AI-POWERED NOTE SUMMARIZATION WEB APPLICATION USING FULLSTACK

A SUMMER INTERNSHIP REPORT

Submitted by

SANJAY M (711723243086)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

KGiSL INSTITUTE OF TECHNOLOGY

JULY 25

KGiSL INSTITUTE OF TECHNOLOGY

BONAFIDE CERTIFICATE

Certified that this Internship report on "AI-POWERED NOTE SUMMARIZATION WEB APPLICATION USING FULLSTACK" at Appin Technology is the bonafide work of **SANJAY M** who belongs to III Year Artificial Intelligence & Data Science “B” during Fifth Semester of Academic Year 2025-2026.

FACULTY INCHARGE **HEAD OF THE DEPARTMENT**

Certified that the candidates were examined by us for AD3512 Summer Internship Viva held on _____at KGiSL Institute of Technology, Saravanampatti, Coimbatore 641035.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deepest gratitude to our **Chairman and Managing Director Dr. Ashok Bakthavachalam** for providing us with an environment to complete our internship project successfully.

We are grateful to our **CEO of Academic Initiatives Mr. Aravind Kumar Rajendran** and our beloved **Secretary Dr. Rajkumar N.** Our sincere thanks to honorable **Principal Dr. Suresh Kumar S** for his support, guidance, and blessings.

We would like to thank **Dr. T. Kumaresan, Head of the Department**, and **Internship Coordinator Ms.H. Sowmiya**, Department of Artificial Intelligence and Data Science for her firm support during the entire course of this Internship and who modelled us both technically and morally for achieving greater success in this project work.

We also thank all the faculty members of our department for their help in making this Internship project a successful one. Finally, we take this opportunity to extend our deep appreciation to our Family and Friends, for all they meant to us during the crucial times of the completion of our project.

ETHER SERVICES

Date: 05.06.2025

ACCEPTANCE LETTER

This is to inform that **Mr SANJAY M [REG NO: 711723243086],** 2nd Year, **B.Tech., ARTIFICIAL INTELLIGENCE AND DATA SCIENCE** student of **KGISL INSTITUTE OF TECHNOLOGY, COIMBATORE** have been approved to do Internship Training in our Organization from 07th June 2025 to 07th July 2025. in the field of **"FULL STACK PYTHON WITH AI".**

We are pleased to confirm that his Internship Training request has been accepted and he is allowed to do Internship Training in our organization.

Thanking You.

For Ether Services

Authorised Signatory

144, 2nd Floor, Sengupta Street, Ram Nagar,
Coimbatore - 641009.
M : +91 99446 44905
www.etherservices.com | E : mohan@etherservices.com

CERTIFIED ISO 9001:2015 COMPANY

NPTEL #startupindia

ETHER SERVICES

Date: 07.07.2025

INTERNSHIP COMPLETION CERTIFICATE

This is to certify that **Mr SANJAY M [REG NO: 711723243086],** 2nd Year, **B.Tech., ARTIFICIAL INTELLIGENCE AND DATA SCIENCE** student of **KGISL INSTITUTE OF TECHNOLOGY, COIMBATORE** has attended Internship Training in our organization from 07th June 2025 to 07th July 2025.

During this Internship Training, he has gained practical knowledge in **FULL STACK PYTHON WITH AI.** He has completed this program successfully and his performance was Excellent.

Thanking You

For Ether Services

Authorised Signatory

144, 2nd Floor, Sengupta Street, Ram Nagar,
Coimbatore - 641009.
M : +91 99446 44905
www.etherservices.com | E : mohan@etherservices.com

CERTIFIED ISO 9001:2015 COMPANY

NPTEL

#startupindia

ABSTRACT

During my summer internship at Appin Technology, I had the opportunity to develop a full-stack AI-powered web application titled “AI-Powered Note Summarization Web Application”. This project was aimed at leveraging artificial intelligence and modern web technologies to help users condense lengthy text content into meaningful summaries in various formats. The primary objective of the application was to provide a user-friendly platform where users could input raw text and receive concise summaries in multiple forms such as paragraphs, bullet points, structured notes, outlines, or essays. The application backend was developed using Python with Flask, while the frontend was designed using HTML and CSS to ensure a responsive and visually appealing user interface.

The core AI functionality was powered by a transformer-based natural language processing model (facebook/bart-large-cnn) integrated using the HuggingFace Transformers library. This allowed the system to generate intelligent, context-aware summaries. Additional features included user authentication, text-length customization (short, medium, long), and the ability to download the summary as a formatted PDF file. The project also involved the use of SQLite for user management and session tracking. The application architecture followed a modular and scalable approach, incorporating clean design, reusable components, and persistent user inputs post-summary generation.

To manage multiple summary styles, the application utilized an Enum class in Python that defined and handled five distinct summary types: Paragraph, Bullet Points, Structured Notes, Outline, and Essay-style. This approach enhanced maintainability and allowed easy expansion for future summary formats. The inclusion of enums ensured clean backend logic and better readability across the summarization module, making the system more organized and extensible.

This internship enabled me to gain valuable experience in real-world full-stack development, artificial intelligence integration, and user-centric application design. It also improved my problem-solving abilities and provided insight into the workflow of building production-ready web applications. Looking ahead, the application can be further enhanced with features such as multilingual support, speech-to-text integration, history saving, and advanced grammar correction tools, making it a versatile solution for students, researchers, and professionals.

INTRODUCTION

In today's information-driven world, the ability to extract meaningful summaries from large volumes of textual data is essential. Whether for academic research, business communication, or personal productivity, summarization tools can significantly reduce time spent on reading while preserving essential content. This project, developed as part of my summer internship at Appin Technology, introduces a web-based platform titled "AI-Powered Note Summarization Web Application Using Fullstack" to address this need using modern web technologies and artificial intelligence.

Domain Focus

- The project falls under the domain of Artificial Intelligence and Web Development, with a specific emphasis on Natural Language Processing (NLP).
- It focuses on solving the real-world problem of information overload by providing a digital tool that generates concise and meaningful summaries in various formats such as paragraphs, bullet points, structured notes, outlines, and essays.
- The system is designed to cater to students, educators, professionals, and researchers who deal with extensive textual content regularly.

Global Relevance and Impact

- As digital learning, research, and remote work continue to rise globally, the need for intelligent content summarization tools becomes increasingly relevant.
- The platform provides a multiform summary generation service, enhancing user accessibility to compressed information regardless of geographical or domain boundaries.
- This AI-enabled application serves as a productivity enhancer and is scalable for broader use across academic institutions and enterprises.

Technology Stack

The application is built using a combination of full-stack technologies and AI libraries:

- **Backend**: Python with Flask framework
- **Frontend**: HTML5 and CSS3 for responsive UI
- **AI Engine**: facebook/bart-large-cnn transformer model via HuggingFace library
- **Database**: SQLite (for user management and session tracking)
- **PDF Export**: ReportLab for generating downloadable formatted summaries
- **Authentication**: Secure user login and registration system using SQLAlchemy ORM

Programming Languages and Tools

- Python is used extensively for backend logic, AI model integration, and summary generation.

