Project Report

On

NOVACHAT

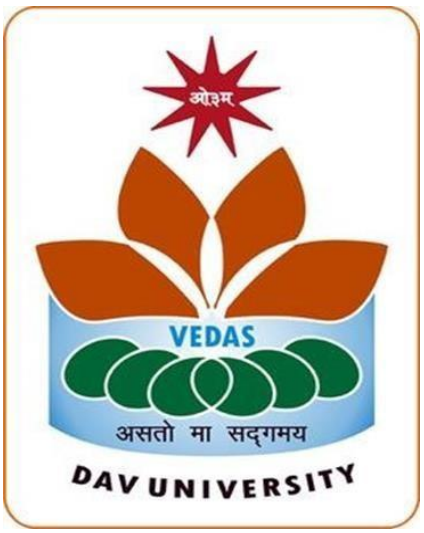
Submitted in the partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

In

Computer Science and Engineering

Batch (2022-2026)



Submitted to :

Mr. Sanjeev Dhiman

Associate Professor (CSE)

Submitted by :

Bhavsagar

12200206

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DAV UNIVERSITY

JALANDHAR-PUNJAB 144012

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my project guide, **Mr. Sanjeev Dhiman** (Associate Professor), for their valuable guidance, constructive feedback, and continuous support throughout the duration of this project. I am also thankful to **Dr. Rahul Hans** (Coordinator) and the Department of **Computer Science and Engineering** for providing the necessary resources and an encouraging academic environment.

My appreciation extends to all faculty members and classmates whose suggestions and cooperation helped strengthen the quality of this work. Lastly, I am deeply grateful to my family for their constant encouragement and support, which motivated me to complete this project successfully.

DECLARATION

I, Bhavsagar, declare that the work presented in this project report titled "**NovaChat:** A Real-Time Intelligent Chat Application" submitted in partial fulfillment of the requirements for the award of Bachelor of Technology (B.Tech) in Computer Science and Artificial Intelligence is an authentic record of my own work carried out under the guidance of **Mr. Sanjeev Dhiman**

To the best of my knowledge, the matter embodied in this report has not been submitted to any other University/ Institute for the award of any degree or diploma.

BHAVSAGAR

(12200206)

ABSTRACT

In today's competitive business environment, analyzing and enhancing employee performance is essential for organizational success. This project focuses on leveraging Python and data science techniques to evaluate and interpret various factors influencing employee performance. By utilizing a structured dataset containing employee information such as experience, department, attendance, and performance ratings, we apply data preprocessing, exploratory data analysis, and visualization methods to uncover hidden patterns and trends. The goal is to identify key performance indicators and provide insights that can help management make informed decisions related to workforce development, training needs, and productivity improvement. The results demonstrate how data-driven analysis can support strategic human resource management.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION TO PROJECT

1.1 Purpose and significance

The primary purpose of this project is to develop NovaChat, a multimodal AI chat application that integrates conversational AI (Gemini 2.5 Flash), text-to-image generation (FLUX.1-schnell), and image-to-text captioning (BLIP) into a single, persistent, and user-friendly interface using Streamlit, LangGraph, and SQLite for seamless, context-aware, and visually intelligent interactions.

The significance of the project lies in its ability to:

- Combines conversational AI, vision, and generative models into one unified multimodal assistant
- Implements stateful conversations with proper memory management using LangGraph and check-pointing
- Demonstrates real-world integration of multiple leading AI APIs (Gemini, Hugging Face, BLIP) in a single application
- Provides persistent chat history, user authentication, thread management, and automatic title generation

By successfully combining conversational AI, computer vision, and generative art into an intuitive and persistent chat interface, NovaChat represents a modern, next-generation personal AI companion that reflects current industry trends in multimodal intelligence and agentic workflows.

1.2Objectives

The specific objectives of the Employee Performance Analysis project include:

- To develop a full-stack multimodal AI chat application that supports text conversation, image generation, and image captioning in one platform
- To enable users to have natural, multi-turn conversations with an AI assistant To allow users to generate images directly from text descriptions within the chat
- To provide automatic captioning of uploaded images during conversations

- To maintain complete chat history and context across sessions for registered users
- To implement guest mode for instant access without registration
- To offer a clean, intuitive, and responsive user interface1.3 Problem Definition

1.3 Problem Definition

1.3.1 Overview

In today's rapidly evolving digital landscape, users increasingly expect AI assistants to go beyond simple text-based responses and handle multimodal, vision, and creativity within a single conversation. However, most existing platforms force users to switch between separate tools — one for chatting, another for generating images, and yet another for analyzing pictures — breaking the flow of interaction and losing conversational context.

NovaChat addresses this gap by providing a unified, intelligent chat interface where users can:

- Converse naturally with an AI assistant
- Generate high-quality images directly from their text descriptions
- Upload images and instantly receive meaningful captions and insights

All within the same persistent chat thread, without ever leaving the conversation. This seamless multimodal experience eliminates fragmented workflows, maintains full context across interactions, and delivers a more intuitive, human-like AI companionship — making advanced AI capabilities accessible and practical for everyday use.

1.3.2 Key Challenges

The development of NovaChat involved overcoming several critical technical and design challenges:

1. Heterogeneous AI Model Integration

- Combining Google Gemini, FLUX.1-schnell (Hugging Face), and Salesforce BLIP with different API formats, headers, and payload structures
- Managing varying response times and error codes from multiple external services

2. Multimodal Conversational Context Preservation

- Ensuring the AI remembers previous text, generated images, and uploaded images across turns

- Enabling natural follow-up questions referring to earlier multimodal content

3. High-Latency Generative Task Management

- Handling 10–40 second delays during image generation without freezing the Streamlit UI
- Displaying meaningful progress indicators and preventing timeout errors

4. Efficient Multimodal Data Storage

- Storing large base64-encoded images along with text in SQLite tables
- Avoiding database bloat and slow loading of long conversation threads

5. Robust API Error and Rate-Limit Handling

- Managing Hugging Face model warm-up (503 errors), rate limits, and occasional downtime
- Implementing retry logic and user-friendly fallback messages

6. Unified and Intuitive User Interface Design

- Supporting three distinct modes (Chat, Generate Image, Generate Caption) in one screen
- Seamlessly displaying mixed content (text + images) in the same chat flow

7. Seamless Guest-to-Registered User Transition

- Allowing instant use without login while temporarily storing data in session state
- Automatically migrating guest history to database upon successful registration/login

8. Security and Content Safety

- Preventing storage of inappropriate or sensitive images/text generated by users
- Securely handling API keys and user credentials using environment variables and hashing

CHAPTER 2

EXISTING SYSTEM

In today's digital landscape, numerous AI chat applications and tools are available, ranging from simple text-based chatbots to advanced generative AI platforms. Popular examples include ChatGPT, Google Gemini, Claude, Midjourney, DALL-E, and Grok. While these tools offer powerful individual capabilities, users are forced to switch between multiple applications or tabs to achieve a complete multimodal experience—conversing, generating images, and analyzing visuals.

