ISSN 2277-2685
IJESR/April-June. 2026/ Vol-16/Issue-2/944-952
A. Pavithra *et. al.*, / International Journal of Engineering & Science Research
Full Length Article

Bookease Smart Novel Summarizer

Mr G Dayakar Reddy[1], A. Pavithra[2], Anepu Pujeetha[3], Jinkala Vyshnavi[4]
[1]Associate Professor; Department Of Computer Science And Engineering Bhoj Reddy Engineering College For Women Hyderabad India.
[2,3,4]B.Tech Students; Department Of Computer Science And Engineering Bhoj Reddy Engineering College For Women Hyderabad India.
Mail Id; jinkalavyshnavi04@gmail.com[4]
Accepted 31-05-2026

Abstract
The increasing availability of digital literary resources has created a need for intelligent tools that can assist readers in understanding and analyzing lengthy textual content efficiently. This study presents BookEase Smart Novel Summarizer, a web-based application designed to facilitate automated novel analysis through Natural Language Processing (NLP) techniques. The system is implemented using the Flask framework in Python with MySQL as the backend database.The proposed application accepts novels in PDF and TXT formats and performs extractive text summarization using frequency-based sentence ranking methods. In addition to summary generation, the system identifies major characters and evaluates the emotional tone of the narrative through VADER sentiment analysis. To provide deeper insight into story progression, an emotional arc visualization is generated by dividing the text into multiple segments and tracking sentiment variations throughout the novel.
The platform further enhances reader engagement by automatically generating multiple-choice comprehension questions from the summarized content. Accessibility features include text-to-speech narration with synchronized sentence highlighting and multilingual translation support for Telugu, Hindi, Tamil, French, Spanish, and German. Furthermore, the application integrates a search facility that enables users to discover literary works from a large collection of public-domain novels available through Project Gutenberg.Experimental evaluation demonstrates that the proposed system effectively combines text summarization, sentiment assessment, quiz generation, and language accessibility within a unified framework. The developed solution serves as an intelligent reading assistant that can support students, researchers, educators, and general readers in exploring and understanding literary texts more efficiently.
Keywords
Natural Language Processing (NLP); Novel Summarization; Sentiment Analysis; VADER; Extractive Summarization; Flask Framework; Text-to-Speech; Machine Learning Applications; Digital Literature Analysis; Web-Based Reading Assistant

Introduction

The rapid growth of digital literature has significantly increased the availability of novels and other long-form textual resources. However, reading and analyzing lengthy literary works often requires considerable time and effort, creating challenges for students, researchers, educators, and general readers who seek quick access to important information. Although recent advances in artificial intelligence have enabled the development of text summarization tools, many existing solutions primarily focus on generating brief summaries and often lack comprehensive literary analysis capabilities. Features such as character identification, sentiment evaluation, emotional progression tracking, comprehension assessment, and personalized content management are generally unavailable within a single integrated platform.To address these challenges, this study presents BookEase Smart Novel Summarizer, an intelligent web-based system developed to facilitate automated analysis of literary texts. The application enables users to upload novels in PDF or TXT formats or retrieve texts from a large collection of publicly available literary works. The system performs extractive summarization to identify key information from the novel while simultaneously generating character insights, sentiment-based evaluations, and visual representations of emotional changes throughout the narrative. Furthermore, the platform enhances user engagement through automatically generated multiple-choice questions that assess understanding of the summarized content. By integrating multiple Natural Language Processing (NLP) functionalities into a unified environment, the proposed system aims to improve reading efficiency and support deeper exploration of literary works.

Proposed System

The proposed BookEase Smart Novel Summarizer is designed as an intelligent reading assistance platform that combines text processing, sentiment analysis, content management, and accessibility features within a single framework. The system employs Natural Language Processing techniques implemented through the Python NLTK library to

perform sentence segmentation, tokenization, stop-word elimination, and frequency-based sentence ranking for extractive summarization. This process enables the identification of the most informative sentences in a novel, thereby generating concise yet meaningful summaries. In parallel, sentiment analysis is conducted using the VADER algorithm to evaluate the emotional characteristics of the text and calculate sentiment scores representing positive, negative, neutral, and compound emotional states.To provide a more comprehensive understanding of narrative development, the novel is divided into multiple textual segments and analyzed individually to generate an emotional arc visualization. This graphical representation illustrates variations in sentiment throughout the storyline and offers insights into the progression of emotional intensity within the text. The platform further supports automatic generation of multiple-choice comprehension questions based on extracted summary content, enabling users to assess their understanding of the novel.