- HTML/CSS are used to build the frontend interface, ensuring a clean and intuitive user experience.
- HuggingFace Transformers, ReportLab, Flask, and SQLAlchemy are key libraries and frameworks that form the core functionality of the application.

Advanced Features

- **Multiple Summary Formats**: Users can choose from Paragraph, Bullet Points, Structured Notes, Outline, and Essay formats.
- **Text Length Options**: The tool supports short, medium, and long summaries based on user preference.
- **PDF Export**: Summaries can be downloaded in PDF format with preserved formatting using embedded fonts.
- **Text Retention**: Inputs and summary preferences persist after generation, improving usability.
- **User Login System**: Secure login and session handling allow for personalized and secure usage.

Development Methodology

1. Data Handling and AI Integration:

- The system uses HuggingFace's summarization pipeline to process user input and return generated summaries with contextual relevance.

2. UX and Responsive Design:

- The interface is designed for ease of use, mobile compatibility, and fast interaction to enhance user retention and accessibility.

3. Performance Optimization:

- The application has been optimized for fast response times, efficient text processing, and lightweight frontend delivery.

Scope for Future Enhancements

While the current version serves as a robust MVP (Minimum Viable Product), several enhancements are planned:

- Speech-to-text input and text-to-speech output
- Multilingual summarization support
- User summary history and cloud storage
- AI-driven grammar correction and keyword extraction
- Mobile-responsive design using frontend frameworks like React or Bootstrap
- Role-based login (admin, user) and analytics dashboard

SYSTEM SPECIFICATIONS

1.Hardware Requirements

For Development:

- **Processor:** Dual-core or multi-core processor (Intel i5 or higher recommended)
- **RAM:** Minimum 8 GB (12–16 GB recommended for smooth AI model execution)
- **Storage:**
  - At least 100 GB of free disk space
  - SSD preferred for faster application runtime and model loading
- **GPU:** Not mandatory (CPU inference is sufficient for BART-based summarization)
- **Internet:** Stable connection required for downloading dependencies (HuggingFace models, Python libraries)

For Deployment / Hosting:

- **Processor:** Multi-core (e.g., Intel Xeon or cloud instance equivalent)
- **RAM:** 8 GB minimum (12+ GB preferred for concurrent user support)
- **Storage:** SSD with at least 100–200 GB for model files, logs, and user data
- **Network:** Reliable 50–100 Mbps bandwidth for moderate traffic handling

2. Software Requirements

Development Environment:

- **Operating System:**
  - Windows 10/11
  - Linux (Ubuntu 20.04 LTS preferred)
  - macOS (latest version)
- **Programming Language:** Python 3.8 or higher
- **Frameworks & Libraries:**
  - Flask (for backend web framework)
  - HuggingFace Transformers (for BART model)
  - SQLAlchemy + SQLite (for database and ORM)
  - ReportLab (for PDF export functionality)
- **Code Editor/IDE:**
  - Visual Studio Code (recommended)
- **Browser (for testing):**
  - Chrome / Firefox / Edge

Production Environment:

- **Operating System:**
  - Ubuntu Server 20.04+ (preferred for deploying Flask apps)
- **Web Server:**
  - Gunicorn + Nginx (recommended combo for Flask production)
- **Database:**
  - SQLite (lightweight and embedded; no separate DB server needed)
- **Other Tools:**
  - Python Virtual Environment (venv)
  - Supervisor (for process management, optional)

MODULE DESCRIPTIONS

1. User Registration and Authentication Module

- **Purpose:**

  Enables secure login and registration functionality for users.
- **Key Features:**
  - User sign-up and login with form validation
  - Session-based authentication using Flask's session management
  - Password encryption using werkzeug.security for enhanced security
- **Technologies Used:**

  Flask, SQLite, HTML/CSS, Werkzeug, Flask-Login

2. Text Input and Preprocessing Module

- **Purpose:**

  Allows users to paste or upload raw text for summarization.
- **Key Features:**
  - Paste text or upload .txt file
  - Basic input sanitization (removal of special characters, trimming spaces)
  - Support for multi-format inputs (paragraphs, notes, long documents)
- **Technologies Used:**

  Flask (Python backend), HTML forms, FileReader

3. Summarization Engine Module

- **Purpose:**

  Processes input text and generates concise summaries using AI.
- **Key Features:**
  - Supports 5 summary types: Paragraph, Bullet Points, Structured Notes, Outline, Essay
  - Multiple length options: Short, Medium, Long
  - Model used: facebook/bart-large-cnn from HuggingFace Transformers
- **Technologies Used:**

  Python, HuggingFace Transformers, Flask

4. Summary Display & Selection Module

- **Purpose:**

  Displays AI-generated summaries to users in a readable format.
- **Key Features:**
  - Summary appears on a dedicated result page
  - Retains user-selected format and length during display
  - Option to re-summarize using different settings
- **Technologies Used:**

  Flask (render templates), Jinja2, HTML/CSS

5. Export to PDF Module

- **Purpose:**

  Allows users to export their summaries as clean, downloadable PDF files.
- **Key Features:**
  - Summary exported in user-friendly fonts (Arial/Helvetica)
  - Dynamic formatting based on selected summary type
  - No dependency on browser plugins or third-party PDF tools
- **Technologies Used:**

  ReportLab (Python PDF generation), Flask

6. User Dashboard Module

- **Purpose:**

  Displays user-specific summaries and options in a centralized area.
- **Key Features:**
  - History of previously generated summaries (optional future feature)
  - Quick access to new summarization
  - Logout and session management
- **Technologies Used:**

  Flask templates, HTML/CSS

7. Contact & Feedback Module

- **Purpose:**

  Enables users to reach out for support, suggestions, or inquiries.
- **Key Features:**
  - Static contact form
  - Email or local storage of feedback (optional backend integration)
  - Clean UI and user validation

- **Technologies Used:**

  Flask, HTML/CSS

8. About and Privacy Pages Module

- **Purpose:**

  Informs users about the application, its features, and privacy policies.
- **Key Features:**
  - "About" section with team/project details
  - "Privacy Policy" outlining data use
  - Scroll-friendly layout with modern design
- **Technologies Used:**

  HTML/CSS, Jinja2 (Flask templates)

9. Notification and Status Indicator Module

- **Purpose:**

  Provides real-time feedback on user actions (e.g., summary completed).
- **Key Features:**
  - Success and error messages for uploads and summaries
  - Loading spinners for long processing tasks
  - Alert messages styled for better UX
- **Technologies Used:**

