CSC1049 Year 3 Project

Functional Specification

Project Title: Finance Coach

Student(s) Name and ID

Aaron O'Shea	**23368161**
Anthony Chukwuemeka Kalu	**23395331**

Supervisor Name: Irina Tal

Module Coordinator: Dr. David Sinclair

Date: 28 November 2025

Table of Contents

1. Introduction

1.1 Overview

The AI-Powered Finance Coach is a mobile application designed to improve financial literacy for students and young adults through interactive lessons, budgeting tools, and a personalized AI tutor. Research indicates that many individuals lack basic financial management skills, such as budgeting, saving, and understanding financial terminology. Traditional financial education is often complex, inaccessible, or unengaging. This system aims to address these challenges by providing a practical, simple, and adaptive learning experience.

The product combines three core components:

1. **Budgeting Module** - where users manually input income, expenses, and financial goals.
2. **Interactive Learning System** - educational modules and quizzes covering essential financial topics.
3. **AI tutor** - a personalized assistant that uses user progress and budgeting patterns to give tailored financial guidance

The system will be built using Flutter for the mobile interface and Django for the backend. The AI tutor will be powered by an external LLM API (e.g., OpenAI API or Microsoft Phi-3), but only receives anonymized user lessons and budgeting summaries. No banking or sensitive financial data is handled.

The application operates independently, though it may optionally integrate with open-banking APIs in the future.

1.2 Business Context

This system is intended for deployment in educational environments, personal finance learning contexts, or as a standalone budgeting and financial literacy tool. It addresses a clear need for accessible financial education among students, young professionals, and adults who want better control over their finances.

While this project is developed in an academic context (School of Computing, Dublin City University), the product has potential for real-world use by:

- Universities providing financial literacy support,
- Financial well-being programs,
- Budgeting and personal development platforms,
- Young adults seeking beginner-friendly financial education tools.

There is no external business sponsor; however, the system aligns with industry trends toward AI-assisted financial planning and educational mobile applications.

2. General Description

2.1 Product / System Functions

The AI-Powered Finance Coach provides the following core functions:

- **Budget Creation:** Users set monthly income, spending categories, and savings targets.
- **Expense Tracking:** Users manually enter expenses which are categorized and stored.
- **Interactive Lessons:** The system includes short, module-based lessons on budgeting, saving, financial terms, and basic investing.
- **Quizzes and Assessments:** Users complete short quizzes that evaluate their understanding.
- **AI-Powered Feedback:** The AI tutor provides explanations, personalized lesson recommendations, and spending insights.
- **Progress Tracking:** Visual dashboards show user performance, spending patterns, and completed lessons.
- **Secure User Accounts:** Authentication and data storage via Django's backend system.

Initially, no external financial API is used; all budgeting data is user-entered to ensure safety, simplicity, and privacy.

The AI-Powered Finance Coach relies on two primary categories of financial data: educational financial content and user-provided budgeting data. The learning modules and interactive lessons use publicly available, non-sensitive educational material sourced from reputable financial literacy resources such as government consumer finance portals, banking education sites, OECD financial literacy frameworks, and publicly accessible financial knowledge platforms such as Investopedia. For simple market-related lessons (e.g., "What is a stock?", "How does crypto volatility work?"), the system may also retrieve general, non-personalised market data from free public APIs such as Yahoo Finance, Alpha Vantage, or CoinGecko. This data is used purely for explanatory and educational purposes and does not provide trading or investment advice.

User-specific financial data comes only from information the user manually inputs into the application, such as income, expenses, budgeting categories, savings goals, and lesson or quiz progress. This information is stored securely in the local application database and is not shared with any third-party financial service. When users interact with the AI tutor, the system only sends anonymised, aggregated insights (for example: "user spends 30% on food" or "user struggles with saving goals") to the AI provider. No names, raw expenses, or personal identifiers are ever transmitted. This ensures that all personalised recommendations remain private, ethical, and compliant with standard data protection expectations. The system therefore combines curated educational content, safe API-based general market data, and locally stored user inputs to provide personalised and responsible financial guidance.

