Quản lý sổ tiết kiệm

Software Architecture Document

Version <1.0>

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 13/005/25 | 1.0 | Initial draft | Trần Thanh Sơn |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[**1. Introduction 5**](#_heading=h.ocuxmo7arlf4)

[**2. Architectural Goals and Constraints 5**](#_heading=h.s05nbmhtf8ey)

[**3. Use-Case Model 5**](#_heading=h.xx4uwwrrp0rn)

[**4. Logical View 6**](#_heading=h.n1t3tlk2upsm)

[4.1. Actors (Ngoại vi hệ thống) 6](#_heading=h.ma916wm6fwij)

[4.2. Containers 6](#_heading=h.q378q19mofgm)

[4.2.1. Web Application (ReactJS) 6](#_heading=h.adwap8x0bvds)

[4.2.2. API Server (Node.js + Express) 7](#_heading=h.dytyytgo2jie)

[4.2.3. Database (MongoDB Atlas) 8](#_heading=h.x7lzvncxoy90)

[4.3. Luồng tương tác chính 8](#_heading=h.ijux97dt7ok6)

[4.4 Component: Frontend 9](#_heading=h.2viu7tcwar32)

[**5. Deployment 12**](#_heading=h.gcyo7qe9mv8j)

[**6. Implementation View 14**](#_heading=h.73z6qmw0og5b)

[6.1 Backend (Node.js + ExpressJS) 15](#_heading=h.vkribycxbnm0)

[6.2 Frontend (ReactJS) 16](#_heading=h.8dfgy1xmeu5g)

[**7. Class Diagrams: 19**](#_heading=h.5ijpq8wdireq)

[**8. DB design: 19**](#_heading=h.8lvmrmwiccog)

[8.1. Bảng đăng nhập chung – account 20](#_heading=h.e8n6ewfhh6yu)

[8.2. Khách hàng – khach\_hang 20](#_heading=h.sc4vorswvef4)

[8.3. Nhân viên – nhan\_vien 21](#_heading=h.o072h3ecoh)

[8.4. Sổ tiết kiệm – so\_tiet\_kiem 21](#_heading=h.h1o47jlne5p6)

[8.5. Giao dịch – giao\_dich 22](#_heading=h.ppjfjwt0wtyt)

[8.6. Quy định (key–value) – rule 22](#_heading=h.e22mlp64we48)

[8.7. Lịch sử ChatBot – lich\_su\_chatbot 23](#_heading=h.6194j5jf9uwj)

[8.8. Yêu cầu dự báo – yeu\_cau\_du\_bao 23](#_heading=h.cvk9ihfhsf31)

[8.9. Log đăng nhập – log\_dang\_nhap 23](#_heading=h.g8kzjg7qggg5)

Software Architecture Document

# Introduction

Tài liệu này mô tả kiến trúc phần mềm cho hệ thống Quản lý sổ tiết kiệm, nhằm đảm bảo thiết kế đáp ứng đúng các yêu cầu chức năng và phi chức năng đã được đề ra trong tài liệu Use-Case Specification

Hệ thống Quản lý sổ tiết kiệm cho phép người dùng thực hiện các thao tác như đăng ký, đăng nhập, mở sổ tiết kiệm, gửi/rút tiền, xem thông tin tài khoản, nhận tư vấn từ ChatBot AI, và nhân viên ngân hàng có thể quản lý sổ, thay đổi quy định và xem báo cáo. Kiến trúc phần mềm được thiết kế để hỗ trợ cả người dùng là khách hàng và nhân viên ngân hàng thông qua nền tảng web

# Architectural Goals and Constraints

* Hệ thống phải triển khai trên nền tảng web.
* Giao diện phải thân thiện, dễ sử dụng cho sinh viên và giảng viên.
* Ứng dụng phải đảm bảo khả năng truy cập từ nhiều thiết bị (PC, laptop, tablet).
* Bảo mật thông tin người dùng (đăng nhập, thông tin cá nhân) là yêu cầu bắt buộc.
* Hệ thống phải hỗ trợ phân quyền truy cập theo vai trò (Quản lý, nhân viên, khách hàng).
* Khả năng mở rộng hệ thống trong tương lai để thêm module mới hoặc tích hợp với hệ thống khác.
* Cần đáp ứng hiệu năng xử lý truy vấn tốt trong điều kiện có nhiều người dùng đồng thời.
* Sử dụng các công nghệ phổ biến, dễ bảo trì và dễ phát triển thêm.
* Cần lưu trữ và truy xuất dữ liệu ổn định, nhất quán (yêu cầu về hệ quản trị cơ sở dữ liệu).
* Hệ thống cần tuân thủ lịch trình phát triển gồm nhiều giai đoạn, với thời gian hoàn thành giới hạn.

# Use-Case Model

Hệ thống Quản lý sổ tiết kiệm bao gồm các tác nhân (actors) chính và các chức năng (use cases) tương tác với nhau. Dưới đây là mô hình tổng quát của hệ thống:

Actors:

* Khách hàng: Người sử dụng dịch vụ tiết kiệm (mở sổ, gửi/rút tiền, tra cứu thông tin, sử dụng AI hỗ trợ...).
* Nhân viên Ngân hàng: Người quản lý và xử lý các yêu cầu từ khách hàng, thay đổi quy định, xem báo cáo...
* Hệ thống AI (GPT API): Cung cấp phản hồi và dự báo tự động.

