**A PROJECT REPORT**

**on**

### ****TRUSTVAULT: A Federated AI Assistant****

**Submitted to:**

**KIIT Deemed to be University**

**In Partial Fulfilment of the Requirement for the Award of**

**BACHELOR’S DEGREE IN**

**COMPUTER SCIENCE AND ENGINEERING**

**BY**

**Indranil Bhowmik [22051770]**

**Aditya Mishra [22051828]**

**Ashish Sinha [22051846]**

**Ayush Aryan [22051847]**

**Prabhupada Samantaray [22052483]**

**Pragaman Kumar Anurag [22052484]**

**UNDER THE GUIDANCE OF**

**Dr. Ajit Kumar Pasayat**

**Assistant Professor,**

**School of Computer Science and Engineering**

**Kalinga Institute of Industrial Technology,Bhubaneswar**

****

**SCHOOL OF COMPUTER ENGINEERING**

**KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY**

**BHUBANESWAR, ODISHA - 751024**

**April 2025**

**CERTIFICATE**

****

**KIIT Deemed to be University**

**School of Computer Engineering**

**Bhubaneswar, ODISHA 751024**

**This is certify that the project entitled**

****TRUSTVAULT: A Federated AI Assistant****

**submitted by**

**Indranil Bhowmik [22051770]**

**Aditya Mishra [22051828]**

**Ashish Sinha [22051846]**

**Ayush Aryan [22051847]**

**Prabhupada Samantaray [22052483]**

**Pragaman Kumar Anurag [22052484]**

**is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Sci-ence & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2024-2025, under our guidance.**

**Date: 08/04/2025**

**Dr. Ajit Kumar Pasayat**

**Assistant Professor,**

**School of Computer Science and Engineering,**

**Kalinga Institute of Industrial Technology,**

**Bhubaneswar**

**ACKNOWLEDGEMENTS**

We are profoundly grateful to **Dr. Ajit Kumar Pasayat** of Affiliation for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

**Indranil Bhowmik [22051770]**

**Aditya Mishra [22051828]**

**Ashish Sinha [22051846]**

**Ayush Aryan [22051847]**

**Prabhupada Samantaray [22052483]**

**Pragaman Kumar Anurag [22052484]**

**ABSTRACT**

Recent advances in NLP have brought powerful language models like ChatGPT to the forefront. However, these models often depend on centralized servers, raising serious concerns about user privacy and data security. To address this, **TrustVault** introduces a privacy-first approach to NLP by enabling users to run and personalize models locally, without sending sensitive data to external servers.

Built on a simulated **Federated Learning** framework, TrustVault allows multiple clients to train models on their private data and share only encrypted updates with a central server. Incorporating **Differential Privacy**, the system ensures that no personal information is exposed during training.

TrustVault supports essential NLP tasks such as **Text Generation**, **Summarization**, and **Question Answering** using efficient transformer models like Phi-2, T5-small, and DistilBERT. Developed in Python with a user-friendly **Streamlit** interface, it is optimized for low resource usage and easy customization.

By combining federated learning, cryptographic techniques, and local model deployment, TrustVault offers a secure, scalable, and user-controlled alternative to conventional LLMs — paving the way for ethical and privacy-conscious AI applications.

**CONTENT**

**1 Introduction**..............................................….……………...........................…….1

**2 Literature Review / Existing System**....................…......………………………..2  
 2.1 **Large Language Models (LLMs)**......................…............…..................…......2  
 2.2 **Federated Learning** ………………………………….……….………………2

2.3 Differential Privacy…………………………………………..……………….2

2.4 Homomorphic Encryption (Future Integration)………………..……………..2

2.5 Hugging Face Transformers…………………………………..………………2

#### **2.6 Natural Language Processing (NLP) Tasks…………………..………………3**

#### **2.7 Existing Systems and Their Limitations……………………..……………….3**

### 3 Methodology / Proposed System ...............................……….…………….….4  3.1 Project Planning..........................................................…….…………………5  3.2 Project Analysis.............................……………………....…...........................5  3.3 System Design ...........................………………………….………….............6

#### 3.3.1 Design Constraints………………………………….……………………6

#### 3.3.2 System Architecture / Block Diagram…………….……………………..7

### 4 Implementation ....................................………………………..........................8  4.1 Methodology ...............................……………………….................................8  4.2 Testing / Verification Plan ..........…………….................................................9  4.3 Result Analysis / Screenshots......…..……....................… ...........................10

### 4.4 Quality Assurance……………………..…………………….………………12

### 5 Conclusion and Future Scope ....................……………….…….....................12  5.1 Conclusion.............………………………………………………..................12  5.2 Future Scope .......…………………………………………………................12

**6 References / Bibliography** .........................................………… ………….........13

**7 Individual Contribution** ............................................………… ………….........14

Plagiarism Report ............................................................… …………….....15

### ****Chapter 1: Introduction****

In recent years, large language models (LLMs) such as ChatGPT have revolutionized human-computer interaction by offering advanced capabilities in natural language generation, question answering, and summarization. These models have become integral to a wide range of applications including education, productivity tools, and virtual assistance. However, despite their growing utility, a fundamental concern continues to surround the centralized nature of these models — particularly in terms of user data privacy and security.