2.2 User Characteristics and Objectives

User Group

The main users include:

- Students aged 16–25
- Young adults with limited financial knowledge
- Individuals seeking to improve budgeting habits
- Beginners to personal finance

User Expertise

- Basic mobile app skills
- Limited or no understanding of financial concepts
- No technical background required

User Objectives

Users aim to:

- Learn how to budget effectively
- Track and manage their expenses
- Improve financial literacy
- Understand basic investing concepts
- Receive personalized guidance on spending habits
- Develop better financial decision-making skills

Desirable (Wish-List) Features

- Automatic bank transaction imports (future enhancement)
- Gamification features (badges, streaks)
- Personalized learning paths
- Voice-based AI guidance
- Multi-language support

Feasible Features for This Project

- Manual budgeting and expense tracking
- Interactive educational modules
- AI-powered personalized explanations
- Secure authentication
- Progress dashboard

2.3 Operational Scenarios

Scenario 1: Creating a Budget

A user launches the app for the first time, enters their monthly income and selects budget categories. They assign spending limits, and the app generates a visual budget overview.

Scenario 2: Adding an Expense

The user buys groceries and opens the app to log the expense (€20). The system updates their Food category and shows the remaining budget for the month.

Scenario 3: Completing a Lesson

The user starts the “Budgeting Basics” lesson, reads the material, and completes a 5-question quiz. The app stores their score and updates their progress.

Scenario 4: Using the AI Tutor

The user types: *“How do I improve my savings?”*

The backend retrieves the user’s spending pattern (e.g., overspending on food) and sends it to the AI API.

The AI responds:“You spent €180 of your €150 food budget. Reducing takeaway purchases could help you save €30–€40 monthly.”

Scenario 5: Setting a Goal

The user sets a goal to save €200 this month. The app tracks user progress and the AI offers motivational tips.

2.4 Constraints

- **Technical Constraints:**
    - Mobile app built in Flutter (Dart)
    - Backend implemented using Django (Python)
    - SQL database for data persistence
    - AI requires stable internet connection to access external LLM API
- **Privacy & Security Constraints:**
    - No handling of sensitive bank credentials
    - All personal data must comply with GDPR
    - AI receives only anonymized spending summaries
- **Resource Constraints:**
    - Limited development time (academic project)
    - No requirement for embedded banking APIs
- **Performance Constraints:**

- AI responses must be near real-time
- App should run smoothly on standard iOS/Android devices

3. Functional Requirements

3.1 User Authentication and Account Management

Functional requirement 1: User Registration

Description: Our system must allow new users to create their account using email address and a password. The registration process must validate the email address format, enforce password strength requirements (minimum 8 characters and a capital letter), and also be able to store credentials safely in the database.

Criticality: Very High, essential for our system to function properly. Without user accounst, personalization, tracking of progress and AI recommendations cannot be provided.

Technical Issues:

Password: Users password must be hashed using a secure algorithm (Django)

Email Validation: To prevent duplicate accounts

Email Verification: To prevent fake accounts being created

Dependancies: None, This is a foundational requirement which other features depend on

3.2 User Login

Functional Requirement 2: User Login/Session Management

Description: Our system must allow registered users to log in using their email address and password. After successful authentication, our system must create a secure session that continues until the user logs out of the application or the session expires.

Criticality: High, Required for accessing personal information/features and maintaining the user's data securely.

Technical Issues:

Session Token Management: Between the frontend and backend Flutter - React.

Token Expiration/ Refresh Mechanism

Secure Storage of Session Tokens on mobile device

Failed Login Attempts: Lockout after multiple failed attempts.

Dependancies: **Functional Requirement 1**

3.3 Core Budgeting Functionality

Functional Requirement 3: Budget/Expense Tracking

Description: Our system must allow users to enter their monthly/yearly income, add their expenses manually with the appropriate amount and what category. Also, the user must be able to set spending limits for each category. Our system must track expenses against the user-set limits, calculating the budget in real-time, and then alert the user when approaching/exceeding limits.

Criticality: High. This functionality enables financial tracking and user control.

Technical Issues:

Input Validation: Income and expense amounts.

Real-Time budget calculations across categories

Database schema design: Income/expenses and category limits.

Notification System: For budget thresholds.

Dependancies: **Functional Requirement 1 & 2**

3.4 Education Content/Learning

Functional Requirement 4: Interactive Educational Content

Description: Our system must provide educational lessons on financial topics including the basics of budgeting, strategies to save money as well as basic investing concepts. Each lesson module must include readable content with examples. Quizzes will also be included, and the system will score automatically based on user responses. These quizzes will provide immediate feedback to the user, and track their progress.

