**COGNITEX: An Intelligent Cognitive Load–Adaptive Learning and Productivity System**

## PROJECTWORKPHASE 2 (REVIEW2)

***Submitted by***

# SANJAY.G 212222230131

***in partial fulfilment for the award of the degree of***

## **BACHELOR OF TECHNOLOGY**

***in***

**Artificial Intelligence and Data Science**

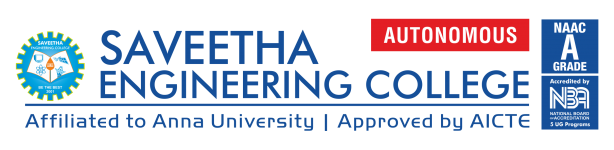


**SAVEETHA ENGINEERING COLLEGE, THANDALAM**

**An Autonomous Institution Affiliated to**

# ANNA UNIVERSITY - CHENNAI 600 025

NOVEMBER 2025

****

ANNA UNIVERSITY, CHENNAI

# BONAFIDE CERTIFICATE

Certified that this Project report **“ COGNITEX: An Intelligent Cognitive Load–Adaptive Learning and Productivity System ”** is the bonafide work of **SANJAY G (212222230131),** who carried out this project work under my supervision.

|  |  |
| --- | --- |
| SIGNATURE **Professor** Dr. K. Michael Mahesh, M.E., Ph.D.,SUPERVISOR Dept of Electronics and Communication Engineering  Saveetha Engineering College, Thandalam, Chennai 602105 | SIGNATURE **Professor**  **Dr. Karthi Govindharaju, M.E., PhD** HEAD OF THE DEPARTMENT Dept of Artificial Intelligence and Data Science ,  Saveetha Engineering College, Thandalam, Chennai 602105. |

DATE OF THE VIVA VOCE EXAMINATION: …………………………

## INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

I would like to express my heartfelt gratitude to our esteemed Founder President **Dr. N. M. Veeraiyan**, our President **Dr. Saveetha Rajesh**, our Director **Dr. S. Rajesh**, and the entire management team for providing the essential infrastructure.

I extend my sincere appreciation to our principal, **Dr. V. Vijaya Chamundeeswari, M.Tech., Ph.D.,** for creating a supportive learning environment for this project.

I am very thankful to our Dean of ICT, **Mr. Obed Otto,** M.E., for facilitating a conducive atmosphere that allowed me to complete my project successfully.

My thanks go to **Dr. Karthi Govindharaju**, Professor and Head of the Department of Computer Science and Engineering at Saveetha Engineering College, for his generous support and for providing the necessary resources for my project work.

I would also like to express my profound gratitude to my Supervisor, **<<Dr.K. Michael Mahesh M.E, Ph.D>>**, and my Project Coordinator **Dr. N.S. Gowri Ganesh**, Associate Professor at Saveetha Engineering College, for their invaluable guidance, suggestions, and constant encouragement, which were instrumental in the successful completion of this project. Their timely support and insights during the review process were greatly appreciated.

I am grateful to all my college faculty, staff, and technicians for their cooperation throughout the project. Finally, I wish to acknowledge my loving parents, friends, and well-wishers for their encouragement in helping me achieve this milestone.

# ABSTRACT

Digital learning and modern workplace environments increasingly demand sustained attention and continuous information processing, which often leads to cognitive overload, mental fatigue, and reduced efficiency. Most existing learning and productivity systems deliver static content and lack the ability to understand a user’s real-time cognitive state. This limitation negatively impacts engagement, learning outcomes, and overall work performance.

This project introduces COGNITEX, an intelligent cognitive load–adaptive learning and productivity platform that continuously monitors user behavior to estimate cognitive load in real time. The system collects software-based behavioral signals such as typing speed, interaction pauses, task completion time, and error frequency, and applies machine learning techniques to classify cognitive load without using sensors or wearable devices.

Based on the predicted cognitive state, COGNITEX utilizes agentic AI to autonomously determine optimal adaptation strategies. These strategies include adjusting content difficulty, modifying task structure, recommending intelligent breaks, and altering presentation formats. Generative AI is then employed to dynamically create personalized learning materials, work assistance, summaries, and real-time feedback tailored to individual users.

The system is designed to operate in both learning and professional work spaces, making it applicable to education, corporate training, and productivity enhancement. Its modular architecture ensures scalability, real-time responsiveness, and seamless integration with existing digital platforms while preserving user privacy.

Overall, the proposed system aims to reduce cognitive overload, enhance engagement, and improve learning efficiency and workplace productivity. By integrating cognitive modeling, agentic decision-making, and generative AI within a continuous feedback loop, COGNITEX provides a future-ready foundation for intelligent adaptive systems and advanced human–AI collaboration.

# TABLE OF CONTENTS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CHAPTER NO.** | | | **TITLE** | **Page**  **Number** |
| **1** |  |  | **INTRODUCTION** |  |
|  | 1.1 | Overview of the project | 10 |
|  | 1.2 | Problem Definition | 11 |
| **2** |  |  | **LITERATURE SURVEY** | 12 |
| **3** |  |  | **SYSTEM ANALYSIS** | 17 |
|  |  |  | Existing System | 17 |
|  | 3.1  3.2  3.3 |  | Disadvantages Proposed System Advantages | 17  18  18 |
|  | 3.4 |  | Feasibility Study | 19 |
|  | 3.5 |  | Hardware Environment | 19 |
|  | 3.6 |  | Software Environment | 19 |
|  | 3.7 |  | Technologies Used | 19 |
|  |  | 3.7.1 | Python | 20 |
|  |  | 3.7.2 | Html, CSS,Javascript | 20 |
|  |  | 3.7.3 | Fast API | 21 |
|  |  | 3.7.4 | Reinforcement Learning | 21 |
|  |  | 3.7.5 | Machine Learning | 22 |
|  |  | 3.7.6 | Generative AI(FLAN-T5) | 22 |
|  |  | 3.7.7 | Vector database(FAISS) | 22 |
|  |  | 3.7.8 | Version control (Git) | 23 |
|  |  | 3.7.9 | Deployment | 23 |
|  |  | 3.7.10 | Deep learning(LSTM) | 24 |
| **4** |  |  | **SYSTEM DESIGN** | 25 |
|  | 4.1 |  | ER- Diagram | 25 |
|  | 4.2 |  | Data Flow Diagram | 26 |
|  | 4.3 |  | UML Diagram | 27 |
|  |  | 4.3.1 | Use Case Diagram | 27 |
|  |  | 4.3.2 | Class Diagram | 28 |
|  |  | 4.3.3 | Sequence Diagram | 29 |
| **5** |  |  | **SYSTEM ARCHITECTURE** | 30 |
|  | 5.1 |  | Architecture Diagram | 30 |
|  | 5.2 |  | Algorithms | 31 |
| **6** |  |  | **SYSTEM IMPLEMENTATION** | 32 |
|  | 6.1 | Module-1Frontend and Backend | 32 |
|  | 6.2 | Module-2 Machine Learning Integration | 32 |
|  | 6.3 | Module-3Agentic Decision Flow and DL Forecasting | 33 |
|  | 6.4 |  | Module-4 Generative Ai Feedback | 33 |
|  | 6.5 |  | Module-5 Vector Memory management & Deployment | 33 |
| **7** |  |  | **SYSTEM TESTING** | 34 |
|  | 7.1 | Black Box Testing | 34 |
|  | 7.2 | White Box Testing | 35 |
|  |  |  |  |  |
| **8** |  |  | **CONCLUSION AND FUTURE ENHANCEMENT** | 36 |
|  | 8.1 | Conclusion | 36 |
|  | 8.2 | Future Enhancement | 37 |
| **9** | 9.1 |  | **APPENDIX-1**  Source Code | 38 |
| **10** |  |  | **APPENDIX-2** | 42 |
|  |  | Sample Output | 42 |
|  | 10.1 | Overload | 42 |
|  |  | OPtimal | 44 |
|  |  |  | UnderLoad | 45 |
| **11** |  |  | **REFERENCES** | 47 |

### LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **FIGURE DESCRIPTION** | **PAGE NO.** |
| 4.1 | Entity Relationship Diagram | 12 |
| 4.2.1 | Level 0 of Data flow Diagram | 13 |
| 4.2.2 | Level 1 of Data flow Diagram | 13 |
| 4.3.1 | Use Case Diagram | 14 |
| 4.3.2 | Class Diagram | 15 |
| 4.3.3 | Sequence Diagram | 16 |
| 5.1 | Architecture Diagram | 17 |
| 5.2. | Algorithms | 18 |
| 10.1 | OverLoad | 36 |
| 10.2 | Optimal | 37 |
| 10.3 | UnderLoad | 38 |

**LIST OF SYMBOLS**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **SYMBOL NAME** | **SYMBOL** |
| 1. | Usecase |  |
| 2. | Actor |  |
| 3. | Process |  |
| 4. | Start |  |
| 5. | Decision |  |
| 6. | Unidirectional |  |
| 7. | Entity set |  |
| 8. | Stop |  |

# Chapter 1 INTRODUCTION

## OVERVIEW OF THE PROJECT

In modern digital learning environments, students are increasingly exposed to large volumes of information, which often results in cognitive overload, reduced engagement, and poor learning outcomes. Traditional e-learning platforms mainly provide static content delivery and rely on manual assessments or predefined rules to evaluate learner progress. Such approaches fail to adapt dynamically to the learner’s real-time mental state and behavioral patterns. To address this limitation, this project proposes **COGNITEX**, an **Agentic Artificial Intelligence–driven adaptive learning system** designed to automatically monitor learner behavior, estimate cognitive load, and deliver personalized instructional interventions in real time. The system integrates automatic keyboard and mouse interaction tracking, machine learning–based cognitive load prediction, multi-agent reasoning, generative AI feedback, and deep learning–based performance forecasting into a unified intelligent framework. COGNITEX operates without requiring explicit user input. Instead, it continuously captures behavioral signals such as typing speed, inactivity duration, task completion time, and error frequency. These features are processed by a trained machine learning model to classify the learner’s cognitive state as Underload, Optimal, or Overload. Based on this prediction, an agentic decision layer autonomously selects suitable learning strategies, while a generative AI module produces personalized guidance. Additionally, a forecasting module anticipates future performance trends, enabling proactive interventions such as difficulty adjustment or break recommendations.

By combining agentic AI, generative models, predictive analytics, and automated behavioral sensing, COGNITEX provides a real-time adaptive learning experience that continuously responds to the learner’s evolving cognitive condition.

#### PROBLEM DEFINITION

In contemporary digital learning environments, students are increasingly exposed to large volumes of instructional content delivered through static platforms that lack awareness of individual cognitive states. Conventional e-learning systems provide uniform learning experiences without considering variations in learner attention, mental fatigue, engagement levels, or task difficulty perception. As a result, many learners experience cognitive overload, reduced motivation, or under-stimulation, ultimately leading to poor learning outcomes, burnout, and disengagement. Existing tutoring platforms primarily rely on manual assessments, periodic quizzes, or self-reported feedback, which fail to capture real-time behavioral patterns and cannot dynamically adapt instructional strategies based on the learner’s evolving cognitive condition.Furthermore, current adaptive learning systems rarely integrate continuous behavioral monitoring, predictive performance analysis, and autonomous intervention within a unified framework. Most solutions operate reactively rather than proactively, addressing learning difficulties only after performance degradation becomes evident. They also lack explainability and personalization, making it difficult for educators and learners to understand system decisions or trust automated recommendations. The absence of real-time cognitive inference, agent-based reasoning, and predictive intervention mechanisms significantly limits the effectiveness of existing educational technologies.Therefore, there exists a critical need for an intelligent, real-time adaptive learning system capable of automatically capturing user behavior, estimating cognitive load, forecasting future learning performance, and executing personalized interventions without manual input. Such a system must integrate machine learning for cognitive estimation, agentic AI for autonomous decision-making, generative models for personalized feedback, and analytics for transparency and performance tracking. Addressing this gap, the proposed COGNITEX platform aims to deliver a proactive, explainable, and self-adaptive learning environment that continuously monitors learner interaction, predicts cognitive risks, and dynamically adjusts instructional strategies to enhance engagement, reduce mental fatigue, and improve overall learning outcomes.

# Chapter 2 LITERATURE SURVEY

## INTRODUCTION

A literature survey or a literature review in a project report is that section which shows various analysis and research made in the field of your interest and the results already published, taking into account the various parameters of the project and the extent of project. Once the programmers start building the tool programmers need a lot of external support. This support can be obtained from senior programmers, books or from the websites. It is the most important part of your report as it gives you a direction in the area of your research. It helps you set a goal for your analysis - thus giving you your problem of statement. Literature survey is the most important sector in the software development process. Before developing the tools and the associated designing the software it is necessary to determine the survey the time factor, resource requirement etc., The consumer needs regarding online customer service differs from person to person. The needs are also based off each persons personal needs. We need to identify and anticipate these needs in order to completely and accurately meet them.

## LITERATURE SURVEY

#### Cognitive Load Theory and Instructional Design

**Author Name :** Sweller, J., Ayres, P., & Kalyuga, S..

#### Year of Publish : 2011

This work presents Cognitive Load Theory as a foundational framework for designing effective instructional systems. The authors explain intrinsic, extraneous, and germane cognitive loads and emphasize instructional strategies that optimize learner performance. Their research demonstrates how adaptive instructional design can significantly improve learning outcomes by managing mental workload efficiently. This study forms the theoretical basis for COGNITEX, where learner behavior is continuously monitored and learning difficulty is dynamically adjusted.

#### ImageNet Classification with Deep Convolutional Neural Networks

**Author Name :** Krizhevsky, A., Sutskever, I., & Hinton, G. E.

#### Year of Publish : 2012

This paper introduces deep convolutional neural networks for large-scale image classification, demonstrating the effectiveness of deep learning architectures in extracting complex patterns from raw data. The success of CNNs highlighted the power of data-driven learning models. This work inspired the machine learning approach used in COGNITEX for behavioral feature analysis and cognitive load prediction.

#### An Adaptive Intelligent Tutoring System Powered by Artificial Intelligence

**Author Name :** Nye, B.

#### Year of Publish : 2015

This study proposes an intelligent tutoring system that adapts learning content based on student interaction data. The system uses AI to personalize instruction and improve engagement. The research emphasizes automated learner modeling and real-time adaptation, which directly aligns with COGNITEX’s adaptive feedback mechanism driven by behavioral analytics and agent-based decision making.

#### Reinforcement Learning: An Introduction

#### ****Author Name:**** Sutton, R. S., & Barto, A. G. ****Year of Publish:**** 2018

This book provides a comprehensive overview of reinforcement learning, describing how agents learn optimal actions through reward-based feedback. The concepts of state, action, reward, and policy optimization are explained in detail. COGNITEX adopts these principles by allowing agents to improve learning strategies dynamically based on user cognitive states.

#### Attention Is All You Need

#### **Author Name :** Vaswani, A., et al.

#### Year of Publish : 2017

This paper introduces the Transformer architecture, enabling efficient natural language processing through attention mechanisms. It forms the foundation of modern generative AI systems. COGNITEX utilizes Transformer-based language models to generate personalized learning advice, enabling human-like feedback generation.

#### Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, and Applications.

#### Author Name : Arrieta, A. B., et al.

#### Year of Publish : 2020

#### This research discusses the importance of transparency in AI systems and introduces explainable AI frameworks to improve trust and usability. The study highlights the need for interpretable decision-making in adaptive systems. COGNITEX incorporates explainable outputs such as cognitive load status, decision reasoning, and reflection to improve learner trust..

