Esoteric Enterprises Financial Platform - Updated Design Document

1. Executive Summary

1.1 Project Overview

Esoteric Enterprises needs an attractive, secure financial website that enables customers to:

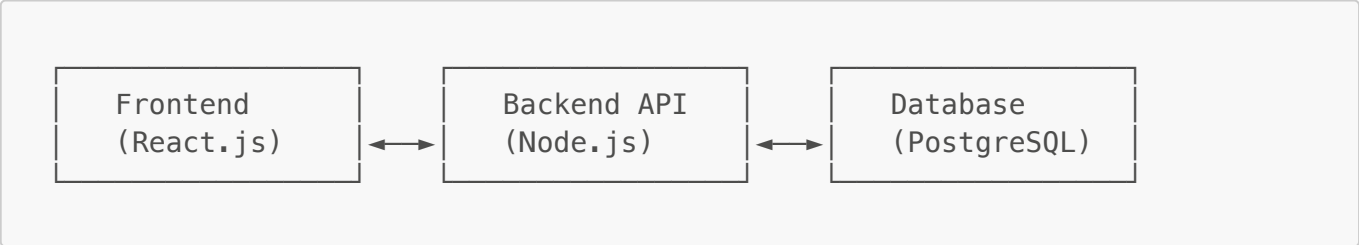
- Create accounts and log in securely with two-factor authentication
- Access a dashboard to view their loan performance and earnings
- View their loan details (principal, monthly payments, bonuses, withdrawals)
- Track growth charts showing bonuses and withdrawals over time
- Access and download important documents
- Schedule meetings with the financial team

1.2 Key Objectives

- Secure financial data protection
- Modern, intuitive user interface
- Scalable architecture for future growth
- Maintainable, well-documented codebase

2. System Architecture

2.1 System Architecture



2.2 Technology Stack

Frontend

- **Framework**: React 19.1.1 with TypeScript
- **UI Library**: Material-UI (MUI) v7.2.0 with Emotion styling
- **Routing**: React Router DOM v7.7.1
- **Charts**: Chart.js 4.5.0 with React-ChartJS-2 integration
- **HTTP Client**: Axios 1.11.0
- **Testing**: React Testing Library with Jest
- **Build Tool**: React Scripts (Create React App)
- **Deployment**: Heroku with serve for production

Backend

- **Runtime**: Node.js with Express 5.1.0
- **Language**: JavaScript with TypeScript support
- **Authentication**: JWT tokens + Speakeasy 2.0.0 for TOTP 2FA
- **Password Security**: Bcryptjs 3.0.2 hashing
- **File Processing**: Multer 2.0.2 for uploads
- **Excel Processing**: XLSX 0.18.5 for bulk operations
- **Security**: Helmet 8.1.0, Express-rate-limit 8.1.0, CORS 2.8.5
- **Validation**: Express-validator 7.2.1
- **Email**: Nodemailer 7.0.6 (configured but notifications disabled)
- **QR Codes**: QRCode 1.5.4 for 2FA setup
- **JWT**: Jsonwebtoken 9.0.2 for secure token handling
- **Testing**: Jest 30.0.5 with Supertest 7.1.4
- **Environment**: Dotenv 17.2.1 for configuration
- **Deployment**: Heroku

Database

- **Database**: PostgreSQL with pg driver 8.16.3
- **Connection**: Connection pooling with SSL support
- **Environment**: Production and test database isolation
- **Backup**: Database safety measures with production protection
- **Testing**: Dedicated test database with automated cleanup

3. Core Features

3.1 User Authentication

- Account registration with validation
- Two-factor authentication (TOTP authenticator apps with QR codes)
- JWT token authentication with configurable expiration
- Secure password hashing with bcrypt
- Session management with token validation
- Rate limiting for login attempts
- Email verification (disabled - database columns remain but functionality removed)

3.2 Customer Portal

Dashboard

- Loan balance overview with Material-UI cards
- Monthly payment tracking with bonus calculations
- Recent transaction display
- Interactive growth chart
- Transaction history with real-time updates
- Responsive design with purple/dark theme

Account Management

- Personal profile management with form validation
- Contact information display
- 2FA setup and management interface
- Account verification status display

Financial Data

- Comprehensive loan transaction history
- Advanced filtering and sorting capabilities
- Real-time loan performance metrics
- Bonus and withdrawal tracking
- Transaction categorization and analytics
- Portfolio dashboard with advanced metrics