Most current LLMs operate on cloud-based servers, requiring users to send their queries and sensitive information to remote data centers for processing. This architecture raises serious privacy concerns, as user data can potentially be intercepted, analyzed, or even stored without explicit consent. Additionally, the reliance on massive centralized datasets often leads to generic outputs, with little personalization or contextual understanding of the individual user. The lack of a privacy-respecting and customizable alternative has left a critical gap in the development of user-centric AI applications.

To address these issues, this project proposes the development of **TrustVault**, a privacy-first large language model system designed to run locally or in a federated environment. By employing a simulated **Federated Learning** approach, the system enables clients to train models on their own devices using personal data while sending only encrypted updates to a central server. This prevents raw data from leaving the user’s device. The integration of **Differential Privacy** techniques further enhances data protection by obfuscating individual data points during the model update process.

TrustVault also includes key NLP functionalities such as **Text Generation**, **Summarization**, and **Contextual Q&A**, all powered by lightweight transformer models such as Microsoft’s Phi-2, T5-small, and DistilBERT. These models have been carefully selected to provide accurate, context-aware responses while maintaining computational efficiency — making them suitable for edge devices or platforms with limited resources.

The structure of this report reflects the logical progression of the project. Following this introduction, Chapter 2 provides a review of existing systems and techniques, highlighting their limitations. Chapter 3 presents the proposed methodology, including system architecture, workflow, and the technologies employed. Chapter 4 discusses the actual implementation of TrustVault, supported by screenshots and code snippets. Chapter 5 evaluates the system’s performance through results and analysis. Finally, Chapter 6 concludes the report with a summary of accomplishments and suggestions for future enhancements.

By introducing a decentralized and privacy-centric approach to language modeling, TrustVault aims to pave the way for more secure, transparent, and personalized NLP systems.

### ****Chapter 2: Literature Review / Existing System****

This chapter presents the foundational concepts, tools, and techniques that underpin the design and development of the TrustVault system. It also explores existing systems and approaches in the field of natural language processing, federated learning, and privacy-enhancing technologies, highlighting their strengths and limitations. The aim is to build a conceptual framework that supports the proposed system and justifies its innovation.

#### ****2.1 Large Language Models (LLMs)****

Large Language Models are AI systems trained on massive text datasets to understand and generate human-like text. They rely on transformer architectures, which enable them to manage long-range dependencies in language. Popular LLMs include GPT, BERT, and their derivatives. TrustVault uses lightweight versions like **Phi-2**, **T5-small**, and **DistilBERT** to maintain efficiency while delivering high performance on language tasks.

#### ****2.2 Federated Learning****

Federated Learning is a decentralized machine learning approach where individual clients train models locally and share only model updates (not raw data) with a central server. This technique preserves user privacy and supports personalized model updates. TrustVault implements a **simulated federated learning** setup to reflect this approach and lay the groundwork for full decentralization in the future.

#### ****2.3 Differential Privacy****

Differential Privacy is a mathematical framework that ensures individual data points cannot be inferred from the outputs of a dataset or model. It adds statistical noise to model updates, making it difficult to reverse-engineer sensitive user data. TrustVault uses this technique to safeguard user inputs during the model training phase, especially in federated environments.

#### ****2.4 Homomorphic Encryption (Future Integration)****

Homomorphic Encryption allows computations to be performed on encrypted data without needing to decrypt it. While not fully implemented in the current prototype, TrustVault is designed to support such encryption techniques in future versions to enhance privacy during training and inference.

#### ****2.5 Hugging Face Transformers****

The Hugging Face Transformers library is an open-source toolkit offering pre-trained models and easy-to-use APIs for NLP tasks. It powers all core models in TrustVault:

* **Phi-2** for context-aware text generation
* **T5-small** for text summarization
* **DistilBERT** for extractive question answering

These models are chosen for their speed, efficiency, and suitability for local or resource-constrained environments.

