Bahria University,

Karachi Campus

A logo with text on it

AI-generated content may be incorrect.

COURSE: CSL-210 OBJECT ORIENTED PROGRAMMING

TERM: SPRING 2025, CLASS: BSE- 2 (A/B/C)

PROJECT NAME

NEO AURA

Submitted By:

|  |  |  |
| --- | --- | --- |
| S.NO | Student Name | Enrollment # |
| 1 | M.WALEED AHMED | 119 |

Submitted to:

Engr.Muhammad Faisal/ Engr. Saniya Sarim

# Table of Contents

1. Introduction

2. Requirements Analysis

3. System Architecture Design

4. Software Development Methodology

5. Risk Management

6. Testing and Validation

7. Documentation and Maintenance  
 WBS   
 Usecase Diagram  
 Flowchart  
 UML diagram   
 Context Diagram

8. Gant Chart  
9. Conclusion

# 1. Introduction

NeoAura is an innovative AI Bot Management System designed for centralized communication with multiple AI bots on a single platform. Built using Java Spring Boot for the backend and JavaFX for the desktop frontend, NeoAura allows users to interact with AI models like ChatGPT, DeepSeek, Bard, Jasper AI, and CodeT5. This document presents a complete software development plan including architecture, methodology, testing, risk management, and maintenance strategy.  
**Features**

USERS:

SIGN UP  
Forget password  
login  
Multiple AI bots  
Real time Poll  
Attractive UI  
Dashboard Features

ADMIN:

Login

User management  
Bot management  
check Bot efficiency  
provide multiple Bot

Realtime Poll creation and deletion

Interactive UI  
  
  
**OOP Concept used in this project**

Exception handling  
. Encapsulation

* Private Fields: All classes use private fields to hide internal state (e.g., JButton, JPanel, and domain-specific fields).
* Public Methods: Controlled access via public methods (e.g., getMainpanel(), updateUserStatus()).
* Data Hiding: Database credentials are encapsulated in Database (DB\_URL, DB\_USER, DB\_PASSWORD are private static final).

2. Inheritance

* Class Extension:
  + Adminpoll extends JPanel.
  + Anonymous inner classes extend ActionListener/ComponentAdapter for event handling.
* Method Overriding:
  + paintComponent() in AdminInterface (custom rendering).
  + actionPerformed() in listeners.

3. Polymorphism

* Interface Implementation:
  + ActionListener (e.g., button clicks).
  + ComponentAdapter (e.g., componentShown() for UI updates).
* Method Overloading:
  + CommonMethods has multiple dimension-calculating methods (e.g., screensize(), halfbuttonpanel()).

4. Abstraction

* Abstract UI Components: Swing classes (JFrame, JPanel) abstract low-level rendering.
* Database Abstraction: Database class hides SQL details behind methods like loginAdmin(), getAllPolls().

5. Association & Composition

* Composition:
  + AdminInterface *has-a* Dashboard, User, and Poll panels.
  + Adminpoll *has-a* JTextField[] for options.
* Aggregation:
  + AdminInterface uses Database to fetch user data.
  + AIChat depends on Google's Client class.

Object creation  
Classes  
Event handling

# 2. Requirements Analysis

**Functional Requirements:**

- Multi-Bot Interaction: Chat with up to 5 AI bots.

- AutoBot: Intelligent bot selection based on user queries.

- Gamified Polls & Case Studies: Enhance user engagement and market research.

- Admin Control Panel: Track user sessions, generate reports (PDF/CSV), create/manage polls.

- Chat History & Badges: Users can view chat logs and earn achievement badges.

**Non-Functional Requirements:**

- Performance: Fast response time for real-time conversations.

- Security: Role-based access for admin and users.

- Scalability: Easy integration of new AI models.

- Maintainability: Modular design using OOP concepts.

- Usability: Intuitive JavaFX interface for desktop usage.

# 3. System Architecture Design

**High-Level Design:**

Frontend: JavaFX-based desktop interface using FXML.  
Backend: RESTful Spring Boot APIs.  
Database: PostgreSQL (custom port 5000).  
Bot Interface Layer: Abstract class APIHandler handles interaction logic for all bots.  
Services: UserService, AdminService, PollService, ReportService, BotService, AutoBotService.

**System Components:**

- User Module: Chat, history, polls, badges.

- Admin Module: Session monitoring, analytics, reports.

- Bot Integrations: ChatGPT, DeepSeek, Bard, Jasper, CodeT5.

**Communication:**

Internal: REST API via Spring Controllers and Services.  
External: API calls to AI bot endpoints using JSON over HTTPS.

# 4. Software Development Methodology

**Chosen Method: Agile**

Due to frequent updates, multiple module interactions, and integration with external APIs, Agile allows flexibility and rapid iteration, ideal for evolving requirements.

