DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY, LONERE

SEMINAR-II REPORT

SUBMITTED BY -

Name of Student	PRN No
Shivani Satish Mali	23068111242056

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SANJAY GHODAWAT INSTITUTE, ATIGRE

ACADEMIC YEAR: 2024-25

Certificate

This is to certify that the Seminar-I has been successfully completed by

Name of Student	PRN No
Shivani Satish Mali	23068111242056

In the partial fulfillment of the Degree in Computer Science and Engineering, Dr. Babasaheb Ambedkar Technological University, Lonere during the academic year 2024-25 under the guidance of

Ms. Dnyanada A Dongare
Mentor

Mr. S. V. Chavan
H.O.D

Dr. V. V. Giri
Director

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SANJAY GHODAWAT INSTITUTE, ATIGRE

ACADEMIC YEAR:2024-25

DR. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY, LONERE

Certificate

This is to certify that **Shivani Satish Mali** with PRN No. **23068111242056** has successfully completed **Seminar-II** (**BTCOS407**) for partial fulfilment towards completion of Degree in Computer Science and Engineering from Sanjay Ghodawat Institute, Atigre

Ms. Dnyanada A Dongare
Mantor

Mr. S. V. Chavan
H.O.D

Dr. V. V. Giri
Director

Date:

Place:

ACKNOWLEDGEMENT

I would like to express my truthful gratitude for providing me all the required valuable suggestions with It is my pleasure to be indebted to various people, who directly or indirectly contributed in the development of this work and who influenced my thinking, behavior, and acts during the course of study.

I am thankful to Dr.V.V.Giri, Director, Sanjay Ghodawat Institute and Mr.S.V.Chavan, HOD of Computer Science and Engineering department for all the support rendered to me during the entire Seminar-II

I also extend my sincere appreciation to Ms. Dnyanada A Dongare, Faculty mentor who consistently provide me all the required valuable suggestions with his precious time in accomplishing my training.

Lastly, I would like to thank the almighty God and my parents for their moral support and my friends with whom I shared my day-to-day experience and received lots of suggestions that improved my quality of work.

INDEX

ABSTRACT

To address this limitation, the proposed project titled **"LitAura – A Book Recommender"** introduces a novel, mood-based book recommendation system that dynamically suggests books according to the user's emotional state.

The system works by allowing users to describe their mood using natural language. This input is then analyzed in real time using **Sentiment.js**, a JavaScript-based sentiment analysis library. The tool evaluates the sentiment of the input text and classifies it into one of three categories: **positive**, **neutral**, or **negative**. Based on the resulting classification, the application determines a suitable book category and queries the **Google Books API** to fetch relevant book suggestions.

From a technical perspective, the front-end interface is built using **HTML**, **CSS**, and **Tailwind CSS** to ensure responsiveness, accessibility, and a clean user experience across devices. The logic for input handling, sentiment analysis, and dynamic content rendering is implemented in **JavaScript**, enabling seamless interaction between the user and the system. The results, which include the book's title, author, description, and cover image, are displayed dynamically without page reloads. Users can also save their favorite books using the **localStorage API**, allowing persistent storage of preferences directly in the browser.

The system follows a modular development roadmap consisting of environment setup, UI design, mood analysis, API integration, results display, and testing. It was developed and tested using tools like **Visual Studio Code**, **Live Server**, **Prettier**, **ESLint**, browser **DevTools**, and **Postman** to ensure code quality, efficient debugging, and smooth API communication.

One of the significant strengths of *LitAura* is its simplicity and usability. It provides an emotionally intelligent interface that can cater to users' current psychological needs, thereby enhancing their engagement with literature. Furthermore, as it does not rely on login systems or databases, it is lightweight and easily deployable on platforms like **GitHub Pages**.

In conclusion, *LitAura – A Book Recommender* demonstrates the effective application of sentiment analysis in enhancing digital reading experiences. By combining web development technologies with emotional intelligence, the project offers a unique solution for personalized content delivery.

INTRODUCTION

As digital content consumption increases, users are continuously seeking more personalized and intuitive tools that cater to their preferences and emotional needs. Reading, a deeply emotional and personal activity, has not yet been fully explored in terms of mood-based recommendations. The project **"LitAura – A Book Recommender"** bridges this gap by integrating sentiment analysis into a web-based application to suggest books aligned with the user's current mood.

