	59814	2670	1.8	0.0
	Aggregated	**2048**	**0**	**19462**	**4**	**59814**	**1859**	**3.7**	**0.0**

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	//	10	11	13	15	19	22	62	290
GET	//predict	44000	46000	47000	48000	50000	52000	55000	60000
	Aggregated	**36**	**36000**	**43000**	**45000**	**48000**	**50000**	**54000**	**60000**


Charts
Total Requests per Second
RPS
Failures/s
6
5
4
3
2
1
0
7:05:43 AM
7:06:38 AM
7:07:33 AM
7:08:28 AM
7:09:23 AM
7:10:18 AM
7:11:13 AM
7:12:08 AM
7:13:03 AM
7:13:58 AM
Response Times (ms)
Median Response Time
95% percentile
60,000
50,000
40,000
30,000
20,000
10,000
0
7:05:43 AM
7:06:38 AM
7:07:33 AM
7:08:28 AM
7:09:23 AM
7:10:18 AM
7:11:13 AM
7:12:08 AM
7:13:03 AM
7:13:58 AM
Number of Users
Users
120
100
80
60
40
20
0
7:05:43 AM
7:06:38 AM
7:07:33 AM
7:08:28 AM
7:09:23 AM
7:10:18 AM
7:11:13 AM
7:12:08 AM
7:13:03 AM
7:13:58 AM

CHAPTER 10
ADVANTAGES & DISADVANTAGES

ADVANTAGES

- Reduces manual work
- More accurate than average human
- Capable of handling a lot of data
- Can be used anywhere from any device

DISADVANTAGES

- Poor in handling illegible handwriting
- Cheques must be scanned and uploaded as a picture
- Prone to occasional errors

CHAPTER 11
CONCLUSION

This project demonstrated a web application that uses machine learning to recognise handwritten numbers and text. Flask, React and a few other technologies were used to create this project. The model predicts the handwritten digit using a CNN network. During testing, the model achieved a 97.% recognition rate. The proposed project is scalable and can easily handle a huge number of users. Since it is a web application, it is compatible with any device that can run a browser. This project is extremely useful in real-world scenarios such as processing bank cheque amounts, numeric entries in forms filled up by hand (tax forms) and so on. There is so much room for improvement, which can be implemented in subsequent versions.

CHAPTER 12
FUTURE SCOPE

This project is far from complete and there is a lot of room for improvement. Some of the improvements that can be made to this project are as follows:

- Add support to detect from digits multiple images and save the results
- Add support to detect multiple digits
- Improve model to detect digits from complex images
- Add support to different languages to help users from all over the world

This project has endless potential and can always be enhanced to become better. Implementing this concept in the real world will benefit several industries and reduce the workload on many workers, enhancing overall work efficiency.

APPENDIX

SOURCE CODE

MODEL CREATION

1. Importing the libraries and the dataset

In [57]:
```
import numpy
import tensorflow
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.layers import Conv2D
from keras.optimizers import Adam
from keras.utils import np_utils
```

In [58]:
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
print(x_train.shape)
print(x_test.shape)
```

```
(60000, 28, 28)
(10000, 28, 28)
```

In [60]:
```
#Reshape the data
x_train = x_train.reshape(60000, 28, 28, 1).astype('float32')
x_test = x_test.reshape(10000, 28, 28, 1).astype('float32')
```

In [61]:
```
# Applying one-hot encoding
classNum = 10
y_train = np_utils.to_categorical(y_train, classNum)
y_test = np_utils.to_categorical(y_test, classNum)
```

In [62]:
```
# Creating the model
model = Sequential()
model.add(Conv2D(64, (3, 3), input_shape = (28, 28, 1), activation = 'relu'))
model.add(Conv2D(32, (3, 3), activation = 'relu'))
model.add(Flatten())
model.add(Dense(classNum, activation = 'softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer = "Adam", metrics = ['accuracy'])
model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs = 5, batch_size = 32)
```

```
Epoch 1/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.2987 - accuracy: 0.9466 - val_loss: 0.0998 - val_accuracy: 0.9681
Epoch 2/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0727 - accuracy: 0.9776 - val_loss: 0.0977 - val_accuracy: 0.9730
Epoch 3/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0522 - accuracy: 0.9840 - val_loss: 0.0851 - val_accuracy: 0.9764
Epoch 4/5
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0405 - accuracy: 0.9869 - val_loss: 0.1012 - val_accuracy: 0.9728
Epoch 5/5
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0325 - accuracy: 0.9897 - val_loss: 0.1154 - val_accuracy: 0.9734
```

Out[62]:

In [63]:
```
metrics = model.evaluate(x_test, y_test, verbose = 0)
print("Test loss & Test Accuracy:")
print(metrics)
```

```
Test loss & Test Accuracy:
[0.11542319506406784, 0.9733999967575073]
```

In [64]:
```
model.save('models/number.h5')
```

In [64]:
```
model.save('models/number.h5')
```

In [72]:
```
# Taking random images and testing the model
from tensorflow.keras.models import load_model
from PIL import Image
import numpy as np

tModel = load_model(r'/content/models/number.h5')
img = Image.open('/content/download.png').convert("L")
img = img.resize((28, 28))
imgArr = np.array(img).reshape(1, 28, 28, 1)
pred = model.predict(imgArr)
print(np.argmax(pred, axis = 1))
```

```
1/1 [==============================] - 0s 22ms/step
[4]
```

GitHub and Project Demo Link

GitHub link- https://github.com/IBM-EPBL/IBM-Project-29046-1660120371
Project Demo Link- https://youtu.be/kYuLewEDcK8