|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Project Gen AI Website Summarizer Web App** | | | | | | |
|  |  |  | | | | |
|  |  |  | | | | |
|  |  |  | | | | |
| R0 | 17/05/25 | Submission | | | | |
| REV | DATE | DESCRIPTION | | | | |
| PURPOSE |  | For information | | | | |
|  | For review/comment | | | | |
|  | For submission | | | | |
| STATUS | | NAME | | Roll Number | DATE | |
| PREPARED | | Mian M. Husnain Akram  Muhammad Abdullah | | 497736  493772 | 17/05/25 | |
| SUBMITTED TO | | Dr Qaiser Riaz | | | | |
| DOCUMENT NUMBER | | | DOCUMENT TITLE | | | REVISION |
| 1 | | | Gen AI based Webisite Summarizer Web App | | | 0 |

Table of Contents

[1 Project Video: 4](#_Toc198496337)

[2 GitHub Link: 4](#_Toc198496338)

[3 Presentation Link: 4](#_Toc198496339)

[4 Executive Summary: 4](#_Toc198496340)

[4.1 Project Overview: 4](#_Toc198496341)

[5 Application OverviewUse Case Definition 4](#_Toc198496342)

[5.1 Core Functionality 4](#_Toc198496343)

[5.2 Key Features 4](#_Toc198496344)

[5.3 User Interface 5](#_Toc198496345)

[6 Technical Architecture 5](#_Toc198496346)

[6.1 Frontend Components 5](#_Toc198496347)

[7 Backend Architecture 5](#_Toc198496348)

[7.1 AWS Bedrock Integration: 6](#_Toc198496349)

[7.2 State Management: 6](#_Toc198496350)

[7.3 Data Flow 7](#_Toc198496351)

[7.3.1 User Input: 7](#_Toc198496352)

[7.3.2 Content Processing: 7](#_Toc198496353)

[7.3.3 AI Processing: 7](#_Toc198496354)

[7.3.4 Output Handling: 7](#_Toc198496355)

[8 Security Features 7](#_Toc198496356)

[9 Configuration Management 8](#_Toc198496357)

[9.1 Environment Settings: 8](#_Toc198496358)

[9.2 AWS Configuration: 8](#_Toc198496359)

[9.3 Streamlit Configuration: 8](#_Toc198496360)

[10 User Interface Screenshots 9](#_Toc198496361)

[10.1.1 Main Application Interface 9](#_Toc198496362)

[10.1.2 About Section 9](#_Toc198496363)

[10.1.3 Settings Panel 10](#_Toc198496364)

[10.1.4 Authentication Screens 10](#_Toc198496365)

[10.1.5 Summary Example 10](#_Toc198496366)s

[11 Performance Considerations 11](#_Toc198496367)

[12 Potential future improvements to the application include: 11](#_Toc198496368)

[13 Conclusion 12](#_Toc198496369)

[14 Future enhancements: 12](#_Toc198496370)

**Table of Figures**

[**Figure 10-1:** Landing Page 9](#_Toc198496373)

[**Figure 10-2:** About Section 9](#_Toc198496374)

[**Figure 10-3:** Settings for Wide View & Theme 10](#_Toc198496375)

[**Figure 10-4:** Signup Panel with proper user management 10](#_Toc198496376)

[**Figure 10-5:** Post Signin UI with History Tab for user 10](#_Toc198496377)

[**Figure 10-6:** Summarized Website & User Credits Info 11](#_Toc198496378)

1. Project Video:

[Click here](https://youtu.be/5T8pIx8onRA) to watch the project video.

1. GitHub Link:

Click here for github repo.

1. Presentation Link:

[Click here](https://prezi.com/view/z6G2PTgVRI2U52q48nPp/) for presentation access.