2.1 Motivation and Background:

The motivation behind this project stems from the observation that most existing book recommendation platforms provide suggestions based on fixed criteria such as genre, author, popularity, or collaborative filtering. These models fail to consider **emotional context**, which plays a vital role in how a person engages with a book. For example, a person feeling down might seek motivational or uplifting literature, while someone feeling happy might look for something light and entertaining.

2] 1.1 Problem Statement

Traditional recommendation systems lack the ability to understand the emotional state of the user, which can lead to less satisfying reading experiences. There is a need for a **lightweight, browser-based solution** that can dynamically understand user mood and provide relevant book suggestions without requiring large datasets or account creation.

2] 1.2 Existing System Drawbacks

- Relies heavily on user history and ratings
- Limited personalization based on real-time emotion
- Requires login/account setups to function optimally
- Often suggests repetitive or trending content, not context-specific

2.2 Objectives of the Proposed System:

The core objective of **LitAura** is to recommend books that match the emotional tone of the user's input text. By leveraging **natural language processing (NLP)** via **Sentiment.js**, and combining it with the vast database of the **Google Books API**, the system aims to create an emotionally aware, user-friendly, and highly accessible book recommender.

2] 2.1 Project Goals

- Analyze the user's emotional state using sentiment analysis techniques.
- Map moods to relevant book categories (e.g., happy → uplifting books, sad → inspirational books).
- Fetch and display book recommendations dynamically using Google Books API.
- Provide the option to save favorite books using browser local storage for easy access.
- Develop a clean, responsive interface suitable for all screen sizes.

2] 2.2 Scope of the Project

The project is designed for:

- **Casual readers** looking for mood-aligned recommendations.
- **Students and educators** as an example of combining sentiment analysis and web development.
- **Developers** interested in lightweight, client-side NLP applications.

