SUMMER TRAINING PROJECT REPORT

(TERM JUNE-JULY 2025)

"VOTE-XPRESS: AN INTERACTIVE ONLINE VOTING SYSTEM"

Submitted By:

Aditya Raj – 12305995

Mihir Anand - 12304225

Course: Advanced Data Structure

Course Code: PETV84

Under the Guidance of

Dr. Sarneet Kaur

School of Computer Science and Engineering

Annexure-II: Student Declaration

To whom so ever it may concern

We, Aditya Raj(12305995) and Mihir Anand(12304225)

hereby declare that the work entitled "Vote-Xpress – Online Voting System" carried out from June 10, 2025 to July 19, 2025 is our original work done under the guidance of Dr. Sarneet Kaur, and has not been submitted elsewhere for any other degree or diploma.

Signatures:

Aditya Raj (12305995)

Mihir Anand (12304225)

Certificate:-

CENTRE FOR
PROFESSIONAL
ENHANCEMENT

Certificate No. 405711

NAAC GRADE A++

Certificate of Merit

This is to certify that Mr./Ms. **Aditya Raj** S/D/W/o **Mr. Upendra Kumar**

student of **School of Computer Science and Engineering** Registration No. **12305995**

pursuing **Bachelor of Technology (Computer Science and Engineering)** completed

skill development course named **Advanced Data Structure**

organized by **Centre for Professional Enhancement** Lovely Professional University

from **10 June 2025** to **19 July 2025** and obtained **A** Grade.

Date of Issue : 13-08-2025
Place of Issue: Phagwara (India)

Prepared by
(Administrative Officer-Records)

Programme Coordinator
Centre for Professional Enhancement

Head of School
School of Computer Applications

Acknowledgement:-

We would like to express our heartfelt gratitude to Dr. Sarneet Kaur, our project guide, for her valuable support, encouragement, and expert advice throughout this project. Her constructive feedback and continuous motivation helped us complete this work successfully.

We are also thankful to all faculty members of the School of Computer Science and Engineering for providing the right environment and resources. Special thanks to our friends and family for their encouragement, patience, and belief in us during this journey.

Table of Contents:-

Chapter 1: Introduction:-

1.1 Background:-

Voting is one of the most important pillars of democracy. Traditionally, voting is carried out through ballot papers or electronic voting machines (EVMs). Although these systems work, they are not always efficient for smaller institutions such as schools, colleges, universities, and organizations. Manual counting, duplicate entries, and lack of interactive features often make the process less engaging.

With the rise of digital technologies, it is now possible to design voting systems that are **fast, secure, interactive, and user-friendly**. This project, **Vote-Xpress**, was created to provide a modern digital platform where users can register, cast their vote, and view results in real-time.

1.2 About the Project:-

Vote-Xpress is a desktop-based interactive voting application developed using Python and Tkinter. It not only allows secure voting but also integrates several innovative features such as:

- One-person-one-vote enforcement
- Live results with charts
- Admin panel for result declaration
- Feedback system
- Civic awareness quiz
- AI-powered chatbot using Gemini API
- Civic facts, slogans, and thoughts of the day

1.3 Importance of the Project:-

- Ensures fair voting for institutions and organizations
- Reduces human errors and resource wastage
- Makes voting more engaging with interactive features
- Educates users about democracy through quizzes and facts
- Integrates modern AI tools for assistance

1.4 Objectives:-

The main objectives of Vote-Xpress are:

1. To provide a secure digital voting system.
2. To ensure vote integrity by preventing multiple votes from one person.
3. To make the system interactive and educational with quizzes and civic insights.
4. To allow real-time result visualization.
5. To integrate an AI chatbot for instant help and support.

Chapter 2: Training Overview:-

2.1 Tools and Technologies Used:-

- Programming Language: Python 3.12
- GUI Framework: Tkinter
- Other Libraries: threading, json, math, datetime, requests
- AI Service: Google Gemini API for chatbot
- IDE Used: Visual Studio Code, IDLE

2.2 Areas Covered During Training:-

During this internship project, we explored:

- GUI Design with Tkinter – creating windows, buttons, labels, and custom layouts.
- Event-driven programming – handling user actions like button clicks and form submissions.
- API Integration – connecting with Gemini API for chatbot responses.
- Multithreading – ensuring the chatbot works without freezing the UI.
- Data structures – using dictionaries, sets, and lists for vote storage, user validation, and quiz handling.

2.3 Weekly Work Summary:-

- Week 1: Project setup, environment installation, and splash screen design.
- Week 2: Development of registration and voting modules.
- Week 3: Result display and admin panel creation.
- Week 4: Feedback, quiz integration, chatbot setup, and testing.

This structured training helped us improve not only our coding skills but also project management and problem-solving skills.

Chapter 3: Literature Review & Existing Systems

3.1 Manual Voting System:-

In manual voting, ballot papers are distributed to voters, who mark their choice and submit it. Although simple, it has problems such as:

- Time-consuming counting
- Risk of invalid or duplicate votes
- Resource intensive (paper, manpower)

3.2 Electronic Voting Machines (EVMs):-

EVMs are widely used in India for government elections. They solved many problems of paper ballots but are still limited:

- Require physical presence of voters
- Expensive to maintain and distribute
- Limited interactivity

3.3 Existing Online Voting Systems:-

Several web-based voting platforms exist, but many of them:

- Do not provide real-time result visualization
- Lack features like feedback collection or quizzes
- Do not integrate modern technologies like AI chatbots

3.4 Research Gap:-

Vote-Xpress fills this gap by creating a lightweight, secure, and interactive desktop application that can be used in schools, colleges, and organizations. It combines voting, learning, and AI assistance in one platform.

Chapter 4: Problem Definition and Objectives

4.1 Problem Definition:-

Conventional voting systems are:

- Prone to errors and duplicate votes
- Time-consuming in result declaration
- Lacking in educational or interactive features

4.2 Objectives:-

The objectives of Vote-Xpress are:

- Ensure secure, single-vote-per-user validation.
- Provide live vote counting and results.
- Offer educational features like quizzes and civic facts.
- Provide AI chatbot support for help and FAQs.
- Collect feedback to improve user experience.

4.3 Scope of the Project:-

- Designed for academic institutions, clubs, and organizations.
- Can be extended to larger organizations with database support.
- Future versions can use blockchain for higher-level security.

Chapter 5: System Analysis and Design:-

5.1 Functional Requirements:-

- User Registration
- Voting Module
- Admin Panel
- Quiz Module
- Feedback Collection
- Chatbot Integration
- Real-time Result Display

5.2 Non-Functional Requirements:-

- Usability: Simple interface for non-technical users.
- Reliability: Prevent duplicate votes.
- Performance: Smooth performance with multithreading.
- Security: Password-protected admin panel.

5.3 System Design:-

Architecture:

1. **Presentation Layer:** GUI built using Tkinter.
2. **Logic Layer:** Python functions for voting, validation, and chatbot calls.
3. **Data Handling Layer:** Dictionaries, sets, and lists for temporary storage.

5.4 Diagrams (described in text):-

- **DFD (Level 1):** Users register → vote → results stored → admin view.
- **UML Class Diagram:** Main classes like VoteApp, SplashScreen, FinalResultsWindow.
- **Use Case Diagram:** Actors: Voter, Admin. Actions: Register, Vote, View Result, Admin Control.