Các Use-Case chính:

* Dành cho Khách hàng:
  + Đăng ký
  + Đăng nhập / Đăng xuất
  + Mở sổ tiết kiệm
  + Gửi tiền vào sổ
  + Rút tiền từ sổ
  + Tra cứu thông tin sổ tiết kiệm
  + Trò chuyện với ChatBot
  + Xem dự báo xu hướng tài chính
* Dành cho Nhân viên Ngân hàng:
  + Đăng nhập / Đăng xuất
  + Mở sổ tiết kiệm thay cho khách hàng
  + Cập nhật thông tin sổ tiết kiệm
  + Đóng sổ tiết kiệm
  + Xem danh sách sổ tiết kiệm
  + Thay đổi quy định
  + Xem báo cáo:
  + Báo cáo doanh số hoạt động
  + Báo cáo mở hoặc đóng sổ theo tháng

# Logical View

Hệ thống được xây dựng theo mô hình 3 lớp, bao gồm các thành phần chính sau:

## 4.1. Actors (Ngoại vi hệ thống)

1. **Khách hàng**
   * Các hành vi:
     + Đăng ký, đăng nhập/đăng xuất (UC1–3)
     + Mở sổ, gửi tiền, rút tiền, tra cứu chi tiết sổ (UC4–7)
     + ChatBot (UC8)
     + Forecast (UC9)
   * Tương tác qua: **Web Application (ReactJS)** qua HTTPS.
2. **Nhân viên ngân hàng**
   * Các hành vi:
     + Mở sổ thay cho khách (UC4)
     + Cập nhật / đóng sổ (UC10–11)
     + Thay đổi quy định (UC13)
     + Xem báo cáo (UC12, UC14–16)
   * Cũng tương tác qua **Web Application**.
3. **Hệ thống AI (GPT API)**
   * Cung cấp hai dịch vụ:
     + ChatBot trả lời các câu hỏi của khách (UC8)
     + Dự báo xu hướng tài chính (UC9)
   * Được gọi từ **Back-end** qua HTTPS.

## 4.2. Containers

### **4.2.1. Web Application (ReactJS)**

* **Mục đích**:
  + Hiển thị hết tất cả các UI, chia thành 4 nhóm component tương ứng use-case:
  + **Auth UI** – Trang Login / Register  
     – Thực hiện UC1–3
  + **Account UI** – Form mở/gửi/rút, bảng danh sách, chi tiết sổ  
     – Thực hiện UC4–7, UC10–11
  + **ChatBot UI** – Khung chat realtime với AI  
     – Thực hiện UC8
  + **Forecast UI** – Form chọn tham số, hiển thị đồ thị, export PNG/CSV  
     – Thực hiện UC9
* **Giao tiếp**:
  + Gọi REST API (JSON over HTTPS) đến **API Server**.

### **4.2.2. API Server (Node.js + Express)**

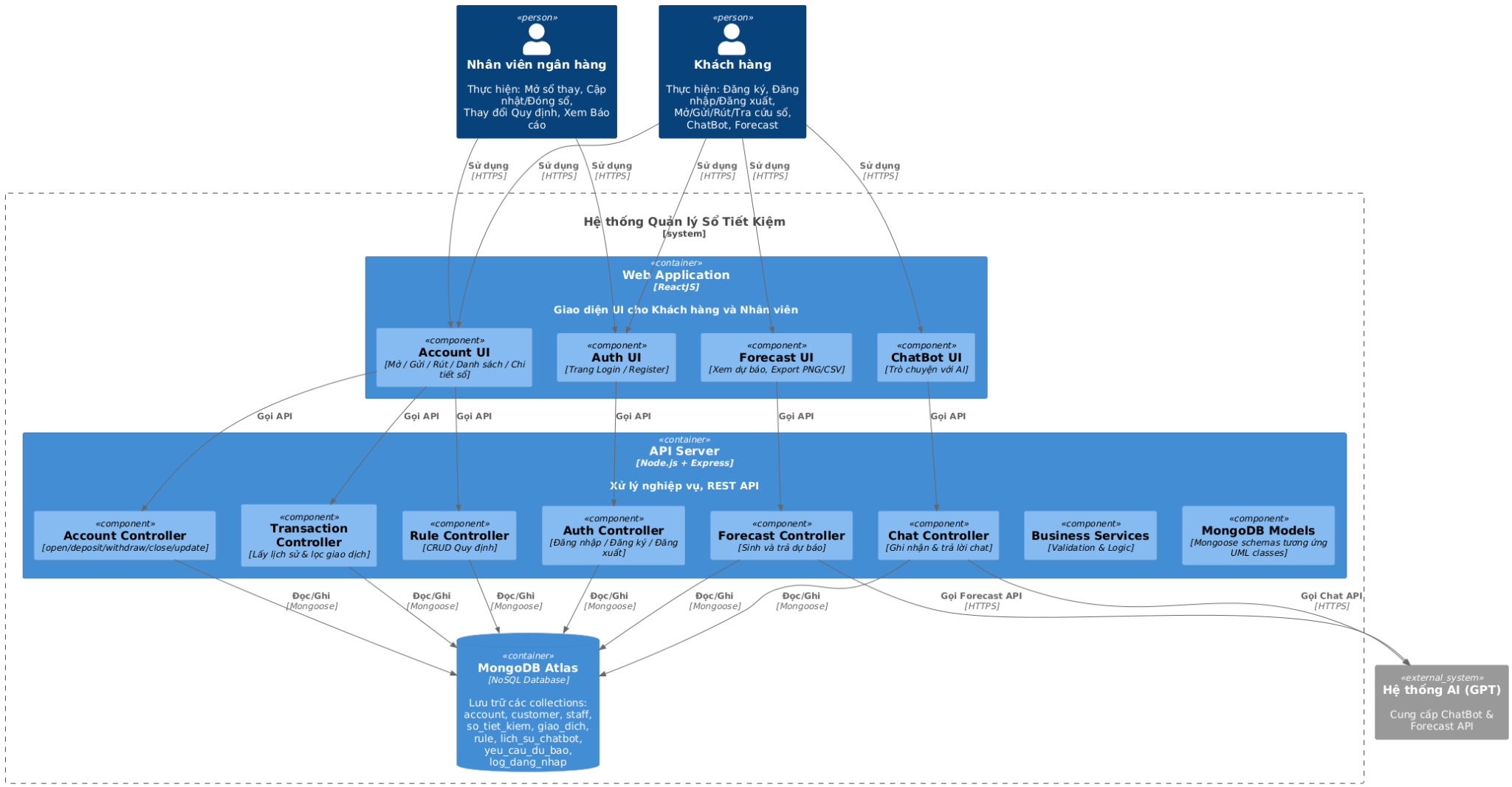
* **Mục đích**:  
  1. Đóng gói toàn bộ **nghiệp vụ** (business logic) của hệ thống, expose dưới dạng RESTful endpoints.
* **Các Component chính**:  
  1. **Auth Controller** – /auth/login, /auth/register, /auth/logout  
      – Thực thi xác thực, tạo Session / JWT  
      – Map tới UML class TaiKhoan, Session, NhatKyDangNhap
  2. **Account Controller** – /accounts/open, /accounts/deposit, /accounts/withdraw, /accounts/close, /accounts/update  
      – Áp quy định (minDeposit, maxTerm…) từ bảng rule  
      – Map tới UML class SoTietKiem, QuyDinh, NhanVien, KhachHang
  3. **Transaction Controller** – /transactions (GET, filter…)  
      – Lấy lịch sử giao dịch GiaoDich, ánh xạ class GiaoDich
  4. **Rule Controller** – /rules CRUD quy định  
      – Map tới class QuyDinh
  5. **Chat Controller** – /chat ghi lịch sử LichSuChatBot, gọi GPT để trả lời  
      – Map tới class LichSuChatBot
  6. **Forecast Controller** – /forecast tính toán / gọi GPT, ghi lịch sử YeuCauDuBao
  7. **Business Services** – Chứa tất cả logic nghiệp vụ phức tạp (validation, fee calculation, interest, term checks…)  
      – Dùng chung bởi nhiều Controller
  8. **MongoDB Models** – Tập hợp các schema Mongoose cho:  
      Account, Customer, Staff, SoTietKiem, GiaoDich, Rule, ChatHistory, ForecastRequest, LoginLog  
      – Phản ánh UML Class Diagram & Database Diagram
* **Giao tiếp**:  
  1. Nhận request từ **Web App** qua HTTP.
  2. Đọc / ghi dữ liệu tới **MongoDB Atlas** qua Mongoose driver.
  3. Gọi GPT API (AI) khi cần ChatBot hoặc Forecast.