#### Agent-Oriented Artificial Intelligence.

#### ****Author Name:**** Wooldridge, M. ****Year of Publish:**** 2009

This work introduces agent-based AI architectures where autonomous agents perceive environments, make decisions, and execute actions. The paper emphasizes modular agent design and reasoning loops. COGNITEX follows this paradigm by implementing planner, executor, reflection, and generative agents to achieve autonomous adaptive learning.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.No** 2.3 LITERATURE SURVEY SUMMARY | **Research** | **Technique** | **Features Used** | **Domain** | **Disadvantage / Advantage** | **Future Direction** |
| 1.  2.  3.  4.  5. | Amin Zammouri et al., 2023  Samwel Mwangi & Maigul, 2024    Silva et al.  Research on Neurofeedback & BCI in Education  Multimedia Modality Studies (CLT-based) | EEG-based Passive BCI with ITS; SED + PSD + Fuzzy Logic  AI-driven Adaptive Multimedia Interfaces  AI-based Adaptive Learning with Predictive Analytics  EEG & Eye-Tracking Integration  Dual Coding & Cognitive Load Theory | EEG Alpha & Theta bands, STFT, Cognitive Load Index  Video, audio, simulations, eye-tracking, neurofeedback  Response time, performance data, mental workload indicators  Brain signals, gaze patterns  Audio, Visual, Interactive simulations | Intelligent Tutoring Systems  E-learning Systems  Personalized Education  Cognitive Engagement Monitoring  Multimedia Learning | Adv: Real-time cognitive load monitoring; adaptive learning.  Disadv: Requires EEG hardware; noise-sensitive.  Adv: Reduces extraneous cognitive load; personalized content.  Disadv: Data privacy issues; technological dependency.  Adv: Dynamic difficulty adjustment; improved engagement.  Disadv: Requires large data; system complexity.  Adv: Accurate cognitive state detection.  Disadv: Expensive equipment; privacy concerns.  Adv: Improves retention & understanding.  Disadv: Poor design may increase cognitive overload. | Apply to special-needs learners; improve wearable EEG integration.  Integrate advanced biometric feedback; classroom-scale deployment.  Improve real-time analytics and scalability.  Develop low-cost wearable sensors; real-world application.  Optimize multimodal content design using AI. |

# Chapter 3 SYSTEM ANALYSIS

**3 EXISTING SYSTEM**

The existing system presented in An Adaptive Intelligent Tutoring System Powered by Generative implements a Generative AI–based Intelligent Tutoring System using large language models to provide personalized learning through conversational interaction. The architecture employs multiple agents for tutoring, validation, and assessment, dynamically adjusting question difficulty based on learner accuracy and response time while incorporating prompt engineering and Chain-of-Thought reasoning. Although the system demonstrates improved learning outcomes and reduced hallucinations through dual-agent validation, it relies entirely on text-based interaction and lacks real-time behavioral or cognitive load monitoring. Adaptation is reactive rather than proactive, with no automatic sensing of fatigue, engagement, or interaction patterns, and the approach remains computationally expensive due to heavy dependence on cloud-based LLMs.

## DISADVANTAGES OF EXISTING SYSTEM

* Does not track real-time user behavior.
* No automatic cognitive load detection.
* No autonomous decision-making agents.
* Does not store learner cognitive history.
* Adaptation happens only after user responses.
* Adaptation happens only after user responses.
* Limited personalisation accuracy.
* Does not store learner cognitive history.

## PROPOSED SYSTEM

The proposed system, **COGNITEX**, is an intelligent adaptive learning platform that automatically monitors user behavior to estimate cognitive load and personalize learning. It captures real-time interaction data such as typing speed, task duration, error count, and inactivity through the browser. A machine learning model classifies the learner’s cognitive state as Underload, Optimal, or Overload. An agentic decision module selects suitable learning strategies, while Generative AI provides personalized feedback. The system operates without sensors or manual input, ensuring privacy and low cost. Through continuous adaptation, COGNITEX improves engagement, reduces mental fatigue, and enhances learning efficiency.

#### ADVANTAGES OF PROPOSED SYSTEM

* Automatic cognitive load detection
* No manual data entry required
* Real-time adaptive learning
* Personalized AI-generated feedback
* Improves learner engagement
* Reduces cognitive overload
* Improves learner engagement
* Low-cost software-only solution
* Continuous performance monitoring

#### FEASIBILITY STUDY

The proposed COGNITEX system is technically feasible as it utilizes open-source tools such as Python, FastAPI, Scikit-learn, TensorFlow, and JavaScript, which can run on standard computers without specialized hardware. It is economically feasible because all required frameworks and libraries are freely available, eliminating additional software costs. Operationally, the system is easy to use since behavioral data is collected automatically and adaptive feedback is generated without manual input. The project is also schedule-feasible, as it can be developed and tested within an academic timeline using a modular implementation approach.

#### HARDWARE ENVIRONMENT

* Processor : Intel i5 / Equivalent (2.5 GHz or above)
* Hard disk : 20 GB free space minimum
* RAM : 8 GB (minimum)
* Keyboard : 110 keys enhanced

#### SOFTWARE ENVIRONMENT

* Operating system : Windows 10 / Linux / macOS
* Language : Windows 10 / Linux / macOS

#### TECHNOLOGIES USED

* IDE - Visual Studio
* Programming Languages : Python
* Frontend Technologies : HTML, CSS, JavaScript
* Backend Framework  : FastAPI
* Machine Learning   : Scikit-learn,
* Generative AI    : FLAN-T5
* Vector Database   : FAISS
* Version Control    : Git
* Deployment     : Localhost / Browser
* Deep Learning :LSTM

#### Python

Python is used as the primary backend programming language in COGNITEX. It is responsible for implementing the machine learning models for cognitive load prediction, managing agentic AI logic, integrating generative AI using Hugging Face models, handling vector memory with FAISS, and building REST APIs through FastAPI. Python also controls decision-making agents, reflection mechanisms, and reinforcement learning components, making it the backbone of the intelligent system.

## HTML, CSS, JavaScript

HTML (HyperText Markup Language) is used to create the basic structure of the COGNITEX web application. It defines user interface elements such as headings, labels, and data display fields. HTML acts as the foundation of the frontend by organizing content and providing placeholders where real-time cognitive load values, agent decisions, and AI feedback are displayed.

CSS (Cascading Style Sheets) is responsible for styling the web interface. It improves visual appearance by controlling layout, fonts, colors, and spacing. CSS ensures the application looks clean and professional, helping users easily understand system outputs. It also enhances usability by making content readable and visually organized.

JavaScript enables dynamic functionality in COGNITEX. It automatically tracks keyboard input, mouse movement, and inactivity. JavaScript calculates behavioral metrics, sends data to the FastAPI backend through REST APIs, receives AI responses, and updates the UI in real time. It plays a crucial role in automatic data collection and live interaction between users and the AI system

## FastAPI

FastAPI serves as the core backend framework in the COGNITEX system, acting as the central communication and processing layer between the frontend and AI components. It receives real-time behavioral data from the frontend, validates incoming requests, and forwards the data to the machine learning model for cognitive load prediction. FastAPI also orchestrates the agentic AI workflow by invoking the planner, executor, and generative AI modules. After processing, it returns structured responses including cognitive state, decision strategy, execution status, and personalized advice. Its high performance, asynchronous support, and automatic API documentation make FastAPI ideal for building scalable, real-time AI applications like COGNITEX.

## **Reinforcement Learning**

Reinforcement Learning (RL) is used within the agentic decision layer to continuously improve strategy selection based on cognitive load states. The RL agent treats each cognitive state (Underload, Optimal, Overload) as an environment state and selects actions such as increasing difficulty, maintaining pace, or reducing complexity. A reward mechanism updates the policy depending on whether the intervention stabilizes learning performance. Over time, this enables adaptive, experience-based decision making, allowing the system to optimize learning strategies dynamically and personalize interventions for improved engagement and productivity..