**Development Process:**

Sprints: 4 total (2 weeks each)  
1. UI & ChatGPT Integration  
2. Multi-bot setup  
3. Admin module and analytics  
4. Testing and optimization  
Milestones: UI prototype, Bot functionality, Admin tools, Deployment-ready build  
Version Control: GitHub repository for code collaboration and version tracking.

# 5. Risk Management

**Potential Risks & Mitigation:**

- API Downtime: Add retry/fallback logic.

- Data Loss: Scheduled PostgreSQL backups.

- Poor UI Performance: Optimize JavaFX rendering.

- Feature Creep: Lock core features during sprint planning.

- Query Misrouting by AutoBot: Enhance NLP & test edge cases.

# 6. Testing and Validation

**Testing Types:**

- Unit Testing: Individual service classes using JUnit.

- Integration Testing: Verify API calls and service layers.

- System Testing: Full workflow including chat and report generation.

- Acceptance Testing: Simulate real-world user behavior.

**Testing Environments:**

Local Development with PostgreSQL and mock APIs.  
Pre-Deployment testing on production-like environment.

**Acceptance Criteria:**

- All five bots respond correctly.

- Reports are generated in CSV/PDF.

- Admin panel shows real-time updates.

- No crash/freeze in UI after continuous use.

# 7. Documentation and Maintenance

**Documentation Includes:**

- User Manual: Basic guide to use NeoAura chat and features.

- Admin Manual: How to track users, generate reports, and create polls.

- API Docs: Swagger/Postman collections with all endpoints.

- System Design: Architecture diagrams and service descriptions.

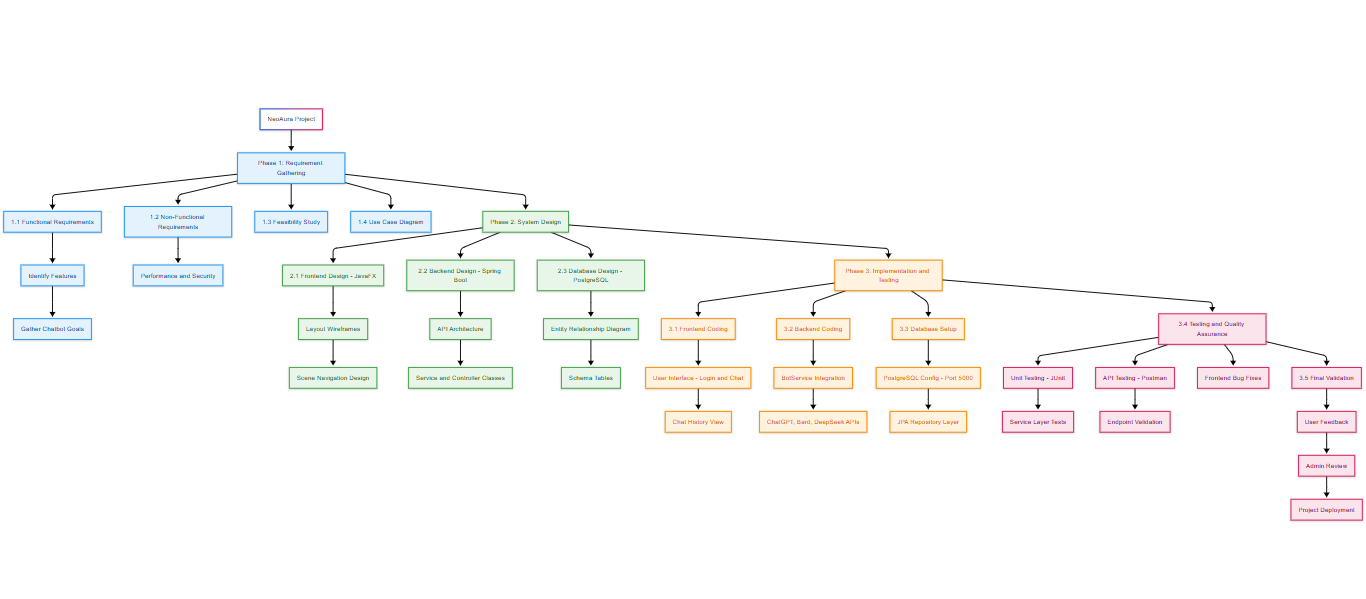
**Maintenance Strategy:**

- Weekly Bug Fixing: Sprint-based fixes after user testing.

- Feature Upgrades: Logged via user feedback form.

- Continuous Integration: Automated builds and testing for new releases.