Chapter 6: Project Details:-

6.1 Title of the Project:-

Vote-Xpress: An Interactive Online Voting System

6.2 System Requirements:-

Hardware Requirements:

- Minimum: Dual Core Processor, 4GB RAM, 500MB free disk space
- Recommended: i5 Processor, 8GB RAM, 1GB free disk space

Software Requirements:

- Operating System: Windows 10 / Linux / MacOS
- Programming Language: Python 3.10+
- Libraries: Tkinter, threading, json, math, datetime, requests
- API: Gemini API for chatbot integration
- IDE: Visual Studio Code / IDLE

6.3 Scope of the Project:-

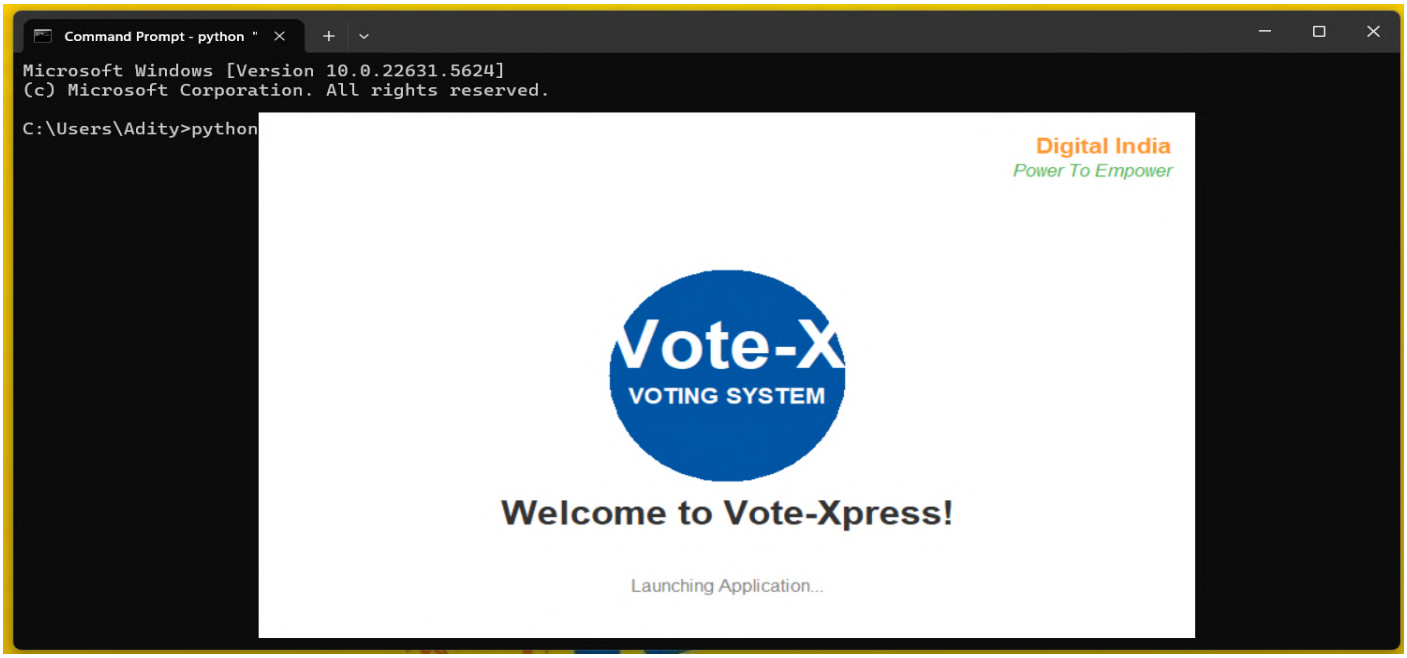
The system can be used by:

- **Colleges and Universities** for student elections
- **Clubs and Societies** for leadership voting
- **Organizations** for team decisions
- **Educational purposes** for teaching civic awareness

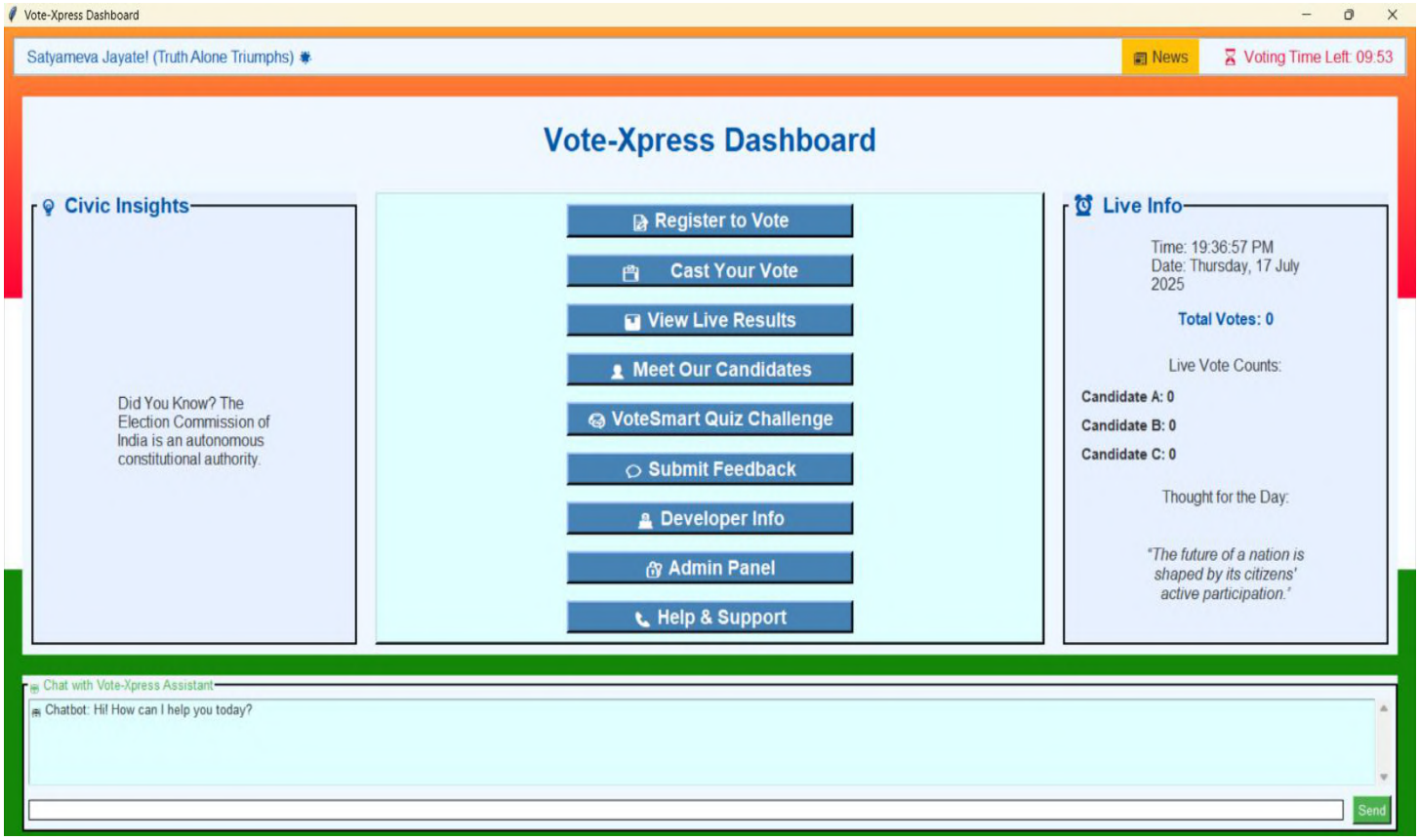
6.4 Features of the System:-

1. **Splash Screen:** Animated welcome with “Digital India” theme.
2. **User Registration:** Secure registration with name and voter ID.
3. **Voting Tab:** Allows casting a single vote per voter.
4. **Result Tab:** Displays real-time voting results with graphs.
5. **Admin Panel:** Password-protected area to declare final results.
6. **Quiz Tab:** Multiple-choice civic quiz to promote awareness.
7. **Feedback Tab:** Collects remarks and ratings from users.
8. **Chatbot Tab:** Provides instant support through Gemini API.
9. **Civic Insights & Slogans:** Motivational content shown on dashboard.
10. **Developer Info:** Details of project developers and their LinkedIn profile.

