iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.79082

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Campus GPT - Your Campus AI Assistant

Rachana Amale, Prof. G. P. Dhomse, Radhika Sharma, Trupti Gunjal, Siddhi Raundal
Department of Computer Engineering, SNJB's Late Sau K. B. Jain College of Engineering, Chandwad, India

Abstract: With the proliferation of digital academic resources, it is becoming increasingly challenging for a student to find a specific piece of information within a large academic document. The existing search mechanisms in academic documents rely heavily on matching keywords and do not consider the actual meaning of the search query. This is making the process of information retrieval inefficient and time-consuming.
In this paper, an academic assistant tool called CampusGPT is proposed. The proposed tool is based on an artificial intelligence algorithm that will enable the student to interact with the document in a natural language. The proposed academic assistant tool is based on a combination of semantic search and a generative artificial intelligence algorithm. The proposed academic assistant tool will process the uploaded document in the PDF format and then convert it into a vector. The proposed academic assistant tool is implemented using the latest technologies. The proposed academic assistant tool will improve the efficiency and speed of the document-based query mechanism.

I. INTRODUCTION

Academic institutions produce a large amount of digital content in various forms such as lecture notes and research papers, etc. This digital content is extremely valuable; however, retrieving a specific piece of information from this content has remained a challenge for students.

Most of the systems use a search facility that uses a keyword search system. However, this system is not effective when the user query does not match the exact words used in the document. In such a case, it is necessary to search the document manually and retrieve the required information, which is a time-consuming process.

However, recent developments in Artificial Intelligence and Natural Language Processing have provided a solution to this problem by introducing various ways to solve this problem more effectively.

CampusGPT is a system that allows a user to pose a query in a more natural way and retrieve accurate results depending upon the content of the document. This system can be defined as a conversation between a user and a system.

The main objectives of this system are:

To provide better access to information in academic institutions

To provide better search time within a document

To provide a user-friendly query interface

II. LITERATURE REVIEW

The application of Artificial Intelligence in educational systems has increased rapidly in recent years, especially in the creation of intelligent academic assistants. Today's intelligent systems based on Artificial Intelligence allow users to interact with information in a more natural way using natural language. Large language models such as GPT models have shown impressive capabilities in understanding user queries and providing meaningful responses [1]. However, these systems are based on pre-trained knowledge and cannot provide accurate answers when users need information from a specific document such as academic notes and research papers.

To overcome this problem in documents, several systems have been designed to focus on text extraction and processing in PDF documents. PyPDF2 is widely used for text extraction in unstructured documents. Although this system enables users to search for text in documents, it is based on keyword matching algorithms. This method is not effective in most cases, as the search query may not be exactly matched in the document.

Recent advancements in semantic search have enabled the development of embedding-based retrieval models. This is based on the vector representation of the text data. Tools such as Sentence Transformers have enabled the ability to comprehend the context of the given text. This makes it easy to search using similarity [3]. Vector databases such as FAISS have become popular in the efficient storage and retrieval of vector data[4]. This has improved the accuracy of search results compared to traditional keyword search. Nonetheless, there is still a problem with handling huge amounts of data for real-time search.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Another significant development is the idea of Retrieval-Augmented Generation (RAG), where the idea of document retrieval and the use of a language model are combined. In this case, the relevant information is retrieved first from the database and then passed to the language model for the generation of accurate responses [5]. This minimizes the chances of the responses being incorrect. Although this idea has been effective in the development of more accurate responses, the use of retrieval systems with the language model has also added more complexity.

Apart from the retrieval and generation mechanisms, there is the incorporation of features like user authentication and data management. This is to improve the usability of the system. The MySQL database is used for the storage of user data and their interactions [6]. Although the features improve the usability of the system, there is a lack of integration of the features with the retrieval mechanisms.

Recently, several platforms have been developed that aim to integrate document processing, semantic search, and conversational AI in a single system. Frameworks such as LangChain have been developed to simplify the integration of language models with external data sources [7]. However, most platforms have been designed to support either document-based queries or AI interactions but not both.

Thus, there is a need to develop a system that incorporates document processing, semantic query, and AI-based response generation, all within one system. This is where the proposed system, CampusGPT, can help bridge this gap by providing a dual-mode system that can process both documents and general AI queries, thus increasing its accessibility.

III. PROPOSED SYSTEM

The proposed system, CampusGPT, is based on the modular architecture for an AI-driven academic assistance system that allows users to interact with educational documents using natural language queries.