  JavaScript, Flask flash messages, CSS animations

EXPERIMENTAL RESULTS

1.CREATE THE BACKEND

//app.py

```
from flask import Flask, render_template, request, redirect, url_for, flash, send_file
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, login_user, login_required, logout_user, current_user
from werkzeug.security import generate_password_hash, check_password_hash
from summarizer import summarize_text
from fpdf import FPDF
from datetime import datetime
from models import db, User, Summary
import docx
import re
import os

app = Flask(__name__)
app.secret_key = "supersecretkey"  # Replace this in production!
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db.init_app(app)

login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = "login"

@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))
def remove_unicode(text):
    return re.sub(r'[^\x00-\x7F]+', '', text)
```

```
def extract_text_from_file(file):
    if file.filename.endswith(".docx"):
        doc = docx.Document(file)
        return "\n".join([para.text for para in doc.paragraphs])
    elif file.filename.endswith(".txt"):
        return file.read().decode("utf-8")
    return ""

@app.route("/")
def home():
    return render_template("index.html")

@app.route("/register", methods=["GET", "POST"])
def register():
    if current_user.is_authenticated:
        return redirect(url_for("summarize"))

    if request.method == "POST":
        username = request.form["username"]
        email = request.form["email"]
        password = request.form["password"]
        hashed_password = generate_password_hash(password)

        if User.query.filter((User.username == username) | (User.email == email)).first():
            flash("Username or Email already exists!", "warning")
            return redirect(url_for("register"))

        user = User(username=username, email=email, password=hashed_password)
        db.session.add(user)
        db.session.commit()
        flash("Account created successfully. Please log in.", "success")
        return redirect(url_for("login"))

    return render_template("register.html")

@app.route("/login", methods=["GET", "POST"])
def login():
```

```
    if current_user.is_authenticated:
        return redirect(url_for("summarize"))

    if request.method == "POST":
        email = request.form["email"]
        password = request.form["password"]
        user = User.query.filter_by(email=email).first()

        if user and check_password_hash(user.password, password):
            login_user(user)
            flash("Login successful!", "success")
            return redirect(url_for("summarize"))
        else:
            flash("Invalid email or password.", "danger")

    return render_template("login.html")

@app.route("/logout")
@login_required
def logout():
    logout_user()
    flash("You have been logged out.", "info")
    return redirect(url_for("home"))

@app.route("/summarize", methods=["GET", "POST"])
@login_required
def summarize():
    summary = ""
    input_text = ""
    selected_summary_type = "Paragraph"
    selected_length = "Medium"

    if request.method == "POST":
        input_text = request.form.get("noteText", "")
        file = request.files.get("file")
        selected_summary_type = request.form.get("summary_type", "Paragraph")
        selected_length = request.form.get("length", "Medium")
```

```
    if file and file.filename:
        input_text = extract_text_from_file(file)

    length_map = {
        "Short": (100, 200),
        "Medium": (200, 350),
        "Long": (350, 600)
    }
    min_len, max_len = length_map.get(selected_length, (100, 250))

    if input_text.strip():
        summary = summarize_text(
            text=input_text,
            summary_type=selected_summary_type.lower().replace(" ", "_"),
            min_length=min_len,
            max_length=max_len
        )

        new_summary = Summary(
            content=input_text,
            summary=summary,
            summary_type=selected_summary_type,
            summary_length=selected_length,
            user_id=current_user.id
        )
        db.session.add(new_summary)
        db.session.commit()

return render_template(
    "summarize.html",
    summary=summary,
    input_text=input_text,
    selected_summary_type=selected_summary_type,
    selected_length=selected_length,
    selected_language="English"
)
```

```
@app.route("/download_pdf", methods=["POST"])
@login_required
def download_pdf():
    summary_text = request.form.get("summary_text", "No summary available.")
    clean_text = remove_unicode(summary_text)

    pdf = FPDF()
    pdf.add_page()
    pdf.set_font("Helvetica", size=12)
    for line in clean_text.splitlines():
        pdf.multi_cell(0, 10, line.strip())

    output_path = "summary_temp.pdf"
    pdf.output(output_path)
    return send_file(output_path, as_attachment=True, download_name="summary.pdf")

@app.route("/about")
def about():
    return render_template("about.html")

@app.route("/contact")
def contact():
    return render_template("contact.html")

@app.route("/privacy")
def privacy():
    return render_template("privacy.html")

if __name__ == "__main__":
    with app.app_context():
        db.create_all()
    app.run(debug=True)
```

//summarize.py

```
from transformers import pipeline
```

```
from enum import Enum
import re

summarizer = pipeline("summarization", model="facebook/bart-large-cnn")
def split_into_chunks(text, max_words=800):
    words = text.split()
    return [' '.join(words[i:i + max_words]) for i in range(0, len(words), max_words)]
def format_paragraph(text):
    return '<b> Paragraph:</b><br><p>' + text.strip() + '</p>'

def format_bullets(text):
    sentences = re.split(r'\.\s+', text.strip())
    sentences = [s.strip().rstrip('.') + '.' for s in sentences if s.strip()]
    return '<b> Bullet Points:</b><br>' + '<br>'.join(f'• {s}' for s in sentences)
def format_structured(text):
    sentences = re.split(r'\.\s+', text.strip())
    sentences = [s.strip().rstrip('.') + '.' for s in sentences if s.strip()]
    return '<b> Structured Notes:</b><br>' + '<br>'.join(f'• {s}' for s in sentences)

# Format: Outline (Roman numerals)
def format_outline(text):
    roman_numerals = ['I.', 'II.', 'III.', 'IV.', 'V.', 'VI.', 'VII.', 'VIII.', 'IX.', 'X.']
    sentences = re.split(r'\.\s+', text.strip())
    sentences = [s.strip().rstrip('.') + '.' for s in sentences if s.strip()]
    return '<b>📋 Outline:</b><br>' + '<br>'.join(f"{roman_numerals[i % len(roman_numerals)]} {s}" for
i, s in enumerate(sentences))

def format_essay(text):
    sentences = re.split(r'\.\s+', text.strip())
    paragraphs = ['. '.join(sentences[i:i + 3]).strip().rstrip('.') + '.' for i in range(0, len(sentences), 3)]
    return '<b>📝 Essay:</b>' + ''.join(f"<p>{p}</p>" for p in paragraphs if p)

class SummaryType(Enum):
    PARAGRAPH = ("Paragraph", format_paragraph)
    BULLET_POINTS = ("Bullet Points", format_bullets)
    STRUCTURED_NOTES = ("Structured Notes", format_structured)
    OUTLINE = ("Outline", format_outline)
```

```
    ESSAY_WRITING = ("Essay Writing", format_essay)

    def __init__(self, label, formatter):
        self.label = label
        self.formatter = formatter

    def format(self, text):
        return self.formatter(text)

    @classmethod
    def choices(cls):
        return [(member.name, member.label) for member in cls]

def summarize_text(text, min_length=30, max_length=100, summary_type="Paragraph"):
    # Limit to 10,000 words
    words = text.split()
    if len(words) > 100000:
        text = ' '.join(words[:10000])

    # Chunk the input
    chunks = split_into_chunks(text)
    summaries = []

    for chunk in chunks:
        try:
            result = summarizer(chunk, min_length=int(min_length), max_length=int(max_length),
do_sample=False)
            summaries.append(result[0]['summary_text'])
        except Exception as e:
            summaries.append(f"[Error summarizing chunk]: {str(e)}")

    full_summary = ' '.join(summaries)
    try:
        summary_enum = SummaryType[str(summary_type).replace(" ", "_").upper()]
    except KeyError:
        summary_enum = SummaryType.PARAGRAPH
```

```
    return summary_enum.format(full_summary)
```