Document Center

- Secure document library with categorization
- Document upload/download with file validation
- Access control and authorization
- File size and type restrictions
- Clean file management system

Request Management

- Withdrawal request creation and tracking
- Meeting request scheduling with admin workflow
- Status updates and approval process
- Request history and management

3.3 Admin Panel

- Comprehensive customer management dashboard
- User account administration and verification
- Loan creation and management tools
- Transaction management and processing
- Document management with batch upload capabilities
- Excel bulk upload for loan data processing
- Advanced user filtering and search
- Request approval and management workflow
- System monitoring and user activity tracking

4. Security Framework

4.1 Authentication Security

- **Two-Factor Authentication**: TOTP via authenticator apps or SMS
- **Password Protection**: Bcrypt hashing with salt rounds

- **Token Authentication**: JWT with configurable expiration
- **Rate Limiting**: Protection against brute force attacks
- **Password Policy**: Strength requirements and validation

4.2 Data Protection

- **Encryption**: TLS/SSL for all communications
- **Input Validation**: Sanitization of user inputs
- **SQL Injection Defense**: Parameterized queries
- **XSS Prevention**: Content security policies

4.3 Monitoring & Backup

- **Performance Monitoring**: Error tracking and analytics
- **Security Logging**: Audit trails for authentication
- **Automated Backups**: Daily encrypted database backups

5. Database Design

5.1 Core Tables

```
-- Users table
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    phone VARCHAR(20),
    role VARCHAR(20) DEFAULT 'user',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    requires_2fa BOOLEAN DEFAULT false,
    last_login TIMESTAMP,
    email_verified BOOLEAN DEFAULT false,
    email_verification_token VARCHAR(255),
    email_verification_expires_at TIMESTAMP,
    account_verified BOOLEAN DEFAULT false,
    verified_by_admin INTEGER REFERENCES users(id),
    verified_at TIMESTAMP
);

-- Loan accounts table
CREATE TABLE loan_accounts (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id),
    account_number VARCHAR(50) UNIQUE NOT NULL,
    principal_amount DECIMAL(15,2) NOT NULL,
    current_balance DECIMAL(15,2) NOT NULL,
    monthly_rate DECIMAL(5,4) DEFAULT 0.01,
    total_bonuses DECIMAL(15,2) DEFAULT 0.00,
```

```
    total_withdrawals DECIMAL(15,2) DEFAULT 0.00,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Loan transactions table
CREATE TABLE loan_transactions (
    id SERIAL PRIMARY KEY,
    loan_account_id INTEGER REFERENCES loan_accounts(id),
    amount DECIMAL(15,2) NOT NULL,
    transaction_type VARCHAR(50) NOT NULL,
    bonus_percentage DECIMAL(5,4),
    description TEXT,
    transaction_date DATE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Documents table
CREATE TABLE documents (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id),
    title VARCHAR(255) NOT NULL,
    file_path VARCHAR(500) NOT NULL,
    file_size INTEGER,
    category VARCHAR(100) NOT NULL,
    upload_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- User sessions tracking
CREATE TABLE user_sessions (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id),
    token_hash VARCHAR(255) NOT NULL,
    expires_at TIMESTAMP NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Two-factor authentication
CREATE TABLE user_2fa (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id),
    secret VARCHAR(255) NOT NULL,
    is_enabled BOOLEAN DEFAULT false,
    backup_codes TEXT[],
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- 2FA attempt tracking
CREATE TABLE user_2fa_attempts (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id) ON DELETE CASCADE,
    attempt_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    success BOOLEAN NOT NULL,
    ip_address INET
```

```
);

-- Payment schedule
CREATE TABLE payment_schedule (
    id SERIAL PRIMARY KEY,
    loan_account_id INTEGER REFERENCES loan_accounts(id),
    payment_date DATE NOT NULL,
    base_amount DECIMAL(15,2) NOT NULL,
    bonus_amount DECIMAL(15,2) DEFAULT 0.00,
    total_amount DECIMAL(15,2) NOT NULL,
    status VARCHAR(20) DEFAULT 'pending',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Withdrawal requests
CREATE TABLE withdrawal_requests (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id) ON DELETE CASCADE,
    loan_account_id INTEGER REFERENCES loan_accounts(id) ON DELETE
CASCADE,
    amount DECIMAL(15,2) NOT NULL,
    status VARCHAR(20) DEFAULT 'pending',
    reason TEXT,
    reviewed_by INTEGER REFERENCES users(id),
    reviewed_at TIMESTAMP,
    completed_by INTEGER REFERENCES users(id),
    completed_at TIMESTAMP,
    notes TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Meeting requests
CREATE TABLE meeting_requests (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id) ON DELETE CASCADE,
    preferred_date DATE NOT NULL,
    preferred_time TIME,
    topic VARCHAR(255) NOT NULL,
    description TEXT,
    status VARCHAR(20) DEFAULT 'pending',
    reviewed_by INTEGER REFERENCES users(id),
    reviewed_at TIMESTAMP,
    scheduled_date TIMESTAMP,
    meeting_link VARCHAR(500),
    notes TEXT,
    phone_number VARCHAR(50),
    location TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Account verification requests
CREATE TABLE account_verification_requests (
    id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(id) ON DELETE CASCADE,
```

```
    status VARCHAR(20) DEFAULT 'pending',
    reviewed_by INTEGER REFERENCES users(id),
    reviewed_at TIMESTAMP,
    notes TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

6. API Endpoints

6.1 Authentication & User Management