### **4.2.3. Database (MongoDB Atlas)**

* **Mục đích**:  
  + Lưu trữ toàn bộ dữ liệu nghiệp vụ:
  + Bảng account (taiKhoan)
  + Bảng khach\_hang, nhan\_vien
  + Bảng so\_tiet\_kiem, giao\_dich
  + Bảng rule
  + Bảng log: lich\_su\_chatbot, yeu\_cau\_du\_bao, log\_dang\_nhap
* **Driver**:  
  + Node.js MongoDB driver / Mongoose

## 4.3. Luồng tương tác chính

1. **Đăng ký / Đăng nhập** – Khách hàng gửi form **Auth UI** → Auth Controller → Mongoose tạo Account + KhachHang → trả token/session.
2. **Mở sổ / Gửi / Rút** – **Account UI** gọi tương ứng → Account Controller  
    – Trong Controller gọi Business Service để:
   * Lấy quy định (min/max…) từ Rule
   * Tính lãi suất mặc định hoặc theo tham số
   * Sinh mã SoTietKiem mới hoặc ghi GiaoDich  
      – Cuối cùng ghi vào MongoDB
3. **Tra cứu / Danh sách sổ** – **Account UI** gọi GET → Account Controller → query SoTietKiem + GiaoDich → trả JSON
4. **ChatBot** – **ChatBot UI** nhập câu hỏi → Chat Controller  
    – Controller lưu LichSuChatBot và call GPT API → nhận phản hồi → trả về UI
5. **Forecast** – **Forecast UI** chọn tham số → Forecast Controller  
    – Controller lưu YeuCauDuBao, gọi GPT API → nhận kết quả dự báo → trả UI
6. **Quy định & Báo cáo (dành cho Staff)** – **Account UI** hoặc riêng **Staff UI** gọi các endpoint trong Rule Controller & báo cáo  
    – Rule Controller CRUD Rule  
    – SalesReport, OpenCloseReport có thể triển khai thêm controller riêng hoặc dùng chung Account Controller + aggregation trên SoTietKiem, GiaoDich



## 4.4 Component: Frontend

**Mô tả:**  
Là lớp giao diện người dùng (UI), được sử dụng bởi cả Khách hàng và Nhân viên ngân hàng thông qua trình duyệt web.

| **Tên Component** | **File/Thư mục** | **Mô tả** | **Phụ thuộc** |
| --- | --- | --- | --- |
| **App Shell** | src/App.jsx | Khung bao ngoài của ứng dụng, chứa <Router> và các layout chung (Header, Sidebar, Footer). | React, React-Router, các Layout component |
| **Auth UI** | src/components/Auth/ | LoginForm.jsx: form đăng nhậpRegisterForm.jsx: form đăng ký | useAuth hook, auth API |
| **Account UI** | src/components/Savings/ | AccountList.jsx: danh sách sổAccountForm.jsx: mở sổ/gửi/rútAccountDetail.jsx: chi tiết sổ | useAccounts hook, accounts API |
| **Transaction UI** | src/components/Transactions/TransactionForm.jsx | Form lọc và hiển thị giao dịch (gửi/rút) | useTransactions hook, transactions API |
| **ChatBot UI** | src/components/Chat/ChatWidget.jsx | Widget chat theo luồng trò chuyện liên tục với GPT | useChat hook, chat API |
| **Forecast UI** | src/components/Forecast/ | ForecastForm.jsx: chọn tham sốForecastChart.jsx: vẽ biểu đồ dự báo | useForecast hook, forecast API, recharts |
| **Layouts** | src/layouts/ | MainLayout.jsx: sidebar + headerAuthLayout.jsx: layout cho trang auth | Shared UI (Header/Footer/Sidebar) |
| **Routing** | src/routes.jsx | Khai báo các route (/, /login, /accounts, /chat, /forecast, …) | React-Router |
| **Context & Hooks** | src/contexts/, src/hooks/ | AuthContext: lưu token, user infoAccountContext: state sổ tiết kiệmCustom hooks: useAuth, useAccounts, useTransactions, useRules, useChat, useForecast | React Context API, axios/fetch |
| **Shared Components** | src/components/Shared/ | Header.jsx, Footer.jsx, Sidebar.jsx, Modal.jsx, Spinner.jsx, Table.jsx | TailwindCSS / shadcn/ui |
| **Static Assets** | public/ | index.html, favicon, logo, các file tĩnh | – |