The application incorporates a secure user management framework that supports account registration, authentication, and personalized content storage. Password security is maintained through bcrypt-based encryption, while email verification is implemented using one-time password (OTP) validation mechanisms. Authenticated users can maintain personal collections of analyzed novels, bookmark preferred titles, and contribute ratings and reviews. From a technical perspective, the system is developed using Python and Flask for backend processing, MySQL for persistent data storage, and HTML, CSS, and JavaScript for frontend interaction. Text extraction from PDF documents is performed using PyPDF2, whereas TXT files are processed through standard UTF-8 text decoding techniques. In addition, a shared community repository enables users to access novels uploaded by other registered members, thereby promoting collaborative learning and expanding the availability of analyzed literary resources. The integration of these functionalities demonstrates the potential of NLP-driven systems to support efficient literary analysis and enhance digital reading experiences.

Requirement Analysis

The functional requirements of the proposed BookEase Smart Novel Summarizer define the primary operations and services provided by the system. The platform incorporates a comprehensive user authentication mechanism that enables account registration, secure login, password recovery, and session management. During registration, users provide basic credentials including name, email address, and password. Passwords are protected through bcrypt-based hashing techniques, while account recovery is supported through email-based One-Time Password (OTP) verification to enhance security and user trust.A core component of the system is the Natural Language Processing (NLP) analysis module, which performs automated literary text processing. The module accepts both PDF and TXT documents and extracts textual content for further analysis. Frequency-driven extractive summarization techniques are employed to identify and present the most informative sentences from the novel. Character identification is performed through analysis of recurring proper nouns and frequently occurring uppercase terms. In addition, sentiment analysis is conducted using the VADER framework to determine the emotional orientation of the text. To provide a visual representation of narrative progression, the novel is divided into multiple segments and analyzed individually to generate an emotional arc depicting sentiment variation throughout the storyline. The system further enhances comprehension by automatically generating multiple-choice questions from the summarized content.

The novel access component allows users to interact with literary resources through multiple channels. Users may upload their own novels in supported formats or search a large collection of public-domain books through external literary repositories. The platform also presents featured titles with associated cover images and maintains a shared community library that enables users to access novels contributed by other members.Several personalized user services are integrated into the platform. Registered users can maintain a private library of analyzed novels, bookmark preferred works, and contribute ratings and reviews through a star-based evaluation system. Accessibility and usability are improved through text-to-speech functionality, which provides narrated reading with synchronized sentence highlighting. Furthermore, multilingual translation capabilities allow literary content to be viewed in multiple languages, thereby extending accessibility to a broader audience.Administrative functionality is incorporated to support platform management and maintenance. Authorized administrators can monitor user activity, manage uploaded novels, remove inappropriate content when necessary, and analyze reading trends to identify popular literary works within the community.

Computational Resource Requirements

The implementation of BookEase Smart Novel Summarizer requires moderate computational resources and can operate effectively on standard personal computing systems. A laptop or desktop computer equipped with a minimum of 4 GB RAM and an Intel i3-class processor or equivalent hardware is sufficient for development and deployment, although higher-performance processors are recommended for improved processing speed. Approximately 1 GB of available

storage space is required for project files, libraries, and database management.

From a software perspective, the application supports deployment on Windows, Linux, and macOS operating systems. The system is developed using Python 3.10 or later and utilizes MySQL as the database management system. Several supporting libraries contribute to the overall functionality of the platform. Flask serves as the web application framework, NLTK provides text processing capabilities, VADER enables sentiment evaluation, and PyPDF2 facilitates text extraction from PDF documents. Additional libraries such as Flask-Bcrypt, PyMySQL, and Flask-Mail are employed for password security, database connectivity, and email-based verification services, respectively.

Software Development Life Cycle Model

The development of BookEase Smart Novel Summarizer follows the Waterfall Software Development Life Cycle (SDLC) model, a structured methodology characterized by sequential progression through distinct development phases. The project commenced with a comprehensive requirement analysis stage during which system objectives, NLP functionalities, authentication mechanisms, database structures, and interface requirements were identified and documented.Following requirement specification, the system design phase focused on defining application architecture, database schemas, processing workflows, Flask routing structures, and user interface layouts. The implementation phase involved the development and integration of backend services, NLP algorithms, database operations, and frontend components. Once development was completed, extensive testing was performed at both module and system levels to verify functionality, performance, and reliability across different user scenarios and literary texts.The deployment phase established the operational environment using a Flask-based application server integrated with a MySQL database. Subsequent maintenance activities include continuous monitoring, error correction, performance optimization, and feature enhancement to ensure long-term system stability and usability.

