**Abstract**

The AI-Powered Daily News Aggregator & Summarizer is a smart system designed to collect, process, and summarize news articles from multiple online sources. Utilizing Natural Language Processing (NLP) and machine learning algorithms, this system curates relevant news, categorizes them into predefined topics, and generates concise summaries for users. The aggregator ensures real-time updates, delivering only the most important information without overwhelming the reader. This project aims to enhance the news consumption experience by reducing the time spent on reading while maintaining the essence of the articles. The system can be further extended to provide personalized news recommendations based on user preferences.

**Project Structure & Required Files:**

1. **Project Root Directory:** AI\_News\_Aggregator/
   * Contains all the necessary files and directories required for the project.
2. **Backend (Server-side Processing):** backend/
   * app.py: Main application file for handling requests and processing news articles.
   * config.py: Configuration settings for API keys, database connections, and server parameters.
   * news\_scraper.py: Scrapes news from multiple sources using APIs (e.g., Google News API, RSS feeds).
   * summarizer.py: Implements NLP techniques (BERT, T5, or GPT-based models) to summarize articles.
   * categorizer.py: Uses machine learning models to classify news into predefined categories.
   * recommendation.py: Implements a recommendation system based on user preferences and browsing history.
   * requirements.txt: Lists dependencies and packages required for backend development.
3. **Frontend (User Interface):** frontend/
   * index.html: Main UI page displaying news articles and summaries.
   * styles.css: Styling for the user interface.
   * script.js: JavaScript file for handling API requests and dynamic UI interactions.
   * components/: Contains reusable UI components (e.g., news cards, filters, etc.).
4. **Database & Storage:** database/
   * database.db: SQLite or MongoDB database for storing user preferences, saved articles, and summaries.
   * models.py: Defines the structure for storing news articles and user data.
5. **Machine Learning Models:** models/
   * text\_summarization\_model.pkl: Pre-trained summarization model.
   * news\_classification\_model.pkl: Model used to categorize articles.
   * recommendation\_model.pkl: Personalized recommendation model.
6. **APIs & Integrations:** api/
   * news\_api.py: Fetches news data from external sources.
   * user\_api.py: Handles user authentication and preferences.
   * summarization\_api.py: Exposes endpoints for summarizing articles.
7. **Testing & Evaluation:** tests/
   * test\_scraper.py: Tests for news scraping functionality.
   * test\_summarizer.py: Unit tests for text summarization accuracy.
   * test\_recommendation.py: Evaluates the recommendation system.
8. **Documentation & Reports:** docs/
   * README.md: Project overview, installation, and usage instructions.
   * design\_doc.pdf: System architecture and flow diagrams.
   * final\_report.pdf: Project report covering implementation, results, and future improvements.
9. **Deployment & DevOps:** deployment/
   * Dockerfile: Defines the container setup for easy deployment.
   * docker-compose.yml: Manages multi-container environments.
   * heroku.yml or aws\_config.json: Deployment configurations for cloud hosting.
10. **Miscellaneous Files:**

* .gitignore: Specifies ignored files in version control.
* LICENSE: Specifies the open-source or proprietary license for the project.
* CONTRIBUTING.md: Guidelines for contributing to the project.