**4.5 Component: Backend**

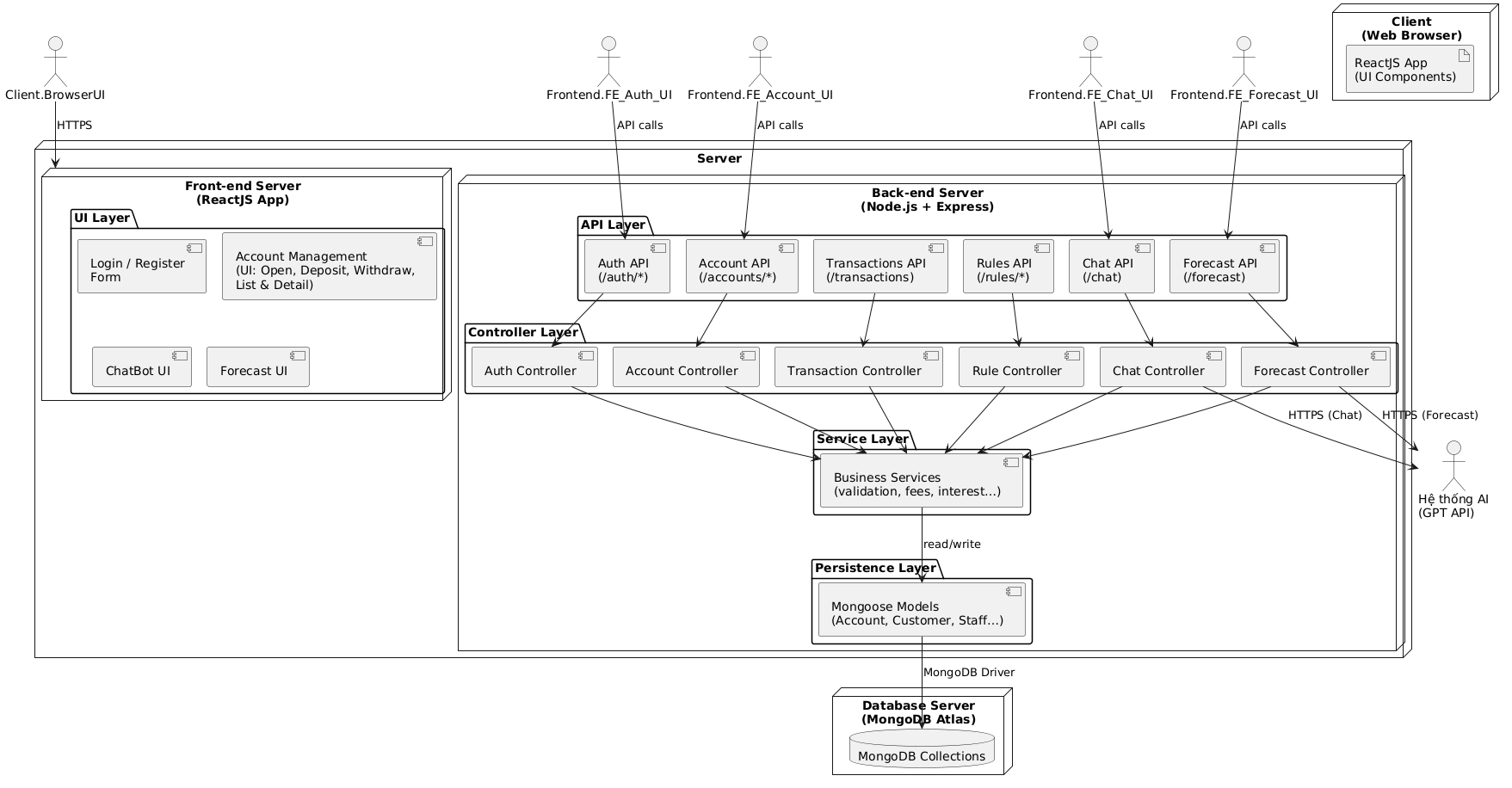
**Mô tả**:

* Là thành phần trung tâm, xử lý toàn bộ nghiệp vụ của hệ thống.

| **Tên Component** | **File/Thư mục** | **Mô tả** | **Phụ thuộc** |
| --- | --- | --- | --- |
| **Entry Point** | src/app.js | Thiết lập Express app, middleware chung (CORS, body-parser), kết nối DB và mount các route | Express, dotenv, mongoose |
| **Config** | src/config/ | db.js (kết nối MongoDB), env.js (đọc biến môi trường), logger.js | dotenv |
| **Routes** | src/routes/ | auth.js: /auth/login, /auth/register, /auth/logoutaccounts.js: /accounts/\*transactions.js, rules.js, chat.js, forecast.js | controllers |
| **Controllers** | src/controllers/ | AuthController.jsAccountController.jsTransactionController.jsRuleController.jsChatController.jsForecastController.js | services |
| **Services (Business)** | src/services/ | AuthService.js: xử lý hash password, JWTAccountService.js: logic mở, gửi, rút, đóng sổTransactionService.js, RuleService.js, ChatService.js, ForecastService.js | models, external GPT client |
| **Models** | src/models/ | Mongoose schema & model cho: Account.js, Customer.js, Staff.js, Saving.js (SoTietKiem), Transaction.js, Rule.js, ChatHistory.js, ForecastRequest.js, LoginLog.js | mongoose |
| **Middleware** | src/middleware/ | authMiddleware.js: bảo vệ route, kiểm JWTerrorHandler.js: catch toàn cục lỗivalidate.js: Joi/express-validator | controllers, services |
| **GPT Client** | src/utils/gptClient.js | Wrapper gọi OpenAI GPT API cho chat và forecast | axios/fetch |
| **Logging & Monitoring** | src/utils/logger.js, src/middleware/metrics.js | Thiết lập Winston hoặc Pino, Prometheus metrics | winston/pino, prom-client |
| **Database** | MongoDB Atlas | Lưu trữ collection tương ứng với UML Class & DB Diagram: accounts, customers, staffs, savings, transactions, rules, chat\_histories, forecasts, login\_logs | mongoose driver |
| **Scripts & Tests** | package.json scripts, tests/ | Scripts chạy migrate, seed, lint, unit/integration tests (Jest/Supertest) | jest, supertest, eslint |