#### ****2.6 Natural Language Processing (NLP) Tasks****

TrustVault supports three major NLP tasks:

* **Text Generation**: Producing coherent, human-like responses based on prompts
* **Summarization**: Condensing long texts into concise summaries
* **Question Answering**: Answering user queries based on given context using transformer-based models

#### ****2.7 Existing Systems and Their Limitations****

* Popular models like ChatGPT and Bard provide exceptional performance but rely heavily on centralized servers. These systems raise concerns related to:
* Data security
* Lack of transparency in data usage
* Inability to personalize model responses effectively  
  Moreover, users have no control over how their data is stored, used, or shared, making them vulnerable to data breaches or misuse.

TrustVault fills this gap by providing a **user-owned, privacy-first** alternative that runs locally and emphasizes secure, decentralized learning.

## ****Chapter 3: Problem Statement / Requirement Specifications****

### ****Problem Statement****

Despite the remarkable advancements in natural language processing through centralized large language models (LLMs) such as ChatGPT, Bard, and others, critical concerns regarding **data security**, **user privacy**, and **control over personal information** persist. These models typically function on cloud-based infrastructure, where user inputs may be logged, processed, or stored on third-party servers—posing serious risks to user confidentiality, especially in sensitive domains like healthcare, education, and legal services.

**TrustVault** is proposed as a **privacy-first, personalized LLM system** that runs locally and uses **simulated federated learning and privacy-preserving techniques** to ensure that all user data remains on the device. The project enables secure, efficient, and customized language understanding and generation while eliminating dependence on cloud platforms.

The solution will allow users to perform tasks such as **Text Generation**, **Summarization**, and **Question Answering**, using lightweight transformer models with privacy controls, while incorporating future-proofing through support for encryption and decentralized learning.

### ****Software Requirement Specification (IEEE Format)****

| **SRS Component** | **Details** |
| --- | --- |
| **Product Name** | TrustVault |
| **Product Category** | Secure NLP Tool / Privacy-Focused LLM |
| **Target Users** | Students, Researchers, Privacy Advocates, Local Users |
| **Functional Requirements** | Text Generation, Summarization, Q&A, Local Auth, Privacy Toggle |
| **Non-Functional Requirements** | Secure Login, Fast Response Time, Local Execution, Privacy Compliance |
| **System Features** | Simulated Federated Learning, Differential Privacy, Streamlit UI |
| **External Interfaces** | File Upload Interface, Optional Internet for Ngrok Tunnel |
| **User Interface** | Streamlit Web App |
| **Hardware Interface** | Laptop/PC with minimum 4GB RAM (tested in Colab for feasibility) |
| **Software Interface** | Python 3.10+, HuggingFace Transformers, Streamlit, Opacus, PyNgrok (optional) |
| **Performance Requirements** | Fast response under 3s, lightweight memory footprint |
| **Design Constraints** | Offline/limited-resource environment support |
| **Security Requirements** | Local authentication, privacy toggles, simulated encryption |

### ****3.1 Project Planning****

The planning phase for TrustVault consists of the following major steps:

**Define core NLP features**:

* Text Generation (Phi-2)
* Text Summarization (T5-small)
* Question Answering (DistilBERT)

**Implement frontend interface** using Streamlit for user interactions.

**Simulate Federated Learning**:

* Train locally on dummy user data
* Send encrypted updates (simulated)
* Apply Differential Privacy (Opacus)

**Design Privacy Toggles** to enable/disable local training and federated sync.

**Add local user authentication** (username/password).

**Testing** the system on Colab (free-tier) and on local machines.

**Optional Extensions**:

* File download/upload
* JWT Authentication
* Docker containerization
* Drive integration

### ****3.2 Project Analysis****

A detailed requirement analysis was conducted to ensure feasibility and practicality of the proposed solution. During the analysis phase:

Ambiguities in federated learning were resolved by **simulating updates** rather than full client-server setups.

Performance trade-offs were addressed by choosing **lightweight models** to ensure usability in low-resource environments.

Threat vectors were analyzed, and **Differential Privacy** was incorporated to safeguard model updates.

User needs were considered, such as **downloadable results**, **file-based inputs**, and **simple UI**, to improve accessibility.

This analysis verified that the system can meet its objectives **without violating user privacy**, even in constrained environments.

### ****3.3 System Design****

#### ****3.3.1 Design Constraints****

**Software Requirements**:

* Python 3.10+
* Hugging Face Transformers
* Streamlit
* Opacus for Differential Privacy
* PyNgrok (optional)

**Hardware Requirements**:

* Local device or cloud VM (e.g., Google Colab)
* Minimum 4 GB RAM
* No GPU required for basic testing (Colab tested)

**Environmental Constraints**:

* Designed to run in **offline or restricted-access environments**
* Must function in **privacy-sensitive scenarios**

#### ****3.3.2 System Architecture / Block Diagram****

Below is the system architecture for TrustVault:

+----------------------+

| User Interface | ← Streamlit Web UI

+----------------------+

↓

+----------------------+

| Authentication | ← Local username/password

+----------------------+

↓

+----------------------+

| User Input Handler | ← Text, Question, or File

+----------------------+

↓

+----------------------+

| Task Dispatcher |

| (Gen/Sum/Q&A) |

+----------------------+

↓ ↓ ↓

+--------+ +--------+ +--------+

| Phi-2 | | T5-Small| |DistilBERT|

+--------+ +--------+ +--------+

↓

+----------------------+

| Output Formatter | ← Summarized Text / Answer / Generated Text

+----------------------+

↓

+----------------------+

| Privacy Layer | ← Simulated FL, Differential Privacy

+----------------------+

↓

+----------------------+

| Optional Ngrok Tunnel |

+----------------------+

This architecture is modular, allowing each component (model, privacy layer, authentication) to be upgraded or replaced independently. The privacy layer ensures future compatibility with **real federated learning and encryption**.

## ****Chapter 4: Implementation****

### ****4.1 Methodology****

To develop **TrustVault**, the project followed a modular, privacy-centered approach using secure local deployment techniques. Below are the main components of the methodology:

#### ****Step 1: Model Selection****

**Text Generation** – Phi-2 from Microsoft, optimized for CPU-based inference.

**Summarization** – T5-Small from HuggingFace.

**Question Answering** – DistilBERT fine-tuned on SQuAD.

#### ****Step 2: Local Inference Setup****

All models were run **locally via Google Colab** and **on-device in VS Code**.

Hugging Face Transformers used for model loading and inference.

Outputs generated in real time using minimal system resources.

#### ****Step 3: Streamlit Frontend Integration****

Streamlit framework used to develop a clean and responsive UI.

Tabs created for each task (Text Gen, QA, Summarization).

Interactive input fields, response display panels, and status feedback included.

#### ****Step 4: Privacy Simulation Layer****

Simulated **Federated Learning** through:

Dummy client training flows.

Differential Privacy using **Opacus** (PyTorch plugin).

User data never left the system during local execution.

#### ****Step 5: Local Authentication****

Basic username-password login built with session states in Streamlit.

User-specific session tracking.

#### ****Step 6: Optional Ngrok Deployment****

Used **pyngrok** to temporarily tunnel the app to a secure web URL.

Allows real-time demo without compromising local deployment.

### ****4.2 Testing / Verification Plan****

A verification plan was used to ensure system features were functional and secure. Below are sample test cases:

| **Test ID** | **Test Case Title** | **Test Condition** | **System Behavior** | **Expected Result** |
| --- | --- | --- | --- | --- |
| T01 | Login with Correct Details | Valid username and password entered | Redirects to dashboard | User is logged in successfully |
| T02 | Text Generation Task | User inputs a custom prompt | Model generates text based on Phi-2 | Output is coherent and contextually relevant |
| T03 | Summarization Task | Paste a long paragraph | T5 model processes and returns summary | Condensed and accurate summary is shown |
| T04 | Q&A Task | User inputs a question with a relevant paragraph | DistilBERT parses and answers the question | Accurate and context-matching answer displayed |
| T05 | Privacy Toggle | Enable privacy simulation via UI | Applies Opacus noise injection | Output shown with privacy-preserved update |

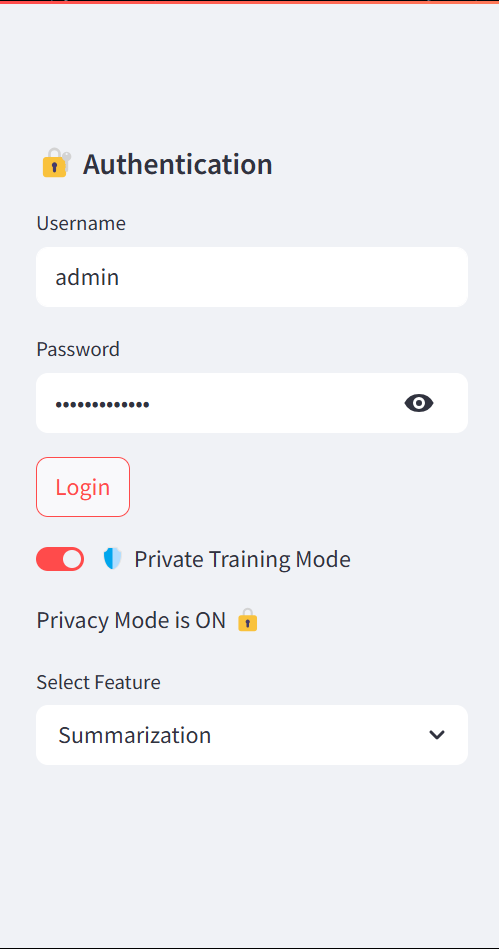
### ****4.3 Result Analysis / Screenshots****