Criticality: High. This is a core feature that differentiates our application from others.

Technical Issues:

Content Management System: For storing/organizing lesson modules in the database.

Design of the Quiz Database: Questions/Answers/Explanations.

Scoring Algorithm: Automatic quiz grading and pass/fail determination.

Progress Tracking: Recording lesson states, and past quiz scores.

Dependancies: Functional Requirement 1, requires an authenticated user.

3.5 AI Powered Features

Functional Requirement 5: AI Powered Personalized Tutor

Description: Our system must integrate with an external LLM API such as OpenAI or Microsoft Phi-3 to provide an AI tutor that answers financial questions asked by the user in natural language. The AI must analyse user spending patterns, their adherence to budgets, and their learning progress. The AI will then suggest specific spending adjustments, tips on saving and relevant lesson recommendations. For privacy, only anonymized, aggregated data will be transmitted to the AI provider. No personally identifiable information will be sent.

Criticality: High. This is a key differentiating feature.

Technical Issues:

LLM API Integration: Authentication and secure API key management.

Data Anonymization Pipeline: Aggregating and anonymizing user data before transmission.

Response Time Optimization: Ensuring Real-Time responses.

Error Handling: Managing API downtime, timeout errors and invalid responses.

Privacy Compliance: Audit logging of all data transmitted to external API's.

Dependancies: Functional Requirement 1, 3 and 4

3.6 Data Visualization and Goals

Functional Requirement 6: Visual Dashboard/Financial Goal Tracking

Description: Our systems must provide a comprehensive visual dashboard which displays user income and expenses by category, their current budget status and spending trends. Users must be able to set savings goals with specific target amounts and deadlines, and the dashboard must display goal progress as a percentage of completion. All visualizations must update in real time.

Criticality: High. Essential for users to quickly understand their financial situation.

Technical Issues

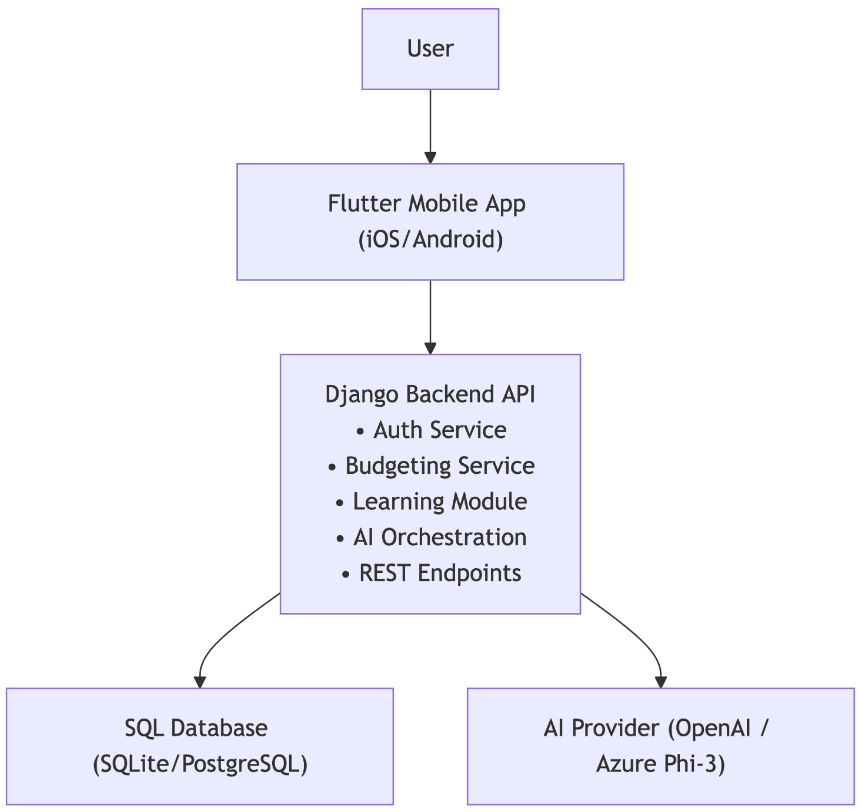
Chart Rendering: Implementing Charts using Flutter’s libraries for finance tracking.

Performance Optimization: Efficient real time calculations and rendering.