Design

System Architecture

System architecture provides a conceptual representation of the overall structure of the proposed application and illustrates the interaction among its major components. It defines how information flows through the system, how requests are processed, and how individual modules collaborate to achieve the desired functionality. A well-designed architecture enhances maintainability, scalability, and system reliability by establishing clear boundaries between processing layers and responsibilities. In the proposed BookEase Smart Novel Summarizer, the architectural framework is divided into two complementary perspectives: Software Architecture and Technical Architecture. These perspectives collectively describe the functional organization and technological implementation of the system.

Software Architecture

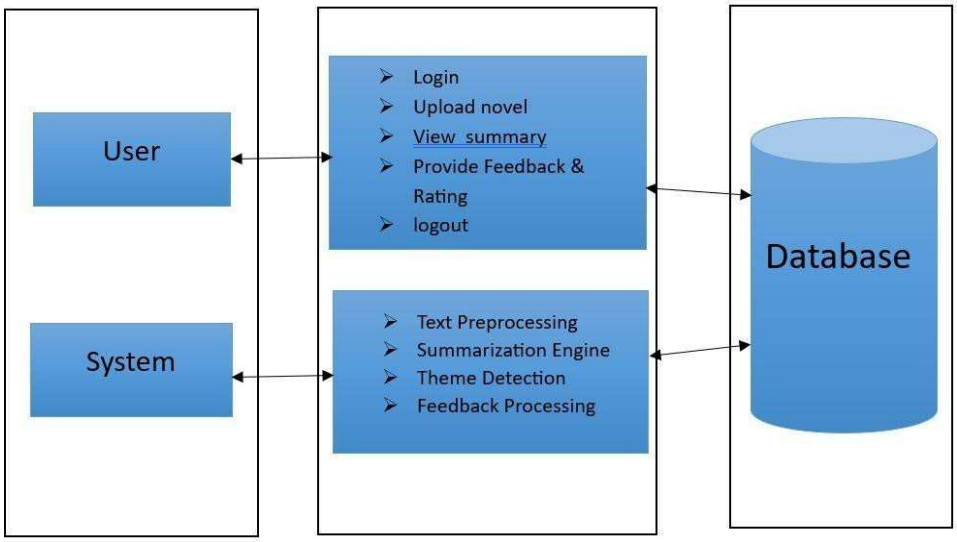


Fig.1 Software Architecture

The software architecture of BookEase Smart Novel Summarizer represents the logical organization of the application and illustrates the interaction between users, processing modules, and data storage components. The architecture is structured around three principal entities: the user interaction layer, the application processing layer, and the database management layer.The user interaction layer serves as the entry point to the system and provides access to functionalities such as account authentication, novel upload, summary visualization, review submission, rating management, and session

termination. Through this interface, users can upload literary texts, access generated analytical results, and interact with community-driven features available within the platform.The application processing layer constitutes the core intelligence of the system and is responsible for executing all computational and analytical operations. This layer performs text preprocessing, tokenization, extractive summarization, character identification, sentiment evaluation, emotional arc generation, and feedback management. Upon receiving a user request, the processing modules analyze the textual content and generate the corresponding outputs that are subsequently presented through the user interface.

The database layer functions as the persistent storage component of the architecture. It maintains user credentials, uploaded novels, generated summaries, sentiment scores, quiz data, ratings, reviews, and other application-related information. Continuous communication between the processing layer and the database ensures efficient retrieval and storage of information. The separation of presentation, processing, and storage responsibilities contributes to improved system organization, simplified maintenance, and enhanced scalability. Figure 3.1 illustrates the software architecture adopted in the proposed system.