# Deployment

Deployment Diagram:



# 

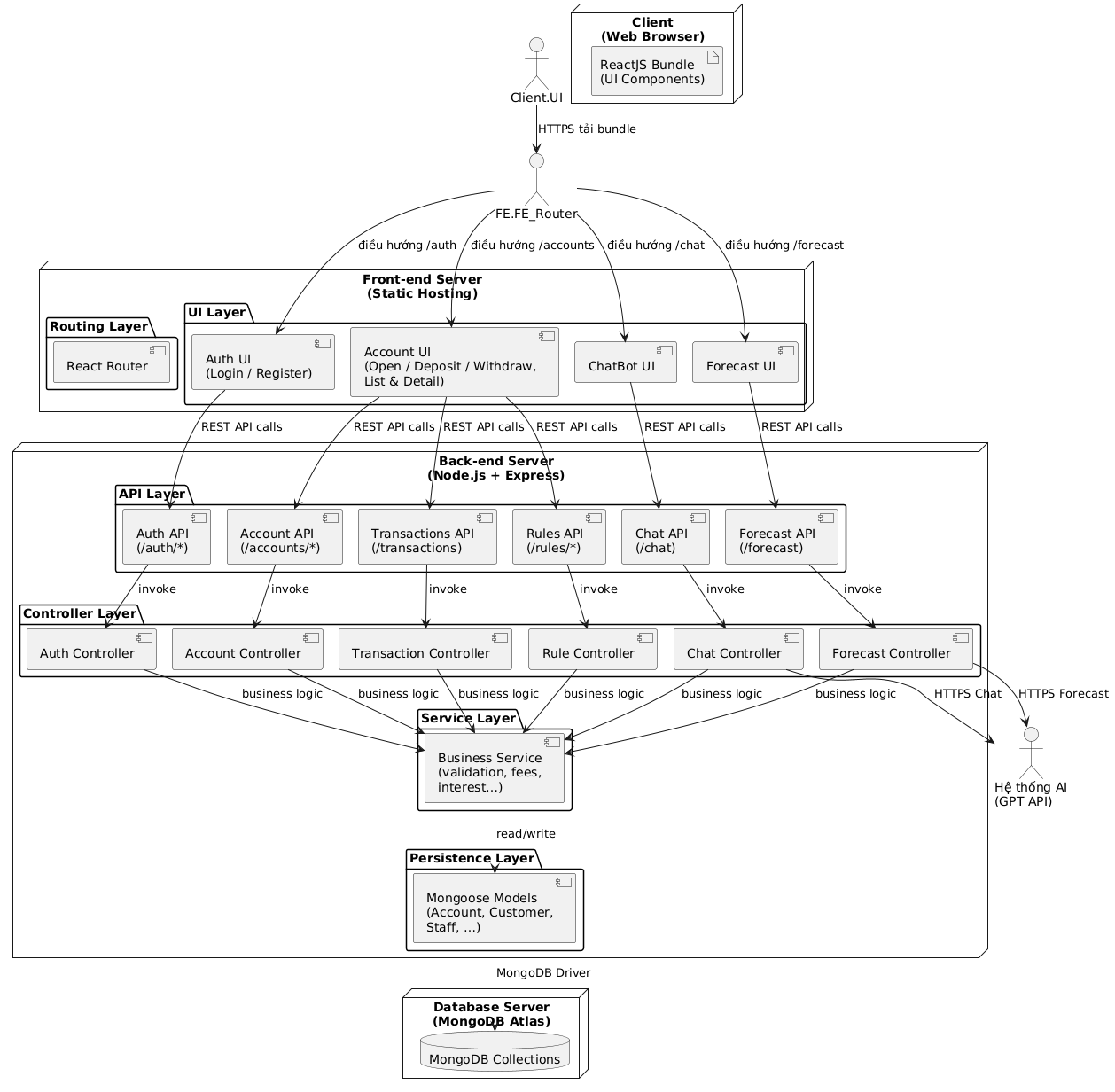
| Thành phần | Mô tả triển khai |
| --- | --- |
| Client (Web Browser) | Chạy trên máy người dùng (PC/Laptop/Mobile). Tải và render ReactJS bundle, thực hiện các tương tác UI (Auth, Account, ChatBot, Forecast) qua HTTPS. |
| Front-end Server (ReactJS App) | Host tĩnh trên VPS/Nginx (hoặc CDN). Phục vụ các file HTML/CSS/JS của ứng dụng React. Giao tiếp với Back-end qua REST API (HTTPS). |
| Back-end Server (Node.js + Express) | Host trên VPS/Nginx. Chạy ứng dụng Express, bao gồm:• API Layer: định nghĩa các endpoint /auth/\*, /accounts/\*, /transactions, /rules/\*, /chat, /forecast• Controller Layer: xử lý request → chuyển đến Service Layer• Service Layer: logic nghiệp vụ (validate, tính phí, sinh mã…)• Persistence Layer: Mongoose Models ánh xạ UML classes, CRUD lên MongoDB. |
| Database Server (MongoDB Atlas) | MongoDB Cloud (Atlas) lưu các collection:accounts, khach\_hang, nhan\_vien, so\_tiet\_kiem, giao\_dich, rule, lich\_su\_chatbot, yeu\_cau\_du\_bao, log\_dang\_nhap. |
| Hệ thống AI (GPT API) | Dịch vụ external của OpenAI. Back-end gọi qua HTTPS để lấy phản hồi ChatBot và dữ liệu Forecast. |

Kết nối giữa các thành phần

Client ↔ Front-end Server:

* Trình duyệt GET/POST các file tĩnh (HTML, JS, CSS) và tải ứng dụng React qua HTTPS.
* Front-end Server ↔ Back-end Server:
* ReactJS App gọi REST API (/auth, /accounts, /chat, /forecast…) qua HTTPS.
* Back-end Server ↔ Database Server:
* Express dùng Mongoose (MongoDB Node.js Driver) để kết nối, đọc/ghi collection qua mạng nội bộ hoặc internet.  
  Back-end Server ↔ Hệ thống AI (GPT API):
* Gửi request chat và forecast qua HTTPS đến OpenAI GPT API, nhận kết quả và lưu log vào MongoDB