//models.py

```
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin
from datetime import datetime

db = SQLAlchemy()
class User(db.Model, UserMixin):
    __tablename__ = 'users'

    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)
    summaries = db.relationship('Summary', backref='user', lazy=True)

    def __repr__(self):
        return f'<User {self.username}>'
class Summary(db.Model):
    __tablename__ = 'summaries'

    id = db.Column(db.Integer, primary_key=True)
    content = db.Column(db.Text, nullable=False)
    summary = db.Column(db.Text, nullable=False)
    summary_type = db.Column(db.String(50))
    summary_length = db.Column(db.String(50))
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)

    def __repr__(self):
        return f'<Summary {self.id} by User {self.user_id}>'
```

//dp.py

```
from flask_sqlalchemy import SQLAlchemy
db = SQLAlchemy()
```

2.CREATE THE FRONTEND

templates/

//index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>NoteSummarizer AI</title>
  <link rel="stylesheet" href="static/style.css">
  <style>
    nav a.auth-link {
      float: right;
      margin-left: 15px;
      text-decoration: none;
      color: #fff;
      background-color: #007BFF;
      padding: 6px 12px;
      border-radius: 6px;
      font-weight: bold;
    }

    nav a.auth-link:hover {
      background-color: #0056b3;
    }
  </style>
</head>
<body>
  <header>
    <nav>
      <a href="/" class="active">Home</a>
      <a href="/summarize">Summarize</a>
      <a href="/about">About</a>
      <a href="/contact">Contact</a>
      <a href="/privacy">Privacy</a>
      <a href="/login" class="auth-link">Login</a>
      <a href="/register" class="auth-link">Register</a>
```

```
    </nav>
  </header>

  <div class="hero hero-split">
    <div class="hero-left">
      <h1>🚀 AI Note Summarizer</h1>
      <p>Upload your notes, lectures, or documents and let AI do the hard work. Get clean, concise summaries in seconds.</p>
      <button onclick="window.location.href='/summarize'">🖍️ Start Summarizing</button>

      <div class="features">
        <div>
          <h3> Fast & Instant</h3>
          <p>Summarize large files in seconds.</p>
        </div>
        <div>
          <h3>📄 Any File Type</h3>
          <p>Works with text, PDFs, or documents.</p>
        </div>
        <div>
          <h3>🎯 Accurate AI</h3>
          <p>Powered by cutting-edge AI models.</p>
        </div>
      </div>

    </div>

    <div class="hero-right">
      <img src="static/ai-illustration.png" alt="AI Illustration">
    </div>
  </div>

  <footer>
    © 2025 NoteSummarizer AI. All rights reserved.
  </footer>
</body>
```

```
</html>
```

//login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Login | AI Note Summarizer</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet" />
  <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.0/css/all.min.css" rel="stylesheet">
  <style>
    body {
      background: url('https://img.goodfon.com/wallpaper/nbig/5/a9/priroda-les-derevia-sumrak.webp') no-repeat center center fixed;
      background-size: cover;
      font-family: 'Segoe UI', sans-serif;
      margin: 0;
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
    }
    .glass-card {
      background: rgba(255, 255, 255, 0.1);
      border-radius: 16px;
      padding: 40px;
      width: 100%;
      max-width: 420px;
      backdrop-filter: blur(12px);
      box-shadow: 0 8px 32px rgba(0, 0, 0, 0.3);
      color: #fff;
    }
    .btn-primary {
      background-color: #6a11cb;
      border: none;
      border-radius: 12px;
```

```
    }
    .btn-primary:hover {
      background-color: #5a0eb9;
    }
    .password-toggle {
      position: absolute;
      right: 15px;
      top: 38px;
      color: #ccc;
      cursor: pointer;
    }
    .form-footer {
      text-align: center;
      margin-top: 20px;
    }
    .form-footer a {
      color: #fff;
      text-decoration: underline;
    }
  </style>
</head>
<body>
  <div class="glass-card">
    <h3> Login to Your Account</h3>
    <form method="POST" action="/login">
      <div class="mb-3">
        <label for="email" class="form-label">Email address</label>
        <input type="email" class="form-control" name="email" id="email"
placeholder="you@example.com" required>
      </div>
      <div class="mb-3 position-relative">
        <label for="password" class="form-label">Password</label>
        <input type="password" class="form-control" name="password" id="password" placeholder="Your
password" required>
        <i class="fa fa-eye password-toggle" onclick="togglePassword('password', this)"></i>
      </div>
      <div class="d-grid">
```

```
      <button class="btn btn-primary">Login</button>
    </div>
    <div class="form-footer">
      <p>Don't have an account? <a href="/register">Register</a></p>
    </div>
  </form>
 </div>

 <script>
  function togglePassword(id, icon) {
    const input = document.getElementById(id);
    if (input.type === "password") {
      input.type = "text";
      icon.classList.remove("fa-eye");
      icon.classList.add("fa-eye-slash");
    } else {
      input.type = "password";
      icon.classList.remove("fa-eye-slash");
      icon.classList.add("fa-eye");
    }
  }
 </script>
</body>
</html>
```

//register.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>Register | AI Note Summarizer</title>
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
 <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.0/css/all.min.css" rel="stylesheet">
 <style>
  body {
    background: url('https://img.goodfon.com/wallpaper/nbig/5/a9/priroda-les-derevia-sumrak.webp') no-
```

```
repeat center center fixed;
      background-size: cover;
      font-family: 'Segoe UI', sans-serif;
      display: flex;
      align-items: center;
      justify-content: center;
      min-height: 100vh;
    }
    .glass-card {
      background: rgba(255, 255, 255, 0.1);
      border-radius: 16px;
      padding: 40px;
      width: 100%;
      max-width: 480px;
      backdrop-filter: blur(12px);
      box-shadow: 0 8px 32px rgba(0, 0, 0, 0.3);
      color: #fff;
    }
    .glass-card h3 {
      text-align: center;
      margin-bottom: 25px;
    }
    .btn-primary {
      background-color: #ff416c;
      border: none;
      border-radius: 12px;
    }
    .btn-primary:hover {
      background-color: #e0355c;
    }
    .password-toggle {
      position: absolute;
      top: 38px;
      right: 15px;
      color: #ccc;
      cursor: pointer;
    }
```