1.) App Open:-



2.)Dashboard of the Application(with all features and chatbot):-



Chapter 7: Implementation

7.1 Methodology

The project was implemented using an event-driven modular approach. Each feature (registration, voting, results, quiz, chatbot, etc.) was developed as a separate module.

7.2 Module Descriptions:-

1. Splash Screen:

- Displays Vote-Xpress logo in a circle.
- Shows "Welcome to Vote-Xpress" with a “Digital India” theme.

- Launches application automatically after 5 seconds.

2. Registration Module:

- Users enter their **Full Name** and **Voter ID**.
- System checks for valid inputs.
- Displays a colorful registration receipt with user details.

3. Voting Module:

- User selects candidate from available options.
- Ensures one person votes only once by checking voter ID.
- Generates a vote receipt with candidate details.

4. Results Module:

- Real-time results shown using bar charts.
- Displays total votes and candidate-wise votes.
- Admin can publish final results with celebration animations like fireworks and confetti.

5. Admin Panel:

- Password-protected (default password: admin123).
- Allows secure access to declare results.

6. Quiz Module:

- Contains 10 multiple-choice questions on Indian democracy.
- Timer-based quiz with scoring system.
- Awards stars and badges for good performance.

7. Feedback Module:

- Collects user opinions and ratings.
- Stores them for improvement analysis.

8. Chatbot Module:

- Integrated with **Google Gemini API**.
- Provides instant answers to user questions.
- Uses multithreading so the chatbot does not freeze the UI.

9. Civic Insights & Slogans:

- Rotating panel shows civic facts like “India is the largest democracy”.
- Displays patriotic slogans like “Mera Bharat Mahan”.
- Shows inspirational “Thought of the Day”.

10. Developer Information & Helpdesk:

- Displays developer names, contact details, and LinkedIn links.
- Provides help contact number and email for support.

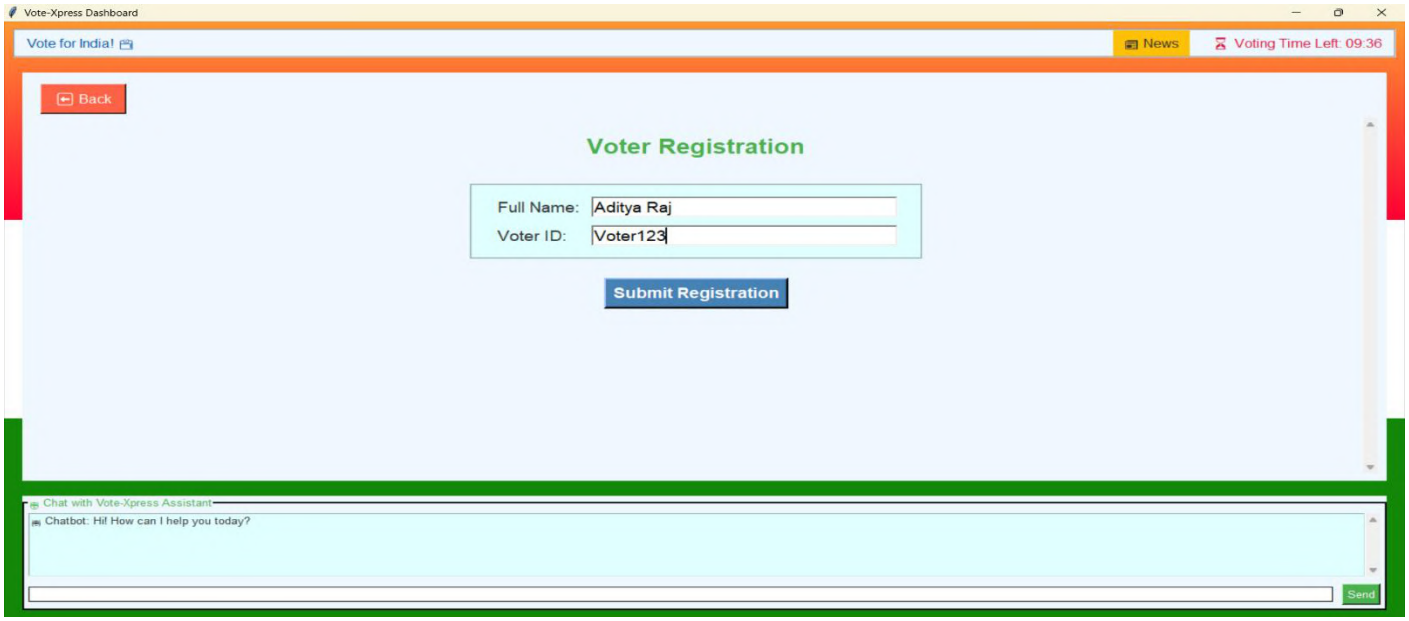
7.3 Color Scheme and UI Design:-

- **Primary Blue:** Branding color for buttons and headers.
- **Green & Orange:** Inspired by the Indian flag.
- **Light Backgrounds:** To keep UI clean and professional.