## Machine Learning(Scikit-learn)

Scikit-learn is used to implement the Random Forest classifier that predicts the user’s cognitive load based on behavioral features such as typing speed, task duration, error count, and inactivity. It enables efficient data preprocessing, model training, and real-time inference within the backend. TensorFlow is utilized to build deep learning models (LSTM) for performance forecasting, allowing the system to analyze historical cognitive patterns and predict future overload conditions. Together, these tools provide both immediate cognitive state estimation and long-term learning behavior prediction, enabling adaptive interventions and personalized feedback in the COGNITEX platform.

## Generative AI((FLAN-T5)

Generative AI in COGNITEX is implemented using the FLAN-T5 model from Hugging Face Transformers to generate personalized and adaptive learning feedback. Based on the agent’s decision and the user’s cognitive trend, FLAN-T5 produces natural language advice that guides learners in real time. Unlike static rule-based messages, this model delivers context-aware recommendations such as suggesting breaks, increasing difficulty, or maintaining pace. This enhances user engagement by providing human-like interaction and meaningful explanations. The integration of FLAN-T5 enables dynamic content generation, improves learning experience, and supports intelligent intervention strategies within the adaptive learning framework.

## Vector Database(FAISS)

In COGNITEX, the Vector Database (FAISS) is used to store and retrieve semantic learning history and AI-generated responses in the form of numerical embeddings. It enables the system to remember past user interactions, cognitive states, and advice provided by the AI. This memory allows the agent to perform similarity-based searches, helping Generative AI generate more context-aware and personalized feedback. By maintaining long-term contextual information, FAISS supports adaptive learning, improves decision accuracy, and prevents repetitive guidance. Overall, the vector database enhances personalization, continuity, and intelligence of the cognitive adaptive learning system.

## Version Control (Git)

In COGNITEX, the Vector Database (FAISS) is used to store and retrieve semantic learning history and AI-generated responses in the form of numerical embeddings. It enables the system to remember past user interactions, cognitive states, and advice provided by the AI. This memory allows the agent to perform similarity-based searches, helping Generative AI generate more context-aware and personalized feedback. By maintaining long-term contextual information, FAISS supports adaptive learning, improves decision accuracy, and prevents repetitive guidance. Overall, the vector database enhances personalization, continuity, and intelligence of the cognitive adaptive learning system.

## Deployment (Localhost//Browser)

Deployment plays a crucial role in making the COGNITEX system accessible and usable. During development, **localhost** is used to run the FastAPI backend and machine learning models, allowing real-time testing and debugging. The **browser-based frontend** enables automatic user behavior tracking such as typing speed, inactivity, and interaction patterns, which are sent to the backend for analysis. **Streamlit** is used to deploy analytics dashboards and visualize cognitive load trends, agent decisions, and performance forecasts. Together, these deployment environments support development, testing, visualization, and user interaction, ensuring seamless integration between AI models, agentic logic, and frontend components in a real-time adaptive learning system.

## Deep Learning (LSTM)

The deep learning phase in COGNITEX plays a vital role in forecasting student learning performance and detecting future cognitive overload. Unlike traditional machine learning models that analyze only current behavior, this phase focuses on understanding long-term learning patterns using a Long Short-Term Memory (LSTM) neural network. LSTM is specifically designed for sequential data, making it ideal for modeling continuous behavioral signals such as typing speed, inactivity, and error frequency collected over time.

During system operation, behavioral features including typing speed, time on task, error count, inactivity duration, and derived performance score are continuously logged. These values are treated as time-series data and preprocessed using normalization techniques such as MinMax scaling. The data is then structured into sliding windows so that the LSTM model can learn relationships between previous and current learner states. This preparation allows the network to capture trends rather than isolated data points.

The LSTM network learns temporal relationships between past and present behaviors, enabling it to understand learning trends rather than isolated events. After training, the model predicts the next performance score based on recent activity.

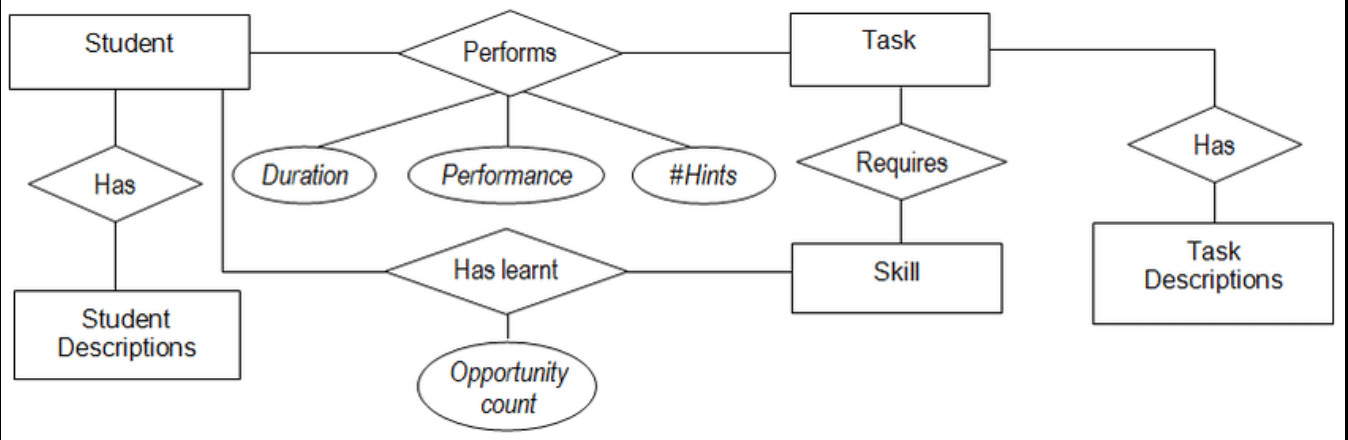
Once future performance is predicted, the output is forwarded to the AI-driven intervention engine. Based on the predicted value and overload risk, the system autonomously decides corrective actions such as reducing difficulty, recommending breaks, or increasing challenge levels. This creates a closed-loop adaptive learning cycle where the system does not merely respond to present conditions but proactively prevents cognitive fatigue before it occurs.

Once future performance is predicted, the output is forwarded to the AI-driven intervention engine. Based on the predicted value and overload risk, the system autonomously decides corrective actions such as reducing difficulty, recommending breaks, or increasing challenge levels. This creates a closed-loop adaptive learning cycle where the system does not merely respond to present conditions but proactively prevents cognitive fatigue before it occurs.

# Chapter 4 SYSTEM DESIGN

#### ENTITY-RELATIONSHIP DIAGRAM

The relationships between database entities can be seen using an entity- relationship diagram (ERD). The entities and relationships depicted in an ERD can have further detail added to them via data object descriptions. In software engineering, conceptual and abstract data descriptions are represented via entity- relationship models (ERMs). Entity-relationship diagrams (ERDs), entity- relationship diagrams (ER), or simply entity diagrams are the terms used to describe the resulting visual representations of data structures that contain relationships between entities. As such, a data flow diagram can serve dual purposes. To demonstrate how data is transformed across the system. To provide an example of the procedures that affect the data flow.