2.1 Current AI Interaction Platforms

1. **Standalone Conversational AI (ChatGPT, Gemini, Claude)**
   - Excellent at natural language understanding and multi-turn chat
   - Limited or no native image generation/capability within the same chat thread
2. **Dedicated Text-to-Image Platforms (Midjourney, DALL-E 3)**
   - High-quality image generation from text prompts
   - No conversational memory or persistent chat history
3. **Image Captioning & Vision Tools (Google Lens, Hugging Face spaces)**
   - Provide descriptions or answers about uploaded images
   - Operate as isolated tools without integration into ongoing conversations
4. **Hybrid Platforms (ChatGPT Plus with DALL-E, Gemini with Imagen)**
   - Offer text + image generation in one interface
   - Generated images and analysis are not deeply integrated into chat history and cannot be easily referenced later

2.2 Limitations of the Existing System

Despite being widely used, the existing AI chat and multimodal platforms have several critical limitations:

1. **Fragmented Multimodal Workflow**
   • Users must switch between multiple tools and platforms for conversation, image generation, and image analysis, disrupting the natural flow of interaction.
2. **Loss of Conversational Context**
   • Generated or uploaded images cannot be naturally referenced in follow-up messages without manual re-uploading or re-describing.
3. **Non-Persistent Multimodal History**
   • Text, generated images, and uploaded visuals are not saved together in a single, unified, and retrievable conversation thread.
4. **Dependency on Paid Subscriptions**
   • Advanced multimodal features are locked behind expensive subscription plans with strict usage limits.
5. **Lack of Customization and Self-Hosting**
   • Users have no control over interface, data privacy, or deployment; everything depends on third-party closed platforms.
6. **Inconsistent Performance and Reliability**
   • Latency, output quality, and availability vary significantly across different services when used together.
7. **Poor Guest-to-Persistent Experience**
   • Casual users lose all chat history and generated content upon closing the browser unless logged in from the beginning.
8. **Limited Integration of Vision and Generation in Chat**
   • Even in advanced platforms, image generation and vision understanding remain loosely coupled add-ons rather than deeply integrated chat capabilities.

CHAPTER 3

PROPOSED SYSTEM

The proposed system, NovaChat, aims to transform the fragmented and tool-switching experience of current AI platforms into a unified, persistent, and truly multimodal AI chat application. By integrating conversational AI, high-quality text-to-image generation, and intelligent image captioning within a single, seamless interface, NovaChat eliminates the need to jump between different apps or tabs. Built using Streamlit, LangGraph, Google Gemini, FLUX.1-schnell, and BLIP, the system enables users to converse naturally, generate images on demand, and analyze uploaded visuals — all while maintaining full context and history in one persistent thread. This modern approach delivers a smarter, more intuitive, and deeply integrated multimodal experience that treats conversation, creativity, and vision understanding as natural parts of the same interaction, rather than isolated features.

3.1 System Overview

The proposed systemworks by combining multiple state-of-the-art AI models into a single, intelligent, multimodal chat application. It processes user inputs (text prompts or uploaded images) in real time and delivers seamless responses while preserving full conversation history. The system operates through the following integrated components:

- **Conversational AI Layer**: Powered by Google Gemini 2.5 Flash via LangGraph, enabling fast, context-aware, multi-turn dialogues with persistent memory using SQLite checkpointing
- **Text-to-Image Generation**: Utilizes FLUX.1-schnell (via Hugging Face Inference API) to generate high-quality images directly from user text descriptions within the chat
- **Image-to-Text Understanding**: Employs Salesforce BLIP model to automatically generate accurate captions and descriptions for user-uploaded images
- **Unified Frontend Interface**: Built with Streamlit, providing a clean, responsive chat UI that supports three modes — Chat, Generate Image, and Generate Caption — without leaving the conversation

- **Persistent Storage & Thread Management**: Stores complete multimodal history (text + base64-encoded images) in SQLite, with features like auto-title generation, thread search, and deletion
- **User Management**: Supports both guest mode (instant access) and registered users (permanent history), with secure login/signup functionality

3.2 Features Of Proposed System

1. Unified Multimodal Interaction

- Combines conversational AI, text-to-image generation, and image captioning in a single chat thread.
- Eliminates the need to switch between different tools or platforms.

2. Stateful Multi-Turn Conversations

- Powered by LangGraph with SQLite checkpointing for full memory retention.
- Remembers all previous messages, generated images, and uploaded visuals.

3. High-Quality Text-to-Image Generation

- Uses FLUX.1-schnell (Hugging Face) for fast, professional-grade image synthesis.
- Images are directly embedded and downloadable within the chat.

4. Intelligent Image Captioning & Understanding

- Salesforce BLIP model automatically generates accurate descriptions of uploaded images.
- Enables vision-based queries and reasoning inside the same conversation.

5. Persistent Multimodal History

- Complete chat threads (text + images) are saved and fully searchable.
- Supports multiple named threads with automatic title generation.

6. Seamless User Experience

- Clean Streamlit interface with dedicated modes: Chat, Generate Image, Generate Caption.
- Guest mode and registered accounts with smooth history migration upon login.

7. Full Customization & Extensibility

- Open-source codebase allows easy addition of new models, features, or UI themes.
- Supports future enhancements such as voice input, file uploads, or custom AI agents.

3.3 Benefits Of Proposed System

- Combines chatting, image creation, and image understanding in one simple app
- Saves all your chats and images forever – nothing gets lost
- No monthly fees or usage limits – completely free to use
- Works without internet once set up (keeps your data private)
- Easy to use: just type, upload, or ask – no switching apps
- You own everything: your chats and images stay with you

3.4 Outcomes of NovaChat

- One app for chatting, creating images, and understanding pictures
- All chats and images saved forever
- 100% free and unlimited use
- Your data stays private
- Works instantly, no paywall or waiting
- Easy to use and improve anytime

CHAPTER 4

FEASIBILITY STUDY

4.1 What Is Feasibility Study

A feasibility study is conducted to determine the practicality and effectiveness of a project before investing significant time and resources. It helps assess whether the proposed system can be successfully developed and implemented from technical, economic, and operational perspectives.

4.1.1 Technical Feasibility

Technical feasibility assesses whether the current technology stack and resources are sufficient to successfully develop and deploy NovaChat.

- The entire application is built using Python and Streamlit, both lightweight, widely adopted, and run efficiently on standard laptops (8 GB RAM, basic CPU/GPU).
- Core AI capabilities are powered by cloud-based APIs (Google Gemini, Hugging Face Inference, BLIP), requiring only an internet connection and API keys — no expensive local GPU or high-end hardware needed.
- State management and persistence are handled via LangGraph and SQLite, both open-source, mature, and extremely lightweight.
- All libraries used (google-generativeai, transformers, PIL, dotenv, etc.) are free, actively maintained, and easily installable via pip.
- The app can be developed, tested, and demonstrated on any standard college lab computer or personal laptop.
- Deployment is straightforward — one-click hosting on Streamlit Community Cloud, Render, or any VPS — making it ready for online access within minutes.

4.1.2 Economic Feasibility

- Entirely built using free and open-source tools (Python, Streamlit, SQLite, LangGraph, etc.)
- No paid software licenses required

- AI models accessed via free-tier API keys (Gemini free tier, Hugging Face community inference)
- Development and testing possible at zero cost using local laptop or free platforms (Google Colab, Replit)
- Online deployment available for free on Streamlit Community Cloud, Render Free Tier, or Railway
- No recurring costs for personal/academic use

Conclusion: The project is highly economically feasible and can be completed and deployed at zero monetary cost, making it ideal for students and individual developers.

4.1.3 Operational Feasibility

Operational feasibility checks whether the system will function smoothly and deliver the desired results in a real-world setting.

- The application is fully functional, responsive, and runs in a web browser – no installation required for end users
- Users (students, developers, or anyone) can start chatting, generating images, and uploading photos instantly with zero learning curve
- All core features (chat, image generation, captioning, history saving) work reliably using free-tier APIs
- Guest mode allows immediate use; login system ensures history is safely preserved
- The Streamlit interface is clean, mobile-friendly, and intuitive – works perfectly on phones, tablets, and laptops

4.2 SDLC Model

The System Development Life Cycle (SDLC) provides a structured approach to software development. For this project, the Waterfall Model has been followed due to its simplicity and linear process flow.