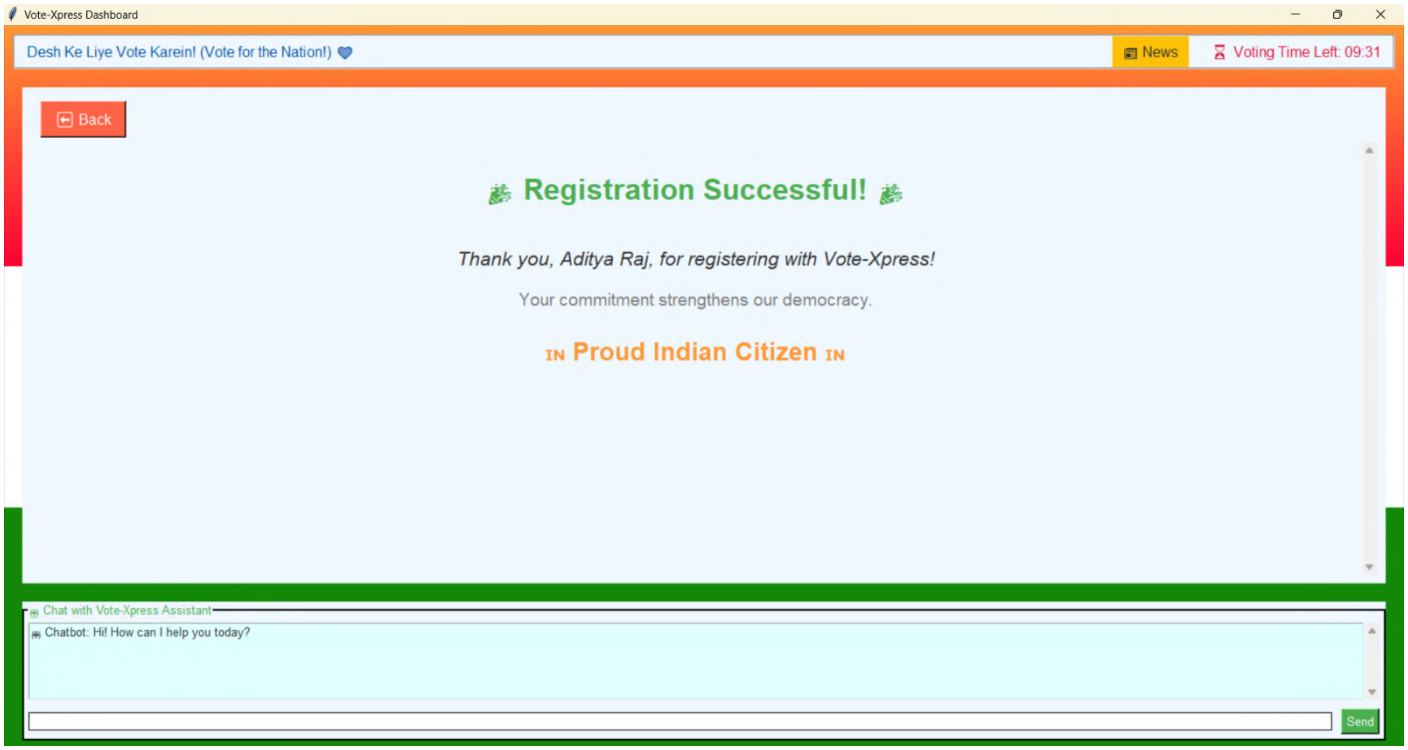
7.4 Coding Style:-

- Code divided into classes (VoteApp, SplashScreen, FinalResultsWindow).
- Constants defined for colors, fonts, slogans, quiz questions.
- Data stored in dictionaries and sets (e.g., votes = {}, voted_users = set()).

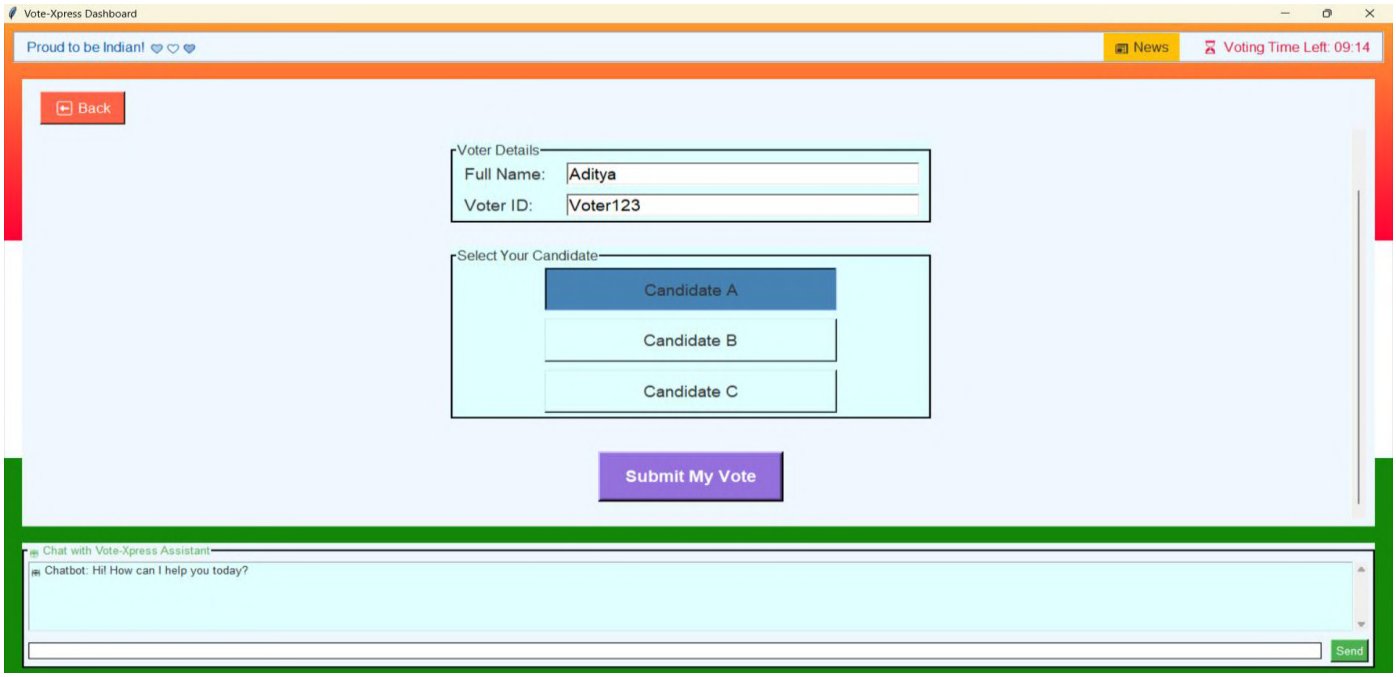
3.) Voter Registration Page:-



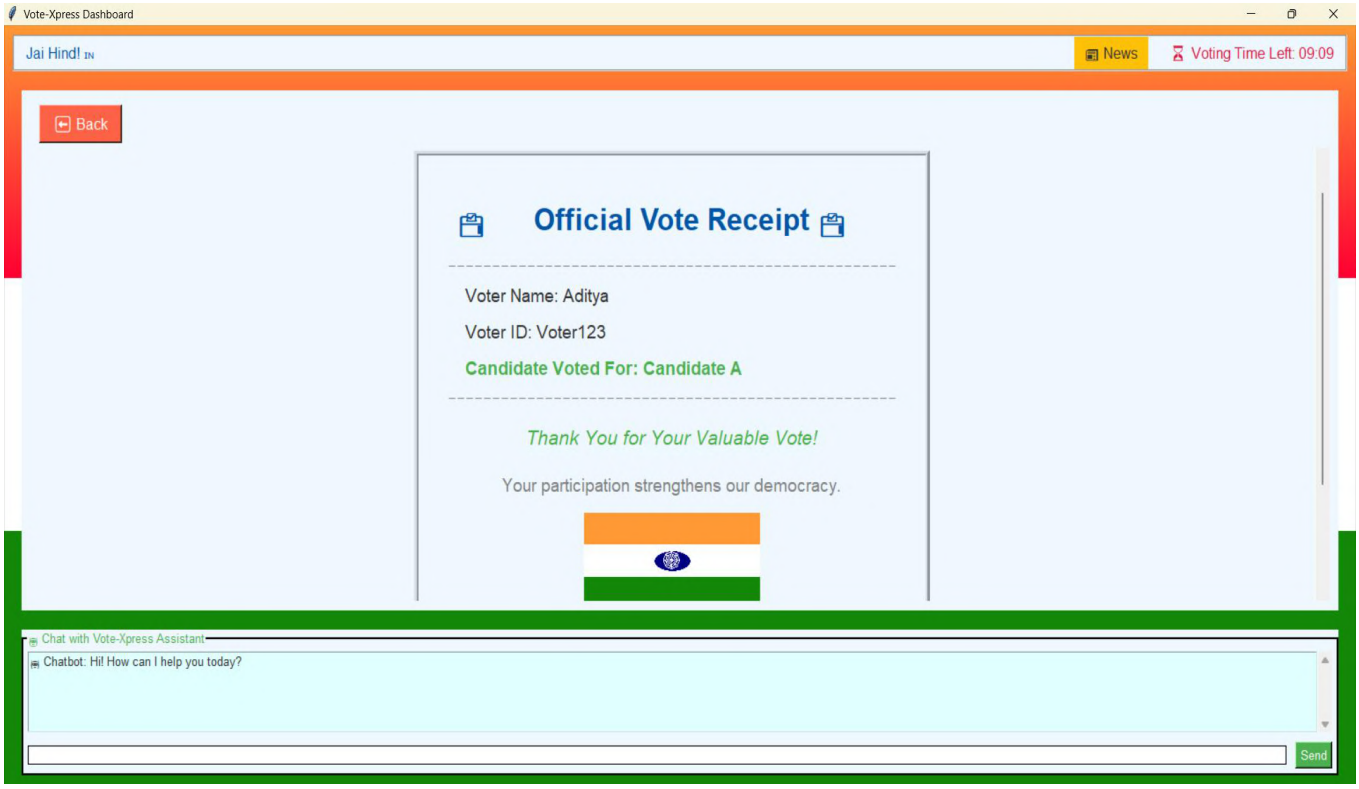
4.)Voter Registered Successfully on Application:-