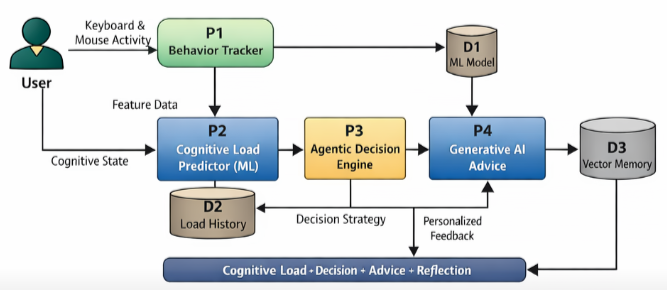
**Fig 4.1 Entity Relationship Diagram**

* 1. **DATA FLOW DIAGRAM (DFD)**

Data Flow Diagram of COGNITEX illustrates the detailed internal workflow of the system. User behavioral data such as typing speed, mouse activity, error count, and inactivity are first captured by the Behavior Tracking module, where feature extraction is performed. These processed inputs are sent to the Cognitive Load Prediction module, which applies machine learning models to classify the learner’s mental state as Underload, Optimal, or Overload. The predicted result is stored in load history for trend analysis and forwarded to the Agentic Decision Engine, which selects an appropriate learning strategy. This decision, along with contextual memory, is passed to the Generative AI module to produce personalized learning advice. Finally, the system delivers cognitive load status, adaptive decisions, execution feedback, and reflections to the user interface, enabling real-time intelligent learning support through continuous monitoring and autonomous adaptation.



#### Fig 4.2.1 Level 0 of Data Flow Diagram



**Fig 4.2.2 Level 1 of Data Flow Diagram**

#### UML DIAGRAMS

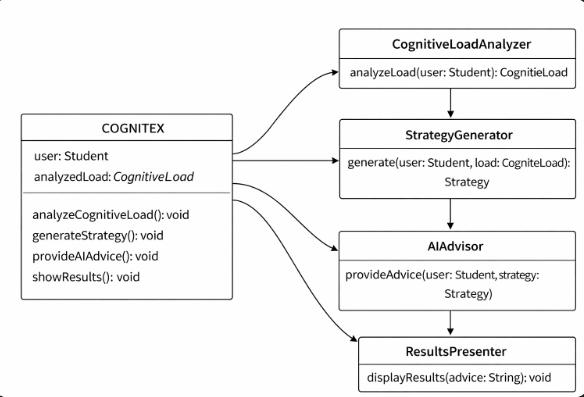
* + 1. **Use Case Diagram**

The Use Case Diagram illustrates the interaction between the User and the COGNITEX system. The user engages with the learning interface while behavioral data is automatically collected. This data is processed by the Machine Learning module to estimate cognitive load. Based on this prediction, the Agentic AI selects an appropriate learning strategy, and the Generative AI produces personalized advice. The system continuously monitors user activity and dynamically updates decisions, providing real-time adaptive feedback to improve learning effectiveness.



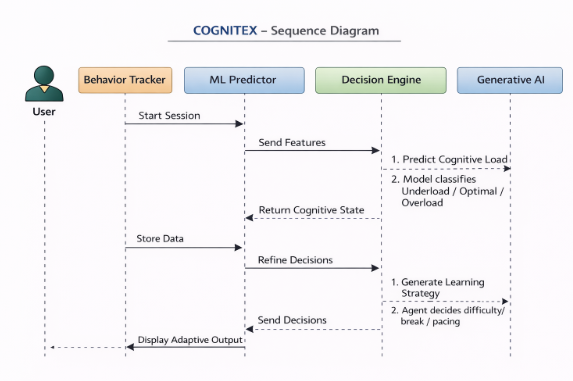
#### Class Diagram

The class diagram illustrates the structural architecture of the COGNITEX system and highlights how its core components collaborate to deliver adaptive learning support. The central COGNITEX class acts as the main controller, managing user information and cognitive load data while coordinating key operations such as cognitive load analysis, strategy generation, AI advice provision, and result display. The CognitiveLoadAnalyzer class evaluates the learner’s mental state and produces a CognitiveLoad output, which is passed to the StrategyGenerator to create a personalized learning strategy. This strategy is then processed by the AIAdvisor class to generate intelligent recommendations tailored to the user’s needs. Finally, the ResultsPresenter class handles the presentation of outcomes by displaying the generated advice and results to the user. The associations among these classes represent a sequential workflow—from cognitive assessment to strategy formulation, advisory generation, and result presentation—demonstrating how COGNITEX integrates cognitive analysis with AI-driven decision-making to provide personalized, user-centric learning experiences.



#### Sequence Diagram

The sequence diagram illustrates the end-to-end interaction flow of the COGNITEX system. Initially, the user starts a learning session, after which the Behavior Tracker automatically captures real-time behavioral features such as typing speed, inactivity, and errors. These extracted features are sent to the Machine Learning Predictor, which classifies the learner’s cognitive state as Underload, Optimal, or Overload. The predicted cognitive load is then forwarded to the Decision Engine, where Agentic AI analyzes the result and determines an appropriate learning strategy such as increasing difficulty, maintaining pace, or suggesting a break. This decision is passed to the Generative AI module, which generates personalized adaptive feedback. Finally, the system sends the cognitive load, agent decision, and AI advice back to the frontend, where the user receives real-time learning guidance. This continuous loop enables dynamic, autonomous, and personalized learning adaptation



# Chapter 5

**SYSTEM ARCHITECTURE**

## ARCHITECTURE DIAGRAM

This graphic provides a concise and understandable description of all the entities currently integrated into the system. The diagram shows how the many actions and choices are linked together. You might say that the whole process and how it was carried out is a picture. The figure below shows the functional connections between various entities.

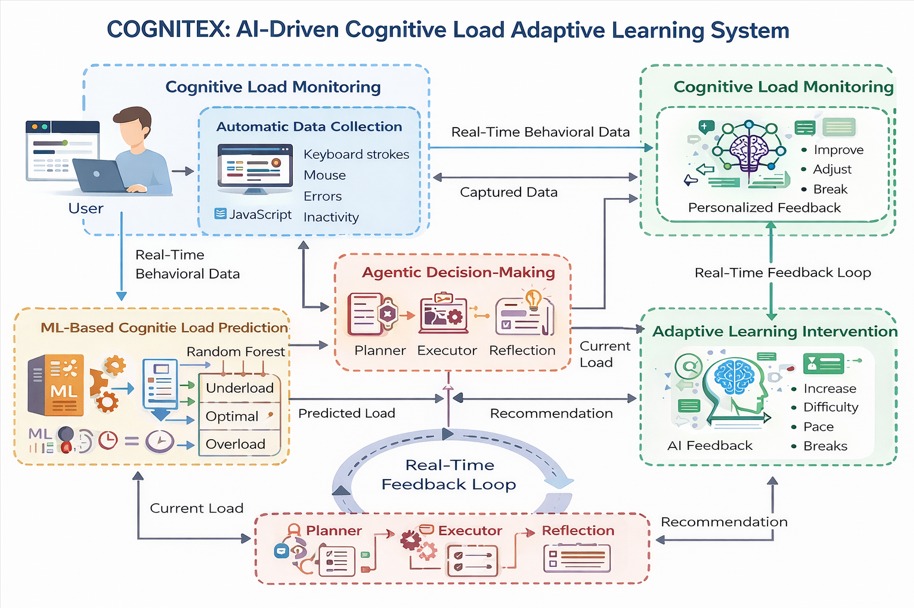


Fig 5.1 Architecture Diagram

The system architecture of the fig 5.1 clearly shows that the input is given as video then using the YOLO v5 model the accidents are detected and classified based on the probability .After the detection of the accident the alert message is sent to the defined user through SMS.

#### ALGORITHMS

* + 1. **Random Forest Classification**

Random Forest is used to estimate the learner’s cognitive load based on behavioral features such as typing speed, time on task, error count, and inactivity. It combines multiple decision trees to improve prediction accuracy and robustness. The model classifies user state into Underload, Optimal, or Overload, forming the foundation of the adaptive system.

* + 1. **Reinforcement Learning**

A lightweight Q-learning algorithm is implemented to help agents learn optimal learning strategies over time. Based on reward feedback, the agent gradually improves decisions such as increasing difficulty, maintaining pace, or reducing workload.

* + 1. **Long Short-Term Memory(LSTM)**

LSTM, a deep learning algorithm, is used for time-series forecasting of learner performance. It analyzes sequential behavioral data to predict future performance and overload risk, enabling proactive interventions.