```
    .form-footer {
      text-align: center;
      margin-top: 20px;
    }
    .form-footer a {
      color: #fff;
      text-decoration: underline;
    }
  </style>
</head>
<body>
  <div class="glass-card">
    <h3> Create Your Account</h3>
    <form method="POST" action="/register">
      <div class="mb-3">
        <label class="form-label">Username</label>
        <input type="text" name="username" class="form-control" placeholder="Username" required>
      </div>
      <div class="mb-3">
        <label class="form-label">Email Address</label>
        <input type="email" name="email" class="form-control" placeholder="you@example.com"
required>
      </div>
      <div class="mb-3 position-relative">
        <label class="form-label">Password</label>
        <input type="password" name="password" id="password" class="form-control" placeholder="Create
password" required>
        <i class="fas fa-eye password-toggle" onclick="togglePassword('password', this)"></i>
      </div>
      <div class="mb-3 position-relative">
        <label class="form-label">Confirm Password</label>
        <input type="password" name="confirm_password" id="confirm_password" class="form-control"
placeholder="Confirm password" required>
        <i class="fas fa-eye password-toggle" onclick="togglePassword('confirm_password', this)"></i>
      </div>
      <div class="d-grid">
        <button type="submit" class="btn btn-primary">Register</button>
```

```
    </div>
    <div class="form-footer mt-3">
      Already have an account? <a href="/login">Login</a>
    </div>
  </form>
 </div>

 <script>
  function togglePassword(id, icon) {
    const input = document.getElementById(id);
    if (input.type === "password") {
      input.type = "text";
      icon.classList.remove("fa-eye");
      icon.classList.add("fa-eye-slash");
    } else {
      input.type = "password";
      icon.classList.remove("fa-eye-slash");
      icon.classList.add("fa-eye");
    }
  }
 </script>
</body>
</html>
```

//summarize.html

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title> AI Note Summarizer</title>
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">
 <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
 <style>
  .drop-box {
    border: 2px dashed #ccc;
    padding: 40px;
    text-align: center;
```

```
      color: #888;
      cursor: pointer;
      border-radius: 10px;
      transition: border 0.3s;
    }
    .drop-box.dragover {
      border-color: #007bff;
      background-color: #f8f9fa;
    }
    .summary-box {
      background: #f9f9f9;
      padding: 20px;
      border-radius: 10px;
      border: 1px solid #ddd;
      margin-top: 20px;
    }
    .save-msg {
      font-size: 14px;
      color: green;
      margin-top: 10px;
    }
    textarea {
      resize: vertical;
    }
    pre {
      white-space: pre-wrap;
      word-wrap: break-word;
      font-family: inherit;
      margin: 0;
    }
  </style>
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-light shadow-sm">
    <div class="container">
      <a class="navbar-brand" href="/">NoteSummarizer AI</a>
      <div class="collapse navbar-collapse">
```

```
      <ul class="navbar-nav ms-auto">
        <li class="nav-item"><a class="nav-link" href="/">Home</a></li>
        <li class="nav-item"><a class="nav-link active" href="/summarize">Summarize</a></li>
        <li class="nav-item"><a class="nav-link" href="/about">About</a></li>
        <li class="nav-item"><a class="nav-link" href="/contact">Contact</a></li>
        <li class="nav-item"><a class="nav-link" href="/privacy">Privacy</a></li>
        {% if current_user and current_user.is_authenticated %}

          <li class="nav-item"><a class="nav-link" href="/logout">Logout</a></li>
        {% else %}
          <li class="nav-item"><a class="nav-link" href="/login">Login</a></li>
          <li class="nav-item"><a class="nav-link" href="/register">Register</a></li>
        {% endif %}
      </ul>
    </div>
  </div>
 </nav>

 <div class="container py-5">
  <h1 class="text-center mb-3"> AI-Powered Note Summarizer</h1>
  <p class="text-center text-muted mb-4">Instantly summarize notes, lectures, or articles using cutting-edge AI.</p>

  <form method="POST" enctype="multipart/form-data">
    <div class="mb-4">
      <div id="drop-area" class="drop-box">
        <p> Drag & Drop or <label for="file-upload" class="text-primary" style="cursor:pointer;">Click to Upload File</label></p>
        <input type="file" name="file" id="file-upload" hidden onchange="showFileName()">
        <p id="file-name" class="mt-2 fw-bold text-dark"></p>
      </div>
    </div>

    <p class="text-center text-muted">Or paste your text below:</p>
    <div class="mb-3">
      <textarea class="form-control" name="noteText" rows="7" placeholder="Paste your text here...">{{ input_text }}</textarea>
```

```
    </div>

    <div class="row g-3 mb-4">
      <div class="col-md-4">
        <label for="summaryType" class="form-label"> Summary Type:</label>
        <select name="summary_type" id="summaryType" class="form-select">

          <option value="paragraph" {% if selected_summary_type == 'paragraph' %}selected{% endif
%}>Paragraph</option>
          <option value="bullet_points" {% if selected_summary_type == 'bullet_points' %}selected{%
endif %}>Bullet Points</option>
          <option value="structured_notes" {% if selected_summary_type == 'structured_notes'
%}selected{% endif %}>Structured Notes</option>
          <option value="outline" {% if selected_summary_type == 'outline' %}selected{% endif
%}>Outline</option>
          <option value="essay_writing" {% if selected_summary_type == 'essay_writing' %}selected{%
endif %}>Essay Writing</option>
        </select>
      </div>

      <div class="col-md-4">
        <label for="length" class="form-label"> Length:</label>
        <select name="length" id="length" class="form-select">
          <option value="Short" {% if selected_length == 'Short' %}selected{% endif %}>Short</option>
          <option value="Medium" {% if selected_length == 'Medium' %}selected{% endif
%}>Medium</option>
          <option value="Long" {% if selected_length == 'Long' %}selected{% endif %}>Long</option>
        </select>
      </div>

      <div class="col-md-4">
        <label for="language" class="form-label"> Language:</label>
        <select name="language" id="language" class="form-select">
          <option value="English" {% if selected_language == 'English' %}selected{% endif
%}>English</option>
        </select>
      </div>
```

```
    </div>

    <div class="d-grid mb-4">
      <button type="submit" class="btn btn-primary btn-lg">✨ Generate Summary</button>
    </div>
  </form>

  {% if summary %}
  <div class="summary-box">
    <h4> AI Structured Summary</h4>
    <pre class="mt-3">{{ summary | safe }}</pre>

    <form method="POST" action="/download_pdf" class="mt-4">
      <input type="hidden" name="summary_text" value="{{ summary | e }}">
      <button type="submit" class="btn btn-success"> Download as PDF</button>
    </form>
    {% endif %}

    {% if saved %}
      <p class="save-msg"> Summary saved to your dashboard!</p>
    {% endif %}
  </div>
  {% endif %}
</div>

<footer class="bg-light py-3 text-center text-muted">
  © 2025 NoteSummarizer AI. All rights reserved.
</footer>

<script>
  const dropArea = document.getElementById('drop-area');
  const fileInput = document.getElementById('file-upload');
  const fileNameDisplay = document.getElementById('file-name');

  function showFileName() {
    if (fileInput.files.length > 0) {
      fileNameDisplay.textContent = ' Uploaded: ' + fileInput.files[0].name;
```

```
    }
  }

  dropArea.addEventListener('dragover', (e) => {
    e.preventDefault();
    dropArea.classList.add('dragover');
  });

  dropArea.addEventListener('dragleave', (e) => {
    e.preventDefault();
    dropArea.classList.remove('dragover');
  });

  dropArea.addEventListener('drop', (e) => {
    e.preventDefault();
    dropArea.classList.remove('dragover');
    const files = e.dataTransfer.files;
    if (files.length > 0) {
      fileInput.files = files;
      showFileName();
    }
  });
 </script>
</body>
</html>
```