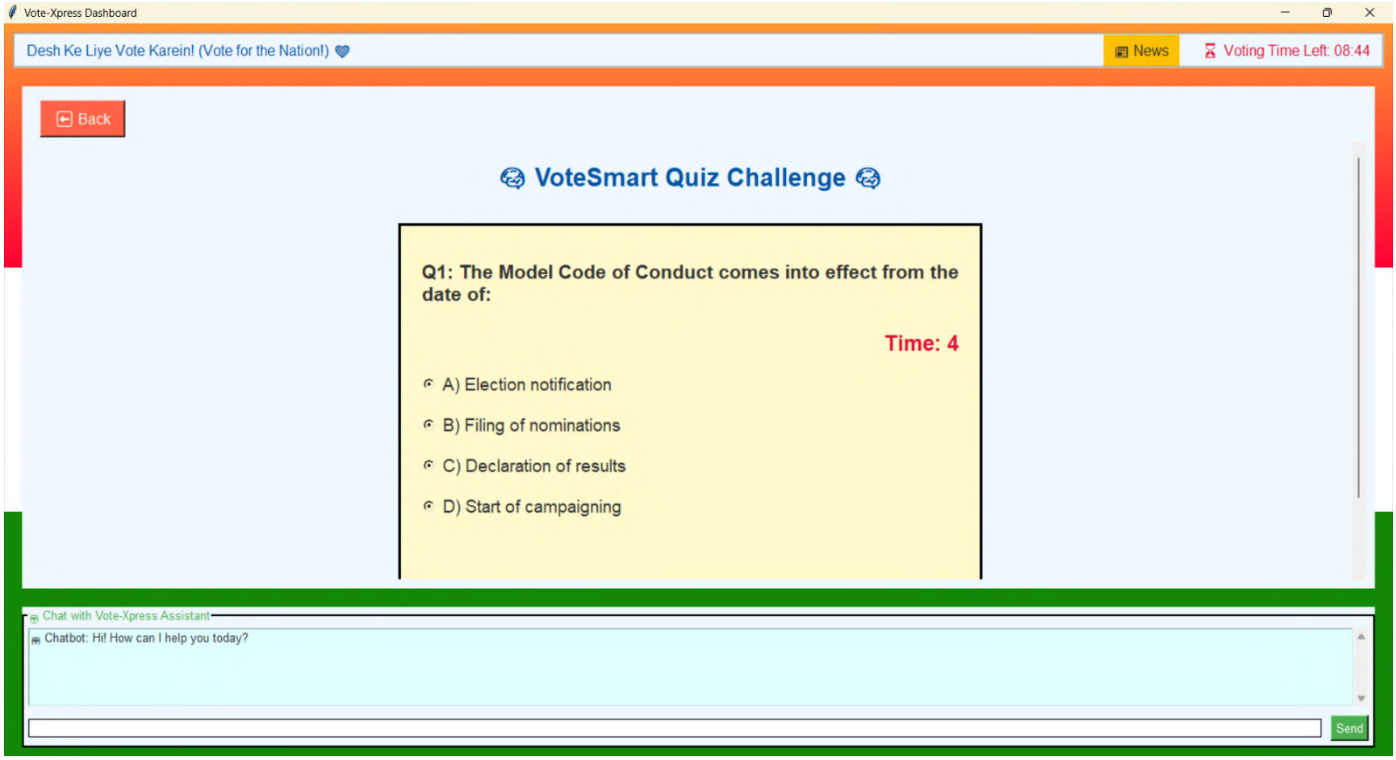
5.) Vote Cast Page:-



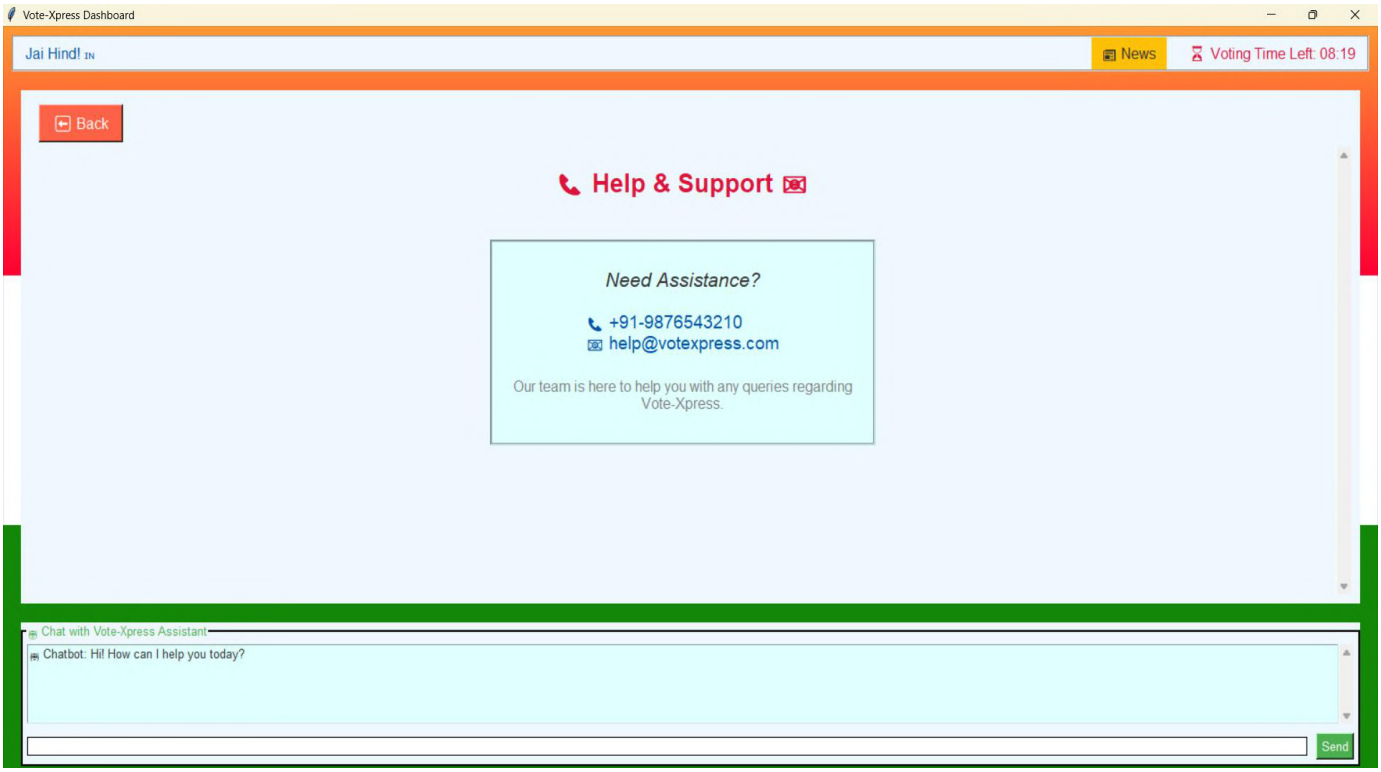
6.) Vote Successfully cast(with receipt):-