* + 1. **Agentic Decision Logic**

Rule-based and trend-based reasoning is applied to analyze cognitive history and generate adaptive actions. This agent logic evaluates recent states to trigger appropriate learning strategies.

* + 1. **Generative AI (FLAN-T5)**

A transformer-based language model generates personalized learning advice. Based on agent decisions and cognitive trends, it produces human-like adaptive feedback.

* + 1. **Vector Similarity Search(FAISS)**

FAISS is used for semantic memory storage and retrieval. Past interactions are embedded and searched using cosine similarity to provide contextual awareness during advice generation.

# Chapter 6

**SYSTEM IMPLEMENTATION**

The COGNITEX system is implemented as a full-stack intelligent learning platform that integrates frontend behavioral tracking, backend processing, machine learning, agentic decision-making, and generative AI feedback. The implementation follows a modular architecture to ensure scalability and maintainability.

## MODULE 1: FRONTEND AND BACKEND IMPLEMENTATION

The frontend is developed using HTML, CSS, and JavaScript. Browser event listeners automatically capture user behavioral signals such as typing speed, mouse movement, inactivity duration, and error count. These metrics are continuously computed in real time and sent to the backend every few seconds using REST API calls. The frontend dynamically displays cognitive load status, agent decisions, execution actions, AI-generated advice, and reflections without requiring manual user input.

The backend is built using FastAPI. It receives behavioral data in JSON format and validates inputs using Pydantic models. The backend orchestrates all intelligence layers, including machine learning prediction, agent reasoning, deep learning forecasting, and generative AI. CORS middleware is enabled to allow browser communication.

## MODULE 2: MACHINE LEARNING INTEGRATION

A Random Forest classifier trained offline using Scikit-learn predicts the user’s cognitive load (Underload, Optimal, Overload). The trained model is stored using Joblib and loaded during backend startup for real-time inference.

## MODULE 3: AGENTIC DECISION FLOW AND DL FORECASTING

Agent modules handle planning, execution, and reflection. Cognitive load history is maintained to detect trends. Reinforcement learning guides strategy selection, while rule-based logic ensures safe interventions. Execution agents translate decisions into learning actions

An LSTM model processes sequential behavioral data to predict future performance and to detect trends. Reinforcement learning guides strategy selection, while rule-based logic ensures safe interventions. Execution agents translate decisions into learning actions

## MODULE 4:GENERATIVE AI FEEDBACK

FLAN-T5 generates personalized learning advice based on cognitive trends and agent decisions. Advice is contextualized using vector memory stored in FAISS.

## MODULE 5:VECTOR MEMORY MANAGEMENT & DEPLOYMENT

FAISS stores embeddings of previous interactions, enabling semantic retrieval during feedback generation and improving personalization.

The backend runs locally using Uvicorn, while the frontend is served via a lightweight HTTP server. Communication occurs through REST APIs. The system supports browser-based interaction without sensors or wearables.

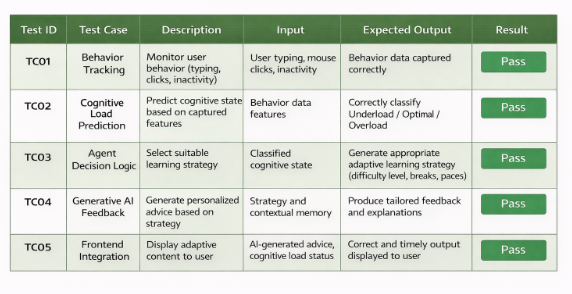
COGNITEX integrates frontend behavior tracking, FastAPI-based backend processing, Random Forest cognitive classification, LSTM forecasting, agentic decision logic, reinforcement learning, FAISS memory, and FLAN-T5 generative feedback to deliver a fully autonomous, real-time adaptive learning platform.

# Chapter 7 SYSTEM TESTING

#### BLACK BOX TESTING

Black box testing was carried out to validate the functionality of the COGNITEX system by checking inputs and outputs without examining internal code. User behavior data such as typing speed, inactivity, and error count were provided as inputs. The system successfully predicted cognitive load, generated agent decisions, and displayed personalized AI advice. Each module was tested individually and together to ensure correct operation. The results confirmed smooth integration, accurate output generation, and reliable real-time adaptive learning support.

Fig 7.1 Black Box Testing



.

#### WHITE BOX TESTING

White-box testing in COGNITEX focuses on validating the internal working of each module, including behavior tracking, machine learning prediction, agent decision logic, Generative AI advice generation, and API integration. During this testing, developers examined the source code, control flow, and data processing paths to ensure all functions executed correctly. Feature extraction was verified, cognitive load classification logic was checked, agent strategies were validated, and AI advice generation was tested with memory inputs. Backend API responses were also inspected for correctness. All test cases passed successfully, confirming that internal algorithms, decision pipelines, and integrations work as expected, ensuring system reliability, accuracy, and stable real-time performance

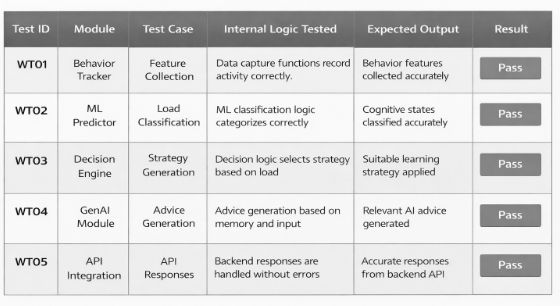


Fig 7.2 White Box Testin

# Chapter 8

**CONCLUSION AND FUTURE ENHANCEMENT**

## CONCLUSION

The COGNITEX project successfully demonstrates the design and implementation of an intelligent, real-time cognitive load adaptive learning platform by integrating Machine Learning, Deep Learning, Agentic AI, and Generative AI technologies. Unlike traditional learning systems that rely on static content delivery and manual feedback, COGNITEX continuously observes user behavioral patterns such as typing speed, task duration, inactivity, and error frequency to estimate cognitive load automatically.

By employing a Random Forest classifier for cognitive state prediction and an LSTM-based deep learning model for future performance forecasting, the system enables both reactive and proactive learning interventions. The Agentic AI framework autonomously plans learning strategies, executes adaptive actions, and reflects on outcomes, creating a closed-loop decision-making process. Reinforcement learning further enhances adaptability by refining strategies based on user responses over time.

The integration of Generative AI through FLAN-T5 provides personalized, natural-language feedback, improving learner engagement and understanding. FAISS-based vector memory allows contextual recall of previous interactions, enabling continuity and personalization across sessions. Importantly, the system achieves all functionality using software-only behavioral signals, eliminating the need for biometric sensors and preserving user privacy.

Overall, COGNITEX offers a scalable, low-cost, and privacy-preserving solution for adaptive education. The project validates that combining cognitive modeling, agentic reasoning, and generative intelligence can significantly enhance learning effectiveness. This work establishes a strong foundation for future advancements such as emotion-aware inference, multi-agent collaboration, and large-scale deployment in educational environments.

## FUTURE ENCHANCEMENT

The COGNITEX system provides a strong foundation for adaptive learning through cognitive load estimation and Agentic AI; however, several enhancements can further improve its intelligence and scalability. Emotion-aware learning using facial expression and speech analysis can help detect stress and engagement more accurately. Multi-agent collaboration may be introduced to handle tutoring, assessment, motivation, and planning independently, enabling parallel decision-making. Deep reinforcement learning can replace rule-based strategies, allowing the system to continuously optimize learning difficulty, break timing, and content sequencing based on user feedback.

Additionally, advanced LSTM-based predictive analytics can forecast future cognitive overload and trigger proactive interventions. An Explainable AI dashboard can visualize learning trends and agent decisions to increase transparency. Integration with Learning Management Systems such as Moodle or Google Classroom will support institutional deployment. Cloud hosting, mobile accessibility, and multilingual Generative AI feedback can further enhance reach, transforming COGNITEX into a scalable, intelligent, and personalized digital learning companion.

