ZAMAN AI Banking Assistant (Iris / Идрис)

Первый исламский цифровой ассистент Казахстана

Технический отчёт / Хакатон ZAMAN BANK

Команда:

Aknur Turakhan — Frontend & System Architecture
DS

Timur Chuiko — Data Engineer
Backend Developer
DS

Nursaya Serikova — Prompt Engineer
ML Engineer
DS

Dana Niyetkliyeva — Backend Developer
Data Engineer
DS

GitHub репозиторий: https://github.com/YourTeamRepo

Октябрь 2025

1 Введение

Проект **ZAMAN AI Banking Assistant (Идрис)** создан в рамках хакатона ZAMAN Bank. Главная цель — разработать человечного, доброжелательного и технически продвинутого ассистента, который помогает клиентам банка понимать исламские принципы финансов, проводить расчёты и использовать продукты банка в удобной, прозрачной и халяльной форме.

Ассистент отвечает на вопросы, рассчитывает доходность и платежи, персонализирует ответы в зависимости от финансового поведения пользователя и повышает вовлечённость клиентов в продукты ZAMAN Bank.

Основные задачи:

- Консультации по продуктам банка;
- Расчёт исламского финансирования (мурабаха, вакала);
- Персонализация общения на основе кластеризации клиентов;
- Голосовое взаимодействие с помощью Whisper AI.

—

2 Архитектура системы

Система построена по модульному принципу (рис. **??**).

Основные компоненты:

1. **Data Layer:** синтетические CSV-базы (user, product, user_product, transaction);
2. **Feature Engineering:** объединение таблиц, создание признаков, нормализация (StandardScaler);
3. **ML Core:** кластеризация клиентов с помощью GaussianMixture, обновление портрета клиента;
4. **AI Engine:** использование LLM API (OpenAI) и Whisper Voice Assistant;
5. **Backend Gateway:** Streamlit-сервер, выполняющий маршрутизацию запросов между GUI и моделью;
6. **Frontend:** React + Vite + Tailwind — чат-интерфейс ассистента;
7. **Feedback Loop:** хранение логов взаимодействий и повторное обучение модели.

Подробное объяснение архитектуры

Архитектура ассистента ZAMAN построена как модульная система, в которой данные, логика машинного обучения, интерфейс и обратная связь клиента объединены в единую экосистему.

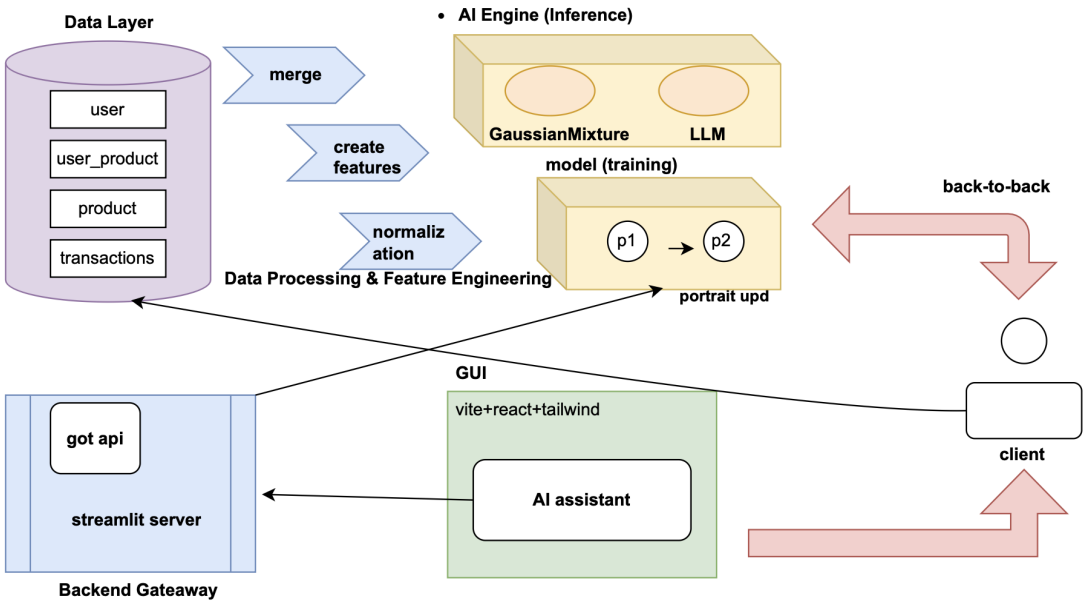

Рис. 1: Подробная архитектура AI-ассистента ZAMAN (окрашенные модули соответствуют уровням системы)

Описание блоков:

1. **Data Layer (фиолетовый блок)** — хранит все исходные CSV-таблицы:
   - `user.csv` — личные данные клиентов (возраст, пол, город, дата регистрации);
   - `user_product.csv` — связи между пользователями и продуктами банка;
   - `product.csv` — список исламских продуктов (мурабаха, вакала и др.);
   - `transactions.csv` — синтетическая история транзакций.

   Данные проходят предобработку и подаются в модуль Feature Engineering.

2. **Data Processing & Feature Engineering (голубые стрелки)** — отвечает за:
   - объединение таблиц (**merge**);

- генерацию признаков (**create features**);
- стандартизацию данных (**normalization** с помощью `StandardScaler`).

После обработки формируется признаковая матрица для ML-моделей.

3. **AI Engine (жёлтый блок)** — состоит из двух частей:
   - **GaussianMixture** — проводит кластеризацию клиентов по признакам RFM (Recency, Frequency, Monetary);
   - **LLM** — крупная языковая модель (OpenAI API), использующая портрет клиента для персонализированных ответов.

   Этот слой отвечает за "мозг"ассистента — вычисления, обучение и формирование контекста ответов.