**7.1 WBS DIAGRAM**



**7.2 USE CASE DIAGRAM**

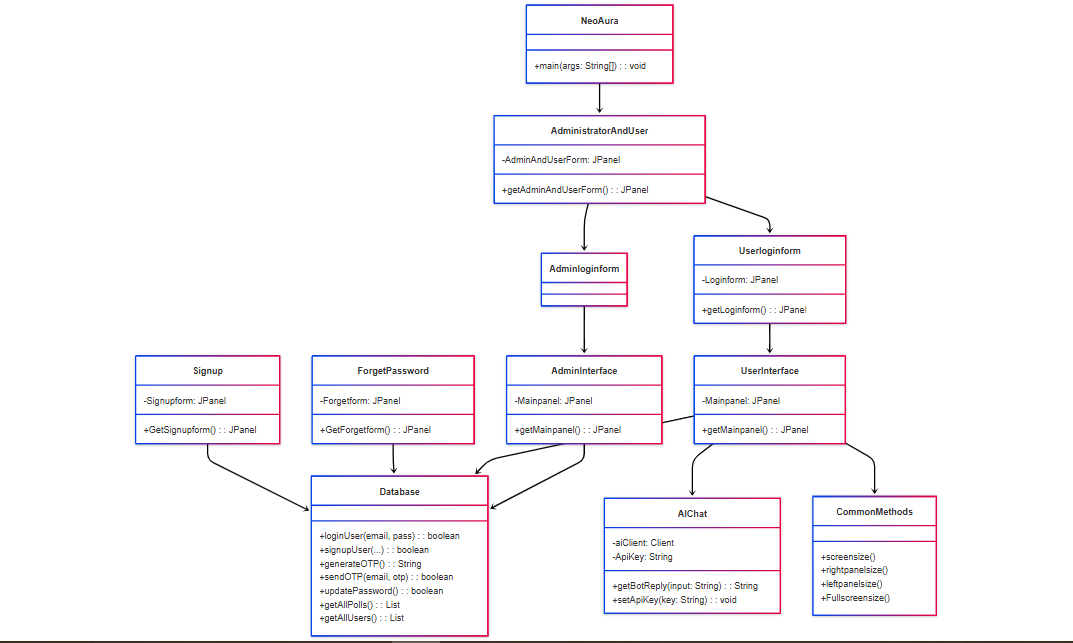
A diagram of a computer network

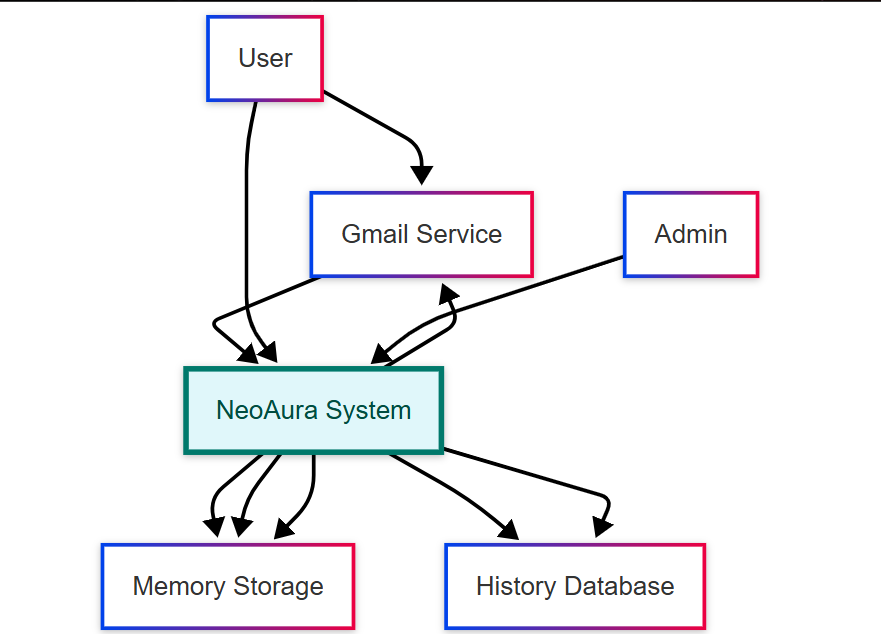
AI-generated content may be incorrect.