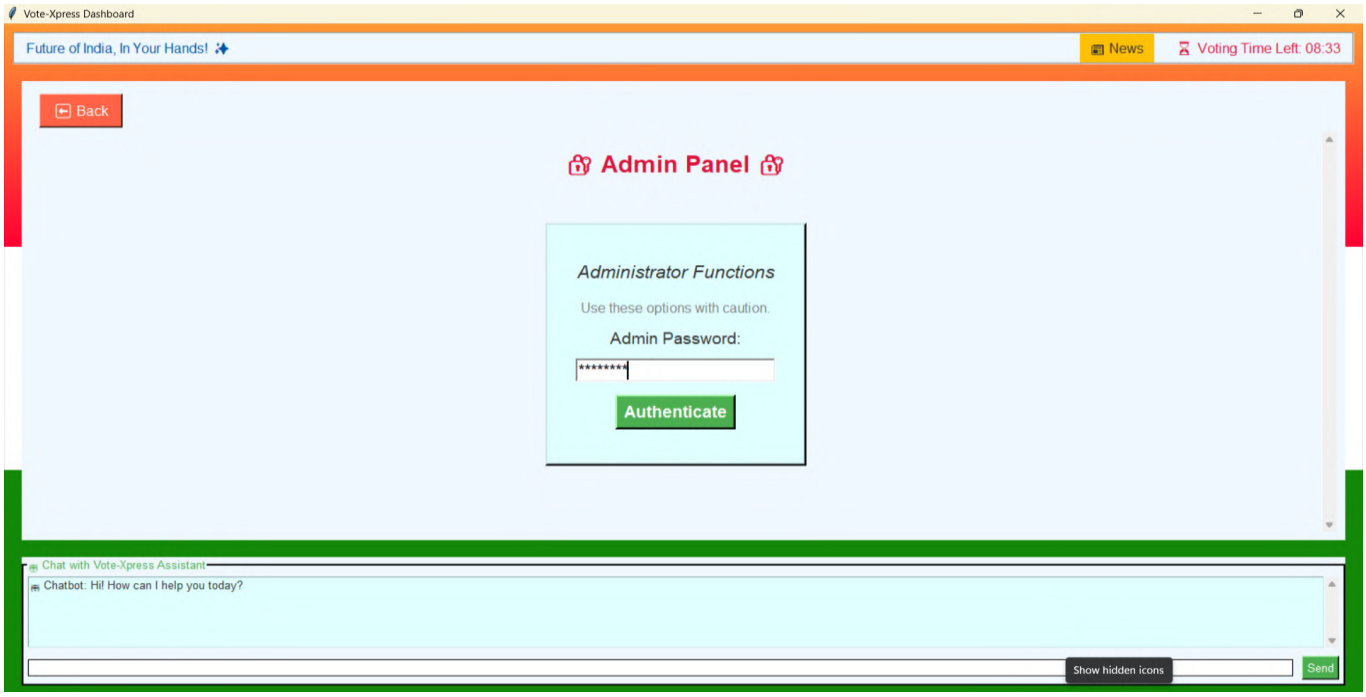
7.) Vote Quiz Game Page:-



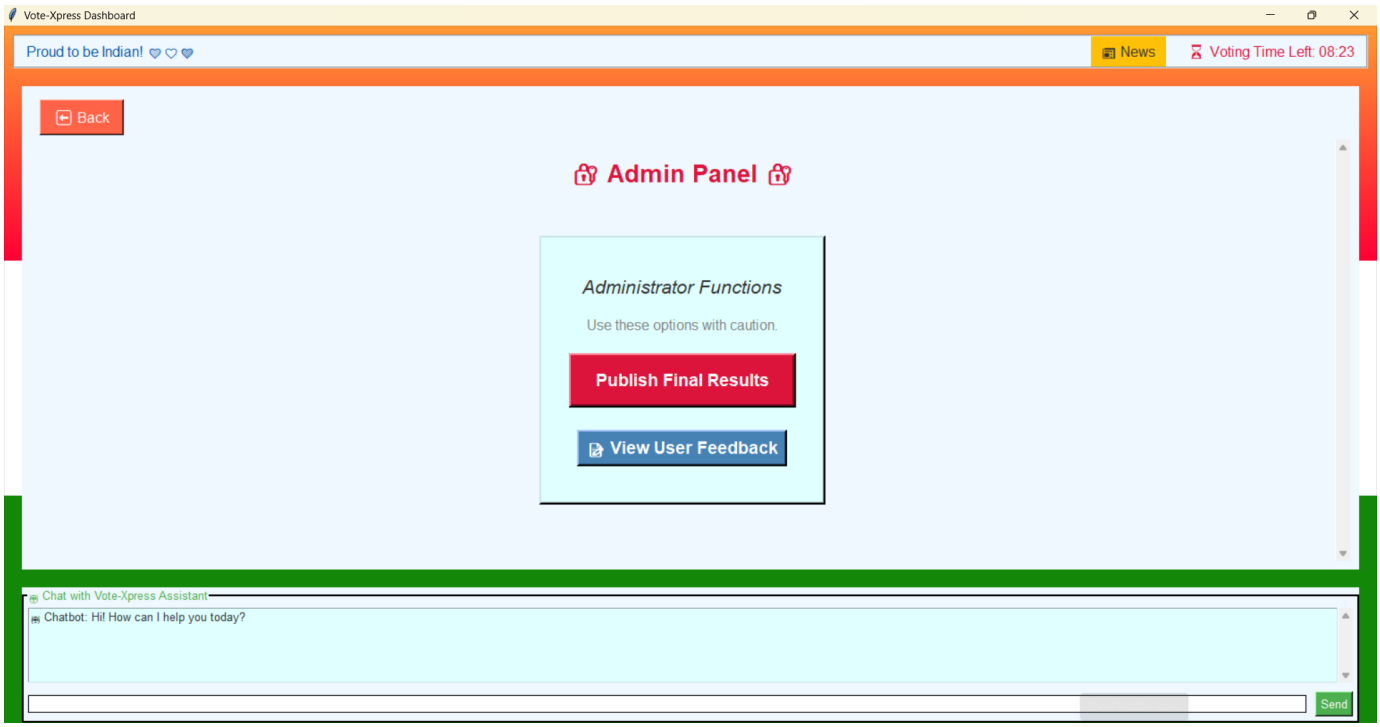
8.) Help and Support Page:-



9.) Admin Panel(Login Page):-



10.) Admin Panel(with Different features):-



11.) Candidates Details Page:-

12.) Developer Information Page:-

Chapter 8: Testing and Validation

8.1 Testing Methods:-

- Unit Testing: Each function tested separately (e.g., vote submission, quiz timer).
- Integration Testing: Checked smooth working of modules together.
- GUI Testing: Verified buttons, inputs, and navigation.
- Performance Testing: Ensured chatbot runs without freezing the system.

8.2 Sample Test Cases:-

Test Case	Input	Expected Output	Result
Voter Registration	Name + Voter ID	Registration receipt	Pass
Duplicate Vote	Same Voter ID	Error: Already Voted	Pass
Quiz Attempt	Answer 10 questions	Score displayed	Pass
Admin Login	Password = admin123	Access granted	Pass
Admin Login	Wrong Password	Access denied	Pass
Chatbot Query	"What is NOTA?"	Correct answer returned	Pass

8.3 Validation:-

The system was validated by simulating multiple users. It correctly prevented duplicate votes, displayed results, and provided interactive learning tools.

