Software Requirements Specification (SRS)

User Registration and Authentication System

Document Control

Item	Description
Document Title	User Registration and Authentication - Software Requirements Specification
Document Version	1.0
Document Status	Final
Last Updated	April 8, 2025
Based on BRD Version	1.0 dated March 20, 2025

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides the technical requirements for implementing the User Registration and Authentication component of the Emtelaak Platform. It translates the business requirements outlined in the BRD into specific system behaviors, interfaces, and technical specifications that developers can directly implement.

1.2 Scope

This SRS covers the detailed technical specifications for:

- User registration process and workflows
- Identity verification (KYC/AML) integration
- Investor accreditation verification
- Authentication and authorization mechanisms
- User profile management
- Role-based access control implementation
- UI/UX requirements with wireframes
- API specifications and data models

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
API	Application Programming Interface
JWT	JSON Web Token
KYC	Know Your Customer
AML	Anti-Money Laundering
MFA	Multi-Factor Authentication
RBAC	Role-Based Access Control
SSO	Single Sign-On
OTP	One-Time Password
UI	User Interface
UX	User Experience
GDPR	General Data Protection Regulation
CCPA	California Consumer Privacy Act

1.4 References

- User Registration BRD (user-registration-brd.docx)
- Project Charter (project-charter.docx)
- Development Approach and Tech Stack (development-approach.docx)
- User Registration Flow (User Registration flow.pdf)

1.5 Overview

The remainder of this document is organized into sections covering:

- System architecture and technical overview
- Functional requirements with detailed specifications
- User interface requirements with wireframes
- External interface requirements for system integrations
- Non-functional requirements including security and performance
- Data models and database schema
- API specifications
- Testing requirements

2. System Overview

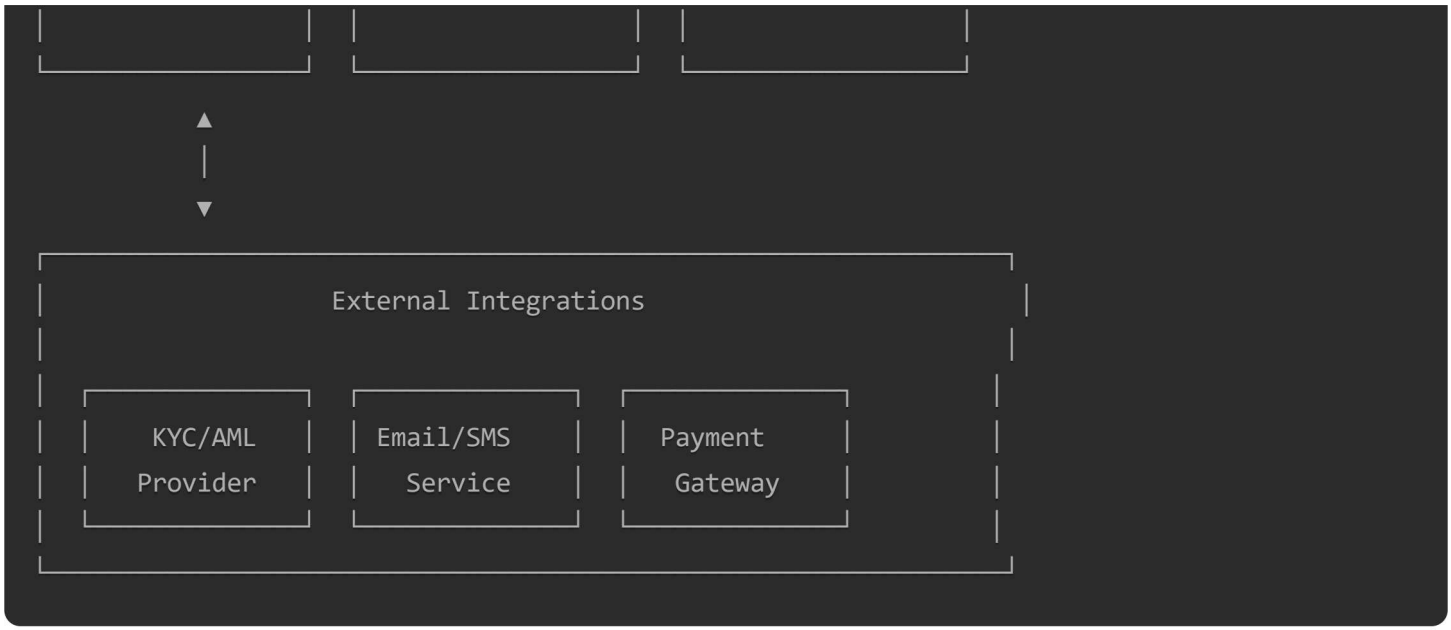
2.1 System Architecture

The User Registration and Authentication system will be implemented as a microservice within the Emtelaak platform architecture following the clean architecture principles outlined in the Development Approach document.

2.1.1 High-Level Architecture Diagram

Copy


Client Layer
Web App (Angular)
Mobile App (Flutter)
API Gateway
Authentication/Authorization
User Registration & Authentication Microservice
Registration API
Identity Verification
User Profile
Auth Service
Accreditation Verification
Admin Service
SQL Database
Document Storage
Cache (Redis)



2.2 System Context

The User Registration and Authentication system will interface with the following components:

1. **Client Applications**
   - Web application (Angular)
   - Mobile application (Flutter)
2. **Internal Systems**
   - Wallet management system
   - Investment processing system
   - Property listing system
   - Notification system
   - Admin dashboard
3. **External Systems**
   - KYC/AML verification service
   - Email service provider
   - SMS service provider
   - Payment processing system

2.3 Technology Stack

Based on the Development Approach document, the system will use:

1. **Backend**
   - ASP.NET Core 8

- Entity Framework Core for data access
- IdentityServer4 for OAuth 2.0 and OpenID Connect implementation
- SQL Server for primary data storage
- Redis for caching and session management
- Azure Blob Storage for document storage

2. **Frontend**
   - Web: Angular with TypeScript
   - Mobile: Flutter with Dart
3. **DevOps**
   - Azure DevOps for CI/CD
   - Docker for containerization
   - Azure App Service for hosting

3. Functional Requirements

3.1 User Registration

3.1.1 Registration Process Flow

FR-REG-01: Step-by-Step Registration Process

- The system shall implement a multi-step registration process with the following sequence:
  1. Basic Information Entry (email, password)
  2. Email Verification
  3. Personal Information Collection
  4. Phone Verification
  5. User Type Selection
  6. Terms Acceptance
  7. Account Creation Confirmation

FR-REG-02: Basic Information Collection

- The system shall collect the following basic information:
  - Email address (unique identifier)
  - Password (meeting security requirements)
  - First name

- Last name
- Country of residence

FR-REG-03: Email Verification

- The system shall:
  - Generate a verification code or link
  - Send verification to the user's email
  - Verify the code/link when provided by user
  - Limit verification attempts and provide expiration mechanism
  - Allow resending of verification emails

FR-REG-04: Phone Verification

- The system shall:
  - Accept phone numbers in international format
  - Send SMS with verification code
  - Verify the code when provided by user
  - Allow resending of verification SMS
  - Limit verification attempts

FR-REG-05: User Type Selection

- The system shall allow users to select one of the following user types:
  - Individual Investor
  - Institutional Investor
  - Property Issuer
  - Different fields will be collected based on user type

FR-REG-06: Terms and Conditions Acceptance

- The system shall:
  - Display terms of service and privacy policy
  - Require explicit acceptance by user
  - Record acceptance date and version of terms accepted
  - Support updates to terms with re-acceptance flow

FR-REG-07: Registration Completion

- The system shall:
  - Create user account upon successful registration
  - Assign appropriate initial role
  - Generate welcome email
  - Redirect to appropriate next steps (KYC, dashboard, etc.)

3.1.2 KYC/AML Verification

FR-KYC-01: KYC Information Collection

- The system shall collect KYC information based on user type:
  - Individual: Full name, date of birth, address, nationality, government ID
  - Institutional: Company name, registration number, business address, authorized representatives, corporate documents

FR-KYC-02: Document Upload

- The system shall:
  - Support upload of identity documents (passport, ID card, driver's license)
  - Support upload of address proof documents (utility bill, bank statement)
  - Validate file formats (JPG, PNG, PDF) and size limits (max 10MB)
  - Securely store documents in encrypted blob storage
  - Support document preview and replacement

FR-KYC-03: Integration with KYC Provider

- The system shall:
  - Transmit user information and documents to third-party KYC provider via API
  - Receive verification results (approved, rejected, additional information needed)
  - Update user KYC status based on provider response
  - Implement error handling for API failures

FR-KYC-04: Verification Status Management

- The system shall manage the following KYC statuses:
  - Not Started
  - In Progress
  - Pending Review

- Additional Information Required
- Approved
- Rejected
- Expired

FR-KYC-05: Re-verification Process

- The system shall:
  - Allow users to resubmit verification if rejected
  - Support periodic re-verification based on risk assessment
  - Notify users of upcoming or required re-verification

3.1.3 Investor Accreditation

FR-ACC-01: Accreditation Information Collection

- The system shall collect accreditation information:
  - Income documentation
  - Net worth verification
  - Investment experience
  - Entity status for institutions
  - Accredited investor certification

FR-ACC-02: Accreditation Document Upload

- The system shall:
  - Support upload of financial documents
  - Validate file formats and sizes
  - Securely store documents
  - Support document management (view, replace, delete)

FR-ACC-03: Accreditation Verification Processing

- The system shall:
  - Support manual review by compliance team
  - Support third-party verification integration
  - Update investor accreditation status based on verification
  - Implement process for handling exceptions

FR-ACC-04: Investor Classification

- The system shall classify investors according to regulatory requirements:
  - Accredited Investor
  - Qualified Investor
  - Non-Accredited Investor
  - Institutional Investor

FR-ACC-05: Investment Limits Enforcement

- The system shall:
  - Assign investment limits based on investor classification
  - Enforce limits during investment process
  - Support override mechanisms with appropriate approvals

3.2 Authentication and Authorization

3.2.1 Authentication

FR-AUTH-01: Login Process

- The system shall:
  - Authenticate users via email/password combination
  - Implement secure cookie handling
  - Support "Remember Me" functionality
  - Implement account lockout after failed attempts
  - Provide password reset functionality

FR-AUTH-02: Multi-Factor Authentication

- The system shall:
  - Support SMS-based MFA
  - Support authenticator app integration (TOTP)
  - Allow users to enable/disable MFA
  - Support MFA recovery process
  - Generate backup codes for recovery

FR-AUTH-03: Token Management

- The system shall:
  - Generate JWT tokens for authenticated sessions
  - Include appropriate claims in tokens
  - Implement token refresh mechanism
  - Support token revocation
  - Implement secure token storage

FR-AUTH-04: Biometric Authentication

- The system shall:
  - Support biometric authentication on mobile app
  - Use device biometric APIs (TouchID, FaceID, fingerprint)
  - Implement fallback authentication method
  - Securely store biometric tokens

FR-AUTH-05: Session Management

- The system shall:
  - Create secure user sessions
  - Implement configurable session timeout
  - Allow users to view and terminate active sessions
  - Track session information (device, location, last activity)

3.2.2 Authorization

FR-RBAC-01: Role Definition

- The system shall define the following roles:
  - Individual Investor
  - Institutional Investor
  - Property Issuer
  - Platform Administrator
  - Compliance Officer
  - Support Staff

FR-RBAC-02: Permission Management

- The system shall:

- Define granular permissions for system functions
- Group permissions into logical sets
- Associate permissions with roles
- Support custom permission configurations

FR-RBAC-03: Role Assignment

- The system shall:
  - Assign initial role based on registration
  - Support role changes by administrators
  - Support multiple role assignments
  - Maintain role assignment history

FR-RBAC-04: Access Control Enforcement

- The system shall:
  - Validate user permissions for all operations
  - Implement access control at API and UI levels
  - Log access control violations
  - Provide clear feedback for unauthorized access attempts

3.3 User Profile Management

3.3.1 Profile Information

FR-PROF-01: Profile Data Management

- The system shall allow users to manage:
  - Personal information
  - Contact information
  - Address information
  - Professional information
  - Profile picture

FR-PROF-02: Profile Completion

- The system shall:
  - Track profile completion percentage

- Prompt users to complete missing information
- Require critical information based on user type
- Reward complete profiles (if gamification implemented)

FR-PROF-03: Profile Privacy

- The system shall:
  - Allow users to control profile visibility
  - Define public/private information
  - Implement privacy settings for different information categories

3.3.2 Settings Management

FR-SET-01: Notification Preferences

- The system shall allow users to manage:
  - Email notification preferences
  - SMS notification preferences
  - Push notification preferences
  - Notification frequency settings

FR-SET-02: Security Settings

- The system shall allow users to manage:
  - Password changes
  - MFA settings
  - Session management
  - Login notifications
  - Device authorization

FR-SET-03: Language Preferences

- The system shall:
  - Support multiple language options
  - Allow users to set preferred language
  - Apply language preference across the platform

FR-SET-04: Display Preferences

- The system shall:
  - Support light/dark mode
  - Support accessibility preferences
  - Allow customization of dashboard elements

3.3.3 Account Management

FR-ACCT-01: Account Status

- The system shall support the following account statuses:
  - Active
  - Suspended
  - Restricted
  - Closed
  - Deleted

FR-ACCT-02: Account Closure

- The system shall:
  - Allow users to request account closure
  - Implement account closure workflow
  - Handle data retention and deletion according to regulations
  - Support account recovery within specified time period

FR-ACCT-03: Account Activity Logs

- The system shall:
  - Track login history and activity
  - Display activity logs to users
  - Alert on suspicious activities
  - Support downloading of activity logs

4. User Interface Requirements

4.1 General UI Requirements

UI-GEN-01: Responsive Design

- All interfaces shall be responsive and optimized for:

- Desktop browsers (min resolution 1024x768)
- Tablets (portrait and landscape)
- Mobile devices (min width 320px)

UI-GEN-02: Accessibility

- All interfaces shall:
  - Conform to WCAG 2.1 Level AA standards
  - Support screen readers
  - Provide sufficient color contrast
  - Implement keyboard navigation

UI-GEN-03: UI Components

- The system shall use consistent UI components:
  - Form controls (inputs, dropdowns, checkboxes)
  - Buttons (primary, secondary, tertiary)
  - Navigation elements
  - Modal dialogs
  - Notification/alert components

UI-GEN-04: Design System

- The UI shall implement Emtelaak design system:
  - Color palette
  - Typography
  - Icons and imagery
  - Spacing and layout
  - Animation and transitions

4.2 Registration Interface

UI-REG-01: Registration Flow

- The registration interface shall:
  - Display current step and total steps
  - Allow navigation between completed steps
  - Validate input in real-time

- Provide clear error messages
- Save progress for later completion

UI-REG-02: Form Validation

- The registration forms shall:
  - Validate input in real-time
  - Display field-specific error messages
  - Highlight required fields
  - Allow submission only when all required fields are valid

UI-REG-03: Mobile-Optimized Registration

- The mobile registration shall:
  - Break complex forms into smaller steps
  - Optimize for touch input
  - Support camera access for document scanning
  - Minimize text input where possible

4.3 Authentication Interface

UI-AUTH-01: Login Screen

- The login screen shall:
  - Support email/password input
  - Provide password reset link
  - Include "Remember Me" option
  - Display appropriate error messages
  - Support social login (if implemented)

UI-AUTH-02: MFA Interface

- The MFA interface shall:
  - Display clear instructions
  - Support numeric code input
  - Include resend code option
  - Provide access to alternative authentication methods

UI-AUTH-03: Password Reset

- The password reset interface shall:
  - Request email address
  - Confirm reset email sent
  - Validate reset token
  - Support new password entry and confirmation
  - Confirm successful password change

4.4 User Profile Interface

UI-PROF-01: Profile Dashboard

- The profile dashboard shall:
  - Display profile completion status
  - Highlight required actions
  - Show verification status
  - Provide quick access to common profile actions

UI-PROF-02: Profile Edit

- The profile edit interface shall:
  - Group related information
  - Support inline editing
  - Validate changes before submission
  - Confirm successful updates

UI-PROF-03: Settings Interface

- The settings interface shall:
  - Organize settings in logical categories
  - Use appropriate controls for different setting types
  - Provide help text and tooltips
  - Save changes immediately or with explicit save action

5. User Interface Wireframes

5.1 Registration Flow Wireframes

5.1.1 Registration Step 1: Basic Information

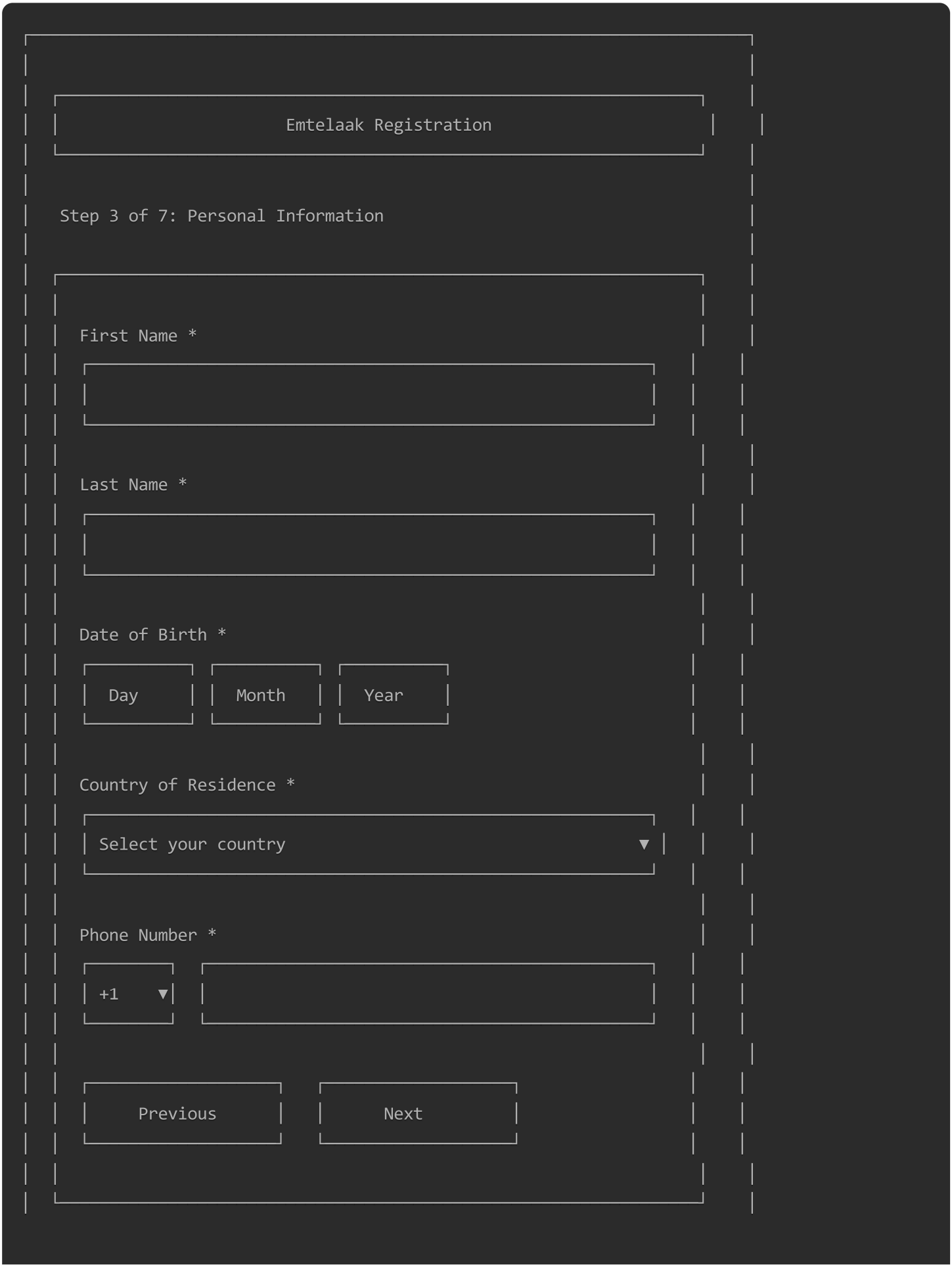
Copy

5.1.2 Registration Step 2: Email Verification

Copy

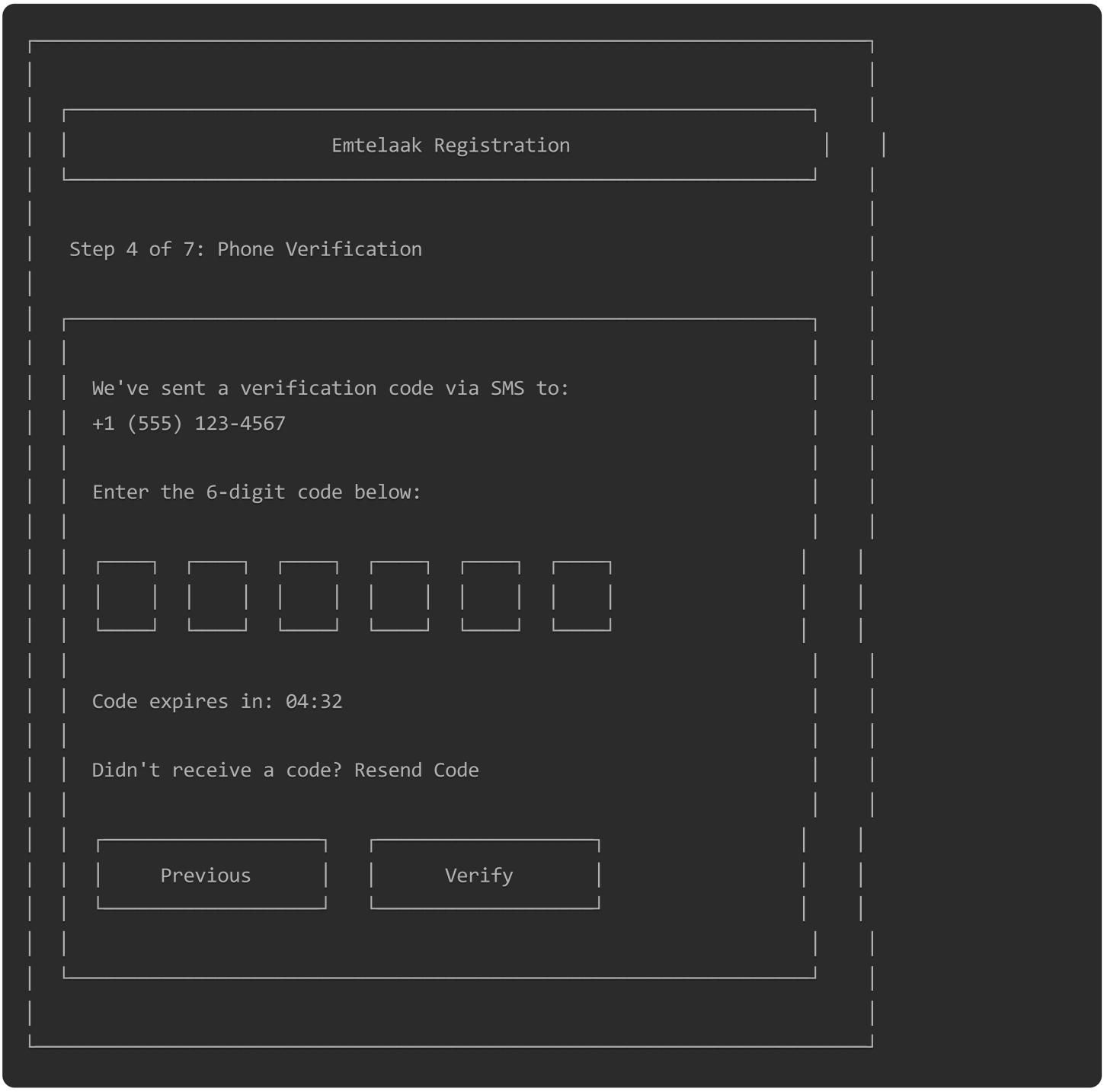
5.1.3 Registration Step 3: Personal Information

Copy


Emtelaak Registration
Step 3 of 7: Personal Information
First Name *
Last Name *
Date of Birth *
Day
Month
Year
Country of Residence *
Select your country
Phone Number *
+1
Previous
Next

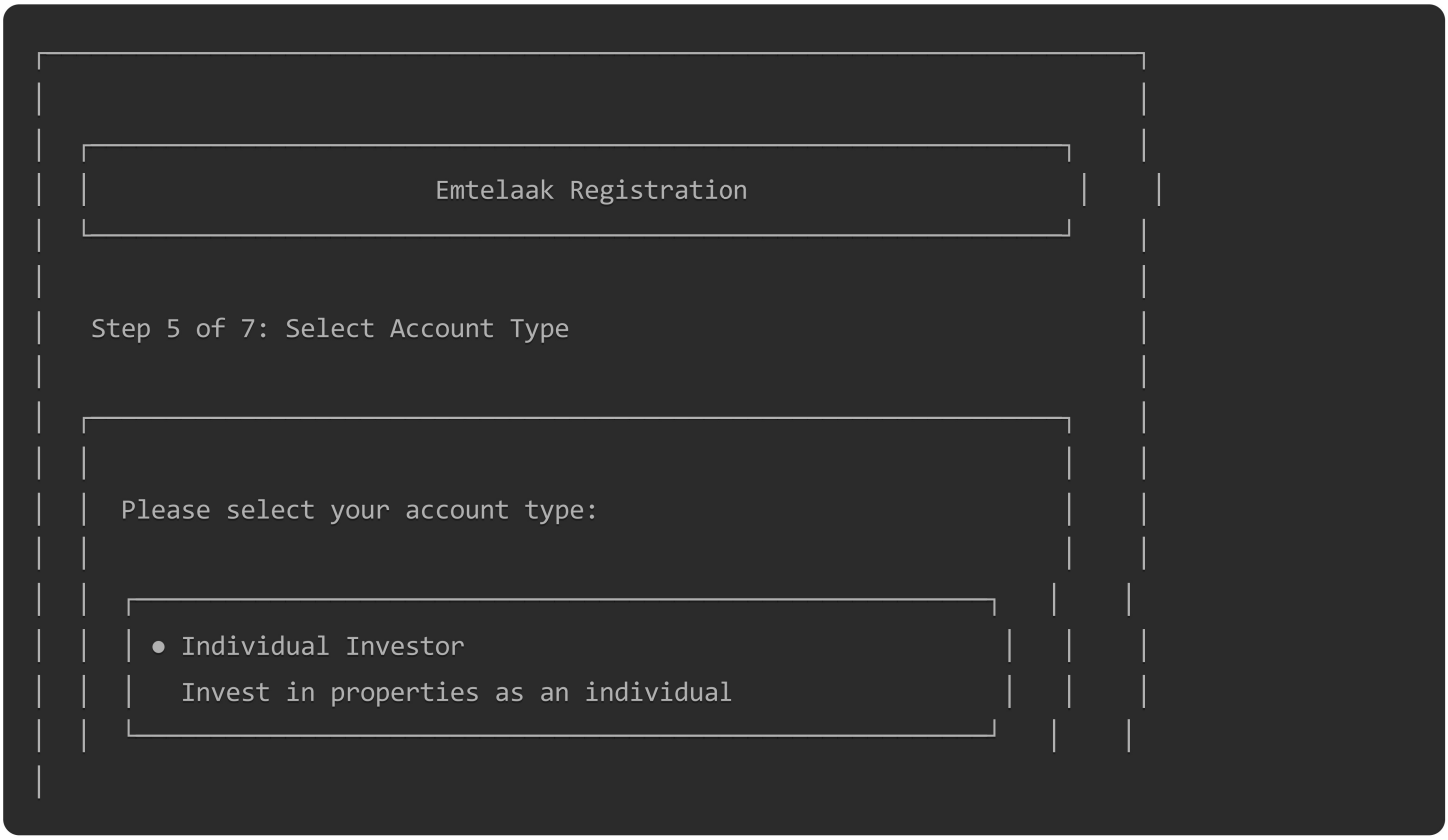
5.1.4 Registration Step 4: Phone Verification

Copy



5.1.5 Registration Step 5: User Type Selection

Copy


Emtelaak Registration
Step 5 of 7: Select Account Type
Please select your account type:
• Individual Investor
Invest in properties as an individual

5.1.5 Registration Step 5: User Type Selection (continued)

Copy

5.1.6 Registration Step 6: Terms and Conditions

Copy


Emtelaak Registration
Step 6 of 7: Terms and Conditions
Please review and accept our terms
Terms of Service
[Scrollable terms of service text appears here]
Privacy Policy
[Scrollable privacy policy text appears here]
☑ I accept the Terms of Service
☑ I accept the Privacy Policy
Previous
Next

5.1.7 Registration Step 7: Completion

Copy

5.2 Authentication Interface Wireframes

5.2.1 Login Screen

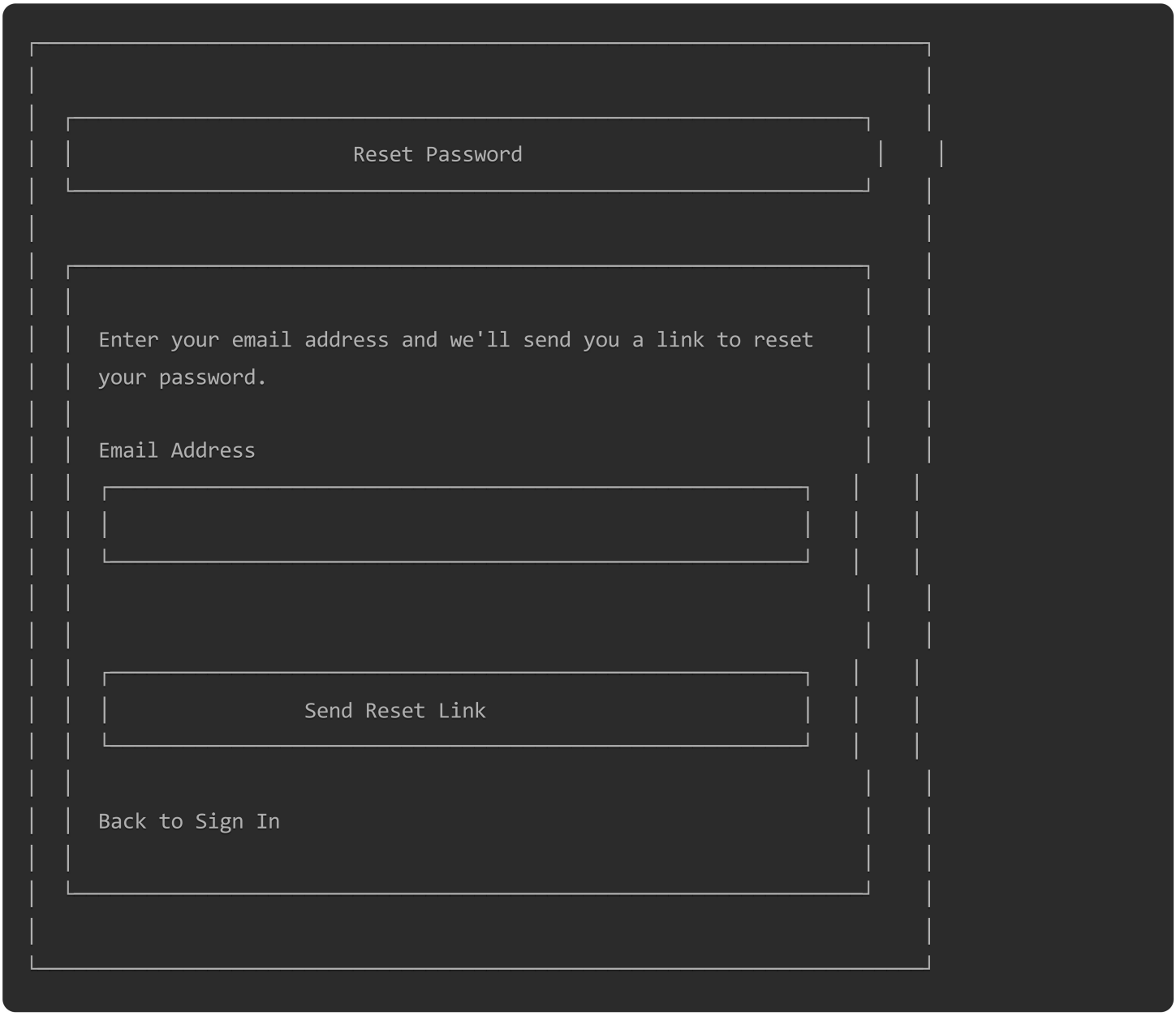
Copy

5.2.2 Multi-Factor Authentication

Copy

5.2.3 Password Reset Request

Copy



5.2.4 New Password Entry

Copy

5.3 KYC Verification Wireframes

5.3.1 KYC Overview Screen

Copy

5.3.2 Document Upload Screen

Copy


Identity Verification
Step 2 of 3: Document Verification
Upload a photo of your government-issued ID
Document Type *
Select ID type
Front Side of ID *
Drag & Drop or Click to Upload
Back Side of ID *
Drag & Drop or Click to Upload
Take photos with mobile camera
Previous
Next

5.4 User Profile Interface Wireframes

5.4.1 Profile Dashboard

Copy


Emtelaak
Logo
Bell
User
Menu
Dashboard > My Profile
My Profile
Photo
John Smith
john.smith@example.com
+1 (555) 123-4567
Edit Profile
Profile Completion
75%
Complete your KYC verification to unlock investing
Personal Info
Name
Contact
Address
Tax Information
Manage
Documents
Identity Documents
Address Proof
Financial Docs
Upload
Verification Status
KYC Verification
In Progress
View Status
Security
Password
2FA Settings
Login Devices
Preferences
Language
Notifications
Theme
Display Settings
Activity
Login History
Recent Activities
Account Changes

6. External Interface Requirements

6.1 API Interfaces

6.1.1 KYC/AML Service Integration

INT-KYC-01: KYC Provider API

- The system shall integrate with the KYC provider's API with the following specifications:
  - REST API with JSON request/response format
  - Authentication using API key and client secret
  - Endpoints for:
    - Document submission
    - Verification status checks
    - User data submission
    - Webhook registration for status updates
  - Error handling with retry mechanism
  - Logging of all API interactions

INT-KYC-02: KYC Webhook Handler

- The system shall implement a webhook handler to:
  - Receive verification status updates
  - Process verification results
  - Update user KYC status
  - Trigger notifications based on status changes
  - Implement security measures for webhook validation

6.1.2 Communication Services

INT-COMM-01: Email Service

- The system shall integrate with email service APIs to:
  - Send verification emails
  - Send notification emails
  - Send password reset emails
  - Track email delivery status
  - Process bounces and failures

INT-COMM-02: SMS Service

- The system shall integrate with SMS service APIs to:
  - Send verification codes
  - Send authentication codes
  - Send notification messages
  - Track message delivery
  - Handle delivery failures

6.2 Internal System Interfaces

INT-SYS-01: Wallet System

- The system shall provide APIs for the wallet system to:
  - Verify user identity status
  - Check user authorization
  - Retrieve user profile information
  - Validate account status

INT-SYS-02: Investment System

- The system shall provide APIs for the investment system to:
  - Verify investor accreditation status
  - Retrieve investor profile data
  - Validate investment eligibility based on user classification
  - Retrieve user KYC status

INT-SYS-03: Notification System

- The system shall provide APIs for the notification system to:
  - Retrieve user notification preferences

- Get user contact information
- Verify user notification permissions
- Update notification status

7. Non-functional Requirements

7.1 Performance Requirements

NFR-PERF-01: Response Time

- The system shall provide the following response times under normal load:
  - Page load time: < 2 seconds
  - Form submission: < 3 seconds
  - Authentication: < 2 seconds
  - API response time: < 500ms

NFR-PERF-02: Throughput

- The system shall support:
  - At least 100 registrations per minute
  - At least 500 authentication requests per minute
  - At least 200 profile updates per minute

NFR-PERF-03: Scalability

- The system shall:
  - Scale horizontally to handle increased load
  - Support at least 10,000 concurrent users
  - Maintain performance metrics during peak usage

7.2 Security Requirements

NFR-SEC-01: Authentication Security

- The system shall:
  - Use HTTPS for all communications
  - Implement OAuth 2.0 and OpenID Connect protocols
  - Support JWT token-based authentication
  - Store passwords using bcrypt with appropriate cost factor

- Implement secure session management

NFR-SEC-02: Data Protection

- The system shall:
  - Encrypt personally identifiable information at rest
  - Implement field-level encryption for sensitive data
  - Use TLS 1.2+ for data in transit
  - Implement secure storage for identification documents
  - Comply with data protection regulations (GDPR, CCPA)

NFR-SEC-03: Access Control

- The system shall:
  - Implement strict role-based access control
  - Validate all authorization for API endpoints
  - Audit all access to sensitive information
  - Implement the principle of least privilege
  - Prevent unauthorized access attempts

NFR-SEC-04: Security Monitoring

- The system shall:
  - Log all security-relevant events
  - Monitor for suspicious activities
  - Implement intrusion detection
  - Alert on potential security incidents
  - Support security audit capabilities

7.3 Reliability Requirements

NFR-REL-01: Availability

- The system shall:
  - Maintain 99.9% uptime (excluding planned maintenance)
  - Implement redundancy for critical components
  - Support failover mechanisms
  - Recover

7.3 Reliability Requirements (continued)

NFR-REL-01: Availability

- The system shall:
  - Maintain 99.9% uptime (excluding planned maintenance)
  - Implement redundancy for critical components
  - Support failover mechanisms
  - Recover from failures automatically when possible

NFR-REL-02: Data Durability

- The system shall:
  - Implement regular database backups
  - Support point-in-time recovery
  - Maintain data integrity across system components
  - Implement transaction durability for critical operations

NFR-REL-03: Error Handling

- The system shall:
  - Implement graceful error handling
  - Provide meaningful error messages to users
  - Log detailed error information for troubleshooting
  - Support automated error alerting for critical failures
  - Implement retry mechanisms for transient failures

7.4 Usability Requirements

NFR-USA-01: User Interface

- The system shall:
  - Provide intuitive navigation
  - Use consistent layout and design
  - Support responsive design for all device types
  - Implement clear visual feedback for user actions
  - Support task completion with minimal steps

NFR-USA-02: Accessibility

- The system shall:
  - Comply with WCAG 2.1 Level AA standards
  - Support keyboard navigation
  - Provide appropriate color contrast
  - Include screen reader support
  - Implement focus management for all interactive elements

NFR-USA-03: Internationalization

- The system shall:
  - Support multiple languages (initially English and Arabic)
  - Use proper internationalization patterns
  - Support right-to-left text display
  - Format dates, times, and numbers according to locale
  - Store all text in resource files for easy translation

7.5 Compliance Requirements

NFR-COMP-01: Regulatory Compliance

- The system shall:
  - Comply with relevant financial regulations
  - Implement KYC/AML requirements
  - Support audit trails for compliance verification
  - Maintain records according to retention policies
  - Support regulatory reporting requirements

NFR-COMP-02: Data Privacy

- The system shall:
  - Comply with GDPR, CCPA, and other applicable privacy regulations
  - Implement data subject rights (access, deletion, portability)
  - Support privacy by design principles
  - Maintain clear records of consent
  - Implement data minimization practices

8. Data Models

8.1 User Entity

Copy

```
User {
    Id: UUID (PK)
    Email: String (unique)
    PasswordHash: String
    PhoneNumber: String
    PhoneVerified: Boolean
    EmailVerified: Boolean
    FirstName: String
    LastName: String
    DateOfBirth: Date
    CountryOfResidence: String
    UserType: Enum (IndividualInvestor, InstitutionalInvestor, PropertyIssuer)
    Status: Enum (Active, Suspended, Restricted, Closed, Deleted)
    CreatedAt: DateTime
    UpdatedAt: DateTime
    LastLoginAt: DateTime
    FailedLoginAttempts: Integer
    LockoutEndDate: DateTime
    TermsAcceptedVersion: String
    TermsAcceptedAt: DateTime
    ReferralCode: String
}
```

8.2 KYC Entity

Copy

```
KYC {
    Id: UUID (PK)
    UserId: UUID (FK to User)
    Status: Enum (NotStarted, InProgress, PendingReview, AdditionalInfoRequired, Approved, Reje
    VerificationId: String (external provider reference)
    SubmittedAt: DateTime
    ApprovedAt: DateTime
    RejectedAt: DateTime
    RejectionReason: String
    ExpiresAt: DateTime
    LastUpdatedAt: DateTime
    RiskLevel: Enum (Low, Medium, High)
    VerificationType: Enum (Basic, Enhanced)
}
```

8.3 Document Entity

Copy

```
Document {
    Id: UUID (PK)
    UserId: UUID (FK to User)
    DocumentType: Enum (IdCard, Passport, DriversLicense, UtilityBill, BankStatement, IncomePro
    FileName: String
    FileSize: Integer
    ContentType: String
    StoragePath: String
    UploadedAt: DateTime
    ExpiryDate: DateTime
    VerificationStatus: Enum (Pending, Verified, Rejected)
    RejectionReason: String
    VerifiedAt: DateTime
}
```

8.4 Accreditation Entity

Copy

```
Accreditation {
    Id: UUID (PK)
    UserId: UUID (FK to User)
    InvestorClassification: Enum (Accredited, Qualified, NonAccredited, Institutional)
    Status: Enum (Pending, Approved, Rejected, Expired)
    ApprovedAt: DateTime
    ExpiresAt: DateTime
    ApprovedBy: UUID (FK to User, admin user)
    IncomeLevel: Decimal
    NetWorth: Decimal
    InvestmentLimitAmount: Decimal
    ReviewNotes: String
    LastUpdatedAt: DateTime
}
```

8.5 UserRole Entity

Copy

```
UserRole {
    Id: UUID (PK)
    UserId: UUID (FK to User)
    RoleId: UUID (FK to Role)
    AssignedAt: DateTime
    AssignedBy: UUID (FK to User, admin user)
    IsActive: Boolean
}
```

8.6 Role Entity

Copy

```
Role {
    Id: UUID (PK)
    Name: String
    Description: String
    IsSystem: Boolean
    CreatedAt: DateTime
    UpdatedAt: DateTime
}
```

8.7 Permission Entity

Copy

```
Permission {
    Id: UUID (PK)
    Code: String (unique)
    Name: String
    Description: String
    Category: String
    CreatedAt: DateTime
}
```

8.8 RolePermission Entity

Copy

```
RolePermission {
    Id: UUID (PK)
    RoleId: UUID (FK to Role)
    PermissionId: UUID (FK to Permission)
    AssignedAt: DateTime
    AssignedBy: UUID (FK to User, admin user)
}
```

8.9 UserSession Entity

Copy

```
UserSession {
    Id: UUID (PK)
    UserId: UUID (FK to User)
    Token: String
    IssuedAt: DateTime
    ExpiresAt: DateTime
    LastActivityAt: DateTime
    IpAddress: String
    UserAgent: String
    DeviceInfo: String
    IsActive: Boolean
    RevokedAt: DateTime
    RevokedReason: String
}
```

8.10 UserPreference Entity

Copy

```
UserPreference {
    Id: UUID (PK)
    UserId: UUID (FK to User)
    Language: String
    Theme: Enum (Light, Dark, System)
    NotificationEmail: Boolean
    NotificationSms: Boolean
    NotificationPush: Boolean
    LoginNotification: Boolean
    TwoFactorEnabled: Boolean
    TwoFactorMethod: Enum (SMS, Authenticator, Email)
    UpdatedAt: DateTime
}
```

8.11 ActivityLog Entity

Copy

```
ActivityLog {
    Id: UUID (PK)
    UserId: UUID (FK to User)
    ActivityType: Enum (Login, Logout, ProfileUpdate, PasswordChange, VerificationSubmit, etc.)
    IpAddress: String
    UserAgent: String
    Timestamp: DateTime
    Details: JSON
    Status: Enum (Success, Failure)
    FailureReason: String
}
```

9. API Specifications

9.1 Authentication API

9.1.1 Registration Endpoint

Endpoint: `POST /api/auth/register`

Description: Creates a new user account with basic information

Request:

json

Copy

```
{
  "email": "string",
  "password": "string",
  "firstName": "string",
  "lastName": "string",
  "phoneNumber": "string",
  "countryOfResidence": "string",
  "userType": "string (IndividualInvestor, InstitutionalInvestor, PropertyIssuer)",
  "termsAccepted": true,
  "privacyAccepted": true
}
```

Response (201 Created):

json Copy

```
{
  "userId": "string (UUID)",
  "email": "string",
  "requiresEmailVerification": true,
  "requiresPhoneVerification": true
}
```

Response (400 Bad Request):

json Copy

```
{
  "errors": {
    "email": ["Email is already registered"],
    "password": ["Password does not meet complexity requirements"]
  }
}
```

9.1.2 Email Verification Endpoint

Endpoint: `POST /api/auth/verify-email`

Description: Verifies user's email with verification code

Request:

json Copy

```
{
  "email": "string",
  "verificationCode": "string"
}
```

Response (200 OK):

json Copy

```
{
  "verified": true,
  "nextStep": "string (phoneVerification, KYC, dashboard)"
}
```

Response (400 Bad Request):

json Copy

```
{
  "verified": false,
  "message": "Invalid or expired verification code",
  "remainingAttempts": 3
}
```

9.1.3 Phone Verification Endpoint

Endpoint: `POST /api/auth/verify-phone`

Description: Verifies user's phone number with verification code

Request:

json Copy

```
{
  "phoneNumber": "string",
  "verificationCode": "string"
}
```

Response (200 OK):

json Copy

```
{
  "verified": true,
  "nextStep": "string (KYC, dashboard)"
}
```

Response (400 Bad Request):

```json
{
  "verified": false,
  "message": "Invalid or expired verification code",
  "remainingAttempts": 3
}
```

9.1.4 Login Endpoint

Endpoint: `POST /api/auth/login`

Description: Authenticates user and issues access token

Request:

```json
{
  "email": "string",
  "password": "string",
  "rememberMe": true
}
```

Response (200 OK):

```json
{
  "accessToken": "string (JWT)",
  "refreshToken": "string",
  "expiresIn": 3600,
  "userId": "string (UUID)",
  "requiresMfa": false
}
```

Response (200 OK with MFA):

json Copy

```
{
  "requiresMfa": true,
  "mfaToken": "string",
  "mfaMethods": ["SMS", "Authenticator"]
}
```

Response (401 Unauthorized):

json Copy

```
{
  "message": "Invalid email or password",
  "remainingAttempts": 3
}
```

9.1.5 Multi-Factor Authentication Endpoint

Endpoint: `POST /api/auth/mfa-verify`

Description: Verifies MFA code and completes authentication

Request:

json Copy

```
{
  "mfaToken": "string",
  "verificationCode": "string",
  "method": "string (SMS, Authenticator)"
}
```

Response (200 OK):

json Copy

```
{
  "accessToken": "string (JWT)",
  "refreshToken": "string",
  "expiresIn": 3600,
  "userId": "string (UUID)"
}
```

Response (401 Unauthorized):

json Copy

```
{
  "message": "Invalid verification code",
  "remainingAttempts": 3
}
```

9.1.6 Password Reset Request Endpoint

Endpoint: `POST /api/auth/forgot-password`

Description: Initiates password reset process

Request:

json Copy

```
{
  "email": "string"
}
```

Response (200 OK):

json Copy

```
{
  "message": "If your email is registered, you will receive a reset link shortly",
  "emailSent": true
}
```

9.1.7 Password Reset Completion Endpoint

Endpoint: `POST /api/auth/reset-password`

Description: Completes password reset with new password

Request:

json Copy

```
{
  "token": "string",
  "newPassword": "string",
  "confirmPassword": "string"
}
```

Response (200 OK):

json Copy

```
{
  "success": true,
  "message": "Password has been reset successfully"
}
```

Response (400 Bad Request):

json Copy

```
{
  "success": false,
  "message": "Token is invalid or expired",
  "errors": {
    "newPassword": ["Password does not meet complexity requirements"]
  }
}
```

9.2 KYC Verification API

9.2.1 KYC Status Endpoint

Endpoint: `GET /api/kyc/status`

Description: Retrieves user's KYC verification status

Response (200 OK):

```json
{
  "status": "string (NotStarted, InProgress, PendingReview, AdditionalInfoRequired, Approved, R
  "submittedAt": "string (ISO date)",
  "expiresAt": "string (ISO date)",
  "rejectionReason": "string",
  "requiredDocuments": [
    {
      "type": "string",
      "status": "string (Pending, Verified, Rejected)",
      "rejectionReason": "string"
    }
  ],
  "nextStep": "string"
}
```

9.2.2 KYC Document Upload Endpoint

Endpoint: `POST /api/kyc/documents`

Description: Uploads a document for KYC verification

Request:

```
multipart/form-data:
- documentType: string (IdCard, Passport, DriversLicense, etc.)
- file: binary data
```

Response (201 Created):

json

Copy

```
{
  "documentId": "string (UUID)",
  "documentType": "string",
  "fileName": "string",
  "uploadedAt": "string (ISO date)",
  "status": "Pending"
}
```

Response (400 Bad Request):

json

Copy

```
{
  "errors": {
    "file": ["File size exceeds maximum allowed (10MB)"],
    "documentType": ["Invalid document type"]
  }
}
```

9.2.3 KYC Information Submission Endpoint

Endpoint: `POST /api/kyc/submit`

Description: Submits KYC information for verification

Request:

json

Copy

```
{
  "personalInfo": {
    "fullName": "string",
    "nationality": "string",
    "dateOfBirth": "string (ISO date)",
    "placeOfBirth": "string",
    "gender": "string"
  },
  "address": {
    "street": "string",
    "city": "string",
    "state": "string",
    "zipCode": "string",
    "country": "string"
  },
  "documentIds": ["string (UUID)", "string (UUID)"]
}
```

Response (202 Accepted):

json

Copy

```
{
  "verificationId": "string",
  "status": "InProgress",
  "submittedAt": "string (ISO date)"
}
```

9.3 User Profile API

9.3.1 User Profile Retrieval Endpoint

Endpoint: `GET /api/users/profile`

Description: Retrieves the current user's profile

Response (200 OK):

json

Copy

```
{
  "userId": "string (UUID)",
  "email": "string",
  "phoneNumber": "string",
  "firstName": "string",
  "lastName": "string",
  "dateOfBirth": "string (ISO date)",
  "countryOfResidence": "string",
  "profilePictureUrl": "string",
  "userType": "string",
  "kycStatus": "string",
  "accreditationStatus": "string",
  "profileCompletionPercentage": 75,
  "createdAt": "string (ISO date)",
  "lastLoginAt": "string (ISO date)"
}
```

9.3.2 User Profile Update Endpoint

Endpoint: `PUT /api/users/profile`

Description: Updates user profile information

Request:

json

Copy

```
{
  "firstName": "string",
  "lastName": "string",
  "phoneNumber": "string",
  "dateOfBirth": "string (ISO date)",
  "countryOfResidence": "string"
}
```

Response (200 OK):

json

Copy

```
{
  "userId": "string (UUID)",
  "updated": true,
  "profile": {
    "firstName": "string",
    "lastName": "string",
    "phoneNumber": "string",
    "dateOfBirth": "string (ISO date)",
    "countryOfResidence": "string"
  }
}
```

9.3.3 User Preferences Update Endpoint

Endpoint: `PUT /api/users/preferences`

Description: Updates user preferences

Request:

json

Copy

```
{
  "language": "string",
  "theme": "string (Light, Dark, System)",
  "notificationEmail": true,
  "notificationSms": true,
  "notificationPush": true,
  "loginNotification": true
}
```

Response (200 OK):

json Copy

```
{
  "updated": true,
  "preferences": {
    "language": "string",
    "theme": "string",
    "notificationEmail": true,
    "notificationSms": true,
    "notificationPush": true,
    "loginNotification": true
  }
}
```

9.3.4 Password Change Endpoint

Endpoint: `PUT /api/users/password`

Description: Changes user password

Request:

json Copy

```
{
  "currentPassword": "string",
  "newPassword": "string",
  "confirmPassword": "string"
}
```

Response (200 OK):

json Copy

```
{
  "updated": true,
  "message": "Password updated successfully"
}
```

Response (400 Bad Request):

```
json
```

Copy

```
{
  "updated": false,
  "errors": {
    "currentPassword": ["Current password is incorrect"],
    "newPassword": ["Password does not meet complexity requirements"]
  }
}
```

9.3.5 Profile Picture Upload Endpoint

Endpoint: `POST /api/users/profile-picture`

Description: Uploads user profile picture

Request:

Copy

```
multipart/form-data:
- file: binary data
```

Response (200 OK):

```
json
```

Copy

```
{
  "profilePictureUrl": "string",
  "uploadedAt": "string (ISO date)"
}
```

10. Security Considerations

10.1 Authentication Security

1. **Password Security**
   - Password complexity requirements: minimum 10 characters, uppercase, lowercase, number, and symbol
   - Password hashing using bcrypt with a cost factor of 12

- Password history maintained to prevent reuse of recent passwords

2. **Token Security**
   - JWT tokens with appropriate expiration (60 minutes for access tokens)
   - Token refreshing mechanism with longer-lived refresh tokens (14 days)
   - Token revocation on logout
   - Secure token storage guidelines for client applications
3. **Session Management**
   - HTTPS-only cookies with Secure and HttpOnly flags
   - SameSite attribute set to Lax or Strict
   - Session timeout after 30 minutes of inactivity
   - Absolute session timeout after 8 hours

10.2 Data Protection

1. **Sensitive Data Handling**
   - PII data encrypted at rest using AES-256
   - Field-level encryption for particularly sensitive fields
   - Secure document storage with access controls
   - Document metadata stored separately from document content
2. **Data Transmission**
   - TLS 1.2+ required for all communications
   - Strong cipher suites required
   - Perfect Forward Secrecy (PFS) supported
   - HTTP Strict Transport Security (HSTS) enabled
3. **Database Security**
   - Connection encryption
   - Parameterized queries to prevent SQL injection
   - Minimal database user privileges
   - Database audit trails for sensitive operations

10.3 API Security

1. **Request Validation**
   - Input validation for all API requests

- Request size limits
- Content-Type enforcement
- CSRF protection for browser clients

2. **Rate Limiting**
   - Rate limits on authentication endpoints
   - Graduated rate limiting based on request frequency
   - IP-based and token-based rate limiting
   - Appropriate 429 Too Many Requests responses

3. **Error Handling**
   - Consistent error format
   - Limited error details in production
   - No stack traces or system details in responses
   - Detailed error logging for internal use

11. Testing Requirements

11.1 Unit Testing

The following components shall be covered by unit tests:

1. **Authentication Service**
   - Password hashing and verification
   - Token generation and validation
   - MFA code generation and validation
   - Password complexity validation

2. **KYC Service**
   - Document validation
   - KYC status management
   - Verification workflow

3. **User Service**
   - Profile management
   - Preference management
   - Role and permission handling

11.2 Integration Testing

The following integration points shall be tested:

1. **KYC Provider Integration**
   - Document submission
   - Verification status updates
   - Error handling
2. **Communication Services**
   - Email delivery
   - SMS sending
   - Template rendering
3. **Internal System Integrations**
   - Wallet system integration
   - Investment system integration
   - Notification system integration

11.3 System Testing

The following end-to-end flows shall be tested:

1. **User Registration Flow**
   - Complete registration process
   - Email and phone verification
   - Edge cases and validation
2. **KYC Verification Flow**
   - Document upload
   - Information submission
   - Approval and rejection scenarios
3. **Authentication Flow**
   - Login with and without MFA
   - Password reset
   - Session management

11.4 Security Testing

The following security aspects shall be tested:

1. **Authentication Security**
   - Password strength enforcement
   - Account lockout
   - Session management
   - MFA effectiveness
2. **Authorization Testing**
   - Role-based access control
   - Permission validation
   - Resource isolation
3. **Penetration Testing**
   - OWASP Top 10 vulnerabilities
   - API security testing
   - Injection attacks
   - Session management vulnerabilities

11.5 Performance Testing

The following performance aspects shall be tested:

1. **Load Testing**
   - Concurrent user registration
   - Authentication under load
   - API endpoint performance
2. **Stress Testing**
   - System behavior under extreme load
   - Recovery from overload
   - Failure scenarios
3. **Endurance Testing**
   - System behavior over extended period
   - Memory management
   - Resource utilization

12. Implementation Guidelines

12.1 Technology Implementation

1. **ASP.NET Core Implementation**
   - Use ASP.NET Core 8 Web API
   - Implement Clean Architecture pattern
   - Apply SOLID principles
   - Use Dependency Injection
2. **Entity Framework Core**
   - Use Code-First approach
   - Implement migrations for database versioning
   - Use repository pattern for data access
   - Implement query optimization techniques
3. **Identity Server 4**
   - Configure OAuth 2.0 and OpenID Connect
   - Implement JWT token issuance and validation
   - Configure identity resources and scopes
   - Implement client credentials for service-to-service authentication

12.2 Project Structure

Copy

```
src/
  Emtelaak.UserRegistration.API/
    Controllers/
    Startup.cs
    Program.cs
    appsettings.json

  Emtelaak.UserRegistration.Application/
    Commands/
    Queries/
    DTOs/
    Interfaces/
    Services/
    Validators/

  Emtelaak.UserRegistration.Domain/
    Entities/
    Enums/
    Events/
    Exceptions/
    ValueObjects/

  Emtelaak.UserRegistration.Infrastructure/
    Data/
    Migrations/
    Repositories/
    ExternalServices/
    Configuration/

  Emtelaak.UserRegistration.Shared/
    Constants/
    Extensions/
    Utils/

tests/
  Emtelaak.UserRegistration.UnitTests/
  Emtelaak.UserRegistration.IntegrationTests/
  Emtelaak.UserRegistration.SystemTests/
```

12.3 Security Implementation

1. **Authentication Implementation**
   - Use Identity Server 4 for OAuth and OIDC
   - Use ASP.NET Core Identity for user management
   - Implement JWT token handling
   - Configure secure cookie options
2. **Authorization Implementation**
   - Implement policy-based authorization
   - Define authorization handlers for complex rules
   - Use role-based authorization attributes on controllers
   - Implement resource-based authorization for data access
3. **Data Protection Implementation**
   - Use ASP.NET Core Data Protection for key management
   - Implement custom encryption for sensitive fields
   - Configure secure storage providers
   - Use Azure Key Vault for key storage in production

13. Acceptance Criteria

13.1 Registration Acceptance Criteria

1. Users can complete the registration process with all required steps
2. Email verification is enforced before account activation
3. Phone verification is successful with valid codes
4. User type selection affects the profile information collected
5. Terms and conditions acceptance is recorded with timestamp
6. Registration metrics meet performance requirements
7. Mobile registration flow provides equivalent functionality

13.2 KYC Verification Acceptance Criteria

1. Users can upload all required document types
2. Document validation rejects invalid formats or sizes
3. KYC information submission sends data to the verification provider
4. Status updates are received and processed correctly
5. Rejection reasons are communicated clearly to users

6. Re-verification process allows document resubmission
7. Mobile document capture works with device cameras

13.3 Authentication Acceptance Criteria

1. Users can login with correct credentials
2. Account lockout occurs after specified failed attempts
3. MFA is enforced when enabled and works correctly
4. Password reset process works end-to-end
5. Session management follows security requirements
6. Remember Me functionality works as specified
7. Biometric authentication works on mobile devices

13.4 User Profile Acceptance Criteria

1. Users can view and edit their profile information
2. Profile completeness is accurately calculated
3. Settings changes are saved correctly
4. Password changes enforce security requirements
5. Profile picture uploads and displays correctly
6. Activity log shows relevant user activities
7. Mobile profile management provides full functionality

14. Appendices

14.1 Glossary

- **KYC**: Know Your Customer - The process of verifying the identity of users
- **AML**: Anti-Money Laundering - Regulations to prevent illegal money activities
- **MFA**: Multi-Factor Authentication - Authentication using multiple verification methods
- **JWT**: JSON Web Token - A compact way to securely transmit information between parties
- **RBAC**: Role-Based Access Control - Access control based on user roles
- **TOTP**: Time-based One-Time Password - Temporary passwords that change based on time
- **PII**: Personally Identifiable Information - Data that can identify a specific individual
- **GDPR**: General Data Protection Regulation - EU data protection regulation
- **CCPA**: California Consumer Privacy Act - California's data privacy law

14.2 References

1. Development Approach and Tech Stack (development-approach.docx)
2. Project Charter (project-charter.docx)
3. User Registration BRD (user-registration-brd.docx)
4. User Registration Flow Diagram (User Registration flow.pdf)
5. Feature Comparison (emtelaak-feature-comparison.docx)
6. Project Timeline (project-timeline.docx)

14.3 Document Revision History

Version	Date	Description	Author
0.1	March 25, 2025	Initial draft	[Author Name]
0.2	March 30, 2025	Added API specifications	[Author Name]
0.3	April 3, 2025	Added wireframes and data models	[Author Name]
0.4	April 6, 2025	Technical review updates	[Author Name]
1.0	April 8, 2025	Final version	[Author Name]