# Implementation View



## 6.1 Backend (Node.js + ExpressJS)

| backend/  ├── src/  │ ├── config/ # Cấu hình môi trường, biến toàn cục  │ │ └── index.js # đọc .env, thiết lập DB URI, JWT secret…  │ │  │ ├── databases/ # Khởi tạo kết nối đến MongoDB  │ │ └── mongoose.js # connect(), mongoose options…  │ │  │ ├── models/ # Mongoose schemas ↔ UML classes & DB tables  │ │ ├── Account.model.js # account collection  │ │ ├── Customer.model.js # khach\_hang  │ │ ├── Employee.model.js # nhan\_vien  │ │ ├── Savings.model.js # so\_tiet\_kiem  │ │ ├── Transaction.model.js # giao\_dich  │ │ ├── Rule.model.js # rule  │ │ ├── ChatLog.model.js # lich\_su\_chatbot  │ │ ├── ForecastReq.model.js # yeu\_cau\_du\_bao  │ │ ├── LoginLog.model.js # log\_dang\_nhap  │ │ └── Session.model.js # session (phiên làm việc)  │ │  │ ├── repositories/ # Data access layer (optional)  │ │ └── account.repo.js # các hàm CRUD, tìm kiếm, giao dịch…  │ │  │ ├── services/ # Business logic ↔ Use-Case  │ │ ├── auth.service.js # dangKy, dangNhap, dangXuat  │ │ ├── customer.service.js # fetch, update customer  │ │ ├── account.service.js # openAccount, deposit, withdraw, closeAccount  │ │ ├── transaction.service.js# list/filter giao dịch  │ │ ├── rule.service.js # CRUD quy định  │ │ ├── chat.service.js # lưu / lấy lịch sử ChatBot  │ │ └── forecast.service.js # ForecastRequest  │ │  │ ├── http/ # HTTP layer: controllers + validators  │ │ ├── controllers/  │ │ │ ├── auth.controller.js  │ │ │ ├── customer.controller.js  │ │ │ ├── account.controller.js  │ │ │ ├── transaction.controller.js  │ │ │ ├── rule.controller.js  │ │ │ ├── chat.controller.js  │ │ │ └── forecast.controller.js  │ │ └── validators/ # Joi / express-validator schemas  │ │ ├── auth.validator.js  │ │ └── account.validator.js  │ │  │ ├── routes/ # Express routers ↔ frontend api/  │ │ ├── auth.routes.js # /api/auth  │ │ ├── customer.routes.js # /api/customers  │ │ ├── account.routes.js # /api/accounts  │ │ ├── transaction.routes.js # /api/transactions  │ │ ├── rule.routes.js # /api/rules  │ │ ├── chat.routes.js # /api/chat  │ │ └── forecast.routes.js # /api/forecast  │ │  │ ├── middleware/ # auth, error-handler, logging…  │ │ ├── auth.middleware.js # JWT verification, role-check  │ │ ├── error.middleware.js # centralized error catcher  │ │ └── logger.middleware.js # request logging  │ │  │ └── app.js # khởi tạo Express, mount routes & middleware  │  ├── .env.example # mẫu env vars (DB\_URI, JWT\_SECRET…)  ├── package.json  └── README.md |
| --- |