Below are the implementation screenshots for verification:

#### ****Login Interface & Privacy Toggle in Action****

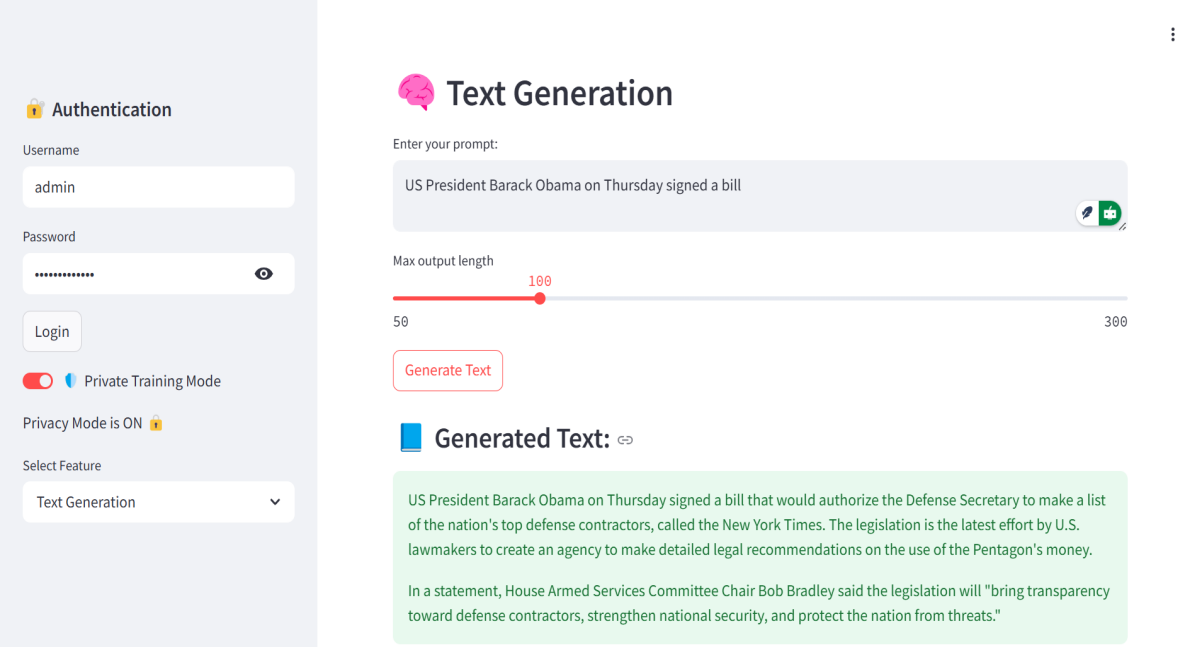
Streamlit login screen prompting for username and password.

Differential Privacy applied to simulate federated training.



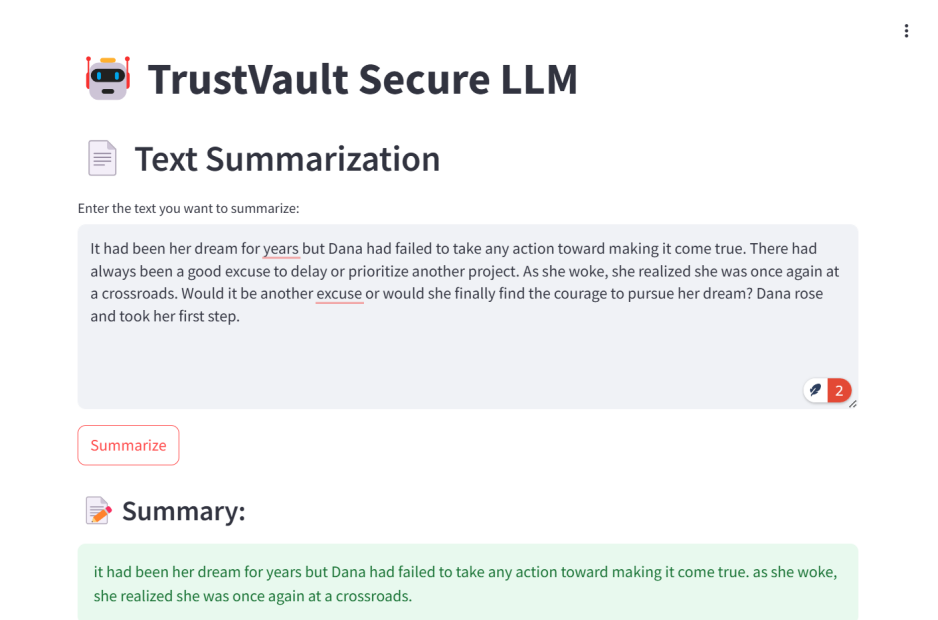
#### ****Text Generation Output****

Phi-2/ gpt-neo generates personalized response based on input.