Chapter 9: Results and Discussion

9.1 Outputs Achieved:-

- Secure Voting: Only one vote per person.
- Real-Time Results: Live bar chart updates.
- Quiz Success: Scoring and badge system working.
- Feedback: Stored successfully.
- Chatbot: Responded with correct answers using Gemini API.
- Admin Panel: Allowed secure result declaration.

9.2 Challenges Faced:-

- Handling chatbot API response delays.
- Preventing Tkinter window from freezing.
- Managing vote storage without a database.
- Designing smooth and attractive UI.

9.3 Discussion:-

Overall, the system achieved all its goals. It can be deployed in colleges and organizations as a reliable alternative to manual voting.

Chapter 10: Challenges and Learnings

10.1 Challenges:-

Technical:-

- API request handling in Python.
- Real-time vote updates without using a database.
- GUI responsiveness during heavy tasks.

Non-Technical:-

- Time management during the internship.
- Coordinating teamwork and task division.

10.2 Learnings:-

- Practical knowledge of Tkinter and event-driven programming.
- API integration and error handling.
- Importance of modular coding.
- Improved problem-solving and debugging skills.
- Teamwork and presentation preparation.

Chapter 11: Future Scope

The project can be further enhanced with:

1. **Database Integration** – Store votes and users in MySQL/Firebase.
2. **Cloud Deployment** – Make the system accessible online.
3. **Blockchain Voting** – Provide higher-level security and transparency.
4. **Multi-language Support** – Support Indian regional languages.
5. **Mobile App Version** – Develop Android/iOS app using Kivy or Flutter.
6. **Advanced Analytics** – Charts showing voter turnout, demographics, etc.

Chapter 12: Conclusion

The project Vote-Xpress: An Interactive Online Voting System has been successfully designed and implemented as part of our summer training. The system fulfills its aim of providing a secure, efficient, and interactive voting platform for academic institutions and organizations.

Through this project, we solved the common problems of traditional voting systems such as manual counting errors, lack of transparency, and limited accessibility. The application was able to combine multiple modules into one integrated system, making the voting process not only smooth but also engaging and educational.

Key Features Achieved in the Project

1. Secure Voting Process
    - Each voter can register using their name and voter ID.
    - The system ensures “one person, one vote” by preventing duplicate entries.
    - A vote receipt is generated to confirm successful voting.
2. Live Result Display
    - Votes are counted in real time.
    - Users and admins can view results in a colorful bar chart.
    - The admin panel allows secure declaration of final results.
    - Celebration animations such as fireworks and confetti make the process more attractive.
3. Quiz for Civic Awareness

- A 10-question quiz was included to promote knowledge of democracy.
- Users receive scores, badges, and stars based on performance.
- This adds an educational value to the application beyond voting.

4. Feedback Module
    - Users can share their opinions and ratings after voting.
    - This helps in improving future versions of the application.
5. AI-Powered Chatbot
    - The chatbot integrated with Gemini API provides instant help.
    - Users can ask questions related to voting, candidates, or general information.
    - The chatbot runs smoothly with multithreading to avoid UI freezing.
6. Civic Insights, Slogans, and Thought of the Day
    - The application displays rotating civic facts (e.g., “India is the world’s largest democracy”).
    - Patriotic slogans keep the users motivated (e.g., “Your Vote, Your Voice”).
    - A daily thought inspires users and makes the platform more interactive.
7. User-Friendly Interface
    - A clean Tkinter-based GUI was created with Indian-themed colors.

    - The system is simple to use even for people with limited technical knowledge.
    - The splash screen and dashboard provide a modern and attractive entry point.
8. Admin Security
    - A password-protected admin panel ensures that only authorized users can declare results.
    - This adds a layer of trust and authenticity to the system.

Learnings and Benefits

- The project gave us practical exposure to Python, Tkinter, multithreading, and API integration.
- We learned how to design a system_that is both functional and user-friendly.
- The experience also improved our skills in problem-solving, debugging, and teamwork.

Real-World Relevance

This project is not just an academic exercise; it has real-world applications:

- Colleges and Universities can use it for student elections.
- Clubs, societies, and organizations can adopt it for leadership selection.
- With further improvements (like database and blockchain integration), it can even be scaled for larger elections.

Final Thoughts

In conclusion, Vote-Xpress is more than just a voting system – it is an interactive civic platform. It ensures fair voting, educates users

about democracy, and integrates modern AI tools to make the process engaging.

This project has shown us how technology can make democratic processes simpler, faster, and more transparent. With its combination of security, interactivity, and education, Vote-Xpress truly represents the spirit of Digital India.

Chapter 13: References:-

1. Python Software Foundation. *Python 3 Documentation*. https://docs.python.org/
2. TkDocs. *Tkinter Reference*. https://tkdocs.com/
3. Google. *Gemini API Overview*. https://ai.google/
4. Election Commission of India. *Handbook & FAQs*. https://eci.gov.in/
5. Chaum, D., et al. (2004). *Secret-Ballot Receipts: True Voter-Verifiable Elections*. IEEE Security & Privacy.

Chapter 14: Appendix

1.) Appendix A:-

A. Quiz Questions (Sample)

- What is the minimum age to vote in India? (Answer: 18 years)
- Which body conducts Lok Sabha elections? (Answer: Election Commission of India)
- What does NOTA stand for? (Answer: None of the Above)

B. Civic Insights

- India is the world's largest democracy.
- Voter turnout in Indian elections often exceeds 60%.

C. Slogans

- "Mera Bharat Mahan IN"
- "Your Vote, Your Voice!"

D. Sample Screens (described)

- Splash Screen: Digital India theme with logo.
- Dashboard: Central menu with side panels.
- Registration Form: Fields for name and voter ID.
- Voting Panel: Candidate selection with radio buttons.
- Results Page: Bar chart showing votes.
- Admin Panel: Final results with fireworks animation.