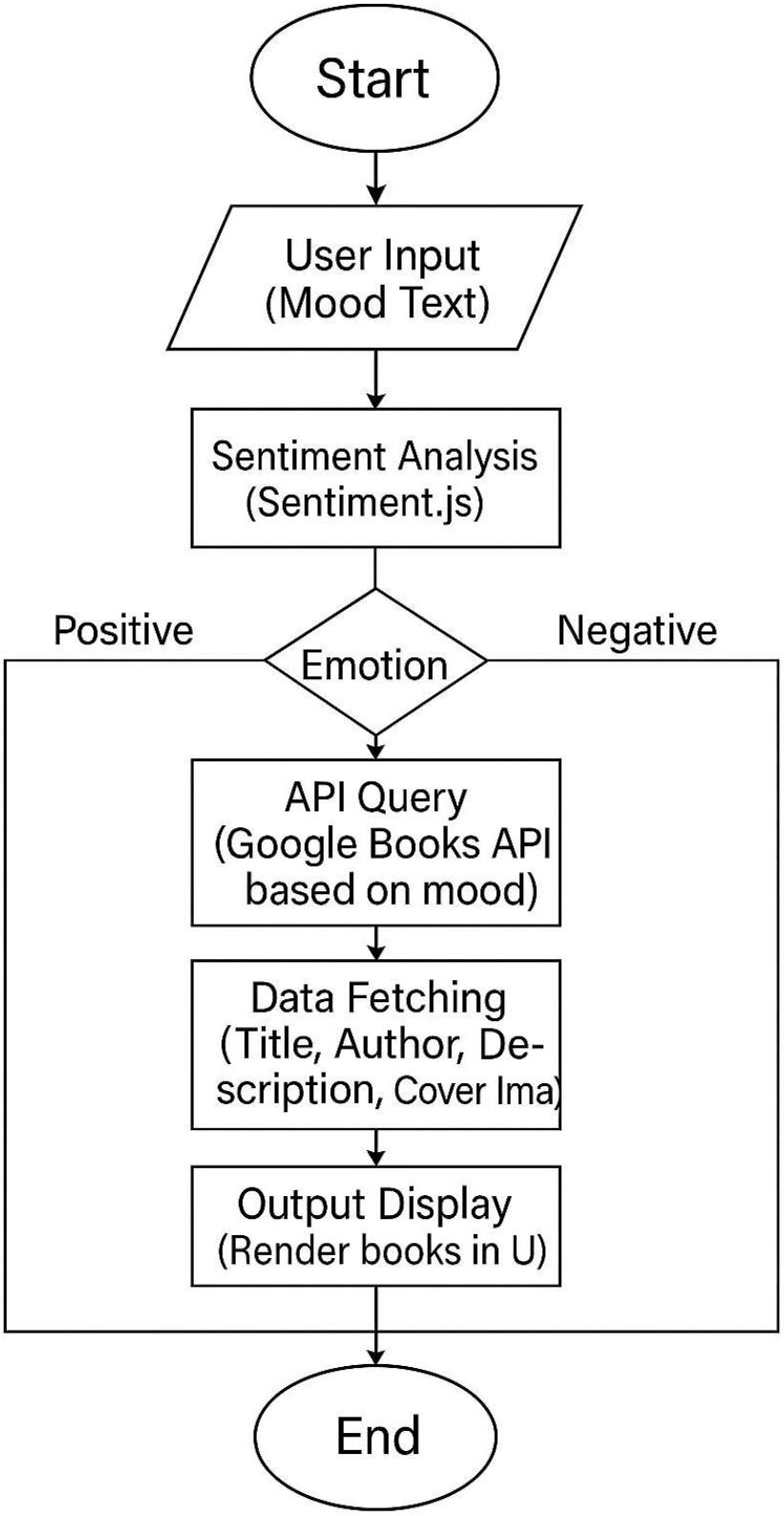
PROPOSED SYSTEM

The proposed system, **LitAura – A Book Recommender**, utilizes mood detection through sentiment analysis to personalize book recommendations. This approach combines front-end development techniques with intelligent text processing using **Sentiment.js**, offering a fast and responsive recommendation experience without the need for user accounts or large backend databases.

The application allows users to enter a text expressing their mood, which is then processed by Sentiment.js to calculate a sentiment score. Depending on the polarity (positive, neutral, or negative), the system dynamically constructs a search query and fetches book data from the Google Books API. These recommendations, along with cover images, titles, authors, and short descriptions, are rendered on the screen using DOM manipulation in JavaScript. Additionally, users can mark books as favourites, which are stored in the browser's local Storage for future reference, offering a personalized reading assistant with zero setup.

This approach is particularly effective in addressing the emotional side of reading, helping users discover content that resonates with their current state of mind—whether they need motivation, peace, or simply entertainment.

3.1 System Architecture / Block Diagram:



Explanation:

1. **User Input** – The user enters a mood or emotion in a text box on the web interface.
2. **Sentiment.js** – The JavaScript library processes this input and identifies the emotional polarity (positive, neutral, negative).
3. **Emotion Classification** – The system maps the mood to specific book categories.
4. **Google Books API** – Based on the classification, a query is sent to the API to fetch relevant books.
5. **Display Layer** – JavaScript dynamically renders the book suggestions on the page.
6. **Favorites Saving** – Users can save desired books using localStorage, allowing persistence even after closing or refreshing the page.

3.2 Hardware and Software Requirements:

Requirement Category	Specification/Details
Hardware	Any modern device (laptop, desktop, mobile phone) with internet access
Operating System	Platform-independent (Windows, macOS, Linux, Android, iOS)
Web Browser	Chrome, Firefox, Safari, or any modern HTML5-compatible browser
Programming Languages	HTML5, CSS3, JavaScript
Styling Framework	Tailwind CSS
Text Processing Library	Sentiment.js (for mood/sentiment analysis)
API Used	Google Books API (for fetching books dynamically)
Development Tools	Visual Studio Code, Prettier, ESLint, Live Server Extension
Deployment Platform	GitHub Pages (used for static web hosting; lightweight and free)
Storage Method	localStorage API (for saving favorites persistently in the browser)

3.3 Advantages and Disadvantages:

Advantages:

- Emotionally Relevant Suggestions: Recommends books based on user mood, enhancing engagement.
- Lightweight & Fast: Entirely client-side with no heavy backend or database.
- User-Friendly UI: Responsive, clean interface built with Tailwind CSS.
- No Account Required: Instant usage without login or personal information.
- Persistence via Local Storage: Favorites are saved locally and retained across sessions.

Disadvantages:

- Limited Sentiment Range: Only detects general sentiment (positive, neutral, negative); lacks nuanced emotion analysis.
- No Personalized History: No long-term user profiling or reading history tracking.
- Internet Dependent: Requires an internet connection to fetch books from the API.
- Static Favourites: Local Storage cannot sync across devices or browsers.