The system architecture is based on the concept of layers. Each layer has been designed for a specific purpose. This concept has been used for scalability and efficient processing of the system.

The system architecture is based on five major layers:

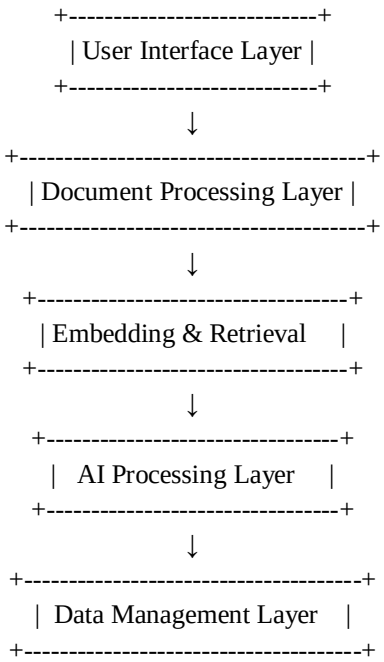


Fig. 1. Layered architecture of the CampusGPT system

A. User Interface Layer

The user interface of the CampusGPT system is developed using Streamlit, providing a simple and interactive environment for users. It allows users to log in, upload PDF documents, enter queries, and view responses in real time.

User authentication ensures secure access and supports smooth interaction between the user and the system.

B. Document Processing Layer

The document processing layer is in charge of handling uploaded PDF documents and retrieving text information. The system utilizes the PyPDF2 library in handling PDF documents and retrieving text information from the documents.The retrieved text is processed and cleaned to eliminate unnecessary formatting problems. Additionally, the text is segmented into smaller chunks using a recursive text-splitting method for efficient retrieval. . This helps the system retrieve relevant parts of the document efficiently.

C. Embedding and Retrieval Layer

At this stage, the data is embedded using pre-trained models from Hugging Face, whereby the data is transformed from text form to numerical form by embedding it in the system. The relationship between sentences is also obtained from these models, and this allows the system to understand the meaning of the query data.

The data is then stored in a database that allows data to be searched based on similarity. When a query is entered into the system, it is converted into a vector and then compared with the data in the database in order to obtain the most relevant data segments. In order to ensure that data retrieval is optimized in the system, similarity is used to ensure that only relevant data is retrieved and passed to the next process.

D. AI Processing Layer

The role of the AI processing layer is to ensure that the response is generated based on the content obtained from the documents or the general knowledge available. In the case of the offline mode, the relevant document segments are obtained, which serve as the context for the language model. The response is generated based on the context obtained from the documents. This is done in order to ensure that the response is based on the actual data available in the documents. In the case of the online mode, the response generation is done using a large language model (LLaMA via Groq API) without the need to upload the documents. The system makes use of structured prompt templates for clear, complete, and relevant responses.

E. Data Management Layer

The data management layer stores user data, login credentials, and chat session history in a MySQL database. This layer stores data securely and retrieves it as required.

The data stored in the database consists of the following:

*User login credentials
* Information regarding documents uploaded by the user
* User query and response session history
* System logs

The system also has a feature for logging user activity.

F. System Workflow Integration

The integration of each level ensures that there is a smooth flow in the system. If a user asks a question, it will be processed in a step-by-step manner from document processing, comparing, and finally generating a response using AI.

The system has the capability to switch from offline mode to online mode based on user requirements. This allows users to obtain both document-related and general knowledge answers under a single system.

The system's modularity gives it flexibility, allowing each module to be changed without affecting the entire system.

IV. DOCUMENT QUERY PROCESSING PIPELINE

A. End-to-End Data Flow

The overall architecture of the CampusGPT system includes a series of well-defined stages to transform input academic documents into a coherent response to a given query. This entire process is efficient, ensuring low response time along with high accuracy. The entire process begins with a user uploading a PDF document to the system. Once this is done, the system extracts text data from the uploaded PDF document. This extracted text data is then subjected to a series of pre-processing activities to maintain consistency in the extracted data. Subsequently, this extracted data is divided into smaller segments to make the overall processing efficient, allowing accurate retrieval of information.

The extracted text data is then transformed into a vector format using a transformer model. This vector format is then stored in a vector database to allow efficient retrieval of data.