## 6.2 Frontend (ReactJS)

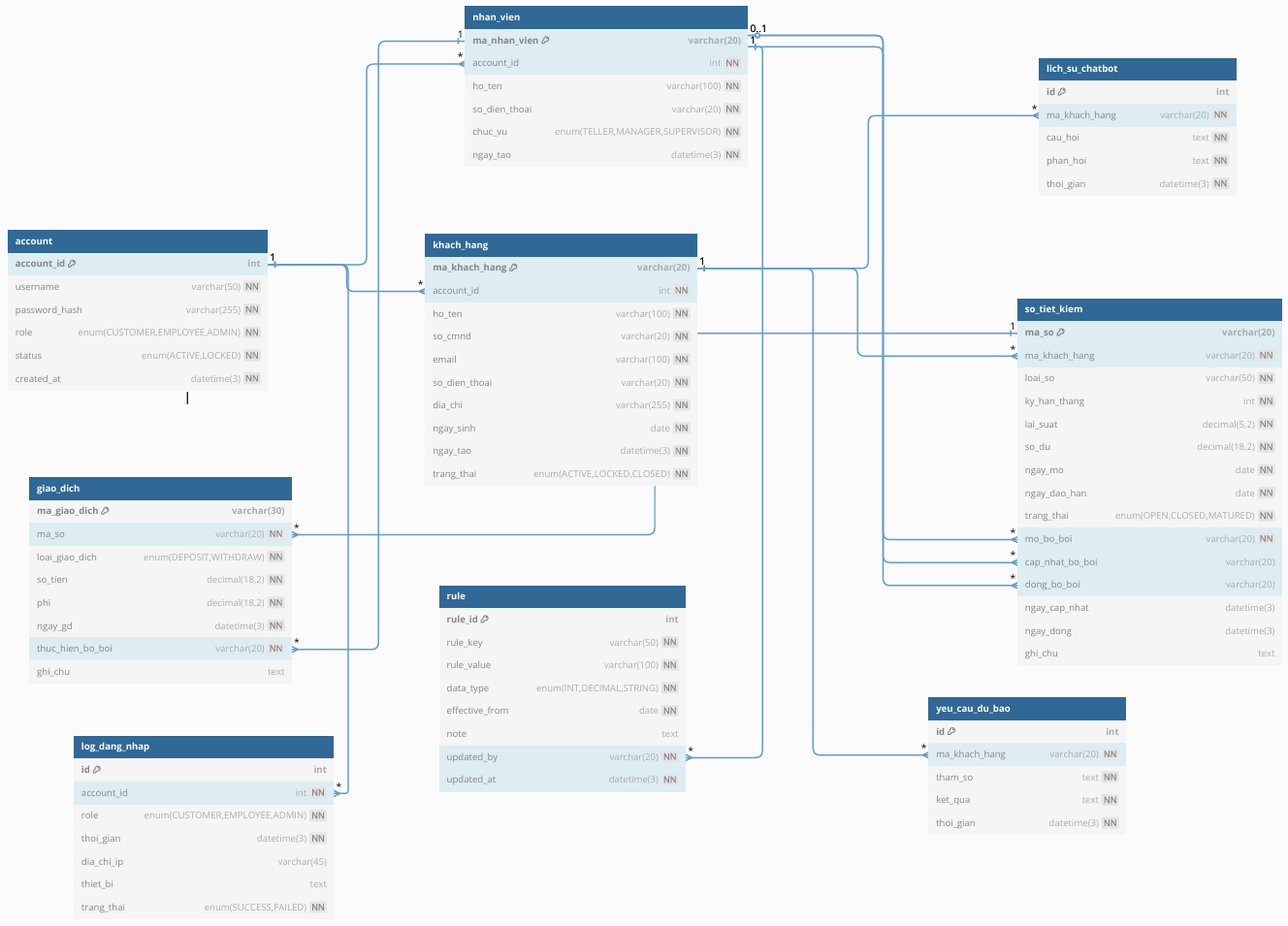
| src/  ├── api/ # Tất cả các cuộc gọi HTTP  │ ├── auth.js # login, register, logout  │ ├── customers.js # fetch/update Customer  │ ├── accounts.js # openAccount, deposit, withdraw, closeAccount  │ ├── transactions.js # list/filter giao dịch  │ ├── rules.js # CRUD quy định  │ ├── chat.js # ChatBot history  │ └── forecast.js # ForecastRequest  │  ├── components/ # Các component tái sử dụng  │ ├── Auth/  │ │ ├── LoginForm.jsx  │ │ └── RegisterForm.jsx  │ │  │ ├── Savings/  │ │ ├── AccountList.jsx  │ │ ├── AccountForm.jsx  │ │ └── AccountDetail.jsx  │ │  │ ├── Transactions/  │ │ └── TransactionForm.jsx  │ │  │ ├── Chat/  │ │ └── ChatWidget.jsx  │ │  │ ├── Forecast/  │ │ ├── ForecastForm.jsx  │ │ └── ForecastChart.jsx  │ │  │ ├── Staff/  │ │ ├── CustomerTable.jsx  │ │ ├── RuleTable.jsx  │ │ ├── ReportCharts.jsx  │ │ └── …  │ │  │ └── Shared/  │ ├── Header.jsx  │ ├── Footer.jsx  │ ├── Sidebar.jsx  │ ├── Modal.jsx  │ ├── Spinner.jsx  │ └── Table.jsx  │  ├── contexts/ # React Contexts cho state toàn cục  │ ├── AuthContext.jsx  │ └── AccountContext.jsx  │  ├── hooks/ # Custom hooks  │ ├── useAuth.js  │ ├── useAccounts.js  │ ├── useTransactions.js  │ ├── useRules.js  │ ├── useChat.js  │ └── useForecast.js  │  ├── pages/ # Các “route pages” tương ứng Use Cases  │ ├── LoginPage.jsx # UC1  │ ├── RegisterPage.jsx # UC2  │ ├── CustomerDashboard.jsx # UC4–7,8,9  │ ├── OpenAccountPage.jsx # UC4  │ ├── DepositPage.jsx # UC5  │ ├── WithdrawPage.jsx # UC6  │ ├── AccountDetailPage.jsx # UC7,10,11  │ ├── ChatPage.jsx # UC8  │ ├── ForecastPage.jsx # UC9  │ ├── CustomerSettings.jsx # UC (edit profile, history)  │ ├── StaffDashboard.jsx # tổng hợp UC12–16  │ ├── CustomerListPage.jsx # UC12  │ ├── AccountManagementPage.jsx # UC12  │ ├── TransactionMgmtPage.jsx # UC12  │ ├── RuleMgmtPage.jsx # UC13  │ ├── SalesReportPage.jsx # UC15  │ └── OpenCloseReportPage.jsx # UC16  │  ├── routes.jsx # React-Router setup  └── App.jsx |
| --- |

# Class Diagrams:

# 

# DB design:

Dưới đây là thiết kế cơ sở dữ liệu của hệ thống quản lý sổ tiết kiệm, được trình bày dưới dạng sơ đồ ERD. Mô hình thể hiện rõ các thực thể (entities), thuộc tính (attributes) và các mối quan hệ (relationships) giữa chúng.



### **8.1. Bảng đăng nhập chung – account**

* **account\_id** (PK, int, tự tăng)
* **username** (varchar(50), NOT NULL, UNIQUE) – tên đăng nhập
* **password\_hash** (varchar(255), NOT NULL) – mật khẩu đã mã hóa
* **role** (enum CUSTOMER|EMPLOYEE|ADMIN, NOT NULL) – phân quyền
* **status** (enum ACTIVE|LOCKED, NOT NULL, default ACTIVE) – tình trạng tài khoản
* **created\_at** (datetime(3), NOT NULL, default now()) – thời điểm tạo

### **8.2. Khách hàng – khach\_hang**

* **ma\_khach\_hang** (PK, varchar(20)) – mã khách hàng
* **account\_id** (int, NOT NULL, UNIQUE) → FK account.account\_id
* **ho\_ten** (varchar(100), NOT NULL) – họ tên
* **so\_cmnd** (varchar(20), NOT NULL, UNIQUE) – CMND/CCCD
* **email** (varchar(100), NOT NULL, UNIQUE) – email
* **so\_dien\_thoai** (varchar(20), NOT NULL) – số điện thoại
* **dia\_chi** (varchar(255), NOT NULL) – địa chỉ
* **ngay\_sinh** (date, NOT NULL) – ngày sinh
* **ngay\_tao** (datetime(3), NOT NULL, default now()) – thời điểm tạo
* **trang\_thai** (enum ACTIVE|LOCKED|CLOSED, NOT NULL, default ACTIVE) – tình trạng

