# System Design Documentation: Filesure Referral & Credit System

## 1. Overview

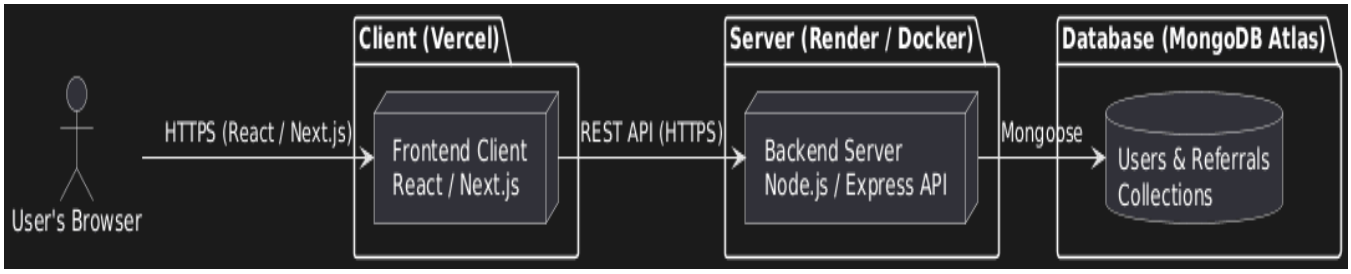
This document outlines the system architecture, data flow, and database design for the FileSure Referral & Credit System.

The objective is to build a scalable full-stack application that allows users to register, refer new users, and earn credits. The system is composed of three primary components:

1. **Client (Frontend):** A Next.js, TypeScript, and Tailwind application.
2. **Server (Backend):** A Node.js, Express, and TypeScript RESTful API.
3. **Database (Persistence):** A MongoDB database for storing user and referral data.

## 2. System Architecture Diagram

This diagram shows the high-level components and their interactions. The system is designed as a classic, decoupled client-server architecture