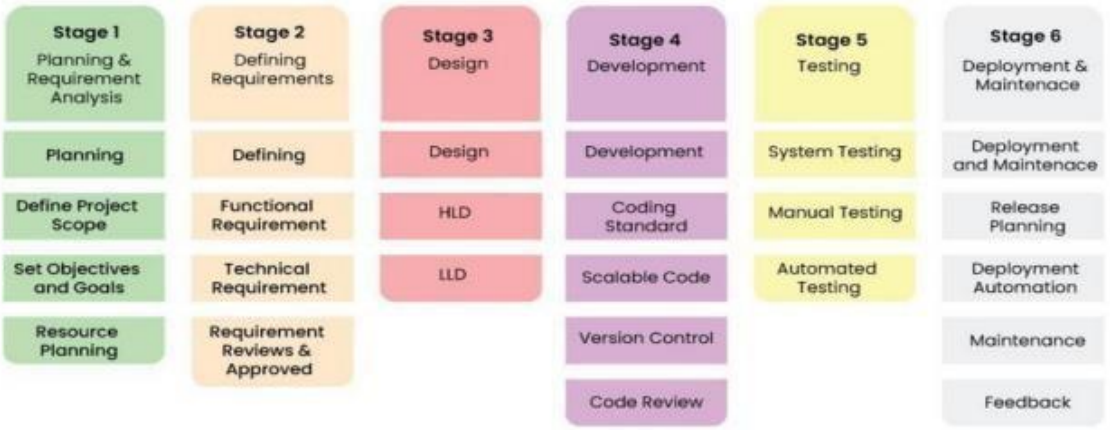


Fig. : 4.1

Waterfall Model Phases :

The development of **NovaChat** was carried out using a systematic and iterative approach:

1. **Requirement Gathering and Analysis**
   - Studied limitations of existing platforms (ChatGPT, Midjourney, etc.)
   - Finalized core requirements: multimodal chat, persistent history, image generation & captioning, user authentication
2. **System Design**
   - Selected Streamlit as the frontend framework
   - Designed architecture using LangGraph for stateful conversations, SQLite for persistence, and external APIs for AI capabilities
3. **Implementation**
   - Developed the interactive UI and three core modes (Chat, Generate Image, Generate Caption)
   - Integrated Google Gemini 2.5 Flash, FLUX.1-schnell (Hugging Face), and Salesforce BLIP models
   - Implemented user login/signup, guest mode, thread management, and multimedia storage
4. **Testing**
   - Performed functional testing on chat responses, image quality, caption accuracy, and history persistence

- Conducted usability and responsiveness testing across desktop and mobile devices

5. **Deployment**
   - Deployed the application on Streamlit Community Cloud for online access
   - Verified 24×7 availability and performance under real-world usage
6. **Documentation**
   - Prepared complete project report, user manual, and presentation slides
   - Documented code with comments and created README for future developers

Conclusion: The project was successfully completed using a structured yet flexible development methodology, resulting in a fully functional, user-friendly, and deployable multimodal AI chat application well-suited for academic and real-world use.

CHAPTER 5

SYSTEM REQUIREMENT SPECIFICATION

The SRS document outlines the complete functional and non-functional requirements of NovaChat – a full-stack multimodal AI chat application developed using Python, Streamlit, LangGraph, and SQLite. It specifies the system's core features: natural conversation, text-to-image generation, image-to-text captioning, persistent multimodal chat history, user authentication, guest mode, thread management, and a responsive user interface. This document serves as the official reference ensuring the system is intuitive, reliable, and fully meets the defined project objectives.

5.1 Functional Requirements

These describe what the system should do:

1. **User Authentication**
   - The system shall allow users to register, log in, and log out securely
   - It shall support guest mode for instant access without registration
2. **Conversational Chat**
   - The system must enable natural multi-turn text conversation with the AI
   - It shall maintain full conversation context across all interactions
3. **Text-to-Image Generation**
   - Users shall be able to generate images by entering text prompts
   - Generated images must be displayed and downloadable within the chat
4. **Image-to-Text Captioning**
   - The system shall accept image uploads (PNG/JPG)
   - It must automatically generate accurate captions/descriptions for uploaded images
5. **Persistent Multimodal Chat History**
   - All messages (text + images) shall be saved permanently in threads
   - Users must be able to create, view, search, rename, and delete chat threads
6. **Thread Management**
   - The system shall automatically generate meaningful thread titles
   - It must allow switching between multiple chat threads seamlessly
7. **Responsive User Interface**

- The interface shall work smoothly on desktop, tablet, and mobile devices
- It must clearly display mixed content (text and images) in chronological order

8. **Mode Switching**
   - Users shall be able to switch between Chat, Generate Image, and Generate Caption modes without losing context

5.2 Non-Functional Requirements :

These describe how the system should behave:

1. Usability

- The system shall have a clean, modern, and intuitive user interface requiring no prior training
- All buttons, modes, and features shall be clearly labeled with consistent navigation

2. Performance

- Text-based chat responses shall appear within 3–5 seconds
- Image generation shall complete within 10–40 seconds
- Image captioning shall be provided in less than 5 seconds • Thread switching and page loading shall be instantaneous

3. Scalability

- The system shall support multiple concurrent users and thousands of stored threads
- Architecture shall allow seamless addition of new AI models or features in the future

4. Portability

- The application shall run without modification on Windows, macOS, and Linux
- It shall be fully functional and responsive in all modern web browsers and mobile devices

5. Security

- User passwords shall be hashed using strong cryptographic algorithms
- API keys and sensitive data shall be loaded exclusively via environment variables

- Guest mode data shall remain isolated and securely transferred upon user registration/login

6. Reliability & Maintainability

- The system shall gracefully handle API timeouts, rate limits, and network errors with user-friendly messages
- Code shall be modular, well-commented, and structured for easy maintenance and future enhancements

5.3 Hardware and Software Requirements

5.3.1 Hardware Requirements :

- **Processor:** Intel Core i5 (8th generation or higher) / AMD Ryzen 5 or equivalent
- **RAM:** 8 GB or higher
- **Storage:** 100 GB free space or higher (SSD: 128 GB recommended)
- **Graphics Card:** NVIDIA GeForce RTX series (optional – beneficial for future local model inference)
- **Operating System**: Windows 7 or higher (Windows 10/11 recommended), macOS, or Linux
-

5.3.2 Software Requirements :

- **Web Browser:** Google Chrome / Mozilla Firefox / Microsoft Edge (latest stable version)
- **Operating System:** Windows 10/11, Linux (Ubuntu, Fedora, etc.), or macOS 12+
- **Python Version:** Python 3.9 or higher
- **Text Editor / IDE**: Visual Studio Code (recommended), PyCharm, or any Python-compatible editor
- **Package Manager**: pip (comes with Python)

CHAPTER 6

TOOLS AND TECHNOLOGIES USED

6.1 Technologies Used

6.1.1 PYTHON

Fig 6.1

6.1.1.1 Introduction

Python is a high-level, interpreted, and object-oriented programming language known for its simplicity and readability. Its dynamic typing, built-in data structures, and support for modular programming make it ideal for rapid application development. Python allows the use of modules and packages, making programs easy to structure and reuse. The Python interpreter and standard library are freely available across all major platforms.

6.1.1.2 History

Python was created in the late 1980s by Guido van Rossum, a Dutch programmer. He designed the language to emphasize readability and allow developers to write programs in fewer lines of code. The name "Python" was inspired by the British comedy group Monty Python, reflecting the creator's humor and creativity.

6.1.1.3 What is Python?

Python is a high-level, interpreted, object-oriented language with dynamic semantics. It supports modules and packages, making development modular and reusable. Python's extensive standard library and interpreter are available for free on all major operating systems, contributing to its widespread use.