1. Executive Summary:
   1. Project Overview:

The Website Content Summarizer is a powerful web application that enables users to generate AI-powered summaries of website content by simply entering a URL. Built with Streamlit and powered by AWS Bedrock's Claude 3 AI model, this tool extracts and analyzes website text to produce concise, informative summaries that capture the main points of any webpage. The application features a user-friendly interface, secure authentication, and a comprehensive history tracking system for previous summaries.

1. Application OverviewUse Case Definition
   1. Core Functionality

The application allows users to::

Enter any website URL

Extract and process the website's textual content

Generate an AI-powered summary using Claude 3 via AWS Bedrock

View and download summaries

Access a history of previous summaries

Manage account settings

* 1. Key Features

Website text extraction with JavaScript rendering support

AI-powered summarization using Claude 3

Downloadable summary reports

History tracking of previous summaries

User authentication system

Customizable settings

Rate limiting and security protections

* 1. User Interface

The application features a clean, responsive design with:

Two-column layout for efficient space utilization

Left column containing URL input and summary display

Right column showing history of previous summaries

Sidebar with about information and settings

Custom styling for better user experience

1. Technical Architecture
   1. Frontend Components

The application utilizes Streamlit to create a responsive web interface with the following components:

**Main Layout**:

Two-column design for optimal space usage

Left column contains URL input and summary display

Right column displays history of previous summaries

**UI Elements**:

URL input field with validation

"Generate Summary" button with loading spinner

Summary display box with formatted text

Download button for saving summaries

History section with expandable entries

Sidebar with about information and settings

User authentication forms (sign in, sign up)

Credits remaining indicator

**Styling**:

Custom CSS for consistent look and feel

Responsive design that adapts to different screen sizes

Color-coded buttons and sections

Loading indicators for better user experience

Light and dark theme options

1. Backend Architecture
   1. **AWS Bedrock Integration**:

python

bedrock = boto3.client(

service\_name='bedrock-runtime',

region\_name='eu-north-1'

)

The application connects to AWS Bedrock using boto3, handling authentication, API calls, model responses, and error states.

**Content Extraction Pipeline**: The application uses a sophisticated extraction system:

Selenium for JavaScript-rendered content

BeautifulSoup for HTML parsing

Text cleaning and formatting algorithms

Content length validation

**AI Processing**: The core summarization functionality:

Formats content for Claude 3

Handles API responses

Processes and formats summaries for display

* 1. **State Management**:

python

if 'history' not in st.session\_state:

st.session\_state.history = []

The application maintains state using Streamlit's session state for:

Summary history

User authentication

Rate limit tracking

**Error Handling**: Comprehensive error handling for:

API connection issues

Content extraction problems

User input validation

Rate limit enforcement

* 1. Data Flow
     1. **User Input**:

User → URL Input → Validation → Processing

* + 1. **Content Processing**:

URL → Selenium → BeautifulSoup → Cleaned Text

* + 1. **AI Processing**:

Cleaned Text → Claude 3 → Formatted Summary

* + 1. **Output Handling**:

Summary → Display → History → Download Option

1. Security Features

**Rate Limiting**: The application enforces a limit of 10 requests per minute to prevent API abuse and ensure fair usage.

**Content Validation**: Content length is validated (maximum 100,000 characters) to prevent resource exhaustion.

**User Authentication**: The application includes a secure login/signup system with:

Password protection

Username/email uniqueness validation

Session management

**AWS Security**:

IAM role-based access

Secure credential management

API key protection

1. Configuration Management
   1. **Environment Settings**:

python

*# config.py*

MAX\_CONTENT\_LENGTH = 100000

RATE\_LIMIT = 10

* 1. **AWS Configuration**:

python

*# bedrock-policy.json*

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"bedrock:InvokeModel"

],

"Resource": [

"arn:aws:bedrock:eu-north-1::foundation-model/\*"

]

}

]

}

* 1. **Streamlit Configuration**:

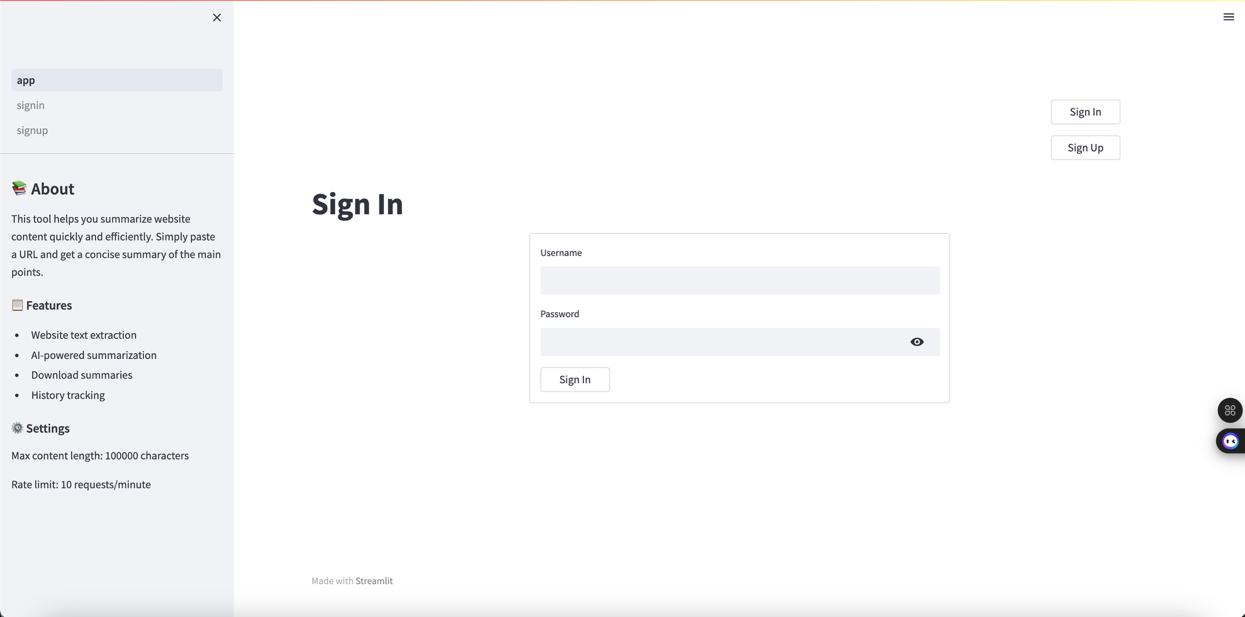
toml

*# .streamlit/config.toml*

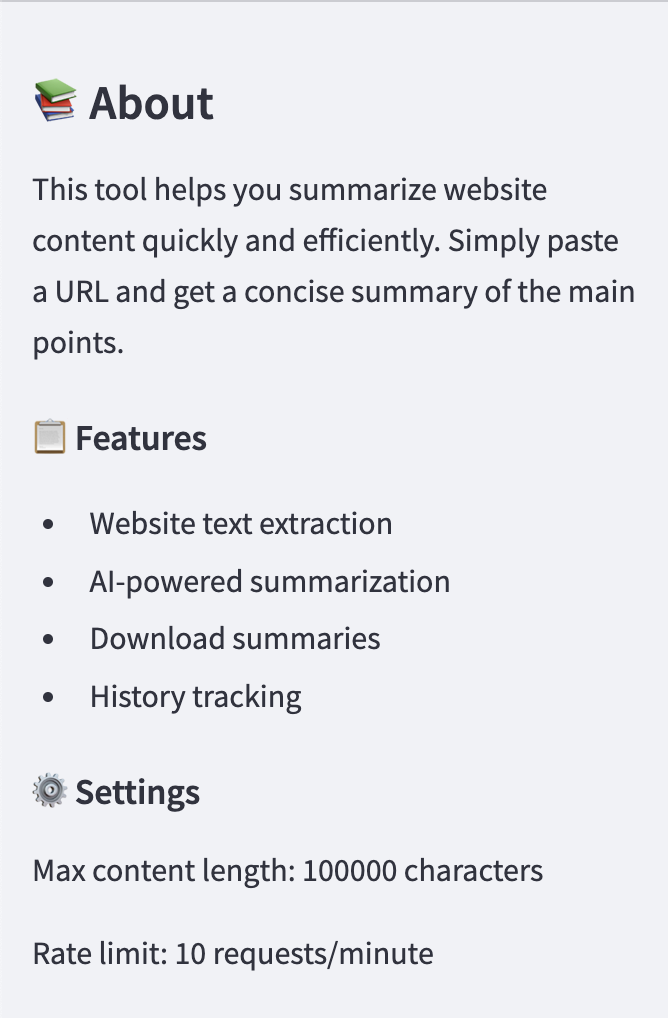
[theme]