METHODOLOGY

4.1 Techniques / Methods to Be Used:

- The application is developed entirely using frontend technologies including HTML, Tailwind CSS, and JavaScript.
- Sentiment analysis is performed using the Sentiment.js library, which analyzes mood-related input from the user.
- The emotional tone (positive, neutral, negative) is classified and mapped to relevant keywords (e.g., "motivational", "healing").
- The Google Books API is dynamically queried based on the selected keyword to fetch real-time book recommendations.
- JavaScript is used to render the fetched books as cards in the user interface.
- Books can be added to a favorites list, which is saved using the browser's localStorage feature.
- Modal windows and pop-up notifications are implemented for interaction and confirmation messages.
- The user interface is made responsive and accessible across devices using Tailwind CSS.

4.2 Algorithm Used:

Step-by-step flow of logic implemented in the JavaScript file:

1. Start
2. User enters a mood or feeling in a text field
3. Sentiment.js processes the input and generates a sentiment score
4. Score is analyzed:
    - If score > 0 → Keyword = "motivational"
    - If score = 0 → Keyword = "neutral"
    - If score < 0 → Keyword = "healing"
5. A dynamic API query is constructed using the selected keyword
6. A request is sent to the Google Books API
7. Book metadata (title, author, description, thumbnail) is fetched
8. Books are displayed on the web page using JavaScript and Tailwind CSS
9. Users can:
    - View book information in cards
    - Save books to favorites (stored via localStorage)
10. End

4.3 Plan of Implementation:

Phase	Activity	Description
1	Project Setup	Initialized GitHub repository and project structure in VS Code
2	UI Design	Created layout and components using Tailwind CSS
3	Sentiment Analysis Integration	Integrated Sentiment.js to evaluate user mood
4	API Connection	Fetched book recommendations from Google Books API
5	Dynamic Rendering	Displayed books as cards with title, author, image, description
6	Favourites Feature	Implemented book saving using local Storage
7	Testing	Cross-browser and mobile responsiveness testing
8	Deployment	Deployed final version using GitHub Pages

4.4 Schedule of Work:

The development of the LitAura – A Book Recommender project was carried out over a period of eight weeks, with each week dedicated to a specific milestone in the implementation process.

In the first week, initial project planning, requirement gathering, and environment setup were completed. The second week focused on designing the user interface using HTML and Tailwind CSS, ensuring a clean and responsive layout. During the third week, the Sentiment.js library was integrated into the system, and sample inputs were tested for mood classification. In the fourth week, the Google Books API was successfully connected to the application, allowing real-time book data retrieval based on mood-based keywords.

The fifth week was dedicated to implementing the dynamic rendering of book cards on the frontend, displaying titles, authors, descriptions, and cover images. In the sixth week, the favourites feature was developed using local Storage, enabling users to save books for later. The seventh week involved thorough testing of the system on different browsers and devices, followed by UI refinements and bug fixing. Finally, in the eighth week, the project was deployed publicly via GitHub Pages, and final documentation was completed.

CONCLUSION

The LitAura – A Book Recommender system successfully demonstrates the practical integration of sentiment analysis and book recommendation using modern frontend web technologies. By leveraging the capabilities of Sentiment.js and the Google Books API, the system is able to interpret a user's mood from their input and provide personalized book suggestions that resonate with their emotional state.

The web application offers a clean, responsive, and user-friendly interface powered by Tailwind CSS, ensuring compatibility across devices. It eliminates the need for user registration and backend infrastructure, making it lightweight and easy to deploy. Features such as mood-based categorization, dynamic book rendering, and local Storage-powered favourites list contribute to an engaging and effective user experience.

This project not only highlights how artificial intelligence techniques like sentiment analysis can enhance user personalization but also demonstrates the effectiveness of API-based application development for real-world usability. Future enhancements could include deeper NLP integration, genre filtering, and user authentication for cross-device syncing.

DECLARATION

I hereby declare that the Seminar II project report entitled “LitAura – A Book Recommender” is the result of my own efforts and research. All the information presented in this report is accurate to the best of my knowledge. Where other sources have been used, proper references and citations have been included. I understand that this work is being submitted as part of my academic curriculum and that academic integrity has been maintained throughout.

My project can be accessed at:

https://shivani-mali.github.io/LitAura-A-Book-Recommender/

Signature

Shivani Mali