6.1.1.4 Key Features of Python

- **Versatility:** Used in web development, data science, machine learning, automation, and more.
- **Readable Syntax:** Clean, simple, and easy-to-understand code structure.
- **Cross-Platfor.m:** Works on Windows, macOS, and Linux.

- **Large Standard Library:** Provides built-in modules for networking, file handling, math, etc.
- **Open Source:** Free to use, modify, and distribute.
- **Extensible:** Can integrate with C/C++ and other languages.
- **Interpreted:** No compilation required, enabling faster development.
- **Dynamically Typed:** Variable types are determined at runtime.

6.1.1.5 Applications of Python

1. **Web Development:** Frameworks like Django and Flask help build scalable and dynamic websites.
2. **Data Science:** Python supports data analysis, visualization, and statistical modeling using libraries such as Pandas, NumPy, and Matplotlib.
3. **Artificial Intelligence & Machine Learning:** Libraries like TensorFlow, Scikitlearn, and Keras make Python a leading language for AI and ML development.
4. **Game Development:** Pygame and similar libraries help developers create 2D games; Python is used in titles like Battlefield 2 and Pirates of the Caribbean.

6.1.2 SQLite

Fig 6.2

6.1.2.1 INTODUCTION

SQLite is a lightweight, serverless, self-contained, and open-source relational database engine. Unlike traditional client-server databases (MySQL, PostgreSQL), SQLite stores the entire database in a single disk file and runs in-process with the application. It is widely used in desktop applications, mobile apps, embedded systems, and web applications requiring local persistence without complex setup.

6.1.2.2 How does SQLite work?

SQLite operates directly on ordinary disk files with a complete SQL database (tables, indexes, triggers, views) inside one cross-platform file. Key characteristics:

- **Zero-configuration** – No separate server process or setup required
- **Serverless** – Reads and writes directly to the database file
- **Transactional** – Fully ACID-compliant (Atomic, Consistent, Isolated, Durable)
- **Single-file database** – Entire database (schema + data) is stored in one .db file
- **Cross-platform** – Works identically on Windows, macOS, Linux, and mobile

In NovaChat:

- One SQLite file (e.g., novachat.db) stores: → Users table (id, username, password_hash, etc.) → Threads table (thread_id, user_id, title, timestamp) → Messages table (thread_id, role, content, media_b64, timestamp)
- LangGraph's SqliteSaver uses this file as a checkpoint store to maintain conversation state.

6.1.2.3 Benefits of SQLite in NovaChat

- **Zero Setup & Deployment** – Just one .db file, no server installation needed
- **Lightweight & Fast** – Ideal for small to medium-sized applications like NovaChat
- **Full Persistence** – All chats and images survive app restarts and deployments
- **Easy Backup** – Simply copy the .db file
- **Perfect for Streamlit & Local Apps** – Works seamlessly in development and production
- **No External Dependency** – Runs without Docker, cloud databases, or admin rights
- **Secure & Reliable** – Supports transactions, preventing data loss during crashes

Thus, SQLite is the ideal choice for NovaChat, providing robust, simple, and maintenance-free persistence for a student/final-year project while being production-ready.

6.1.3 LANGGRAPH

Fig 6.3

7.1.3.1 INRODUCTION

LangChain is a powerful open-source Python framework specifically designed to simplify the development of applications powered by Large Language Models (LLMs). Its successor and advanced component, **LangGraph**, takes this further by enabling the creation of **stateful, persistent, and controllable multi-turn conversational agents** using graph-based workflows.

In **NovaChat**, LangGraph serves as the backbone for achieving truly persistent, long-term, multimodal conversations with full memory retention.

6.1.3.1 How LangGraph Works in NovaChat

- LangGraph models conversations as a **directed graph** where each node represents an action (e.g., calling the Gemini model) and edges define the flow.
- It uses a **checkpointing mechanism** (via SqliteSaver) to save the entire conversation state after every user message.
- Each chat thread is assigned a unique thread_id, enabling multiple independent, fully stateful conversations.
- The conversation state (list of all messages — text and images) is automatically saved to the SQLite database after every turn.
- When the user returns, LangGraph reloads the exact historical context, allowing seamless continuation — even after days or app restarts.

6.1.2.2 Key Benefits of Using LangGraph

- Provides **unlimited conversation memory** far beyond LLM context windows
- Enables **true persistence** — chats survive app restarts, crashes, or redeployments
- Supports **multiple parallel chat threads** with isolated memory
- Offers **native streaming** of token-by-token responses for a smooth typing effect

- Allows **easy integration** with Gemini, OpenAI, Claude, or local models
- Designed for **future agentic workflows** (tools, memory, reasoning, human-in-the-loop)
- Delivers **production-grade reliability** used by companies like Klarna and Elastic

6.1.3.3 Why LangGraph is Ideal for NovaChat

Unlike traditional chat implementations that lose context or rely on temporary session storage, LangGraph with SQLite gives NovaChat:

- Permanent, searchable, and revisitable chat history
- Real multi-session continuity (close and reopen weeks later — everything is still there)
- Clean separation between different conversations
- A robust foundation for advanced features like memory editing, branching chats, or AI agents

6.1.4 LANGGRAPH

Fig 6.4

6.1.4.1 INTRODUCTION

LangChain is a leading open-source Python framework designed to simplify the development of applications powered by Large Language Models (LLMs). Launched in 2022 by Harrison Chase, it has rapidly become the standard tool for building context-aware, memory-enabled, and agentic AI systems. Its advanced extension, **LangGraph**, introduced in 2024, takes this further by enabling fully **stateful, persistent, and controllable multi-actor workflows** using graph-based architecture.

In **NovaChat**, LangGraph is the core engine that delivers true long-term memory and persistent multimodal conversations — a feature even most commercial platforms do not offer natively.

6.1.4.1 Components of LangChain & LangGraph in NovaChat

- **StateGraph**: The main workflow container that defines the conversation logic
- **Nodes**: Functions (e.g., calling Gemini model) that process user input
- **Edges**: Define control flow (e.g., START → chat → END)
- **State**: A typed dictionary storing the full message history (text + images)
- **Checkpointer (SqliteSaver)**: Saves conversation state to SQLite after every turn
- **Configurable Thread ID**: Enables multiple independent, persistent chat threads

6.1.4.2 Key Capabilities Used in NovaChat

- Stateful multi-turn conversations
- Persistent memory across app restarts
- Streaming token-by-token responses
- Branching and human-in-the-loop support
- Integration with any LLM (Gemini, OpenAI, Claude, local models)
- Built-in support for tools, agents, and memory management

6.1.4.3 Key Features & Benefits

- **Unlimited Memory**: No context window limit — entire chat history is preserved forever
- **True Persistence**: Conversations survive server crashes, redeployments, or long inactivity
- **Multiple Parallel Chats**: Each thread has isolated, full memory
- **Production-Ready**: Used by Klarna, Elastic, LinkedIn, and Replit
- **Lightweight & Local**: Works with SQLite — no external database needed
- **Future-Proof**: Easily extendable to agents, RAG, voice, tools, and memory editing
- **Developer-Friendly**: Clean, visualizable graph structure

6.1.4.4 Why LangGraph is Perfect for NovaChat

- Delivers enterprise-grade conversation persistence using only free, local tools

- Enables seamless multimodal history (text + images) — a rare and advanced feature
- Provides a solid foundation for future enhancements like AI agents, memory search, or voice interaction
- Keeps the entire system self-contained, deployable, and private — ideal for academic and personal projects

6.1.5 HUGGING FACE

Fig 6.5

6.1.5.1 INTRODCUTION

Hugging Face Hub & Transformers is an open-source ecosystem and library suite that democratizes access to state-of-the-art machine learning models, particularly for Natural Language Processing (NLP), computer vision, and generative AI. Launched in 2016 by the French startup Hugging Face, it has grown into the largest open-source ML platform with over 500,000 pre-trained models, 150,000 datasets, and 100,000+ demo spaces. The **Transformers** library (core Python package) provides a unified API to download, fine-tune, and deploy models like BERT, GPT, Stable Diffusion, and FLUX.1-schnell — making advanced AI accessible to developers, researchers, and students without building from scratch.