4. **Model (Training)** — отдельный обучающий модуль, который обновляет портреты клиентов (`portrait upd`) после серии взаимодействий. Он анализирует историю общения, пересчитывает показатели RFM и обновляет сегмент пользователя.

5. **Backend Gateway (синий блок)** — серверная часть, реализованная на **Streamlit**. В ней находятся API-коннекторы (модуль `got api`), которые обеспечивают обмен данными между:
   - GUI-интерфейсом (frontend);
   - LLM API (OpenAI);
   - ML-моделью (кластеризация и обновление данных).

   Этот слой — связующее звено между визуальной частью и моделями.

6. **GUI (зелёный блок)** — создан на базе **Vite** + **React** + **TailwindCSS** и интегрирован с Streamlit-чатом. Интерфейс отображает:
   - чат с ассистентом;
   - результаты расчётов (например, аннуитет, доходность);
   - короткие подсказки, рекомендации и пояснения;
   - элементы для голосового взаимодействия.

7. **Client Interaction (розовые стрелки)** — пользователь общается с ассистентом в режиме **back-to-back**. Это значит, что ответы AI учитывают предыдущие сообщения клиента, а также его сегмент из модели GaussianMixture.

8. **Feedback Loop (возвратные стрелки)** — после серии взаимодействий данные из GUI возвращаются в Data Layer. На их основе происходит повторное обучение и обновление портретов клиентов, обеспечивая постоянное самообучение ассистента.

Почему выбрана такая структура:

- Модульность позволяет масштабировать систему — легко добавить, например, новый тип данных (поведенческий или геоаналитический).
- Чёткое разделение ролей между Backend, ML и Frontend обеспечивает безопасность и гибкость.
- Использование GaussianMixture даёт более мягкую сегментацию, чем KMeans, что важно при работе с человеческим поведением.
- Интеграция Streamlit облегчает внедрение AI-логики без сложного DevOps, что ускоряет итерации во время хакатона.

Результат: такое построение архитектуры позволяет ассистенту «учиться» на пользовательских диалогах, корректировать портрет клиента, и давать не просто математически точные, но и человечные, этически выверенные ответы.

—

3 Обработка данных и ML pipeline

Синтетические данные

Так как банк не предоставлял реальных данных, команда сгенерировала синтетические CSV-файлы с реальной структурой: `user.csv`, `product.csv`, `user_product.csv`, `transaction.csv`.

Данные обрабатываются в pandas DataFrame, очищаются, нормализуются и объединяются по ключу `user_id`.

Признаки (features)

Используются RFM-признаки:

- **Recency (R):** сколько дней прошло с последней транзакции;
- **Frequency (F):** количество транзакций за период;
- **Monetary (M):** общая сумма трат.

Кластеризация

Для сегментации клиентов используется **GaussianMixture**:

$$\text{pdf}(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

где K — количество кластеров, π_k — вероятность кластера, μ_k, Σ_k — среднее и ковариация.

Интерпретация кластеров:

- Saver — склонен к накоплению;
- Investor — выбирает долгосрочные и доходные продукты;
- Spender — активно использует платёжные сервисы.

Retraining Loop

После каждой серии взаимодействий портрет клиента обновляется. Новые данные (recency, frequency, monetary) вычисляются и подаются в модель повторно, что позволяет ассистенту адаптировать рекомендации к изменениям поведения пользователя.

—

4 Формулы и расчёты в исламском финансировании

Аннуитетный платёж (без процентов)

$$A = P \cdot \frac{i(1+i)^n}{(1+i)^n - 1}$$

где:

- P — сумма финансирования;
- n — срок (в месяцах);
- $i = \frac{r}{12}$ — месячная ставка (на основе наценки r);

Общая сумма выплат:

$$T = A \cdot n$$

Наценка (прибыль банка):

$$O = T - P$$

Капитализация:

$$S = P(1 + i)^n$$

или без капитализации:

$$S = P + P \cdot r \cdot \frac{n}{12}$$

Пояснение: Эти формулы отражают принципы халяльной прибыли — банк получает фиксированную наценку, а не процент (риба). Все расчёты основаны на договорной модели (мурабаха или вакала), и прибыль распределяется между клиентом и банком по справедливым правилам.

—

5 Backend и Frontend

Backend (Streamlit + API Gateway):

- принимает сообщения из GUI;
- обращается к LLM API (OpenAI);
- обрабатывает текстовые и голосовые запросы (Whisper1);
- сохраняет сессии и логи взаимодействий.

Frontend (React + Tailwind + Streamlit Chat):

- отображает чат с ассистентом;
- позволяет рассчитать доходность и ежемесячные платежи;
- визуализирует результаты;
- имеет тёплый и человечный UI.

—

6 Этический и исламский контекст

Ассистент полностью соответствует принципам исламского банкинга:

- отсутствие риба (процентной прибыли);
- прозрачность условий;
- халяльные направления инвестиций;

- уважительное и доброжелательное общение.

Речь ассистента построена на балансе: 70% информативности и 30% теплоты. Используются короткие эмодзи , чтобы создать доверительную атмосферу.

—

7 Будущее развитие

- Подключение реальных клиентских данных;
- Обучение на исторических транзакциях;
- Интеграция с API банка при соблюдении норм безопасности;
- Развитие голосового ассистента Whisper1;
- Масштабирование продукта для других исламских банков;
- Возможность патентования и коммерциализации.

—

8 Заключение

ZAMAN AI Assistant (Идрис) объединяет машинное обучение, этику и исламские принципы, создавая инновационный, безопасный и человечный цифровой банк будущего. Этот проект показывает, как AI может быть не просто алгоритмом, а партнёром, уважающим культуру и ценности пользователей.

«Инновации с душой — путь к справедливому цифровому будущему.»