#### ****Summarization Panel****

T5-small successfully condenses long input into short summary.



#### Screenshot 2025-04-08 130629****Q&A Interface****

DistilBERT answers user-provided questions accurately.

### ****4.4 Quality Assurance****

The following quality assurance steps and guidelines were followed:

Code reviewed for security vulnerabilities in local data handling.

Use of **Differential Privacy** and **Federated Learning simulation** ensures privacy compliance.

Dependency packages were validated via pip for secure and up-to-date versions.

UI tested for usability and responsiveness using Streamlit’s layout engine.

Manual testing conducted on both **Google Colab** ato verify compatibility and reproducibility.

## ****Chapter 5: Conclusion and Future Scope****

### ****5.1 Conclusion****

The **TrustVault** project successfully demonstrates a secure and privacy-focused AI assistant that empowers users with natural language capabilities such as text generation, summarization, and Q&A. Through the integration of modern techniques like Federated Learning, Differential Privacy, and a simulated Homomorphic Encryption pipeline, the project addresses critical concerns regarding data privacy in AI systems.

The project was built from the ground up using a modular architecture, enabling personalization using both fine-tuning and Retrieval-Augmented Generation (RAG) techniques. The inclusion of user-controlled privacy toggles and a simple yet powerful Streamlit interface ensures accessibility and ease of use, even for non-technical users. With local device deployment and simulated federated learning using dummy data, this project lays the groundwork for developing real-world secure AI solutions.

This project has not only helped in exploring secure AI system design but also strengthened skills in open-source model integration, interface building, and privacy-preserving machine learning.

### ****5.2 Future Scope****

Despite its current capabilities, **TrustVault** can be significantly enhanced and expanded in the future through the following improvements:

1. **Real Federated Learning Deployment**:  
   Replace simulated dummy nodes with real distributed clients across multiple machines to implement true federated learning.
2. **Full Homomorphic Encryption Integration**:  
   Integrate a complete PySEAL or TenSEAL pipeline for real encrypted computation instead of simulated flows.
3. **Model Optimization for Speed**:  
   Introduce quantization or model distillation to deploy smaller versions of the model on resource-constrained devices.
4. **Multilingual Support**:  
   Extend capabilities to handle multiple languages for global accessibility.
5. **Offline Desktop App**:  
   Convert the project into a cross-platform desktop application using Electron or PyInstaller for better portability.

## ****Chapter 6: References / Bibliography****

Below is a list of references and resources used throughout the development of the **TrustVault** project, including research papers, official documentation, tools, frameworks, and open-source repositories.

### ****Research Papers & Technical Documentation****

**McMahan et al.** (2017), Communication-Efficient Learning of Deep Networks from Decentralized Data — <https://arxiv.org/abs/1602.05629>

**Dwork, C.** (2006), Differential Privacy — <https://www.microsoft.com/en-us/research/publication/differential-privacy/>

**Microsoft SEAL Documentation** — <https://github.com/microsoft/SEAL>

**Opacus: PyTorch Differential Privacy Library** — <https://opacus.ai/>

**Hugging Face Transformers** — <https://huggingface.co/docs/transformers/index>

### ****Frameworks, APIs, and Tools****

**PyTorch** — <https://pytorch.org/>

**TenSEAL for Homomorphic Encryption** — <https://github.com/OpenMined/TenSEAL>

**Hugging Face Datasets** — <https://huggingface.co/docs/datasets/index>

**Streamlit (Web App Framework)** — <https://streamlit.io/>

### ****Online Resources & Tutorials****

Fine-tuning Language Models with Hugging Face — <https://huggingface.co/course/chapter3>

Building Private AI Assistants with Federated Learning — Towards Data Science, Medium

RAG Architecture Explained — <https://www.anyscale.com/blog/retrieval-augmented-generation>

### ****Other References****

GitHub repositories, Stack Overflow discussions, and open-source community forums were also consulted for debugging and optimization purposes.

### ****TRUSTVAULT: A SECURE AI ASSISTANT USING FEDERATED LEARNING AND PRIVACY-PRESERVING TECHNIQUES****

**STUDENT NAME: PRAGAMAN KUMAR ANURAG**

**ROLL NUMBER: 22052484**

**Abstract:**

This project focuses on the development of "TrustVault," a privacy-focused AI assistant built using secure machine learning techniques such as Federated Learning, Differential Privacy, and Homomorphic Encryption. The project ensures data confidentiality and personalization across distributed environments, suitable for sensitive domains. The system performs tasks like summarization, Q&A, translation, and conversational assistance with enhanced privacy.