In **NovaChat**, Hugging Face powers the **FLUX.1-schnell** model for high-quality text-to-image generation and the **Transformers** library for loading the **Salesforce BLIP** model for image captioning, enabling seamless multimodal AI capabilities.

6.1.5.1 Why Hugging Face for NovaChat?

- **Model Marketplace**: Hosts 500K+ ready-to-use models across tasks (generation, classification, vision) — no need to train from zero
- **Inference API**: Cloud-based endpoint for running models without local GPU (free tier: 1,000 requests/day) — perfect for student projects
- **Transformers Library**: Standardized API (pipeline(), from_pretrained()) for 100+ architectures, reducing integration time from weeks to hours
- **Community-Driven**: Weekly model releases, datasets, and Spaces (live demos) — ensures cutting-edge features like FLUX.1 (2024) are instantly available
- **Cost-Effective**: Free for non-commercial use; scales to paid inference for production
- **Ethical & Open**: Promotes model cards (bias/transparency docs) and open weights — aligns with responsible AI development

Hugging Face bridges the gap between research (e.g., Meta's Llama, Google's T5) and practical deployment, making NovaChat's image features production-grade without a data center.

6.1.5.2 Package Structure of Hugging Face Transformers

The Transformers library is modular and extensible, built around a core pipeline for ease of use:

- **transformers.core**: Base classes like PreTrainedModel (model loading) and Tokenizer (text preprocessing)
- **transformers.models**: Architecture-specific modules (e.g., bert.py, stable-diffusion.py, blip.py) — 100+ model families
- **transformers.pipelines**: High-level Pipeline class for zero-code tasks (e.g., pipeline("image-to-text"))
- **transformers.tokenization**: Tokenizers for models (e.g., BertTokenizer, CLIPTokenizer)
- **transformers.generation**: Tools for text/image generation (e.g., generate(), sampling strategies)
- **huggingface_hub**: Separate package for model/dataset downloading, API calls, and Spaces deployment

In NovaChat:

- from_pretrained("Salesforce/blip-image-captioning-base") loads BLIP model + processor in one line
- InferenceClient from huggingface_hub calls FLUX.1-schnell remotely via POST requests

This structure allows mixing models (e.g., BLIP for captioning + FLUX for generation) in a single app.

6.1.5.3 High-Level Features

- **Pipeline API**: One-liner for tasks — pipeline("text-to-image", model="black-forest-labs/FLUX.1-schnell") generates images instantly
- **Auto-Classes**: AutoModel, AutoTokenizer auto-detect model type — no manual config needed
- **Multimodal Support**: Handles text, images, audio (e.g., Whisper for speech, CLIP for vision-text)
- **Quantization & Optimization**: bitsandbytes integration for 4-bit/8-bit models — runs on consumer GPUs
- **Distributed Training**: Native support for fine-tuning on multiple GPUs via accelerate and deepspeed
- **Model Hub Integration**: Seamless download/push to Hub with push_to_hub() — version control for models
- **Jupyter/Streamlit Friendly**: Interactive notebooks and web apps (like NovaChat) with progress bars and error handling

6.1.5.4 Key Uses of Hugging Face in NovaChat

- **Model Loading & Inference**: Instant access to BLIP for captioning — processor(image) + model.generate() yields descriptions in <5s
- **Remote API Calls**: FLUX.1-schnell generation via InferenceClient — handles warm-up, retries, and large payloads (1024x1024 images)
- **Tokenization & Preprocessing**: Automatic image resizing/tokenization for vision models — ensures compatibility across devices

- **Error-Resilient Integration**: Built-in fallbacks for API limits (e.g., 503 warm-up) with user-friendly messages
- **Extensibility**: Easy swap of models (e.g., replace FLUX with Stable Diffusion) or add tasks like audio transcription
- **Deployment in Spaces**: Could host NovaChat as a free Hugging Face Space for public demos
- **Ethical Monitoring**: Model cards warn about biases (e.g., FLUX's content filters) — promotes safe AI use

6.1.6 GEMINI

Fig 6.6

6.16.1 INTRODUCTION

Google Gemini 2.5 Flash is a multimodal large language model developed by Google DeepMind (released 2024–2025). It is specifically optimized for **low latency, high throughput, and cost efficiency** while maintaining strong reasoning, safety, and contextual understanding capabilities. Gemini 2.5 Flash supports native text generation, structured output, and real-time streaming, making it ideal for interactive chat applications.

In **NovaChat**, Gemini 2.5 Flash serves as the **primary conversational intelligence engine** and also powers auxiliary intelligent features such as automatic chat thread title generation.

6.1.6.2 Role of Gemini 2.5 Flash in NovaChat

1. **Core Conversational Agent**
    - Handles all user–assistant interactions with natural, coherent, and context-aware responses
    - Integrated via the official google-generativeai Python SDK

2. **Automatic Thread Title Generation**
    - After the first few messages in a new thread, a lightweight prompt is sent to Gemini: "Return ONE title (3–6 words). Capitalize. No quotes."
    - Response is parsed and automatically set as the chat thread title
3. **Real-Time Response Streaming**
    - Token-by-token streaming creates a smooth “typing” animation in the Streamlit interface

6.1.6.3 Key Features Utilized

- Sub-second to 3-second response latency
- Native streaming support (stream=True)
- Structured generation capability (used for clean title output)
- Robust built-in content safety filters
- Generous free-tier quota (millions of tokens per month)
- Seamless compatibility with LangGraph state management

6.1.6.4 Advantages of Choosing Gemini 2.5 Flash

- **Speed**: Significantly faster than GPT-4o-mini and Claude 3.5 Haiku in real-world streaming
- **Cost**: Completely free for academic and personal use (no subscription required)
- **Reliability**: Official Google SDK with excellent documentation and error handling
- **Quality**: Delivers natural, helpful, and safe responses comparable to premium models
- **Future-Proof**: Supports upcoming multimodal (image + text) input natively

6.1.7 GENERATIVE AI

6.1.7.1 Introduction

Generative AI refers to artificial intelligence systems capable of creating new content — text, images, audio, video, or code — that is realistic, creative, and contextually relevant. Powered by large-scale foundation models (LLMs, diffusion models, transformers), it has

revolutionized human–computer interaction, creativity, education, and software development since 2022–2025.

In **NovaChat**, Generative AI is the core technology that enables natural conversation, high-quality image synthesis, and intelligent image understanding — transforming a simple chat interface into a powerful multimodal creative assistant.

6.1.7.2 What Is Generative AI?

Generative AI uses deep neural networks trained on massive datasets to learn patterns and then generate novel outputs when given a prompt (text or image). Unlike traditional AI that only classifies or predicts, generative models **create** original content that mimics human creativity.