Technical Architecture

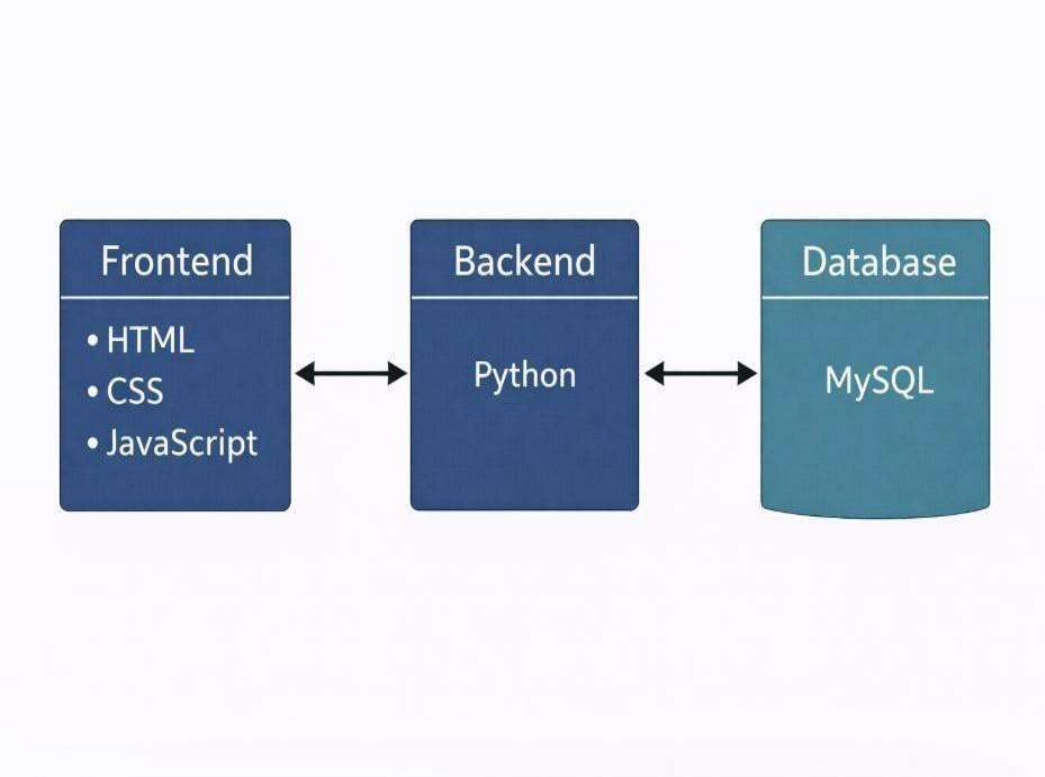


Fig.2 Technical Architecture

The technical architecture describes the implementation framework and technological layers utilized to develop and deploy the BookEase Smart Novel Summarizer. The architecture follows a layered design approach in which responsibilities are distributed across presentation, application, and data management tiers. This separation improves modularity, simplifies system upgrades, and facilitates future feature expansion.The presentation layer is developed using standard web technologies including HTML, CSS, and JavaScript. This layer is responsible for rendering the graphical user interface and providing interactive elements through which users access system services. All user requests, including authentication, novel uploads, searches, and review submissions, originate from this layer and are transmitted to the backend for processing.The application layer is implemented using Python and the Flask web framework. It acts as the central processing unit of the platform by managing routing operations, business logic execution, session control, NLP-based analysis, and communication with external services. Core functionalities such as text extraction, summarization, sentiment analysis, character recognition, translation, quiz generation, and text-to-speech processing are executed within this layer.

The data layer utilizes MySQL as the relational database management system for persistent information storage. User accounts, authentication records, uploaded literary content, analysis outputs, bookmarks, ratings, reviews, and community-

contributed resources are stored within structured database tables. The Flask application interacts continuously with the database through read and write operations, ensuring that user activities and analytical results are accurately maintained.The layered interaction among the presentation, application, and data management components establishes a robust and efficient architecture capable of supporting multiple users and complex NLP operations. This design promotes flexibility, maintainability, and reliable system performance while providing a scalable foundation for future enhancements. Figure 3.2 presents the technical architecture of the proposed system.

Implementation

The implementation of BookEase Smart Novel Summarizer was carried out using a full-stack web development approach that integrates Natural Language Processing (NLP), database management, and interactive web technologies within a unified platform. The system was developed using Python as the primary programming language due to its extensive ecosystem of libraries supporting machine learning, text analytics, and web application development. The backend architecture was implemented using the Flask framework, while MySQL was employed for persistent data storage. The user interface was designed using HTML, CSS, and JavaScript to provide an intuitive and responsive reading environment.

Python serves as the core processing engine of the application and manages request handling, database communication, file operations, and NLP workflows. Flask provides lightweight yet powerful web application functionality, enabling efficient routing, session management, authentication control, and template rendering. The framework facilitates seamless integration between the frontend interface and backend analytical modules.The text analysis component relies on the Natural Language Toolkit (NLTK) and the VADER sentiment analysis framework. NLTK performs sentence segmentation, tokenization, stop-word elimination, and frequency analysis to generate extractive summaries from literary texts. VADER evaluates the emotional characteristics of the content and produces sentiment scores that quantify positive, negative, neutral, and compound emotional states. These scores are subsequently utilized to generate emotional arc visualizations that illustrate sentiment progression throughout the narrative. Data persistence is achieved through a MySQL relational database containing structured information related to user accounts, uploaded novels, reviews, bookmarks, ratings, and community-shared content. Communication between the Flask application and the database is established using the PyMySQL connector, ensuring efficient data retrieval and storage operations. Text extraction from uploaded PDF documents is performed using the PyPDF2 library, whereas plain-text documents are processed through standard UTF-8 decoding procedures.