Data Refresh: Implementing pull to refresh and automatic updates.

Dependancies: Functional Requirement 1, 3 and 4

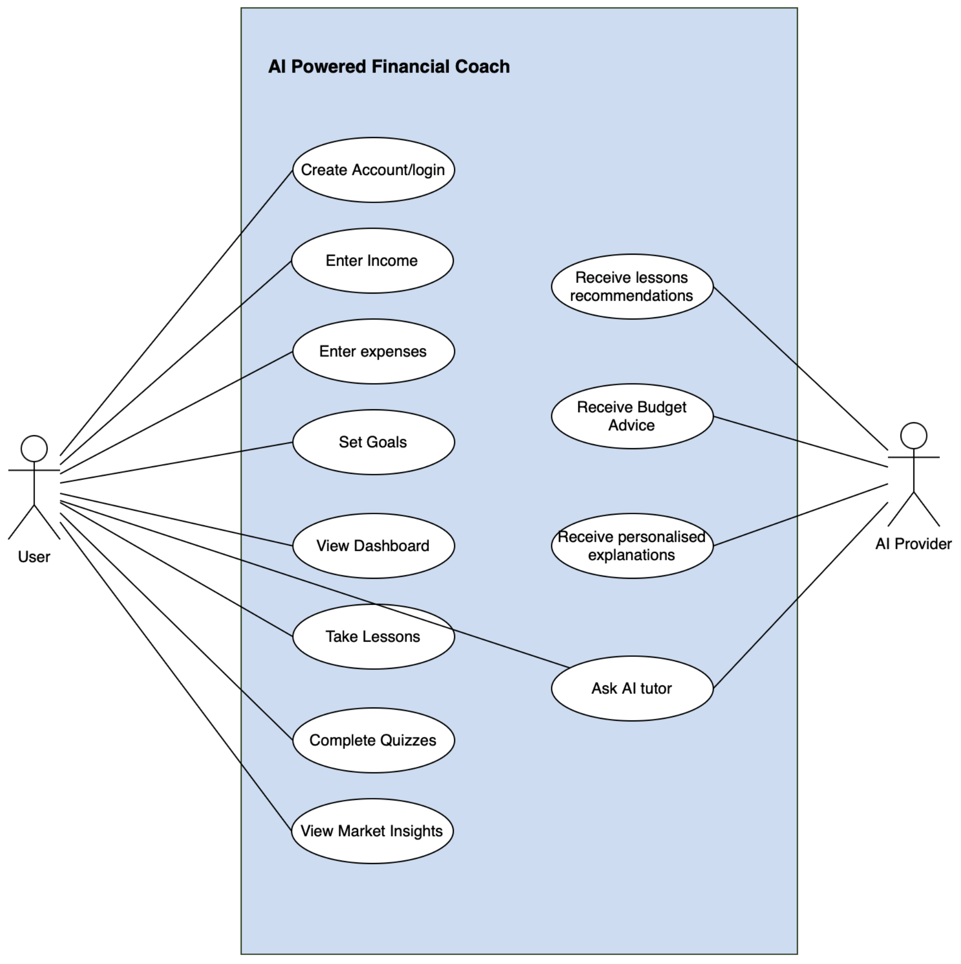
4. System Architecture



5. High level design

5.1 Use cases

The following section outlines the key use cases for the AI-Powered Finance Coach system. Use cases describe the interactions between external actors and the system in order to achieve specific goals. They help define the functional requirements of the application by identifying what the system must allow users and external services to do. Each use case captures a distinct piece of functionality, such as budgeting, learning, or interacting with the AI tutor, and illustrates the sequence of actions required to complete that task. Together, these use cases provide a comprehensive overview of the system's expected behaviour and serve as the foundation for later design, implementation, and testing activities.