When a user queries the system, it is also transformed into a vector format, which is then compared to the vector format of the stored data to retrieve accurate information. This information is then passed to a language model to generate a coherent response to the query, which is then displayed to the user in real time.

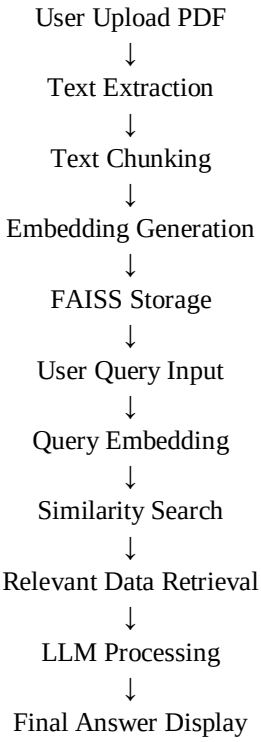


Fig. 2. End-to-end document query processing pipeline in CampusGPT.

B. Pipeline Stage Latency

The various stages in the system's pipeline have a role in determining the system's response time. The system is designed in a way that it balances the system's response time and its accuracy in producing a response. The system's response time is determined by various factors such as document size, number of segments stored in the system, and query used in the system. In most cases, the system is able to produce its initial response within a time frame of two to three seconds. The final response is produced within a time frame of three to four seconds. The various stages in the system's pipeline that have a role in determining its response time include document parsing, producing a response, producing a response using a language model, text segmentation, and similarity search. The major portion of the system's response time is determined by producing a response using a language model. However, the system's response time is not greatly affected due to efficient vector search

Stage	Operation	Time
Parsing	PDF extraction	200–400 ms
Chunking	Text split	100–200 ms
Retrieval	FAISS search	200–500 ms

Response	LLM generation	800–1500 ms
Total	Full pipeline	2–4 sec

TABLE I
Processing Time by Pipeline Stage

C. Response Time Model

The total response time for the system can be represented as the summation of individual response times for the stages in the system. All stages in the system operate sequentially to generate the total response time for the user. The response time for the system can be represented as follows:

$T_{total} = T_{parse} + T_{chunk} + T_{embed} + T_{retrieve} + T_{generate} + T_{display}$

In this response time model for the system, individual stages in the system include document parsing, segmentation, embedding generation, response retrieval, response generation, and display.

The response time for the system is normally expected to be in the range of approximately 2.5 to 3.5 seconds. This means that the system is able to deliver near real-time responses for the user in handling queries based on documents.

D. Computational Complexity

The efficiency of this system can also be analyzed in terms of computational complexity. Let 'n' denote the number of document segments, and 'q' denote the length of the user query.

The complexity of text extraction and segmentation is linearly related to the size of the document. Likewise, the complexity of generating embeddings is linearly related to the number of segments in a document.

The complexity of similarity search is optimized through vector indexing methods, which can handle large data sets efficiently. The complexity of generating responses, which relies on a transformer model, is linearly related to the length of input sequences, implying a high computational complexity compared to other components.

The memory requirement of this system is related to the number of stored embeddings as well as the size of the data being processed. Efficient management of embeddings ensures scalability of this system.

E. Optimization Strategies

Several optimization strategies have been integrated in the optimization of the system's performance, and this in turn reduces the response time of the system.

The incorporation of the chunk-based approach in the system optimizes the system's performance, especially in the handling of large documents, as it limits the amount of text being processed at any given time. The incorporation of the vector-based retrieval also optimizes the system's performance as it reduces the time taken in the retrieval of the required information from a given set of documents.

The use of selective retrieval optimizes system performance as it ensures that only relevant information is passed to the language model, which in turn improves system response quality. Furthermore, the system has been optimized to allow dual-mode operation, where document-based retrieval is only necessary in certain instances, thus avoiding extra overhead in general query mode. The optimization strategies have been incorporated to ensure that system performance is optimized while at the same time ensuring system response quality.

F. Workflow Summary

The workflow of the CampusGPT system begins with document upload and text extraction. The extracted text is segmented and converted into embeddings, which are stored in a vector database.

When a query is submitted, it is converted into an embedding and compared with stored vectors. The most relevant data is retrieved and passed to the language model, which generates a response displayed to the user.

V. PERFORMANCE EVALUATION

A. Experimental Setup

The performance of the CampusGPT system was carried out in a normal condition of use with a regular desktop setup. Various academic PDF files of different sizes were used, with both simple and complex queries.