```
POST /api/auth/register                    - User registration with
validation
POST /api/auth/login                       - Login with 2FA support
POST /api/auth/logout                      - Secure logout
POST /api/auth/complete-2fa-login          - Complete 2FA login flow
GET  /api/user/profile                     - Get user profile
PUT  /api/user/profile                     - Update user profile
POST /api/user/request-account-verification - Request account verification
```

6.2 Two-Factor Authentication

```
POST /api/2fa/setup                        - Initialize 2FA setup with QR
code
POST /api/2fa/verify-setup                 - Verify and enable 2FA
GET  /api/2fa/status                       - Get current 2FA status
POST /api/2fa/generate-backup-codes        - Generate backup codes
POST /api/2fa/disable                      - Disable 2FA
```

6.3 Loan Management

```
GET  /api/loans                            - Get user loans
GET  /api/loans/:loanId/transactions       - Get loan transaction history
GET  /api/loans/:loanId/analytics          - Get loan performance
analytics
```

6.4 Document Management

```
GET    /api/documents                      - Get user documents
GET    /api/documents/:id/download         - Download document
POST   /api/admin/documents/upload         - Admin document upload
GET    /api/admin/documents/:id/download   - Admin document download
DELETE /api/admin/documents/:id            - Delete document
```

6.5 Request Management

```
POST /api/withdrawal-requests              - Create withdrawal request
GET  /api/withdrawal-requests              - Get user withdrawal requests
POST /api/meeting-requests                 - Create meeting request
GET  /api/meeting-requests                 - Get user meeting requests
```

6.6 Admin Operations

```
GET  /api/admin/users                      - Get all users with filtering
GET  /api/admin/users/:id/documents        - Get user documents
GET  /api/admin/users/:id/loans            - Get user loans
GET  /api/admin/users/:id/transactions     - Get user transactions
PUT  /api/admin/users/:id/verify           - Verify user account
POST /api/admin/create-loan                - Create new loan
GET  /api/admin/loans                      - Get all loans
PUT  /api/admin/loans/:id                  - Update loan
POST /api/admin/loans/:id/transactions     - Add loan transaction
GET  /api/admin/loans/:id/transactions     - Get loan transactions
DELETE /api/admin/loans/:id                - Delete loan
GET  /api/admin/withdrawal-requests        - Get all withdrawal requests
PUT  /api/admin/withdrawal-requests/:id    - Update withdrawal request
POST /api/admin/withdrawal-requests/:id/complete - Complete withdrawal
GET  /api/admin/meeting-requests           - Get all meeting requests
PUT  /api/admin/meeting-requests/:id       - Update meeting request
GET  /api/admin/verification-requests      - Get verification requests
PUT  /api/admin/verification-requests/:id  - Update verification request
POST /api/admin/loans/excel-upload         - Bulk loan upload via Excel
GET  /api/admin/loans/excel-template       - Download Excel template
DELETE /api/admin/documents/:id            - Delete document
```

6.7 System Health