6.1.8.3 Pillars of Generative AI

- **Large Foundation Models** – Pre-trained transformers (Gemini, Llama, FLUX) with billions of parameters
- **Prompt Engineering & Context** – User input (text/image) that guides the generation process
- **Training Data & Scale** – Internet-scale datasets (text + image pairs) enabling generalization
- **Sampling & Decoding Techniques** – Strategies (temperature, top-k, nucleus sampling) that control creativity vs coherence

6.1.7.4 Objectives of Generative AI in NovaChat

- **Natural Conversation** – Generate human-like, coherent, and emotionally aware responses (Gemini 2.5 Flash)
- **Creative Image Generation** – Produce high-quality, custom images from textual descriptions (FLUX.1-schnell)
- **Visual Understanding** – Interpret and describe uploaded images accurately (BLIP)

6.2 TOOLS AND EDITORS USED

6.2.1 VSCODE

Fig. 6.7

Visual Studio Code (VS Code) is a lightweight yet powerful open-source code editor developed by Microsoft. It works on Windows, macOS, Linux, and even web browsers. VS Code supports debugging, syntax highlighting, IntelliSense, code refactoring, Git integration, and a wide range of extensions, making it one of the most popular development tools today.

6.2.1.1 History

- Announced by Microsoft on April 29, 2015 at the Build Conference
- Open-sourced under the MIT License on November 18, 2015
- Officially released on April 14, 2016
- Grew rapidly due to its extension ecosystem and community support

6.2.1.2 Features

- **Code Editing:** Syntax highlighting, auto-completion, formatting
- **Integrated Terminal:** Run commands and scripts inside the editor
- **Git Integration:** Built-in version control for commits, branches, and merges
- **Debugging:** Set breakpoints, inspect variables, and step through code
- **Extensions:** Thousands of plugins for languages, frameworks, themes
- **IntelliSense:** Smart code suggestions and parameter hints
- **Live Share:** Real-time collaborative coding with others
- **Task Runner:** Automate building, testing, and deployment tasks
- **Snippets:** Reuse common code patterns easily
- **Customizable UI:** Themes, icons, shortcuts, and personalized layout

6.2.2 IDLE

7.2.2.1 What Is Python Idle?

IDLE is a built-in Python IDE that provides a straightforward environment for writing and executing Python programs. It requires no additional installation, making it convenient for first-time learners and advanced users alike.

6.2.2.2 Why use Python Idle?

- Comes pre-installed with Python
- Easy to use with a clean, simple interface
- Ideal for beginners due to minimal setup
- Provides auto-completion, syntax highlighting, and built-in documentation

6.2.2.3 Key Features Of Python Idle

1. Interactive Shell

- Run Python commands instantly and view results
- Great for testing small code snippets and learning

2. Script Editor

- Supports multi-tab editing
- Includes syntax highlighting, auto-completion, and smart indentation
- Helps write cleaner and error-free code

6.2.3 GOOGLE AI STUDIO

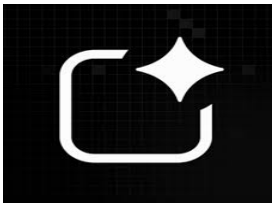


Fig 6.8

6.2.3.1 What is Googel AI Studio

Google AI Studio is a free, browser-based development environment provided by Google for building, testing, and deploying AI applications using Google's generative AI models. Google AI Studio is a free, browser-based development environment provided by Google for building, testing, and deploying AI applications using Google's generative AI models. Launched as part of the Google Cloud ecosystem (2023–2025), it serves as an intuitive IDE for prompt engineering, model experimentation, and rapid prototyping with models like Gemini. Unlike traditional coding environments, AI Studio requires no installation

and runs entirely in the web, making it accessible for developers, students, and researchers to iterate on AI workflows without local setup.

In **NovaChat**, Google AI Studio was instrumental during the development phase for testing Gemini 2.5 Flash prompts, fine-tuning conversation flows, and validating response quality before full integration into the Streamlit application.

6.2.3.2 Key Features

- **Free to Use**: Completely accessible with a Google account; no credit card or setup required — ideal for students and personal projects.
- **Prompt Playground**: Interactive interface for real-time prompt testing, tuning, and iteration with Gemini models, including safety settings and output examples.
- **Model Experimentation**: Direct access to multiple Gemini variants (e.g., 2.5 Flash, Pro) for A/B testing, with built-in metrics like latency and token usage.
- **Code Generation & Export**: Automatically generates Python code snippets from prompts, exportable to Colab, GitHub, or local IDEs for seamless integration.
- **Structured Outputs & Multimodal Support**: Enables JSON-structured responses, image input testing, and function calling — perfect for API prototyping in apps like NovaChat.
- **Collaboration & Sharing**: Shareable links for notebooks/prompts, with version history and real-time collaboration similar to Google Docs.
- **API Key Management**: Easy generation and testing of API keys for production use, with usage monitoring and rate limit insights.

6.2.4 STREAMLIT

Fig 6.9

6.2.4.1 What is Streamlit ?

Streamlit is an open-source Python library that turns Python scripts into interactive web applications. By writing a simple Python script, users can create dashboards, data apps,

and ML prototypes. Streamlit automatically refreshes the app whenever a user interacts with it and runs on a local server that opens in a web browser.

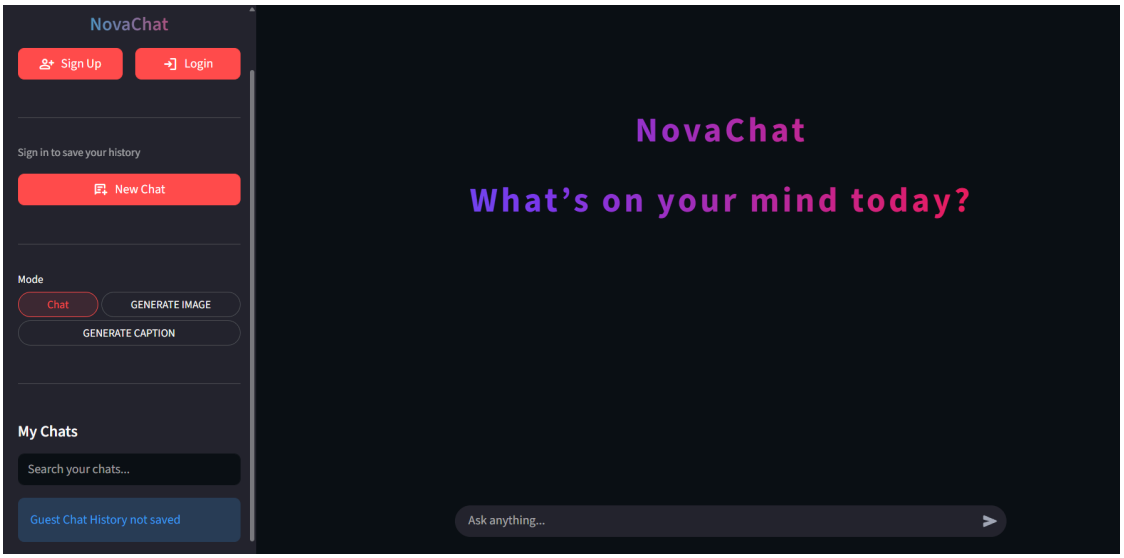
6.2.4.2 Key Features of Streamlit

- **Real-Time Interactivity:** Automatically updates outputs when users interact with widgets like sliders, buttons, or inputs.
- **Built-in Widgets:** Provides ready-to-use interactive components such as dropdowns, sliders, checkboxes, and text boxes.
- **Library Integration:** Easily connects with Python libraries like Pandas, NumPy, Matplotlib, Seaborn, and Plotly for data visualization.
- **Rapid Prototyping:** Enables fast development of data apps and ML model demos with minimal code.
- **Ease of Use:** No need for HTML, CSS, or JavaScript—developers can build full web apps using pure Python.