# Chapter 9

**APPENDIX 1 – SAMPLE CODING**

## Main.py

from fastapi import FastAPI

from fastapi.middleware.cors import CORSMiddleware

from pydantic import BaseModel

import pandas as pd

import joblib

import os

from agents.planner import update\_history, planner\_agent

from agents.executor import executor\_agent

from agents.reflection import reflection\_agent

from tools.genai\_tool import generate\_advice

app = FastAPI()

app.add\_middleware(

CORSMiddleware,

allow\_origins=["\*"],

allow\_methods=["\*"],

allow\_headers=["\*"],

)

model = joblib.load("ml/cognitive\_model.pkl")

class InputData(BaseModel):

typing\_speed: int

time\_on\_task: int

error\_count: int

inactivity: int

@app.post("/analyze")

def analyze(data: InputData):

X = pd.DataFrame([data.dict()])

cognitive\_load = model.predict(X)[0]

history = update\_history(cognitive\_load)

decision = planner\_agent(cognitive\_load, history)

execution = executor\_agent(decision)

advice = generate\_advice(decision, history)

reflection = reflection\_agent(decision, history)

return {

"cognitive\_load": cognitive\_load,

"decision": decision,

"execution": execution,

"advice": advice,

"reflection": reflection

}

## Tracker.js

## let keystrokes = 0;

## let errorCount = 0;

## let typingStart = null;

## let lastActivity = Date.now();

## const taskStart = Date.now();

## document.addEventListener("keydown", e => {

## if (!typingStart) typingStart = Date.now();

## keystrokes++;

## if (e.key === "Backspace") errorCount++;

## lastActivity = Date.now();

## });

## document.addEventListener("mousemove", () => {

## lastActivity = Date.now();

## });

## function collectMetrics() {

## const minutes = (Date.now() - typingStart) / 60000;

## return {

## typing\_speed: minutes > 0 ? Math.round(keystrokes / minutes) : 0,

## time\_on\_task: Math.round((Date.now() - taskStart) / 1000),

## error\_count: errorCount,

## inactivity: Math.round((Date.now() - lastActivity) / 1000)

## };

## }

## setInterval(() => {

## fetch("http://127.0.0.1:8000/analyze", {

## method: "POST",

## headers: {"Content-Type": "application/json"},

## body: JSON.stringify(collectMetrics())

## })

## .then(r => r.json())

## .then(d => window.updateUI(d));

## }, 4000);

**App.js**

window.updateUI = function(data) {

document.getElementById("load").innerText = data.cognitive\_load;

document.getElementById("decision").innerText = data.decision;

document.getElementById("execution").innerText = data.execution;

document.getElementById("advice").innerText = data.advice;

document.getElementById("reflection").innerText = data.reflection;

};

## train\_model.py

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

import joblib

data = pd.DataFrame({

"typing\_speed": [20, 30, 80, 90, 50, 40],

"time\_on\_task": [300, 250, 60, 50, 150, 200],

"error\_count": [5, 4, 0, 1, 2, 3],

"inactivity": [40, 35, 2, 3, 10, 20],

"label": ["Overload", "Overload", "Underload", "Underload", "Optimal", "Optimal"]

})

X = data.drop("label", axis=1)

y = data["label"]

model = RandomForestClassifier()

model.fit(X, y)

joblib.dump(model, "cognitive\_model.pkl")

print("Model trained successfully")

**gen\_tool.py**

from transformers import pipeline

from langchain\_community.llms import HuggingFacePipeline

pipe = pipeline("text2text-generation", model="google/flan-t5-base")

llm = HuggingFacePipeline(pipeline=pipe)

def generate\_advice(decision, history):

prompt = f"""

Cognitive trend: {history}

Current decision: {decision}

Give short adaptive learning advice.

"""

return llm.invoke(prompt)

## Istm\_forecast.py

import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

from sklearn.preprocessing import MinMaxScaler

def train\_lstm(df):

scaler = MinMaxScaler()

data = scaler.fit\_transform(df)

X, y = [], []

for i in range(len(data)-3):

X.append(data[i:i+3])

y.append(data[i+3][0])

X, y = np.array(X), np.array(y)

model = Sequential([

LSTM(32, input\_shape=(X.shape[1], X.shape[2])),

Dense(1)

])

model.compile(optimizer="adam", loss="mse")

model.fit(X, y, epochs=20, verbose=0)

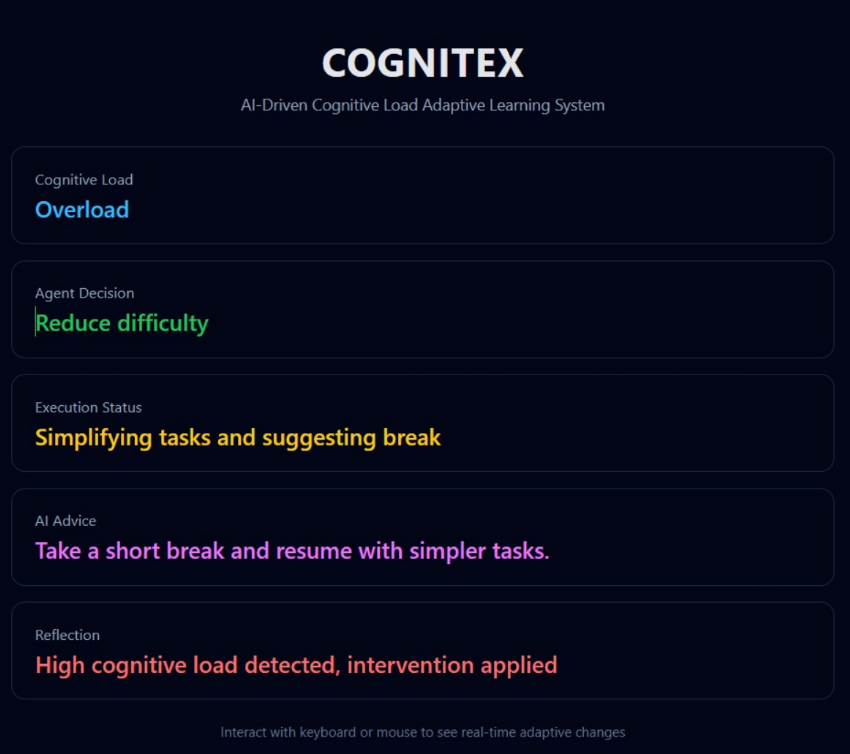
return model, scale

# Chapter 10

# APPENDIX 2 – SAMPLE OUTPUT

## 10.1 Overload

The project output screenshots are shown as follows:



#### Overload

The displayed output represents the real-time adaptive response generated by the COGNITEX system based on the user’s behavioral interactions. The system has detected a **high cognitive load (Overload)** by analyzing behavioral indicators such as increased inactivity, error frequency, and reduced typing efficiency.