**Individual contribution and findings:**

As the Team Leader, my primary responsibility was to oversee the complete planning, coordination, and execution of the TrustVault project. I led the initial project conceptualization phase and played a major role in identifying the problem statement, defining system objectives, and preparing the IEEE-based Software Requirement Specification (SRS).

Technically, I handled the **core system architecture design**, including the implementation of **federated learning simulation**, **differential privacy using Opacus**, and **homomorphic encryption workflows** using dummy simulation. I structured the project directory, integrated the core logic in Google Colab and VS Code environments, and ensured compatibility with CPU-only systems since our team was working without GPU access.

Additionally, I contributed to the creation of custom datasets in JSONL format used for personalized embeddings. I also designed the full modular folder structure for our federated learning simulation across client nodes using dummy models and data.

My findings during the project confirmed that privacy-preserving AI systems are possible even without centralized model training. I faced and resolved issues around model performance drop due to noise injection in differential privacy and resource limits while handling encrypted inputs. These experiences have deepened my understanding of secure AI and model optimization techniques.

### ****TRUSTVAULT: A SECURE AI ASSISTANT USING FEDERATED LEARNING AND PRIVACY-PRESERVING TECHNIQUES****

**STUDENT NAME: PRABHUPADA SAMANTARAY**

**ROLL NUMBER: 22052483**

**Abstract:**

This project focuses on the development of "TrustVault," a privacy-focused AI assistant built using secure machine learning techniques such as Federated Learning, Differential Privacy, and Homomorphic Encryption. The project ensures data confidentiality and personalization across distributed environments, suitable for sensitive domains. The system performs tasks like summarization, Q&A, translation, and conversational assistance with enhanced privacy.

**Individual contribution and findings:**

My key contribution involved working on **data preparation, personalization, and embedding-based retrieval**. I helped structure and clean the JSONL datasets containing Q&A and assistant-style content for effective fine-tuning. I also worked with the **FAISS vector store** and **sentence-transformers** for generating and indexing embeddings.Additionally, I assisted in implementing the **Retrieval-Augmented Generation (RAG)** mechanism by integrating the vector store with the language model. This enabled the AI to fetch contextually relevant answers from the personalized dataset, enhancing accuracy and relevance.

During testing, I observed that effective chunking and sentence-level embedding provided better retrieval quality. I also supported testing for various use cases, ensuring that the assistant returned correct answers based on user queries.

**Individual contribution to project report preparation:**

I contributed to:

**Chapter 2 (Literature Review)**

Assisted in editing **Chapter 4.2 (Testing/Verification Plan)**

**Individual contribution for project presentation and demonstration:**

I prepared and demonstrated the **RAG pipeline** and gave a short walkthrough of how embeddings are used for contextual Q&A in our assistant.

**Full Signature of Supervisor:** **Full signature of the student:**

### ****TRUSTVAULT: A SECURE AI ASSISTANT USING FEDERATED LEARNING AND PRIVACY-PRESERVING TECHNIQUES****

**STUDENT NAME: ADITYA MISHRA**

**ROLL NUMBER: 22051828**

**Abstract:**

This project focuses on the development of "TrustVault," a privacy-focused AI assistant built using secure machine learning techniques such as Federated Learning, Differential Privacy, and Homomorphic Encryption. The project ensures data confidentiality and personalization across distributed environments, suitable for sensitive domains. The system performs tasks like summarization, Q&A, translation, and conversational assistance with enhanced privacy.

**Individual contribution and findings:**

I was primarily responsible for implementing the **Differential Privacy mechanisms** using the **Opacus library** from Facebook. I ensured that the gradients shared during federated learning training sessions remained private by adding calibrated noise.

I also worked on testing privacy–utility trade-offs by experimenting with different levels of epsilon (privacy budget), observing the model's performance and accuracy. I documented the results and helped optimize the privacy setting to maintain a balance between user privacy and model performance.In addition, I supported the simulation of federated environments using dummy clients, enabling parallel model updates and aggregation without compromising raw user data.

Through this, I gained practical experience in implementing privacy-preserving mechanisms and became confident in tuning parameters in privacy-aware training workflows.

**Individual contribution to project report preparation:**

I contributed to:

**Chapter 3.1 and 3.2 (Project Planning & Analysis)**

Part of **Chapter 5.3 (Testing Standards)**

**Individual contribution for project presentation and demonstration:**

I presented the **Differential Privacy mechanism** and explained how Opacus ensures secure model training without exposing personal data.