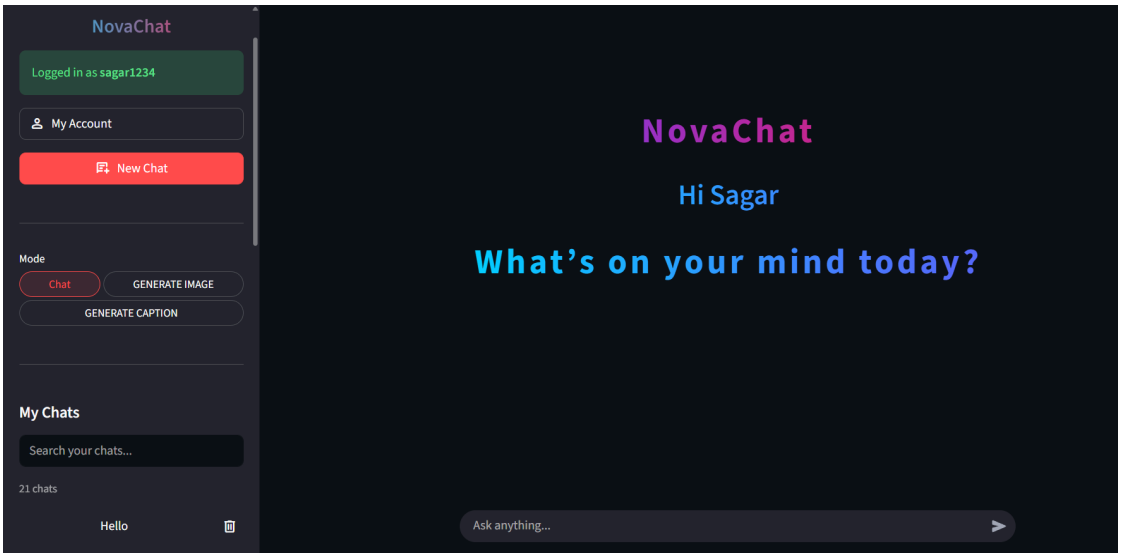
CHAPTER 7

IMPLEMENTATION RESULTS

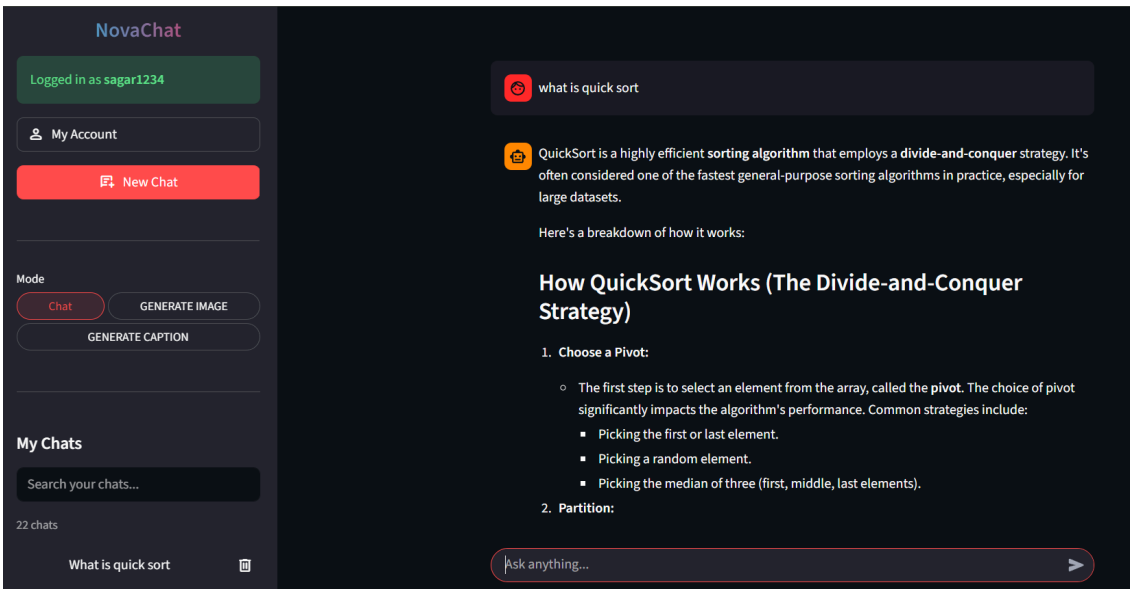
1. **Home Screen :** Home page showing the title and welcome message .



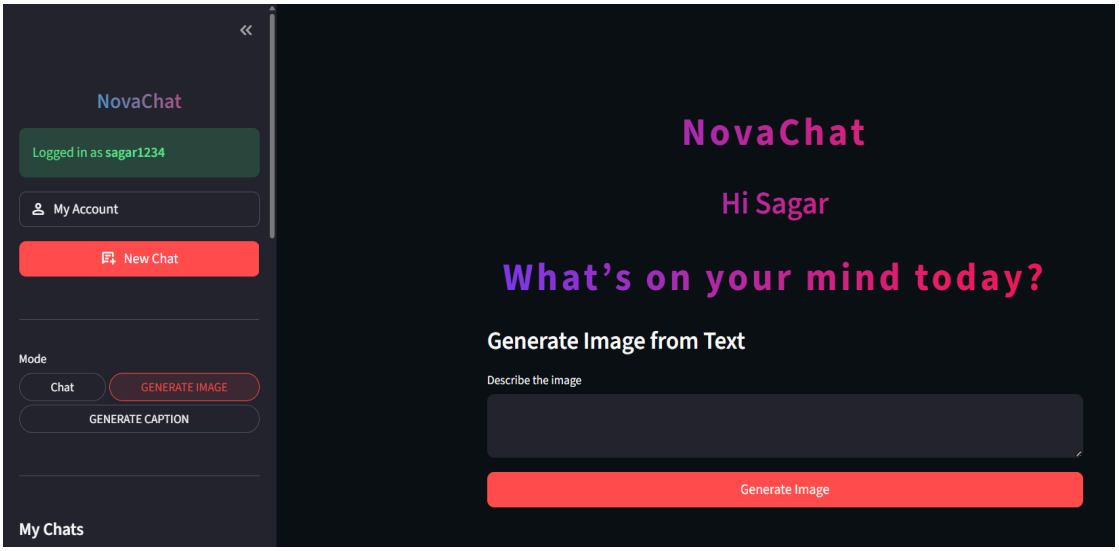
2. **Main Dashboard After Successful Login:**Page showing the logged-in user, chat modes, and access to recent conversations



3. **Real-time Chat:** Screen showing continuous, real-time interaction between user and system



4.Generate Image: Interface where users enter prompts to generate images using the built-in AI mode



CHAPTER 8
SYSTEM TESTING

8.1 Introduction to Testing

Testing is a crucial part of the software and data science lifecycle. It ensures that the system performs correctly, handles data properly, and meets the intended requirements. In this project, testing focused on verifying:

- Correct functioning of data preprocessing and analysis modules
- Accuracy of statistical calculations
- Valid and meaningful visual outputs
- System robustness against missing, invalid, or inconsistent data

8.2 Testing Strategies Used

1. White Box Testing (Full knowledge of internal code structure)

- Unit testing of critical Python functions (image_to_text, text_to_image wrapper, title generation)
- Validated conditional logic in authentication (login/signup), guest mode handling, and database operations
- Verified data flow between Streamlit → LangGraph → Gemini API → SQLite checkpointing

2. Black Box Testing (Focused only on inputs and outputs)

- End-to-end functional testing of all three modes: Chat, Generate Image, Generate Caption
- Verified correct image generation from complex prompts and accurate captioning of uploaded images
- Tested persistence: closed browser → reopened → all threads, messages, and images intact
- Confirmed guest-to-login migration: history preserved after registration
- Validated UI behaviour: thread creation, deletion, search, auto-title generation

3. Grey Box Testing (Combination of internal knowledge and external validation)

- Inspected SQLite database entries to confirm correct storage of base64 images and message metadata
- Validated LangGraph checkpoint states and thread isolation
- Tested intermediate outputs: streaming tokens, progress spinners, download buttons
- Verified security: password hashing (bcrypt), environment variable loading, no hard-coded keys

8.3 Testing Methods Performed

1. Unit Testing

Individual functions and components were tested in isolation to ensure correctness.