Several security mechanisms were incorporated to protect user information and maintain platform integrity. Passwords are encrypted using Flask-Bcrypt before storage in the database, minimizing the risk of unauthorized access. In addition, Flask-Mail is used to implement email-based One-Time Password (OTP) verification, providing a secure password recovery mechanism.The presentation layer employs Jinja2 templates integrated with HTML, CSS, and JavaScript to create a visually appealing book-oriented interface. Interactive features such as tab navigation, quiz evaluation, text-to-speech functionality, translation services, dynamic review loading, and rating systems are implemented through client-side scripting. The platform further extends its capabilities through integration with the Gutendex API, which provides access to a large collection of public-domain literary works derived from Project Gutenberg. This integration enables users to search, retrieve, and analyze thousands of novels directly within the application environment.

Testing

Software testing plays a critical role in validating system functionality, reliability, security, and overall quality. As software applications increasingly support educational, research, and information-intensive activities, rigorous testing becomes essential to ensure accurate processing and a positive user experience. In the context of BookEase Smart Novel Summarizer, testing activities were conducted to verify the correctness of NLP algorithms, authentication procedures, database operations, user interface components, and system integrations.The primary objectives of testing were to identify implementation defects, validate compliance with system requirements, strengthen application security, and enhance user satisfaction. Through systematic testing, potential issues could be detected and resolved before deployment, thereby improving the overall robustness of the platform.

Testing Dimensions

Testing was performed across multiple dimensions to ensure comprehensive system validation. Evaluation included database operations, application programming interfaces, and user interface components. Different levels of testing were considered, ranging from individual functions to integrated system workflows. Functional, performance, and security aspects were examined using both manual and structured testing methodologies. This multidimensional approach enabled the identification of defects at different stages of development while ensuring complete coverage of application features.

Testing Stages

Unit Testing

Unit testing focused on validating the behavior of individual software components in isolation. Key modules examined during this phase included the extractive summarization algorithm, sentiment analysis engine, character identification module, quiz generation component, password encryption mechanism, and text-processing functions. Each module was evaluated independently to confirm that expected outputs were generated for a variety of input conditions. The modular testing approach facilitated rapid identification and correction of implementation errors during development.

Integration Testing

Integration testing assessed the interaction between interconnected modules after successful unit validation. Particular attention was given to the communication between file upload services, text extraction routines, NLP processing modules, database storage mechanisms, and result visualization interfaces. Testing confirmed that analytical outputs generated by the processing layer were accurately transmitted to the database and subsequently presented to users through the web interface. The successful integration of all components demonstrated effective coordination among system modules.

System Testing

System testing evaluated the application as a complete and fully integrated software product. End-to-end workflows were executed to verify compliance with functional requirements and quality standards. The testing process included user registration, authentication, novel upload, text analysis, sentiment visualization, quiz generation, community interaction, and administrative operations. Multiple document formats and literary genres were utilized to ensure that the system maintained consistent performance under diverse operating conditions.

Acceptance Testing

Acceptance testing represented the final validation stage prior to deployment. Prospective users interacted with the platform to determine whether the implemented features satisfied practical requirements and usability expectations. Evaluation confirmed that functionalities such as summarization, sentiment analysis, multilingual translation, text-to-speech narration, community sharing, and personalized libraries operated effectively and provided meaningful value to readers. The successful completion of acceptance testing indicated readiness for operational deployment.

Testing Methodologies

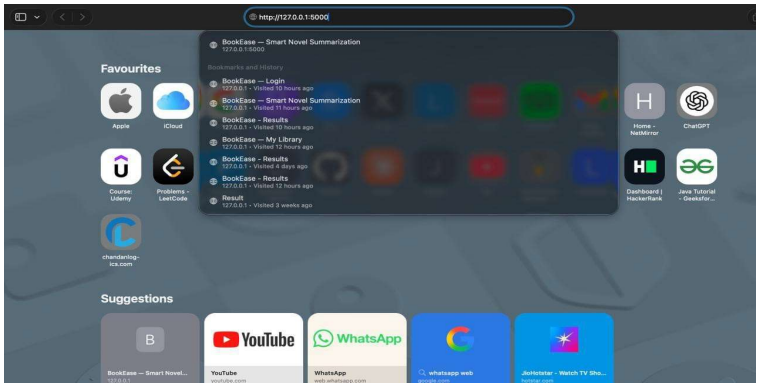
Two complementary testing methodologies were adopted during system validation. Black-box testing was employed to examine application behavior from the user's perspective without consideration of internal implementation details. This approach verified the correctness of outputs generated from actions such as novel uploads, summary generation, sentiment analysis, and quiz creation. White-box testing was used to evaluate the internal logic of critical software components, including summarization algorithms, sentiment computation procedures, and database interaction routines. Statement coverage, branch coverage, and path coverage techniques were applied to improve confidence in code quality and execution correctness.