Use case 1: Create Account / Login	
Field	**Description**
Primary Actor	User
Secondary Actor	None
Description	User creates an account or logs into the system to access personalised features
Precondictions	User has a valid email or device
Criticality	High
Dependencies	None

Use case 2: Enter Income	
Field	**Description**
Primary Actor	User
Description	User enters their monthly/weekly income into the budgeting system
Preconditions	User is logged in
Postconditions	Income is stored in the database
Criticality	High
Dependencies	UC1

Use case 3: Enter Expenses	
Field	**Description**
Primary Actor	User
Description	User records expenses manually and categories them
Preconditions	User is logged in
Postconditions	Expense is saved and included in dashboard calculations
Criticality	High
Dependencies	UC1, UC2

Use case 4: Set Financial Goals	
Field	**Description**
Primary Actor	User
Description	User defines savings or budgeting goals
Preconditions	User is logged in. Income entered
Postconditions	Goals stored in the database
Criticality	Medium
Dependencies	UC1, UC2

Use case 5: View Budget Dashboard	
Field	**Description**
Primary Actor	User
Description	System displays budgets, spending progress, and financial summaries.
Preconditions	User has entered income/expenses
Postconditions	User sees updated dashboard
Criticality	High
Dependencies	UC2, UC3

Use case 6: Take Interactive lessons	
Field	**Description**
Primary Actor	User
Description	User reads lesson content on budgeting, saving, and markets.
Preconditions	User is logged in
Postconditions	Lesson progress stored.
Criticality	High
Dependencies	UC

Use case 7: Complete Quizzes	
Field	**Description**
Primary Actor	User
Description	User answers lesson questions to test their understanding
Preconditions	Lesson opened
Postconditions	Quiz scores recorder
Criticality	Medium
Dependencies	UC6

Use case 8: Receive Lesson Recommendations	
Field	**Description**
Primary Actor	User
Secondary Actor	AI provider
Description	System recommends lessons based on profile, quiz results, or spending behaviour.
Preconditions	User has completed lessons/quizzes
Postconditions	Recommended content shown
Criticality	Medium
Dependencies	UC6, UC7

Use case 9: Ask AI Tutor Questions

Field	**Description**
Primary Actor	User
Secondary Actor	AI Provider
Description	User asks a financial question and receives an AI-generated explanation
Preconditions	User is logged in
Postconditions	AI response displayed
Criticality	High
Dependencies	UC1

Use case 10: View market Insights

Field	**Description**
Primary Actor	User
Secondary Actor	Market Data API
Description	User views simple educational market insights retrieved via API
Preconditions	Internet connection
Postconditions	Market data shown
Criticality	Low
Dependencies	UC1

6. Preliminary Schedule

This section outlines the main phases for developing our AI Powered Finance Coach Application and an indicative timeline.

Major Tasks and Timeline

Submission of Project Proposal - 17th October 2025

Presentation of Project Proposal - 31st October 2025

Submission of Functional Specification - 28th November 2025

Backend Implementation (Django) - 28th November 2025 - 1st January 2026

Mobile App Implementation (Flutter) - 28th November 2025 - 1st January 2026

AI Tutor Integration - 28th November - 15th January 2026

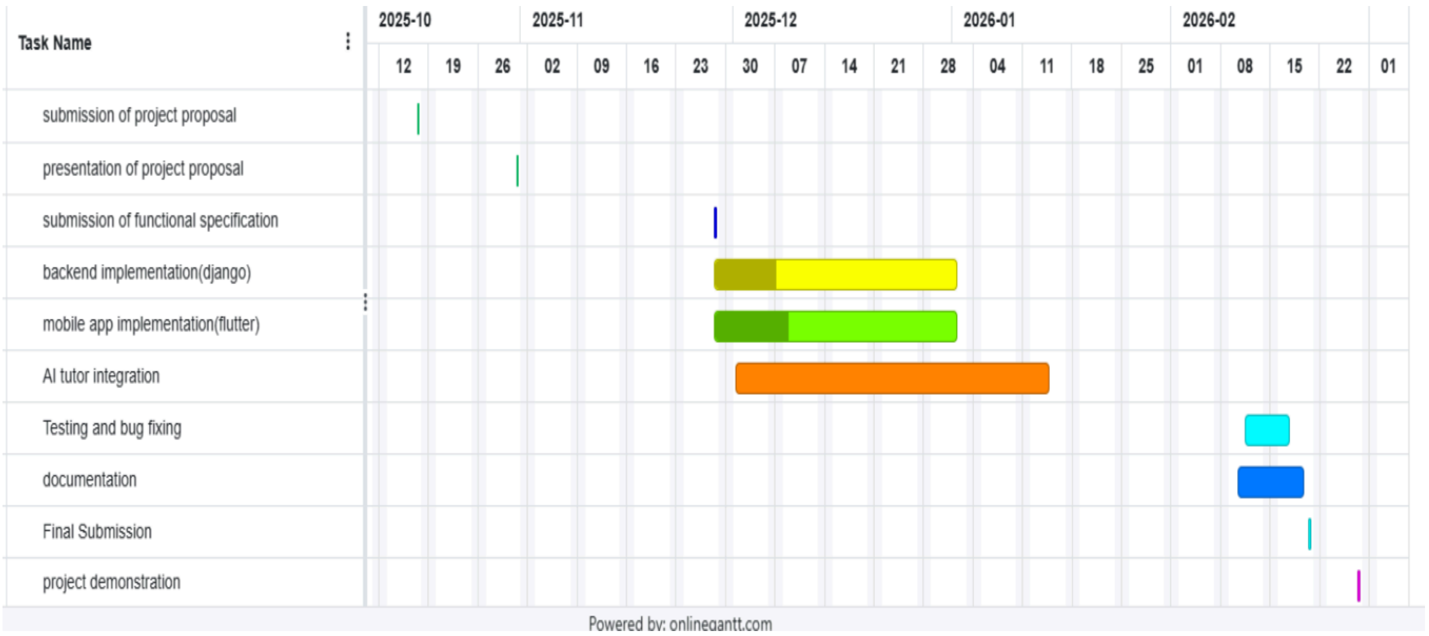
Testing and Bug Fixing - 11th February 2026 - 17th February 2026