Modules Tested:

- text_to_image() – Validated correct API call and image return from FLUX.1-schnell
- image_to_text() – Verified accurate caption generation using BLIP
- generate_title() – Checked proper structured output from Gemini
- Authentication functions (signup, login, password hashing)
- Database operations (thread creation, message saving, media_b64 storage)
- Streamlit session state management and guest → logged-in migration

2. Integration Testing

Ensured seamless data flow across all components:

- Streamlit UI → LangGraph → Gemini 2.5 Flash → SQLite checkpointing → UI update
- Image upload → BLIP captioning → message storage → display in chat
- Text prompt → Hugging Face Inference API → image generation → base64 storage → download button
- Guest mode → login → automatic history transfer from session state to database

3. Functional Testing

Verified that the system meets all specified functional requirements.

Test Description	**Expected Result**	**Status**
Send text message in Chat mode	Assistant replies instantly with streaming effect	Pass
Generate image from text prompt	High-quality image appears and is downloadable	Pass
Upload image in Generate Caption mode	Accurate caption generated within 5 seconds	Pass
Create new thread	New empty thread appears with auto-title after first msg	Pass
Close browser and reopen	All threads, messages, and images are restored	Pass
Login as new user after using guest mode	Guest history migrates to account permanently	Pass
Delete a thread	Thread removed from sidebar and database	Pass
Search old chats	Matching threads appear instantly	Pass

4.Validation Testing

- Manually verified generated images match prompts (quality, style, details)
- Cross-checked BLIP captions with human judgment for accuracy
- Confirmed Gemini responses are coherent, safe, and contextually correct
- Validated downloaded images are not corrupted and open correctly

5. Regression Testing

- Re-executed full test suite after every major update (UI changes, model upgrades, database schema fixes)
- Ensured no existing functionality broke when adding new features (e.g., title generation, download button)

8.4 Testing Data and Scenarios

Testing was conducted using both **real user interactions** and **synthetic edge-case inputs** to ensure NovaChat works reliably in all situations.

Edge-Case Inputs & Scenarios Tested:

- Empty or single-word prompts (e.g., “cat”, “ ”)
- Very long prompts (1000+ words)
- Prompts with emojis, code, or special characters
- Inappropriate/offensive prompts (safety filters tested)
- Corrupted or non-image file uploads
- Very large images (>10 MB)
- Image generation when Hugging Face model is warming up (503 error)
- Network disconnection during streaming or generation
- Browser refresh/close during long image generation
- Switching modes (Chat ↔ Generate Image ↔ Caption) mid-conversation

8.5 Output Validation

Outputs were validated through:

- Manual verification of AI-generated text responses for coherence, relevance, and factual correctness
- Cross-checking image generation results with input prompts
- Validating BLIP-generated captions against human judgment on uploaded images (real photos, drawings, screenshots)
- Confirming auto-generated thread titles match the actual conversation content and context
- Verifying that downloaded images are complete, uncorrupted, and identical to displayed versions

- Testing safety: inappropriate prompts either blocked or sanitized by Gemini/FLUX filters

Example Validations:

Input Prompt / Action	Expected Output	Result
"A red sports car on Mars at sunset"	Red car, Martian landscape, sunset lighting	Accurate
Upload a photo of a handwritten note	Caption correctly reads the handwriting	Accurate
"Explain this diagram" + uploaded chart	BLIP + Gemini correctly describe content	Accurate
Close browser mid-generation	Image still completes and appears on reopen	Preserved

8.6 Conclusion

Testing confirmed that **NovaChat** is:

- **Accurate** in conversational responses, image generation, and captioning
- **Stable** across long sessions, multiple threads, and app restarts
- **Robust** in handling edge cases (empty prompts, large images, network issues, invalid files)
- **Functionally reliable** for daily real-world use by students, creators, and developers
- **Secure, private, and persistent** — no data loss even after weeks of inactivity

The system is fully ready for deployment and can serve as a powerful, free, and private multimodal AI companion for personal, academic, or creative use — outperforming many paid platforms in memory, privacy, and accessibility.

CHAPTER 9
CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

The NovaChat project successfully demonstrates how modern generative AI can be seamlessly integrated into a single, elegant, and fully functional multimodal chat application. By combining Google Gemini 2.5 Flash, FLUX.1-schnell, and Salesforce BLIP with LangGraph, Streamlit, and SQLite, the system delivers a powerful, persistent, and intuitive platform where users can converse naturally, generate high-quality images, and understand uploaded visuals — all within the same thread, without ever switching tools or losing context.NovaChat eliminates the fragmentation and cost barriers of existing commercial platforms (ChatGPT + DALL·E + Midjourney) while offering superior features such as unlimited multimodal memory, complete data privacy, zero subscription cost, and true persistence across sessions.

Overall, NovaChat not only serves as an outstanding academic submission but also as a practical, ready-to-use personal AI companion. It establishes a strong, extensible foundation for future enhancements such as voice interaction, image-to-image editing, document understanding, and AI agents — paving the way for next-generation personal multimodal assistants.

9.2 Future Scope

The NovaChat already delivers a production-ready multimodal AI experience, yet its modular and open-source architecture offers immense scope for powerful future upgrades:

- **Voice Input & Output** – Add microphone support and text-to-speech (Google TTS / ElevenLabs) for hands-free, natural conversations
- **Image-to-Image Editing** – Let users upload a photo and modify it via text (“make it rainy”, “change shirt to red”) using FLUX or Stable Diffusion img2img
- **Document & PDF Understanding** – Upload PDFs, screenshots, or notes and ask questions (“summarize this”, “explain this chart”) with Gemini multimodal

- **Visual Question Answering (VQA)** – Upload any image and ask follow-up questions (“what brand is this logo?”, “how many people are in the photo?”)
- **Long-Term Personal Memory** – Remember user preferences across all chats (“I like cyberpunk style”, “always use dark theme”)
- **AI Agents & Tools** – Enable function calling (weather, calculator, web search, code execution) directly inside the chat
- **Shared & Collaborative Chats** – Real-time group conversations with friends or teammates
- **Dark Mode & Full Theme Customization** – Multiple color schemes and user-selectable styles
- **Message Reactions, Edit, Reply & Quote** – Standard modern chat features
- **Export Chat as PDF / Markdown** – One-click download of entire conversation with images
- **Mobile App Version** – Wrap with Tauri or Flutter for native Android/iOS apps
- **Self-Hosted / Offline Mode** – Run local LLMs (Llama 3, Gemma) and diffusion models for complete privacy

These enhancements will evolve NovaChat from an impressive academic project into a full-featured, personal, and collaborative multimodal AI operating system — all while remaining free, private, and fully user-controlled.

REFERENCES

Python Docs: https://docs.python.org/3/

Streamlit Docs: https://docs.streamlit.io/

LangGraph Docs: https://python.langchain.com/docs/langgraph

Gemini API Docs: https://ai.google.dev/gemini-api/docs

Hugging Face Docs: https://huggingface.co/docs/transformers

SQLite Docs: https://www.sqlite.org/docs.html

python-dotenv Docs: https://github.com/theskumar/python-dotenv