Test Results and Validation

Comprehensive testing was performed across all major functional modules, including user registration, authentication, password recovery, file upload, novel analysis, character extraction, sentiment evaluation, quiz generation, translation services, text-to-speech functionality, novel search, dashboard management, bookmarking, review submission, community library operations, administrative controls, and session management. The results indicated that all evaluated test cases successfully met their expected outcomes without critical failures. The consistent success of these tests demonstrates the reliability, stability, and functional correctness of the proposed BookEase Smart Novel Summarizer platform.

Overall, the testing outcomes confirm that the developed system satisfies both functional and non-functional requirements while delivering accurate literary analysis, secure user management, and a responsive user experience.

Screenshots



ISSN 2277-2685
IJESR/April-June. 2026/ Vol-16/Issue-2/944-952

Fig 1 Run in the Terminal

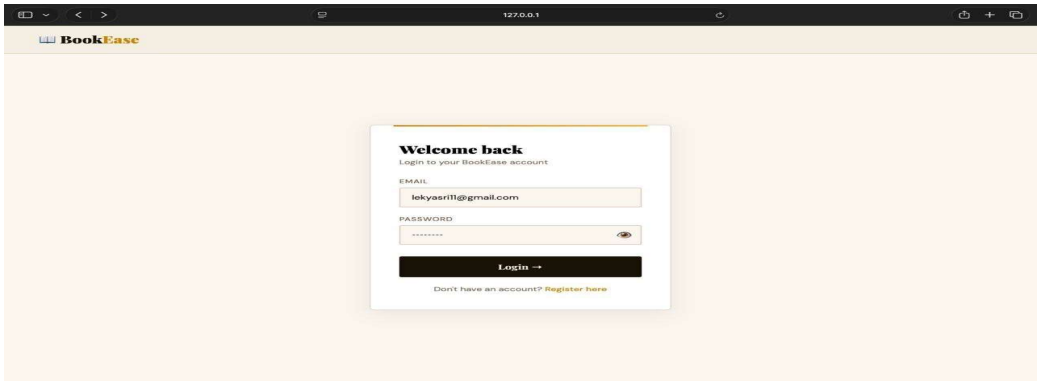


Fig 2 Login Page

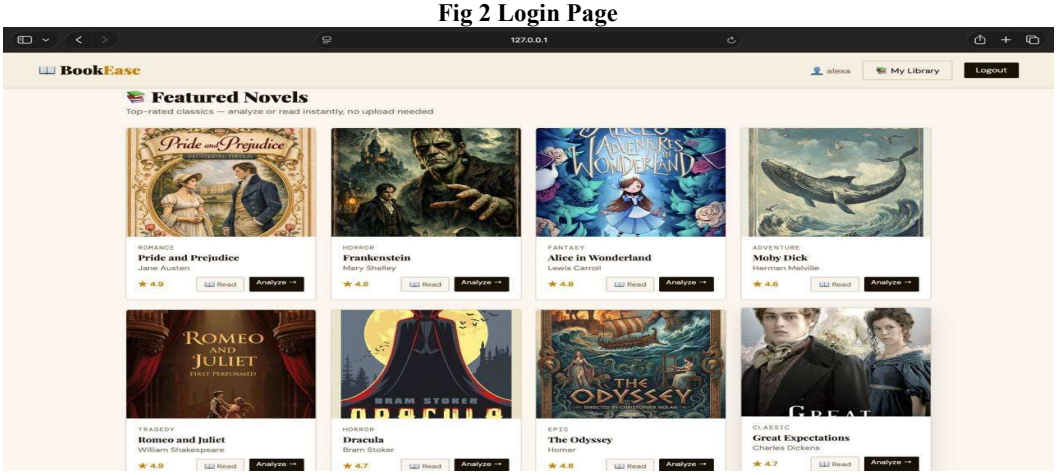


Fig 3 Featured Novels Page

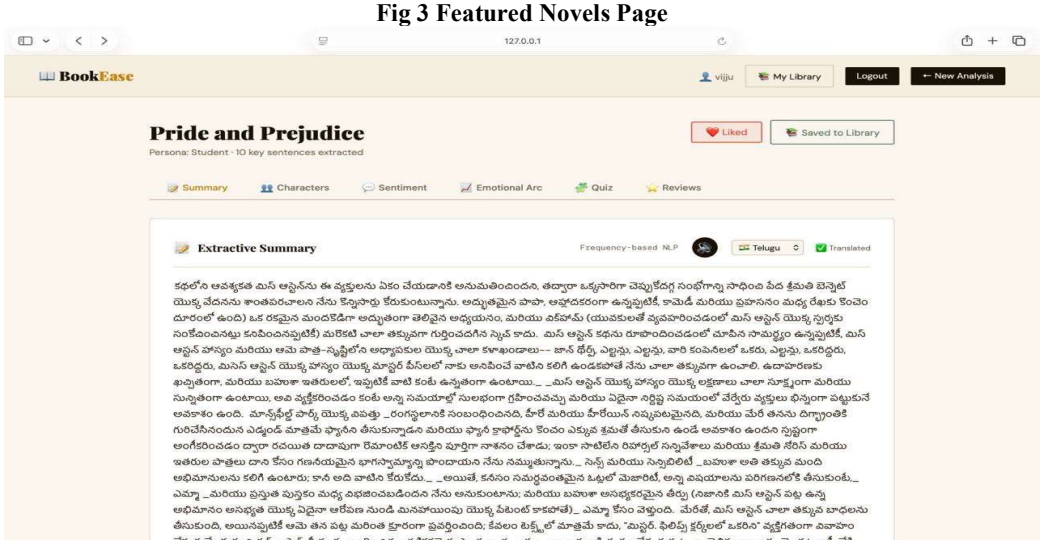


Fig 4 Translated Summary Page

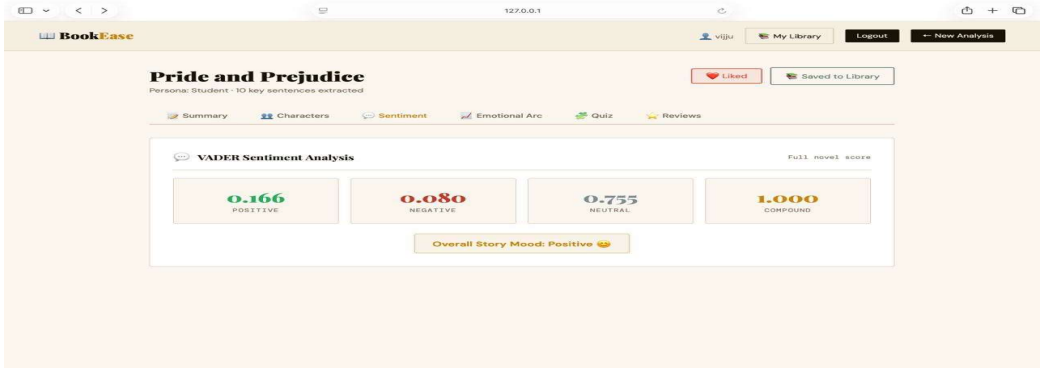


Fig 5 Sentiment Analysis Page

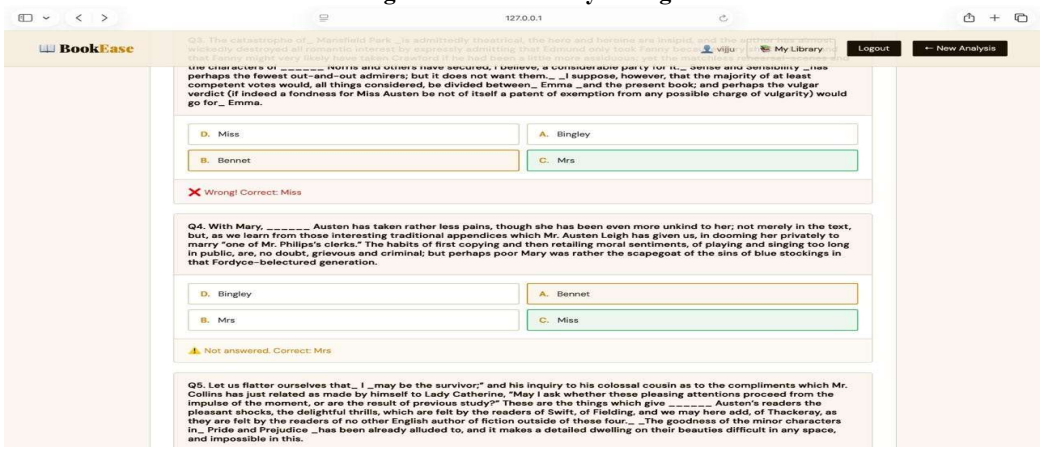


Fig 6 Generated Quiz Questions Page

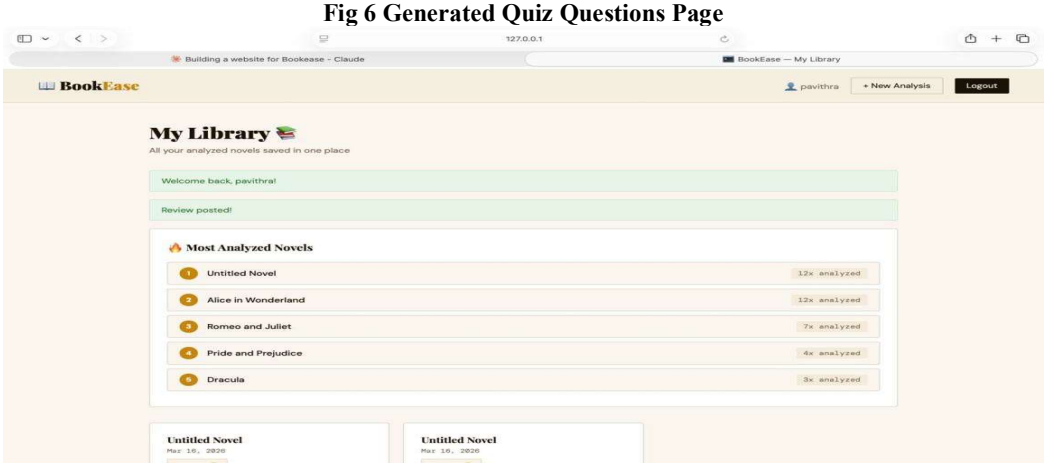


Fig 7 Most Analysed Novels Page

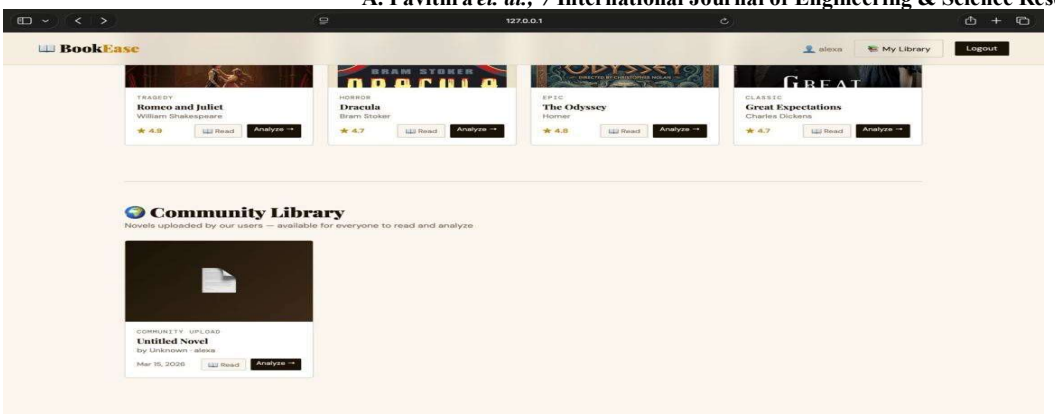


Fig 8 Community Library Page

Conclusion:

The Smart Novel Summarization and Analysis System uses NLP techniques to generate concise summaries and literary insights from novels. It reduces reading time, improves comprehension, and supports interactive features.The system provides an efficient and user-friendly solution for academic and literary analysis

Future scope:

BookEase can be enhanced with AI-powered summarization using Large Language Models (LLMs) for more accurate and context-aware analysis. Future plans include mobile app development, collaborative reading features, personalized book recommendations based on reading history, real-time multi-user review discussions, and integration with advanced NLP models for deeper character relationship analysis and plot structure detection.

References

[1] S. Bird, E. Klein, and E. Loper, "Natural Language Processing with Python," O'Reilly Media, 2009. Available at: https://www.nltk.org/book/

[2] C. J. Hutto and E. Gilbert, "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text," Proceedings of the International AAAI Conference on Web and Social Media, 2014.

[3] M. Grinberg, "Flask Web Development: Developing Web Applications with Python," O'Reilly Media, 2nd Edition, 2018.

[4] A. Géron, "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow," O'Reilly Media, 2019.

[5] NLTK Documentation, "Natural Language Toolkit," Available at: https://www.nltk.org/

[6] Flask Documentation, "Flask — A Python Microframework," Available at: https://flask.palletsprojects.com/

[7] PyMySQL Documentation, "PyMySQL — Pure Python MySQL Driver," Available at: https://pymysql.readthedocs.io/