2.)Appendix B :- Complete Source Code of Vote-Xpress:-

```
import tkinter as tk
from tkinter import messagebox
import threading
import json
import math
import datetime
import requests
import random

# ----------------------------
# Application Constants
# ----------------------------
APP_NAME = "Vote-Xpress"
APP_VERSION = "1.0"

# Color Palette
PRIMARY_COLOR = "#0d6efd"
SECONDARY_COLOR = "#198754"
DANGER_COLOR = "#dc3545"
WARNING_COLOR = "#ffc107"
LIGHT_COLOR = "#f8f9fa"
DARK_COLOR = "#212529"

# Civic Slogans and Thoughts
SLOGANS = [
    "Your Vote, Your Voice!",
    "Mera Bharat Mahan ɪɴ",
    "Digital India – Strong India",
    "Every Vote Counts!"
]
```

```
THOUGHTS = [
    "Democracy is not just voting, it is participation.",
    "Be the change you want to see in the world.",
    "Voting is the expression of our commitment to society."
]


# Quiz Questions
QUIZ_QUESTIONS = [
    {"question": "What is the minimum age to vote in India?",
     "options": ["16 years", "18 years", "21 years"], "answer": "18 years"},
    {"question": "Who conducts Lok Sabha elections?",
     "options": ["Supreme Court", "Parliament", "Election Commission"], "answer": "Election Commission"},
    {"question": "When was the first general election held in India?",
     "options": ["1947", "1951", "1962"], "answer": "1951"},
    {"question": "What does NOTA stand for?",
     "options": ["None Of The Above", "National Organization for Transparency Act", "National Overseas Training Agency"], "answer": "None Of The Above"},
    {"question": "Which Article of the Constitution gives voting rights in India?",
     "options": ["Article 326", "Article 21", "Article 356"], "answer": "Article 326"},
]


# ----------------------------
# Voting Data Structures
# ----------------------------
votes = {}
voted_users = set()


# ----------------------------
# Splash Screen Class
# ----------------------------
class SplashScreen:
```

```
    def __init__(self, root, callback):
        self.root = root
        self.callback = callback
        self.root.overrideredirect(True)
        self.root.geometry("500x300+500+200")
        self.root.configure(bg=PRIMARY_COLOR)

        label = tk.Label(root, text="Welcome to Vote-Xpress",
                         font=("Arial", 20, "bold"), bg=PRIMARY_COLOR, fg="white")
        label.pack(expand=True)

        slogan = tk.Label(root, text=random.choice(SLOGANS),
                          font=("Arial", 12), bg=PRIMARY_COLOR, fg="white")
        slogan.pack(side="bottom", pady=20)

        # Auto close after 3 seconds
        self.root.after(3000, self.close)

    def close(self):
        self.root.destroy()
        self.callback()

# ----------------------------
# Main Application Class
# ----------------------------
class VoteApp:
    def __init__(self, root):
        self.root = root
        self.root.title(APP_NAME)
        self.root.geometry("800x600")
```

```
    # Notebook (tabs)
    self.tabs = tk.Frame(root, bg=LIGHT_COLOR)
    self.tabs.pack(fill="both", expand=True)

    self.create_dashboard()

def create_dashboard(self):
    lbl = tk.Label(self.tabs, text="Vote-Xpress Dashboard",
                font=("Arial", 22, "bold"), fg=PRIMARY_COLOR)
    lbl.pack(pady=20)

    # Buttons for modules
    tk.Button(self.tabs, text="Register & Vote", command=self.registration_module,
            width=20, height=2, bg=PRIMARY_COLOR, fg="white").pack(pady=10)
    tk.Button(self.tabs, text="View Results", command=self.results_module,
            width=20, height=2, bg=SECONDARY_COLOR, fg="white").pack(pady=10)
    tk.Button(self.tabs, text="Quiz", command=self.quiz_module,
            width=20, height=2, bg=WARNING_COLOR).pack(pady=10)
    tk.Button(self.tabs, text="Feedback", command=self.feedback_module,
            width=20, height=2, bg=DANGER_COLOR, fg="white").pack(pady=10)
    tk.Button(self.tabs, text="Chatbot", command=self.chatbot_module,
            width=20, height=2, bg=DARK_COLOR, fg="white").pack(pady=10)

# ----------------------------
# Registration Module
# ----------------------------
def registration_module(self):
    reg_win = tk.Toplevel(self.root)
    reg_win.title("Voter Registration")

    tk.Label(reg_win, text="Full Name:").pack()
```

```
    name_entry = tk.Entry(reg_win)
    name_entry.pack()

    tk.Label(reg_win, text="Voter ID:").pack()
    id_entry = tk.Entry(reg_win)
    id_entry.pack()

    tk.Label(reg_win, text="Choose Candidate:").pack()
    candidate_var = tk.StringVar()
    for c in ["Candidate A", "Candidate B", "Candidate C"]:
        tk.Radiobutton(reg_win, text=c, variable=candidate_var, value=c).pack()

    def submit_vote():
        name = name_entry.get()
        voter_id = id_entry.get()
        candidate = candidate_var.get()
        if not name or not voter_id or not candidate:
            messagebox.showerror("Error", "All fields are required")
            return
        if voter_id in voted_users:
            messagebox.showerror("Error", "You have already voted!")
            return
        votes[candidate] = votes.get(candidate, 0) + 1
        voted_users.add(voter_id)
        messagebox.showinfo("Success", f"Vote cast for {candidate}")

    tk.Button(reg_win, text="Submit Vote", command=submit_vote, bg=PRIMARY_COLOR,
fg="white").pack(pady=10)

# ----------------------------
# Results Module
```

```
    # ----------------------------
    def results_module(self):
        res_win = tk.Toplevel(self.root)
        res_win.title("Voting Results")

        if not votes:
            tk.Label(res_win, text="No votes yet!").pack()
            return

        total_votes = sum(votes.values())
        tk.Label(res_win, text=f"Total Votes: {total_votes}", font=("Arial", 14,
"bold")).pack(pady=10)

        for candidate, count in votes.items():
            percentage = (count / total_votes) * 100
            tk.Label(res_win, text=f"{candidate}: {count} votes ({percentage:.2f}%)").pack()

    # ----------------------------
    # Quiz Module
    # ----------------------------
    def quiz_module(self):
        quiz_win = tk.Toplevel(self.root)
        quiz_win.title("Civic Awareness Quiz")

        score = {"value": 0}
        q_index = {"value": 0}

        def show_question():
            for widget in quiz_win.winfo_children():
                widget.destroy()
            if q_index["value"] >= len(QUIZ_QUESTIONS):
```

```
            tk.Label(quiz_win, text=f"Quiz Finished! Your Score:
{score['value']}/{len(QUIZ_QUESTIONS)}",
                     font=("Arial", 14, "bold")).pack(pady=20)
            return
        q = QUIZ_QUESTIONS[q_index["value"]]
        tk.Label(quiz_win, text=q["question"], font=("Arial", 12, "bold")).pack(pady=10)
        for option in q["options"]:
            tk.Button(quiz_win, text=option, command=lambda opt=option:
check_answer(opt)).pack(pady=5)

    def check_answer(selected):
        q = QUIZ_QUESTIONS[q_index["value"]]
        if selected == q["answer"]:
            score["value"] += 1
        q_index["value"] += 1
        show_question()

    show_question()

# ----------------------------
# Feedback Module
# ----------------------------
def feedback_module(self):
    fb_win = tk.Toplevel(self.root)
    fb_win.title("Feedback")

    tk.Label(fb_win, text="Your Feedback:").pack()
    fb_text = tk.Text(fb_win, height=5, width=40)
    fb_text.pack()

    def submit_feedback():
        feedback = fb_text.get("1.0", "end").strip()
```

```
        if feedback:
            messagebox.showinfo("Thank You", "Feedback submitted successfully!")
        else:
            messagebox.showerror("Error", "Please enter some feedback")

    tk.Button(fb_win, text="Submit", command=submit_feedback, bg=PRIMARY_COLOR,
fg="white").pack(pady=10)

  # ----------------------------
  # Chatbot Module
  # ----------------------------
  def chatbot_module(self):
    chat_win = tk.Toplevel(self.root)
    chat_win.title("Vote-Xpress Chatbot")

    chat_display = tk.Text(chat_win, height=15, width=50, state="disabled")
    chat_display.pack()

    entry = tk.Entry(chat_win, width=40)
    entry.pack(side="left", padx=5, pady=5)

    def send_message():
      msg = entry.get()
      if not msg.strip():
        return
      chat_display.config(state="normal")
      chat_display.insert("end", f"You: {msg}\n")
      chat_display.config(state="disabled")
      entry.delete(0, "end")

      # Run chatbot in separate thread
```

```
        threading.Thread(target=lambda: get_response(msg)).start()


    def get_response(user_msg):
        try:
            # Dummy response (replace with Gemini API call)
            answer = f"Bot: You asked '{user_msg}', here is a helpful answer."
        except Exception:
            answer = "Bot: Sorry, I am unable to respond now."
        chat_display.config(state="normal")
        chat_display.insert("end", answer + "\n")
        chat_display.config(state="disabled")


    tk.Button(chat_win, text="Send", command=send_message, bg=SECONDARY_COLOR,
fg="white").pack(side="right", padx=5)


# ------------------------------
# Run Application
# ------------------------------
def launch_app():
    root = tk.Tk()
    app = VoteApp(root)
    root.mainloop()


if __name__ == "__main__":
    splash = tk.Tk()
    SplashScreen(splash, launch_app)
    splash.mainloop()
```