```
GET /api/health                            - Health check endpoint
```

7. User Interface Design

7.1 Design Approach

- Modern purple and black color scheme
- Professional design for financial confidence
- Fast loading with sub-second performance

7.2 Key Pages

1. **Login/Register**

2. **Dashboard**
3. **Transactions**
4. **Documents**
5. **Profile**

7.3 Dashboard Design

Color Palette: Custom purple (#6f5cf2) with matte black (#1F2937).

Design Features:

- Purple gradient headers with shadows
- Matte black backgrounds
- Purple accents with hover animations
- White typography for contrast
- Card-based layout with rounded corners
- Modern icons and micro-interactions

8. Development Timeline

Phase 1: Foundation (Weeks 1-2)

- Set up development environment
- Create basic React app with routing
- Set up Node.js backend with database connection
- Implement user registration and login

Phase 2: Core Features (Weeks 3-4)

- Build user dashboard
- Create transaction display and filtering
- Implement basic account management
- Add simple charts for financial data

Phase 3: Document Management (Weeks 5-6)

- File upload system for admins
- Document categorization and storage
- Download functionality for users
- Admin panel for customer management

Phase 4: Polish & Deploy (Weeks 7-8)

- Security hardening and testing
- UI/UX improvements
- Performance optimization
- Production deployment and testing

9. Budget Breakdown

Development Costs

- **Frontend Development**: $2,500
- **Backend Development**: $3,000
- **Security Implementation (inc. 2FA)**: $2,000
- **Loan Management System**: $1,000
- **Payment Processing**: $500
- **Database Setup & Design**: $600
- **Testing & Deployment**: $400

Total Development: $10,000

Monthly Operational Costs

- **Backend Hosting** (DigitalOcean): $20/month
- **Database Hosting**: $15/month
- **Frontend Hosting** (Netlify/Vercel): Free
- **SSL Certificate**: Free (Let's Encrypt)
- **Domain**: $15/year
- **Calendly**: $10-$20/month
- **DocuSign** $10-$20/month **Total Monthly**: ~$60-70

10. Future Features

These are some other features that you could consider implementing in the future, but wouldn't be in scope for the current website.

Advanced Features (Future Phases)

- Advanced analytics and reporting
- Mobile applications
- Automated compliance reporting
- Advanced monitoring and alerting
- Load balancing and auto-scaling
- Microservices architecture

Enterprise Security Features

- Penetration testing
- Advanced intrusion detection
- Compliance certifications
- Advanced audit logging
- Encryption at rest

11. Success Metrics

Technical Goals

- Page load times under 3 seconds
- 99% uptime

- Zero security incidents
- All core features working properly

Business Goals

- Easy customer onboarding
- Customers can access their data quickly
- Documents are easily accessible
- Admins can manage customer data efficiently

12. Testing Infrastructure (Implemented)

12.1 Comprehensive Test Suite

- **Jest Testing Framework**: 30.0.5 with Supertest for API testing
- **Database Safety**: Production database protection with isolated test database
- **Test Coverage**: Enhanced authentication, user profiles, loans, API integration
- **Security Testing**: Rate limiting, XSS prevention, SQL injection protection
- **File Upload Testing**: Document upload validation with automatic cleanup
- **Environment Isolation**: Dedicated test environment configuration

12.2 Test Suite Coverage

```
auth-enhanced.test.js          - Complete authentication flows
user-profile.test.js           - Profile management and validation
loans-basic.test.js           - Loan operations and transactions
robust-api.test.js            - API integration and error handling
admin-comprehensive.test.js   - Admin operations and management
security-advanced.test.js     - Security measures validation
performance-load.test.js      - Performance and load testing
api-integration-edge.test.js  - Edge cases and integration
excel-upload.test.js          - Excel bulk upload functionality
```

12.3 Database Safety Features

- **Production Protection**: Hard-coded prevention of test execution against production
- **Test Database Isolation**: Automatic esoteric_loans_test database usage
- **Environment Validation**: Tests fail if NODE_ENV !== 'test'
- **Clean State Management**: Automated database cleanup between tests
- **Connection Validation**: Verification of correct database connections

13. Risk Mitigation

Technical Risks (Addressed)

- **Data Loss**: Database safety measures and test isolation implemented
- **Security Breach**: Comprehensive security testing and validation
- **Test Data Contamination**: Production database protection implemented

- **Code Quality**: Extensive test coverage with automated validation

Business Risks

- **Scope Creep**: All major features successfully implemented
- **Security Compliance**: 2FA, rate limiting, and security measures in place
- **Maintainability**: Well-documented codebase with comprehensive tests