Static/

//Style.css

```
/* Global Styles */
body {
 margin: 0;
 font-family: Arial, sans-serif;
 background: linear-gradient(135deg, #eef2f3, #f5f7fa);
 color: #333;
 min-height: 100vh;
 display: flex;
 flex-direction: column;
```

```
}

/* Navigation */
header nav {
  background-color: #ffffff;
  padding: 15px 30px;
  box-shadow: 0 2px 8px rgba(0,0,0,0.1);
  text-align: center;
}

header nav a {
  color: #333;
  text-decoration: none;
  margin: 0 15px;
  font-weight: bold;
  font-size: 16px;
  transition: color 0.3s ease;
}

header nav a.active,
header nav a:hover {
  color: #007bff;
  border-bottom: 2px solid #007bff;
  padding-bottom: 4px;
}
.hero button:hover {
  background-color: #0056b3;
  transform: scale(1.05);
}

/* Features Section */
.features {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
  margin-top: 30px;
}
```

```
.features div {
  background: #fff;
  padding: 15px;
  border-radius: 8px;
  box-shadow: 0 2px 6px rgba(0,0,0,0.1);
  flex: 1;
  min-width: 180px;
  text-align: center;
}

/* Summarize Section */
.summarize-container {
  max-width: 850px;
  margin: 40px auto;
  padding: 30px;
  background: #fff;
  border-radius: 12px;
  box-shadow: 0 4px 15px rgba(0,0,0,0.1);
  text-align: center;
}

.intro-text {
  font-size: 16px;
  color: #555;
  margin-bottom: 20px;
}

.upload-section, .upload-box {
  border: 2px dashed #ccc;
  padding: 25px;
  border-radius: 10px;
  margin-bottom: 15px;
  background-color: #fafafa;
}

.upload-label {
```

```
  color: #007bff;
  font-weight: bold;
  cursor: pointer;
}
.options-box {
  display: flex;
  flex-wrap: wrap;
  gap: 15px;
  margin: 20px 0;
  justify-content: center;
}

.option-group select {
  width: 100%;
  padding: 8px;
  border-radius: 6px;
  border: 1px solid #ccc;
}

.generate-button, .download-button {
  padding: 12px 30px;
  border-radius: 25px;
  border: none;
  background-color: #007bff;
  color: white;
  font-size: 16px;
  cursor: pointer;
  transition: background-color 0.3s ease, transform 0.2s ease;
}

.generate-button:hover, .download-button:hover {
  background-color: #0056b3;
  transform: scale(1.05);
}

/* Summary Display */
.summary-result {
```

```
  margin-top: 30px;
}

.summary-box {
  text-align: left;
  background: #f9f9f9;
  padding: 15px;
  border-radius: 10px;
  border: 1px solid #ddd;
  white-space: pre-wrap;
}

/* Contact Cards */
.contact-cards {
  display: flex;
  justify-content: center;
  gap: 20px;
  flex-wrap: wrap;
  margin-top: 30px;
}

.contact-card h3 {
  margin-bottom: 10px;
}

.contact-card a {
  color: #007bff;
  text-decoration: none;
  font-weight: bold;
}

.contact-card a:hover {
  text-decoration: underline;
}

/* Responsive */
@media (max-width: 768px) {
```

```
  .hero-split {
    flex-direction: column;
    text-align: center;
  }

  .hero-left, .hero-right {
    padding: 10px;
  }

  nav a {
    display: block;
    margin: 10px 0;
  }

  .options-box {
    flex-direction: column;
    align-items: center;
  }
}
```

//summarize.css

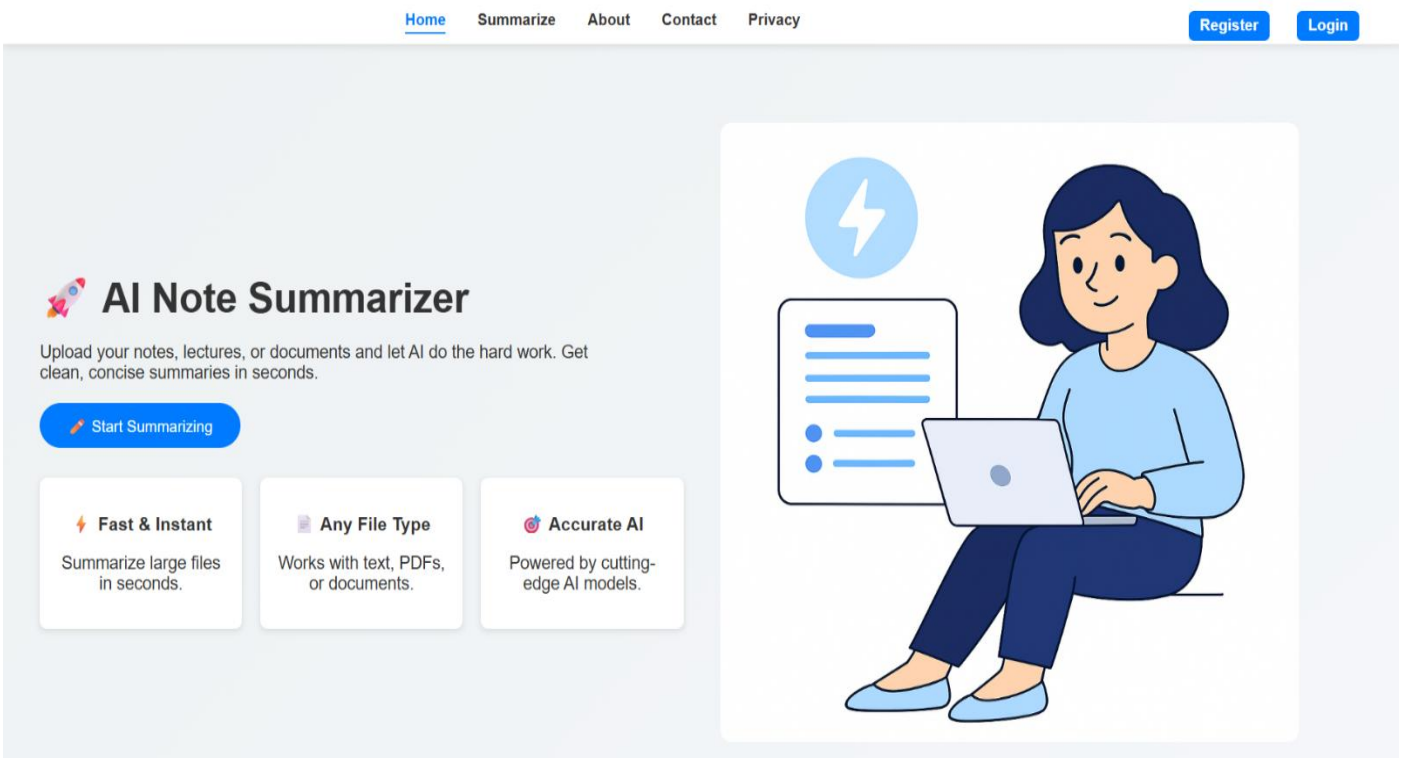
```
body {
  margin: 0;
  font-family: Arial, sans-serif;
  background: linear-gradient(135deg, #f0f7ff, #e8f5e9);
  color: #333;
  display: flex;
  flex-direction: column;
  align-items: center;
  padding: 40px 20px;
  min-height: 100vh;
}

.container {
  background: #fff;
  padding: 40px 30px;
  border-radius: 15px;
```