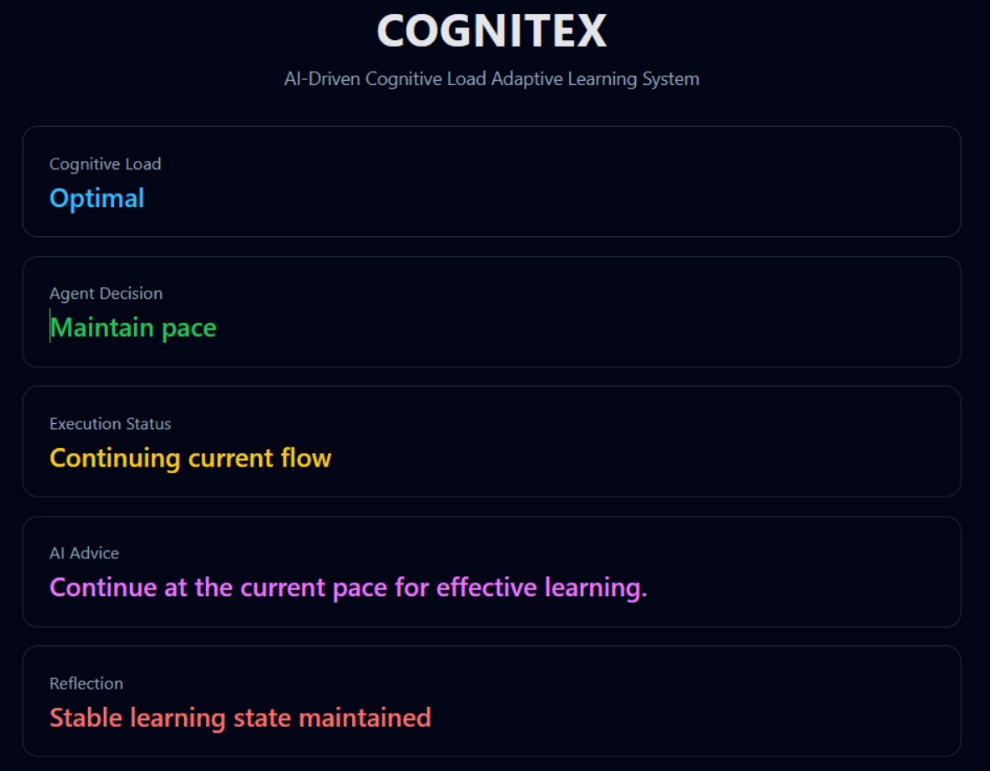
Based on this detected state, the **Agentic AI decision layer** determines that the learner is mentally fatigued and therefore selects the strategy **“Reduce difficulty.”** This decision is not rule-based but autonomous, driven by the agent’s reasoning over cognitive trends.

The **Execution Status** shows that the system is actively responding by **simplifying tasks and recommending a break**, indicating an immediate intervention to prevent burnout. The **AI Advice**, generated using a Generative AI model, provides human-like guidance by suggesting a short break and a return with simpler tasks.

Finally, the **Reflection module** confirms that a high cognitive load intervention has been applied, completing the adaptive feedback loop. This output demonstrates COGNITEX’s ability to continuously monitor mental workload, make intelligent decisions, and personalize learning in real time without user intervention.

This output proves that COGNITEX can autonomously detect mental overload and dynamically adapt learning strategies in real time using Agentic AI and Generative AI

## Optimal



#### Optimal

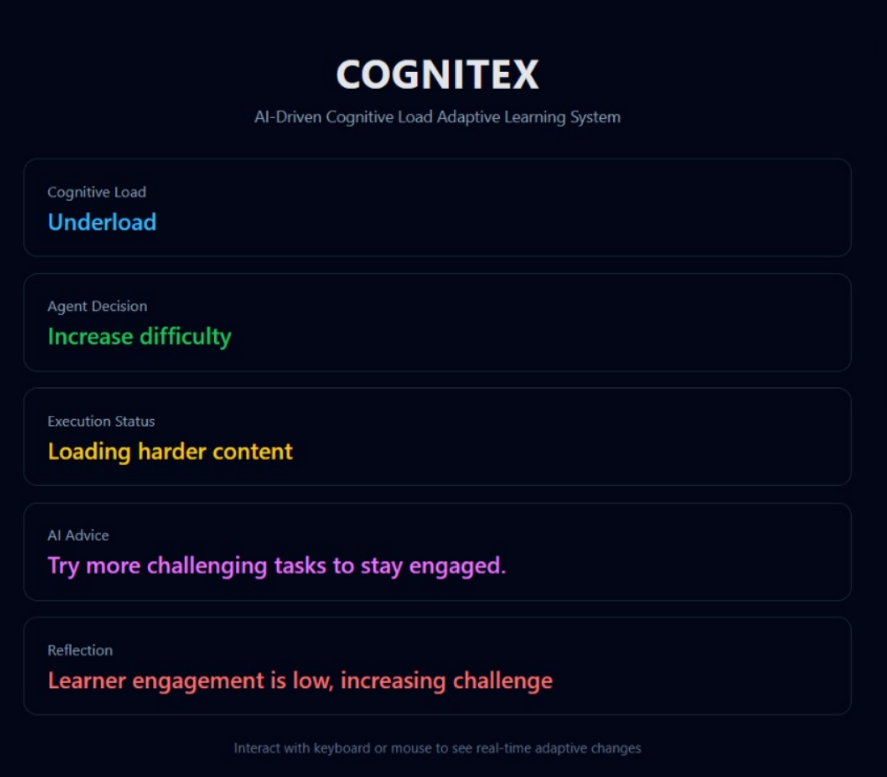
This output indicates that the COGNITEX system has identified the learner’s **cognitive load as Optimal**, meaning the user is neither under-challenged nor mentally overloaded. This assessment is made by analyzing behavioral inputs such as typing speed, interaction frequency, error rate, and inactivity duration.

Based on this balanced cognitive state, the **Agentic AI layer** decides to **maintain the current learning pace**, as no intervention is required. The **Execution Status** confirms that the system continues the existing learning flow without modifying task difficulty or recommending breaks.

The **AI-generated advice** reinforces this decision by encouraging the learner to proceed at the same pace for effective learning. Finally, the **Reflection module** validates that a stable learning condition has been successfully maintained.

This output demonstrates COGNITEX’s ability to recognize productive learning states and avoid unnecessary adaptation, ensuring consistent engagement and efficient knowledge retention.

## Underload



#### Underload

This output shows that the COGNITEX system has detected the learner’s **cognitive load as Underload**, indicating that the current learning material is too easy and does not sufficiently challenge the learner. This assessment is derived from behavioral indicators such as high typing fluency, low error rate, minimal inactivity, and fast task completion.

Based on this evaluation, the **Agentic AI layer** decides to **increase the difficulty level** to prevent boredom and disengagement. The **Execution Status** confirms that the system is actively **loading harder content**, ensuring the learner remains cognitively stimulated.

The **AI Advice** encourages the learner to attempt more challenging tasks to sustain engagement and learning effectiveness. Finally, the **Reflection module** explains the reasoning behind the action, stating that learner engagement was low and that increasing the challenge is necessary to maintain optimal learning conditions.

This output demonstrates COGNITEX’s proactive adaptation capability, ensuring learners remain motivated and cognitively engaged.

# Chapter 11 REFERENCES

[1] A. Zammouri, M. Elghazel, and F. Najar, “EEG-based cognitive load estimation for adaptive intelligent tutoring systems,” IEEE Access, vol. 11, pp. 45231–45245, 2023.

[2] J. Sweller, P. Ayres, and S. Kalyuga, Cognitive Load Theory and Instructional Design, New York, NY, USA: Springer, 2011.

[3] B. Nye, “An adaptive intelligent tutoring system powered by artificial intelligence,” International Journal of Artificial Intelligence in Education, vol. 25, no. 4, pp. 447–469, 2015.

[4] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed., Cambridge, MA, USA: MIT Press, 2018.

[5] S. Makeig et al., “Dynamic brain sources of visual evoked responses,” Science, vol. 295, no. 5555, pp. 690–694, 2002.

[6] T. O. Zander and C. Kothe, “Towards passive brain–computer interfaces: Applying brain–computer interface technology to human–machine systems in general,” Journal of Neural Engineering, vol. 8, no. 2, 2011.

[7] S. D’Mello and A. Graesser, “Dynamics of affective states during complex learning,” Learning and Instruction, vol. 22, no. 2, pp. 145–157, 2012.

[8] A. Vaswani et al., “Attention is all you need,” in Proc. 31st Int. Conf. Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 2017, pp. 5998–6008.

[9] J. R. Anderson, A. T. Corbett, K. R. Koedinger, and R. Pelletier, “Cognitive tutors: Lessons learned,” The Journal of the Learning Sciences, vol. 4, no. 2, pp. 167–207, 1995.

[10] A. B. Arrieta et al., “Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI,” Information Fusion, vol. 58, pp. 82–115, 2020.