### **8.3. Nhân viên – nhan\_vien**

* **ma\_nhan\_vien** (PK, varchar(20)) – mã nhân viên
* **account\_id** (int, NOT NULL, UNIQUE) → FK account.account\_id
* **ho\_ten** (varchar(100), NOT NULL) – họ tên
* **so\_dien\_thoai** (varchar(20), NOT NULL) – số điện thoại
* **chuc\_vu** (enum TELLER|MANAGER|SUPERVISOR, NOT NULL) – chức vụ
* **ngay\_tao** (datetime(3), NOT NULL, default now()) – thời điểm tạo

### **8.4. Sổ tiết kiệm – so\_tiet\_kiem**

* **ma\_so** (PK, varchar(20)) – mã sổ
* **ma\_khach\_hang** (varchar(20), NOT NULL) → FK khach\_hang.ma\_khach\_hang
* **loai\_so** (varchar(50), NOT NULL) – loại sổ
* **ky\_han\_thang** (int, NOT NULL) – kỳ hạn (tháng)
* **lai\_suat** (decimal(5,2), NOT NULL) – lãi suất (%)
* **so\_du** (decimal(18,2), NOT NULL, default 0) – số dư
* **ngay\_mo** (date, NOT NULL) – ngày mở
* **ngay\_dao\_han** (date, NOT NULL) – ngày đáo hạn
* **trang\_thai** (enum OPEN|CLOSED|MATURED, NOT NULL) – tình trạng
* **mo\_bo\_boi** (varchar(20), NOT NULL) → FK nhan\_vien.ma\_nhan\_vien – nhân viên mở
* **cap\_nhat\_bo\_boi** (varchar(20)) → FK nhan\_vien.ma\_nhan\_vien – nhân viên cập nhật
* **dong\_bo\_boi** (varchar(20)) → FK nhan\_vien.ma\_nhan\_vien – nhân viên đóng
* **ngay\_cap\_nhat** (datetime(3)) – thời điểm cập nhật
* **ngay\_dong** (datetime(3)) – thời điểm đóng
* **ghi\_chu** (text) – ghi chú

### **8.5. Giao dịch – giao\_dich**

* **ma\_giao\_dich** (PK, varchar(30)) – mã giao dịch
* **ma\_so** (varchar(20), NOT NULL) → FK so\_tiet\_kiem.ma\_so
* **loai\_giao\_dich** (enum DEPOSIT|WITHDRAW, NOT NULL)
* **so\_tien** (decimal(18,2), NOT NULL) – số tiền
* **phi** (decimal(18,2), NOT NULL, default 0) – phí
* **ngay\_gd** (datetime(3), NOT NULL) – thời gian giao dịch
* **thuc\_hien\_bo\_boi** (varchar(20), NOT NULL) → FK nhan\_vien.ma\_nhan\_vien – nhân viên thực hiện
* **ghi\_chu** (text) – ghi chú

### **8.6. Quy định (key–value) – rule**

* **rule\_id** (PK, int, tự tăng) – mã quy định
* **rule\_key** (varchar(50), NOT NULL) – khóa
* **rule\_value** (varchar(100), NOT NULL) – giá trị
* **data\_type** (enum INT|DECIMAL|STRING, NOT NULL)
* **effective\_from** (date, NOT NULL, default now()) – có hiệu lực từ
* **note** (text) – ghi chú
* **updated\_by** (varchar(20), NOT NULL) → FK nhan\_vien.ma\_nhan\_vien – người cập nhật
* **updated\_at** (datetime(3), NOT NULL, default now()) – thời điểm cập nhật

### **8.7. Lịch sử ChatBot – lich\_su\_chatbot**

* **id** (PK, int, tự tăng)
* **ma\_khach\_hang** (varchar(20), NOT NULL) → FK khach\_hang.ma\_khach\_hang
* **cau\_hoi** (text, NOT NULL)
* **phan\_hoi** (text, NOT NULL)
* **thoi\_gian** (datetime(3), NOT NULL)

### **8.8. Yêu cầu dự báo – yeu\_cau\_du\_bao**

* **id** (PK, int, tự tăng)
* **ma\_khach\_hang** (varchar(20), NOT NULL) → FK khach\_hang.ma\_khach\_hang
* **tham\_so** (text, NOT NULL)
* **ket\_qua** (text, NOT NULL)
* **thoi\_gian** (datetime(3), NOT NULL)

### **8.9. Log đăng nhập – log\_dang\_nhap**

* **id** (PK, int, tự tăng)
* **nguoi\_dung** (varchar) – tên đăng nhập (khách hàng / nhân viên)
* **loai\_nguoi\_dung** (varchar) – CUSTOMER / EMPLOYEE
* **thoi\_gian** (datetime(3), NOT NULL, default now())
* **dia\_chi\_ip** (varchar(45)) – IPv4/IPv6
* **thiet\_bi** (text)
* **trang\_thai** (enum SUCCESS|FAILED, NOT NULL, default SUCCESS)

**Các mối quan hệ chính**

1. **Khách\_hang ↔ So\_tiet\_kiem** – 1 Khách hàng có thể có nhiều Sổ tiết kiệm (1 → \*).
2. **Nhan\_vien ↔ So\_tiet\_kiem** – Mỗi sổ được mở/cập nhật/đóng bởi 1 Nhân viên (0..1 → \*).
3. **So\_tiet\_kiem ↔ Giao\_dich** – 1 Sổ có nhiều Giao dịch (1 → \*).
4. **Nhan\_vien ↔ Giao\_dich** – Mỗi giao dịch do 1 Nhân viên thực hiện (1 → \*).
5. **Nhan\_vien ↔ Quy\_dinh** – Mỗi bản ghi Quy định được cập nhật bởi 1 Nhân viên (1 → \*).
6. **Khach\_hang ↔ Lich\_su\_chatbot** – 1 Khách hàng có thể có nhiều phiên ChatBot (1 → \*).
7. **Khach\_hang ↔ Yeu\_cau\_du\_bao** – 1 Khách hàng có thể gửi nhiều yêu cầu dự báo (1 → \*).
8. **Log\_dang\_nhap** – Không cần FK, lưu **nguoi\_dung** và **loai\_nguoi\_dung** để audit mọi lần đăng nhập.