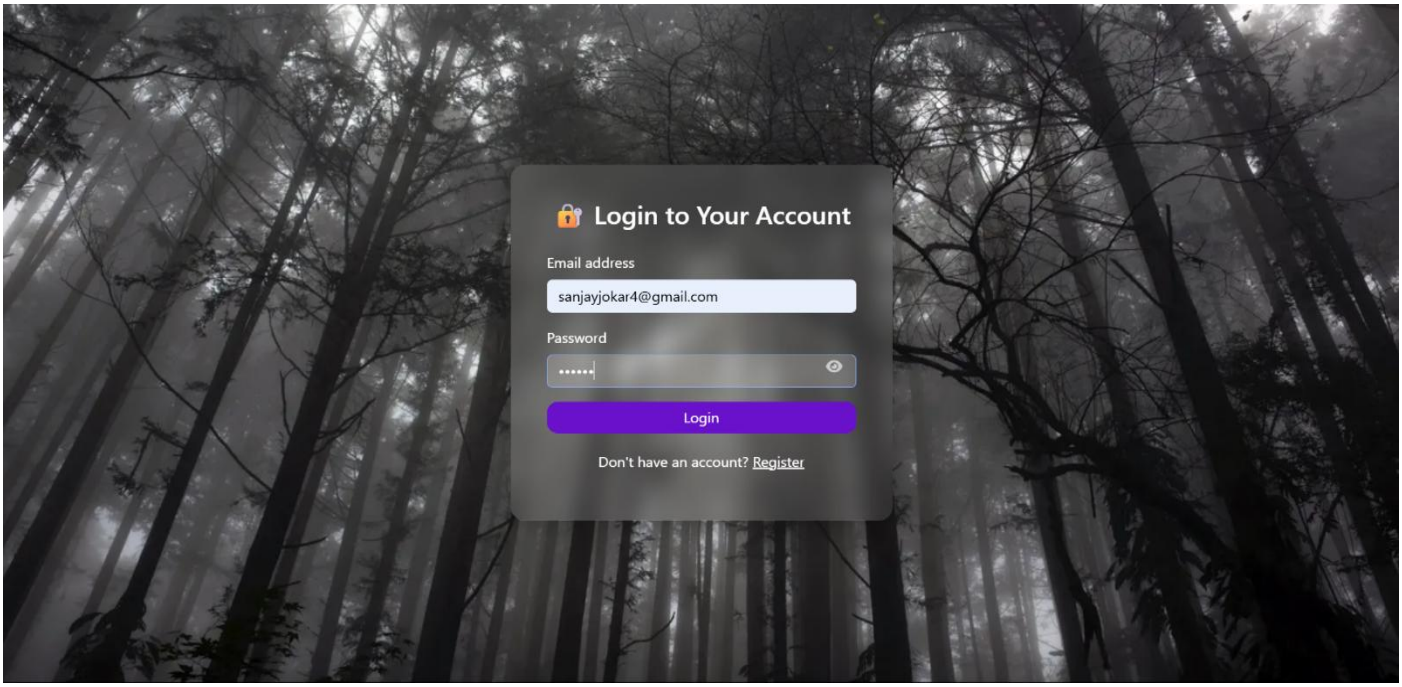
```
  box-shadow: 0 0 20px rgba(0,0,0,0.1);
  max-width: 800px;
  width: 100%;
}

h1 {
  font-size: 36px;
  margin-bottom: 20px;
}

.upload-box {
  border: 2px dashed #ccc;
  border-radius: 10px;
  padding: 20px;
  text-align: center;
  margin-bottom: 20px;
  cursor: pointer;
  color: #888;
}

.upload-box:hover {
  background-color: #f9f9f9;
}

textarea {
  width: 100%;
  height: 150px;
  margin-top: 10px;
  padding: 10px;
  border-radius: 8px;
  border: 1px solid #ccc;
  font-size: 16px;
}

.options {
  display: flex;
  flex-wrap: wrap;
```