**7.3 FLOWCHART**

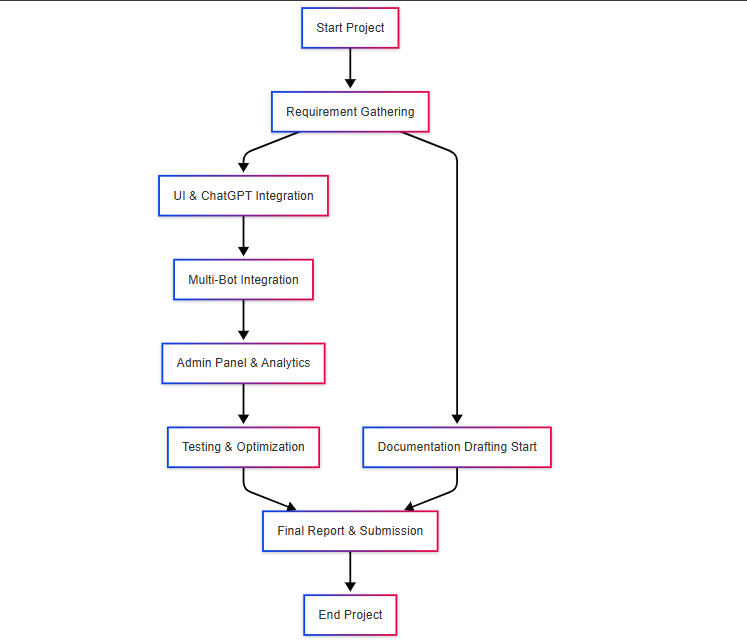
A diagram of a diagram

AI-generated content may be incorrect.

**7.4 UML DIAGRAM  
**

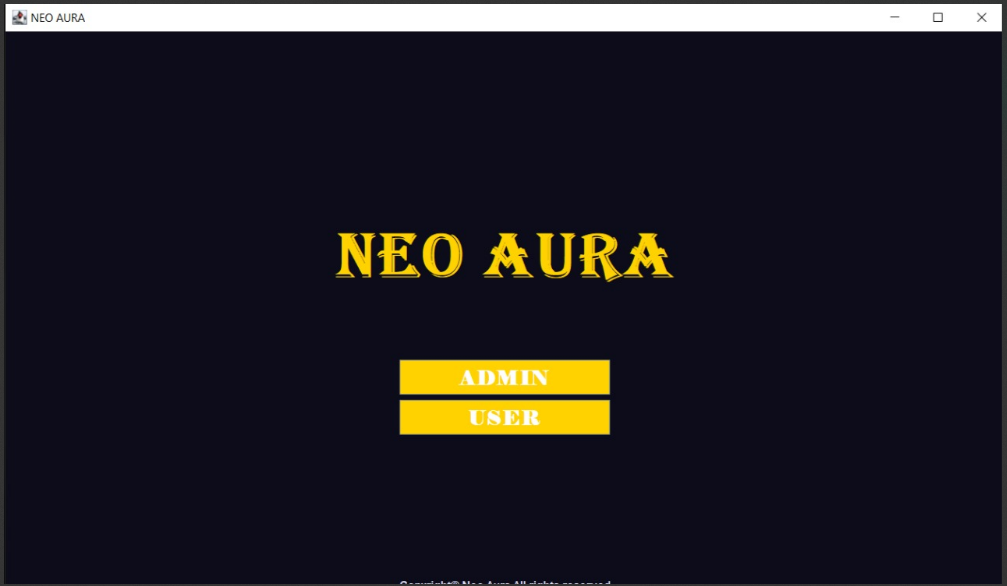
**7.5 CONTEXT DIAGRAM  
**

**7.6 Pert Chart**

****

**8. OUTPUTS**

**MAIN SCREEN**

****

**ADMIN LOGIN SCREE**

**A screenshot of a computer

AI-generated content may be incorrect.**

**ADMIN INTERFACE**

**A screenshot of a computer

AI-generated content may be incorrect.**

# 

**ADMIN ACCOUNT CONTROL  
A screenshot of a computer

AI-generated content may be incorrect.**

# 

# ADMIN POLLS CREATION SCREEN

**A screenshot of a computer

AI-generated content may be incorrect.**

# USER LOGIN SCREEN

# A screenshot of a computer AI-generated content may be incorrect.

# USER FORGET PASSWORD SCREEN A screenshot of a computer AI-generated content may be incorrect.

# USER SIGN UP SCREEN

**A screenshot of a login screen

AI-generated content may be incorrect.**

**USER INTERFACE**

**A screenshot of a computer

AI-generated content may be incorrect.**

**USER CHAT SCREEN**

**A screenshot of a computer

AI-generated content may be incorrect.**

**USER BOT SELECTION SCREEN**

**A screenshot of a computer

AI-generated content may be incorrect.**

**USER POLL SCREEN**

**A screenshot of a computer

AI-generated content may be incorrect.**

# 9. Conclusion

NeoAura offers a centralized solution for managing multi-AI chat environments with both user and admin perspectives. Using a strong object-oriented Java base, clean modular architecture, and agile development, NeoAura balances performance, scalability, and usability. The outlined plan ensures not only a robust system but also a maintainable and extensible platform for future enhancements.