primaryColor = "#4CAF50"

1. User Interface Screenshots
   * 1. Main Application Interface



* + - 1. Landing Page
    1. About Section

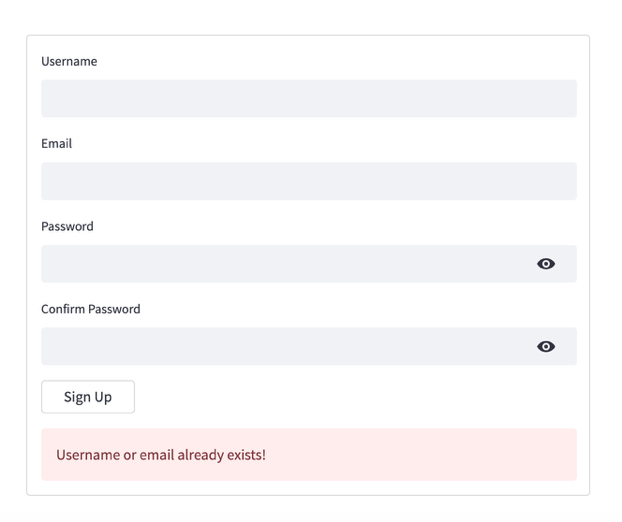


* + - 1. About Section
    1. Settings Panel

A screenshot of a computer

AI-generated content may be incorrect.

* + - 1. Settings for Wide View & Theme
    1. Authentication Screens



* + - 1. Signup Panel with proper user management
    1. Summary Example

A screenshot of a computer

AI-generated content may be incorrect.

* + - 1. Post Signin UI with History Tab for user

A screenshot of a computer

AI-generated content may be incorrect.

* + - 1. Summarized Website & User Credits Info

1. Performance Considerations

**Caching**:

Streamlit's built-in caching mechanisms

Session state management for efficient data handling

Optimized data structures for history tracking

**Resource Management**:

Chrome driver cleanup after extraction

Memory management for large websites

Connection pooling for AWS API calls

**Error Recovery**:

Graceful failure handling for API issues

User feedback for errors

State recovery mechanisms

## Future Enhancements

1. Potential future improvements to the application include:

**Enhanced Summarization Options**:

Adjustable summary length

Topic-focused summaries

Multilingual support

**Advanced Extraction**:

PDF support

Image and chart analysis

Video transcript summarization

**Collaboration Features**:

Shared workspaces

Team accounts

Summary annotations

**Integration Options**:

Browser extension

API for third-party applications

Mobile app version

1. Conclusion

The Website Content Summarizer represents a powerful tool for quickly extracting and summarizing web content using advanced AI technology. Its combination of robust content extraction, sophisticated AI processing, and user-friendly interface makes it an ideal solution for researchers, students, professionals, and anyone who needs to quickly understand the key points of web content without spending time reading entire articles.

1. Future enhancements:

Dos/DDOs Protection

LinkedIn Profiles Summarization & Validations for HRs

Research Papers smart analysis with Q&A