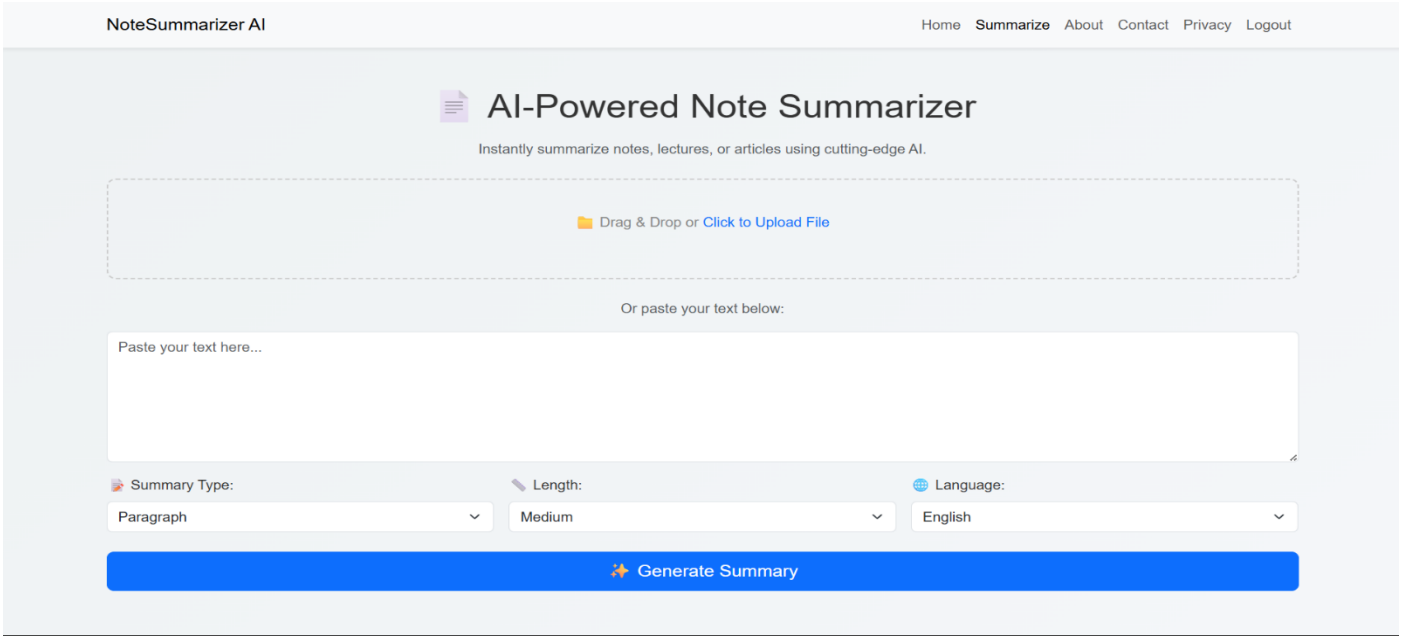
```
  gap: 20px;
  margin-top: 20px;
  justify-content: center;
}

.options label {
  display: block;
  font-weight: bold;
  margin-bottom: 5px;
}

select {
  padding: 8px;
  border-radius: 5px;
  border: 1px solid #aaa;
}

.generate-button, .download-button {
  margin-top: 20px;
  padding: 12px 30px;
  border-radius: 25px;
  background: #007bff;
  border: none;
  color: white;
  cursor: pointer;
  font-size: 16px;
}

.generate-button:hover, .download-button:hover {
  background: #0056b3;
}

.summary-box {
  margin-top: 40px;
  background: #f7f7f7;
  padding: 20px;
  border-radius: 10px;
```

```
  text-align: left;
}
```

OUTPUT







AI Structured Summary

Paragraph:

Cricket is one of many games in the "club ball" sphere that involve hitting a ball with a hand-held implement. Others include baseball, golf, hockey, tennis, squash, badminton and table tennis. In cricket's case, a key difference is the existence of a solid target structure, the wicket, that the batter must defend. It is generally believed that cricket originated as a children's game in the south-eastern counties of England, sometime during the medieval period. One possible source for the sport's name is the Old English word "cryce" (or "cricc") meaning a crutch or staff. In Samuel Johnson's Dictionary, he derived cricket from cryce, Saxon, a stick. In Old French, the word "criquet" seems to have meant a kind of club or stick. According to Heiner Gillmeister, a European language expert of Bonn University, "cricket" derives from the Middle Dutch phrase for hockey, "met de (krik ket)sen" ("with the stick chase").[17] Gill meister has suggested that not only the name but also the sport itself may be of Flemish origin. The earliest definite reference to cricket being played comes from evidence given at a court case in Guildford in January 1597 (Old Style, equating to January 1598 in the modern calendar). The case concerned ownership of a certain plot of land, and the court heard the testimony of a 59-year-old coroner, John Derrick, who gave witness that he was a scholler in the ffree schoole of Guldeford hee. Given Derrick's age, it is certain that cricket was being played c. 1550 by boys in Surrey.

Download as PDF

AI Structured Summary

Structured Notes:

- Cricket is one of many games in the "club ball" sphere that involve hitting a ball with a hand-held implement.
- Others include baseball, golf, hockey, tennis, squash, badminton and table tennis.
- In cricket's case, a key difference is the existence of a solid target structure, the wicket, that the batter must defend.
- It is generally believed that cricket originated as a children's game in the south-eastern counties of England, sometime during the medieval period.
- One possible source for the sport's name is the Old English word "cryce" (or "cricc") meaning a crutch or staff.
- In Samuel Johnson's Dictionary, he derived cricket from cryce, Saxon, a stick.
- In Old French, the word "criquet" seems to have meant a kind of club or stick.
- According to Heiner Gillmeister, a European language expert of Bonn University, "cricket" derives from the Middle Dutch phrase for hockey, "met de (krik ket)sen" ("with the stick chase").[17] Gill meister has suggested that not only the name but also the sport itself may be of Flemish origin.
- The earliest definite reference to cricket being played comes from evidence given at a court case in Guildford in January 1597 (Old Style, equating to January 1598 in the modern calendar).
- The case concerned ownership of a certain plot of land, and the court heard the testimony of a 59-year-old coroner, John Derrick, who gave witness that he was a scholler in the ffree schoole of Guldeford hee.
- Given Derrick's age, it is certain that cricket was being played c.
- 1550 by boys in Surrey.

Download as PDF

CONCLUSION

The **AI-Powered Note Summarizer Web Application** is a full-stack intelligent platform developed using Python (Flask), HTML/CSS, and the Hugging Face Transformers library. Designed as part of a summer internship project at Appin Technology, the application focuses on solving a common real-world challenge processing and condensing large volumes of text into readable, well-structured summaries. With support for five summary types - Paragraph, Bullet Points, Structured Notes, Outline, and Essay Writing the application caters to a diverse user base, including students, teachers, researchers, writers, and corporate professionals. The core summarization engine is powered by the transformer-based facebook/bart-large-cnn model, allowing for accurate, context-aware summarization across various content types. Users can also select the length of the summary short, medium, or long - based on their specific needs or time constraints.

The frontend of the application has been built with a clean, modern, and responsive design to provide an intuitive user experience on both desktop and mobile platforms. The backend, built with Flask, manages user input routing, form processing, session management, and integration with the AI model. A secure login and registration system ensures authenticated access, while user data is stored using SQLite and SQLAlchemy ORM for efficient and scalable database management. To enhance usability, the application includes the ability to export generated summaries into well-formatted PDF files using the ReportLab library, ensuring that users can easily save, print, or share their outputs offline. Additionally, backend logic includes the use of Python Enums for organized summary type selection and maintains state persistence post-submission to improve overall workflow continuity.

This project not only demonstrates effective integration of AI and web development technologies but also emphasizes user-centric design, modular architecture, and practical problem-solving. It reflects a thorough understanding of real-world software requirements such as scalability, accessibility, and content personalization. The application has potential for future enhancements, including PDF and DOCX file summarization, multilingual summarization, integration with cloud storage (Google Drive, Dropbox), history tracking, voice input (speech-to-text), and deployment on cloud platforms like AWS, Render, or Heroku. This internship provided valuable exposure to both theoretical and practical aspects of full-stack development and AI integration, strengthening skills in software design, debugging, teamwork, and delivering production-ready solutions.