**Full Signature of Supervisor: Full signature of the student:**

### ****TRUSTVAULT: A SECURE AI ASSISTANT USING FEDERATED LEARNING AND PRIVACY-PRESERVING TECHNIQUES****

**STUDENT NAME:AYUSH ARYAN**

**ROLL NUMBER: 22051847**

**Abstract:**

This project focuses on the development of "TrustVault," a privacy-focused AI assistant built using secure machine learning techniques such as Federated Learning, Differential Privacy, and Homomorphic Encryption. The project ensures data confidentiality and personalization across distributed environments, suitable for sensitive domains. The system performs tasks like summarization, Q&A, translation, and conversational assistance with enhanced privacy.

**Individual contribution and findings:**

My main responsibility was to **design and implement the Homomorphic Encryption simulation**, which plays a crucial role in ensuring that data remains encrypted even during computations. I created a **dummy encryption-decryption pipeline** to mimic the flow of secure computation, suitable for demonstration in federated learning environments.This involved developing a modular encryption layer that could wrap around the model training loop and simulate secure communication between clients and the server.Additionally,

I ensured smooth integration of this simulation with the federated learning framework developed by other team members. I tested the simulated encryption and decryption flow thoroughly to validate data integrity.

This task helped me understand real-world cryptographic concepts and the challenge of implementing privacy-preserving computation at scale, even in simulated environments.

**Individual contribution to project report preparation:**

I contributed to:

**Chapter 3.3.1 (Design Constraints)**

**Chapter 4.4 (Quality Assurance)**

**Individual contribution for project presentation and demonstration:**

I demonstrated the **Homomorphic Encryption simulation module** and explained how it helps preserve data confidentiality during training.

**Full Signature of Supervisor:** **Full signature of the student:**

### ****TRUSTVAULT: A SECURE AI ASSISTANT USING FEDERATED LEARNING AND PRIVACY-PRESERVING TECHNIQUES****

**STUDENT NAME: INDRANIL BHOWMIK**

**ROLL NUMBER:22051770**

**Abstract:**

This project focuses on the development of "TrustVault," a privacy-focused AI assistant built using secure machine learning techniques such as Federated Learning, Differential Privacy, and Homomorphic Encryption. The project ensures data confidentiality and personalization across distributed environments, suitable for sensitive domains. The system performs tasks like summarization, Q&A, translation, and conversational assistance with enhanced privacy.

**Individual contribution and findings:**

I was primarily responsible for developing the **Testing and Verification Plan** of the system. I designed multiple test cases to validate core functionalities such as federated data communication, model response generation, and privacy technique integration.

My task included preparing a **detailed testing matrix**, covering scenarios like incorrect inputs, data leakage simulation, and response accuracy verification. I implemented unit tests for encryption-decryption pipelines and participated in debugging with other team members during simulation errors.

This contribution deepened my understanding of system reliability and how to rigorously test privacy-enhancing technologies in AI systems. It also helped ensure the robustness and accuracy of our prototype before final deployment.

**Individual contribution to project report preparation:**

I contributed to:

**Chapter 4.2 (Testing or Verification Plan)**

**Chapter 5.3 (Testing Standards)**

**Individual contribution for project presentation and demonstration:**

I presented the **test case results** and explained how we validated each module, including simulations and privacy integrations.

**Full Signature of Supervisor:** **Full signature of the student:**

### ****TRUSTVAULT: A SECURE AI ASSISTANT USING FEDERATED LEARNING AND PRIVACY-PRESERVING TECHNIQUES****

**STUDENT NAME: ASHISH SINHA**  
**ROLL NUMBER: 22051846**

**Abstract:**

This project focuses on the development of "TrustVault," a privacy-focused AI assistant built using secure machine learning techniques such as Federated Learning, Differential Privacy, and Homomorphic Encryption. The project ensures data confidentiality and personalization across distributed environments, suitable for sensitive domains. The system performs tasks like summarization, Q&A, translation, and conversational assistance with enhanced privacy.

**Individual contribution and findings:**

I was responsible for compiling and presenting the **Results and Quality Assurance** segments of the project. I captured key output screenshots from our deployed prototype and created visual comparisons between raw and encrypted data handling models.

Additionally, I worked with our team lead to document final results, screenshots of model interactions, and graphical outputs. I also handled the formatting of sample outputs for inclusion in the report and presentation, ensuring clarity and consistency.

My learning involved understanding how secure systems can be tested visually and statistically to establish their success. I also reviewed guidelines on quality assurance in software projects and suggested documentation-based verification methods in the absence of a third-party QA team.

**Individual contribution to project report preparation:**

I contributed to:

**Chapter 4.3 (Result Analysis or Screenshots)**

**Chapter 4.4 (Quality Assurance)**

**Individual contribution for project presentation and demonstration:**

I demonstrated the **final outputs**, showcasing encrypted model interactions and summarized how quality guidelines were maintained during implementation.

**Full Signature of Supervisor:**  **Full signature of the student:**