The performance of the system was based on the response time, accuracy of the retrieved information, and overall stability of the system during continuous interaction.

B. Retrieval Performance

The system uses an embedding search method, which allows it to comprehend the meaning of the query asked by the user instead of relying on matching words. This increases the accuracy of the retrieved results.

The retrieval process is fast even with multiple documents uploaded, indicating that the system performs well.

C. Response Generation Performance

The response generation is based on a language model and is accurate and relevant to the query asked. The response generation time depends on the complexity of the query and is within an acceptable range.

D. End-to-End System Performance

The system usually responds within 2 to 4 seconds. When the documents are shorter, the system responds faster than when the documents are longer.

In general, the system performance is stable across the scenarios.

E. Scalability Analysis

The system can support multiple documents with no major performance problems. The more data the system processes, the more memory is used.

The system can scale well due to its modular design.

VI. IMPLEMENTATION DETAILS

A. System Environment

The CampusGPT system is implemented using Python as the primary programming language. The application is developed using the Streamlit framework, which provides a simple and interactive interface for users.

The system runs on a standard computing environment with moderate hardware requirements, making it suitable for academic use.

B. Technologies Used

The system uses PyPDF2 for extracting text from PDF documents. The extracted text is processed and converted into embeddings using transformer-based models.

FAISS is used for efficient storage and retrieval of embeddings, enabling fast similarity search. The response generation is performed using a large language model accessed through an API.

A MySQL database is used to store user data, login credentials, and chat history.

C. System Integration

All components of the system are integrated in a modular structure. The document processing, embedding, retrieval, and AI modules work together to generate responses.

This modular design ensures flexibility and allows future improvements without affecting the entire system.

VII. RESULTS AND DISCUSSION

A. System Performance Results

The CampusGPT system was tested with various academic documents and user queries. The results showed that the system can generate relevant and meaningful responses to queries in a short time.

The embedding-based retrieval method helps the system to understand the context of queries instead of focusing on the keyword. Therefore, the system can retrieve relevant information even if the word used in the query does not match the word used in the document.

B. Response Quality Analysis

The quality of the response generated by the system was evaluated based on the relevance, clarity, and completeness of the response. The results showed that the system generated accurate and clear responses.

The language model used by the system could successfully retrieve relevant information and present it clearly and in a way that could be easily understood. This makes the system suitable for academic purpose documents using natural language queries. The system has simplified the retrieval of relevant information by using document processing, semantic search, and artificial intelligence-based response generation.

The results have demonstrated that the system is able to generate accurate responses within a short period. The use of embedding-based retrieval has improved the search results, thus enhancing the efficiency of the system.

The dual functionality of the system has improved flexibility since users can easily switch between document-based queries and artificial intelligence-based queries.However, The system demonstrates certain limitations since it can only support PDF files. Additionally, users need to have an active internet connection when using the system for artificial intelligence-based queries.

C. Comparison with Traditional Methods

CampusGPT, in comparison with traditional methods of keyword search, offers better results in terms of accuracy and contextual understanding. In traditional methods, the user has to search for information and then interpret it, while in CampusGPT, the information is retrieved in the form of an answer.

This saves time for the user and offers a better user experience.

D. Discussion

The results show that the efficiency of the system is greatly enhanced by the integration of semantic search with AI-based response generation.

However, it has been observed that the efficiency of the system relies on the quality of the uploaded documents and the accuracy of the embedding model, especially for large datasets.

Overall, CampusGPT offers a useful tool for interacting with academic documents

VIII. CONCLUSION

The CampusGPT system demonstrates an effective solution for interacting with academic documents using natural language queries. By combining document processing, semantic search, and AI-based response generation, the system simplifies the process of retrieving relevant information.

The results show that the system provides accurate and context-aware responses within a short time. The use of embedding-based retrieval enhances search quality compared to traditional keyword-based methods.

The system also offers flexibility through its dual-mode functionality, allowing both document-based and general AI queries.

However, the system currently supports only PDF documents and depends on internet connectivity for AI-based processing. Future work can focus on supporting additional file formats and improving system scalability.

Overall, CampusGPT highlights the practical use of artificial intelligence in academic environments and its potential to improve learning experiences.