Documentation - 8th February 2026 - 19th February 2026

Final Submission - 20th February 2026

Project Demonstration - 27th February 2026

Below is a Gantt chart illustrating these tasks, durations, overlaps and dependencies.



7. Other Nonfunctional Requirements

7.1 Performance Requirements

- Dashboard and budgeting data should load within 2 seconds.
- AI Tutor responses should return within 3–5 seconds.
- Backend API responses should complete within 300–500 ms.
- The app must run efficiently on standard iOS and Android devices.

7.2 Safety Requirements

- All financial content must remain educational only, not investment advice.
- Data must not be lost during normal use; integrity checks required.
- System must prevent harmful or misleading AI output through prompt filtering.
- App must fail gracefully with user-friendly error messages.

7.3 Security Requirements

- All communication must use HTTPS/TLS encryption.
- User passwords must be hashed using Django’s secure authentication system.
- Only anonymised summaries may be sent to AI providers, never raw financial data.
- Users must be able to delete their account and associated data.
- Access to personal data requires authentication; no third-party sharing.

7.4 Software Quality Attributes

- Usability: Simple, intuitive UI for beginners.
- Reliability: Aim for high availability and stable backend operation.
- Maintainability: Modular code structure for easy updates.
- Portability: Cross-platform (iOS/Android) via Flutter.
- Scalability: Should handle more lessons, users, and data without redesign.

7.5 Business Rules

- Only registered users can use budgeting and AI features.
- Users must manually enter income and expenses (no bank connections).
- AI outputs must be labelled as educational guidance.
- Lesson recommendations depend on user progress and quiz performance.
- Admins manage lesson and quiz content; users cannot modify it.

8. Appendix

Appendix 1 - Resources

- **Similar Sites:**
  - https://mint.intuit.com/
  - https://www.investopedia.com/
  - https://web.meetcleo.com/

- **Research Tools**:
  - https://www.sqlcourse.com/
  - https://www.coingecko.com/en/api?utm_source=chatgpt.com
  - https://www.khanacademy.org/college-careers-more/personal-finance

Appendix 2 - Glossary

Term	Definition
AI Tutor	An artificial intelligence system that provides personalized explanations and recommendations based on user data.
Budgeting Module	The part of the system where users enter income, expenses, and goals.
LLM (Large Language Model)	An AI model capable of understanding and generating natural language.
Financial Literacy	Understanding money management, including budgeting, saving, and basic investing.
User Progress Data	Data stored about completed lessons, quiz scores, and learning performance.
Manual Financial Entry	User-provided input of expenses and income, as opposed to automated bank imports.
Open Banking API	APIs (e.g., Plaid, TrueLayer) that allow apps to access financial transactions with user consent.
REST API	A standard method for communication between the mobile app (frontend) and the server (backend).
Database Schema	The structure of the stored data (tables, fields, and relationships).

Data Anonymisation	Removing personal identifiers from data before sending it to the AI model.