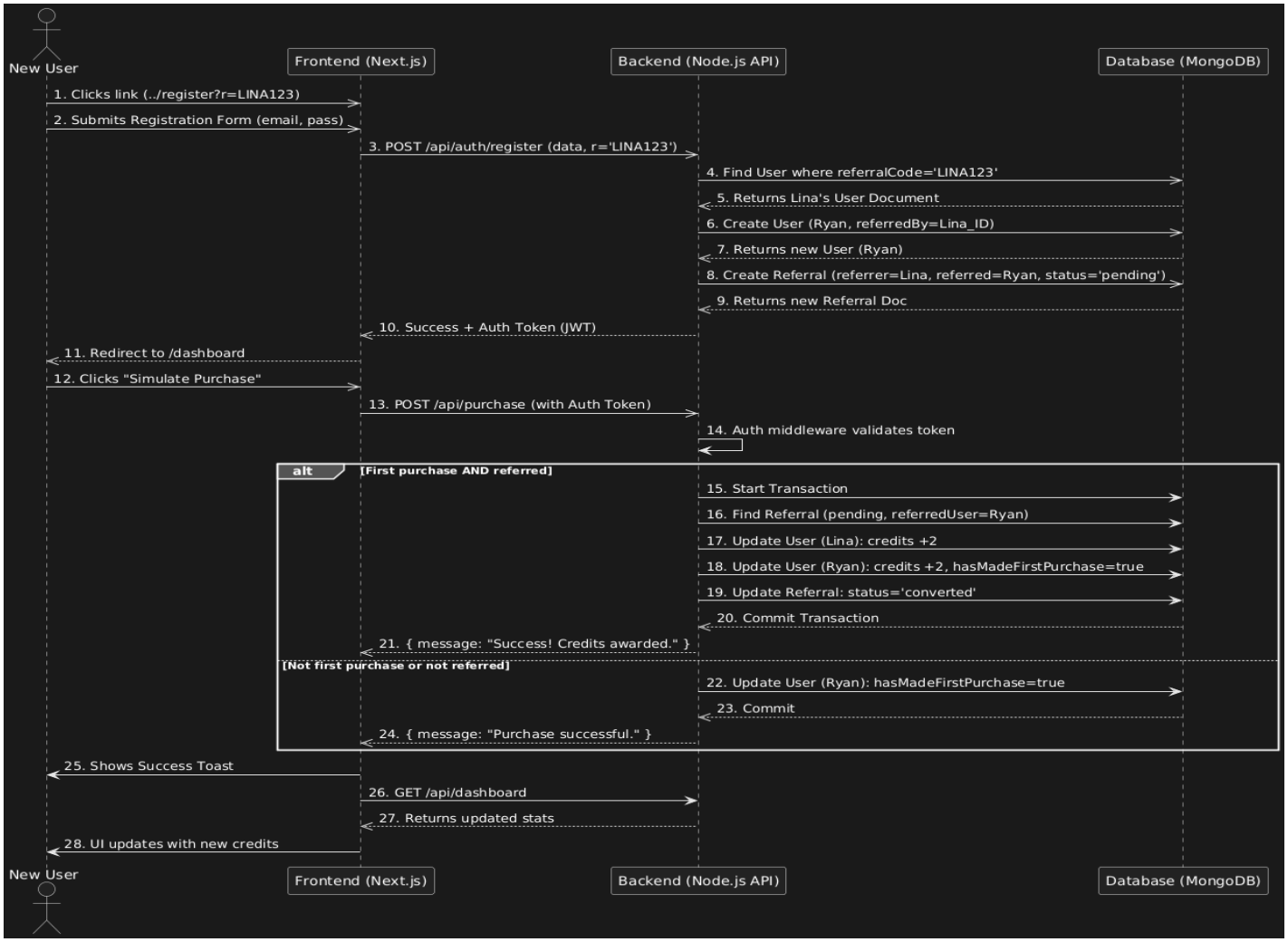



**Component Responsibilities:**

- **Frontend Client:**
  - Renders the UI (Dashboard, Login, Register).
  - Manages client-side state (Zustand).
  - Handles user input and validation.
  - Communicates with the Backend Server via REST API calls.
- **Backend Server:**
  - Handles all business logic (registration, login, purchase simulation).
  - Secures endpoints and manages user authentication (JWT).
  - Performs server-side validation.
  - Interfaces with the MongoDB database to persist and retrieve data.
- **MongoDB Database:**
  - Stores all persistent data, specifically the users and referrals collections.
  - Ensures data integrity (e.g., unique constraints on email and referral codes).

### 3. Core Data Flow (UML Sequence Diagram)

This diagram illustrates the most critical business logic: a new user signing up via a referral link and completing their first purchase .




## 4. Database Schema

The database design consists of two core collections as designed in Step 1.

### `users` Collection

Stores user information, authentication details, and their referral status.

- `email` (String, Required, Unique): User's login email.
- `passwordHash` (String, Required): Hashed password.
- `referralCode` (String, Required, Unique): The user's own unique code (e.g., `LINA123`).
- `credits` (Number, Default: 0): The total number of credits earned.
- `referredBy` (ObjectId, Ref: 'User'): A link to the user who referred them (if any).

- `hasMadeFirstPurchase` (Boolean, Default: false): Critical for tracking the "first purchase only" rule.
- `createdAt` (Date): Timestamp of registration.

#### `referrals` Collection

Explicitly tracks the relationship between a referrer and a referred user to manage status and prevent double-crediting.

- `referrer` (ObjectId, Ref: 'User', Required): The user who owns the link (Lina).
- `referredUser` (ObjectId, Ref: 'User', Required, Unique): The user who signed up (Ryan).
- `status` (String, Enum: ['pending', 'converted'], Default: 'pending'): Tracks if the referral has successfully led to a purchase.
- `createdAt` (Date): Timestamp of the referral.

## 5. API Endpoint Design

The API is structured as a RESTful service.

| Method | Endpoint | Access | Description |
|---|---|---|---|
| POST | /api/auth/register | Public | Registers a new user. Can optionally include a `referralCode` in the body. |
| POST | /api/auth/login | Public | Authenticates a user and returns a JWT. |
| GET | /api/dashboard | Private | **(Protected)** Fetches all dashboard statistics [15] for the authenticated user. |
| POST | /api/purchase | Private | **(Protected)** Simulates a purchase [16] for the authenticated user and triggers the credit award logic. |

---