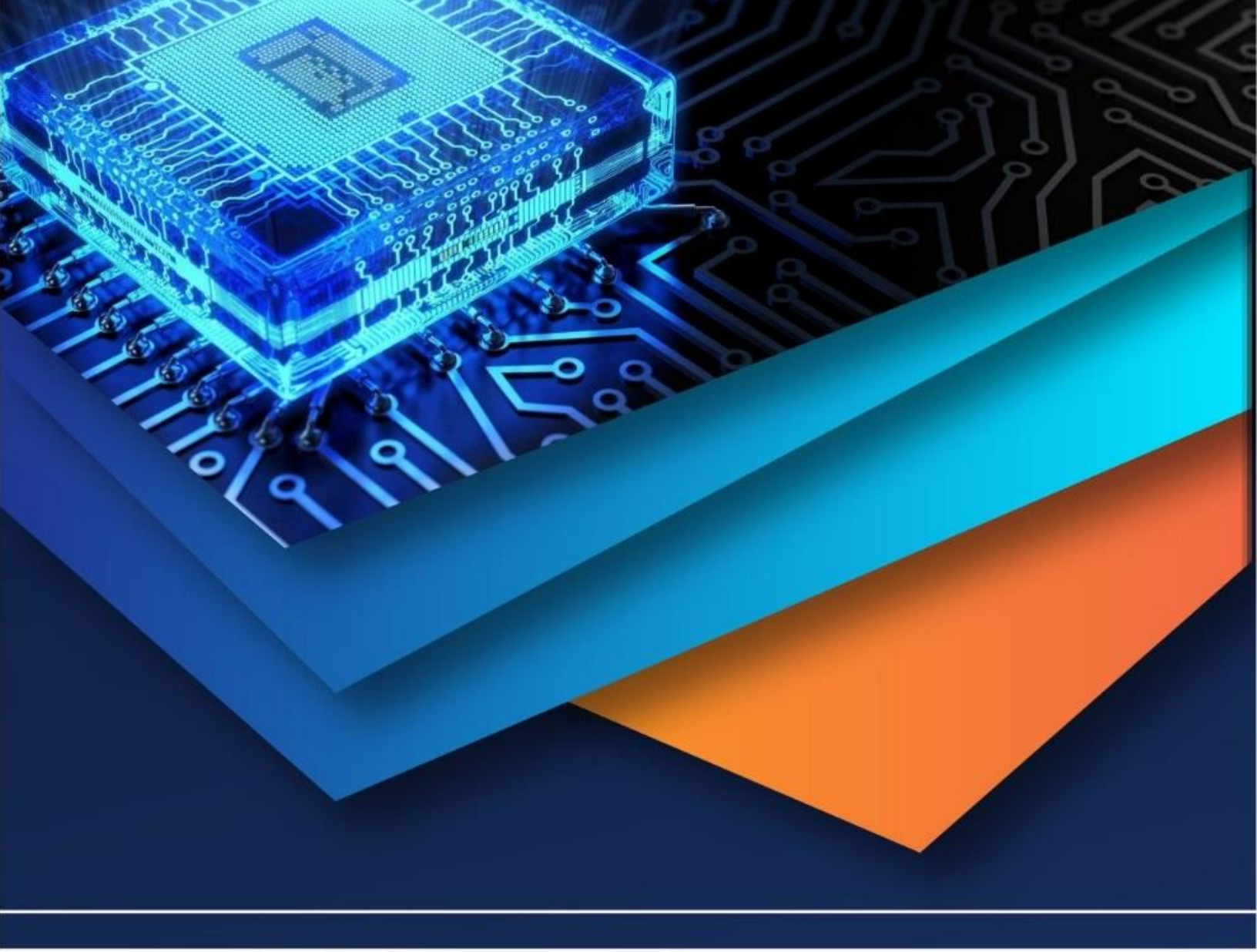
IX. ACKNOWLEDGMENT

The authors thank SNJB's Late Sau. K. B. Jain College of Engineering, Chandwad

REFERENCES

[1] T. Brown et al., "Language Models are Few-Shot Learners," NeurIPS, 2020.
[2] PyPDF2 Documentation, Python PDF Processing Library.
[3] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT," 2019.

[4] J. Johnson et al., “FAISS: A Library for Efficient Similarity Search,” Facebook AI Research, 2017.
[5] P. Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” 2020.
[6] MySQL Documentation, Oracle Corporation.
[7] LangChain Documentation, 2023.
[8] A. Vaswani et al., “Attention Is All You Need,” NeurIPS, 2017.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY