Authentication API Specification - Revised

Version: 1.2.0 | Last Updated: June 26, 2025

Overview

Comprehensive authentication and user management endpoints for the CodeBlooded API, providing secure user registration, authentication, profile management, preferences, and password management capabilities.

Authentication Method

- **Token Type:** PASETO v3.local tokens
- **Header:** `Authorization: Bearer <token>`
- **Token Expiration:** Configurable (default: 24 hours)

Endpoints

1. User Registration

Endpoint: `POST /api/auth/register`

Description: Creates a new user account with validated credentials.

Request Body:

```
{
  "full_name": "Jane Doe",
  "username": "jane.doe",
  "email": "jane@example.com",
  "password": "SecretPass@123"
}
```

Validation Rules:

- `full_name`: Must contain exactly first and last name (letters only)
- `username`: 3-15 characters, lowercase letters, dots, underscores only

- `email`: Valid email format
- `password`: Min 8 chars, must include uppercase, lowercase, number, special character

Success Response (201 Created):

```
{
  "status": "success",
  "timestamp": "2025-06-24T14:30:00.000Z",
  "message": "User registered successfully.",
  "data": {
    "user": {
      "id": 1,
      "full_name": "Jane Doe",
      "username": "jane.doe",
      "email": "jane@example.com"
    }
  }
}
```

Error Response (400 Bad Request):

```
{
  "status": "error",
  "errors": [
    {
      "field": "username",
      "message": "Username must be between 3 and 15 characters."
    }
  ]
}
```

2. User Login

Endpoint: `POST /api/auth/login`

Description: Authenticates user and returns secure PASETO token.

Request Body:

```
{
  "username": "jane.doe",
  "password": "SecretPass@123"
}
```

Success Response (200 OK):

```
{
  "status": "success",
  "data": {
    "user": {
      "id": 1,
      "username": "jane.doe"
    },
    "token": "v3.local.encrypted_token_here",
    "expires_at": "2025-06-25T14:30:00.000Z"
  }
}
```

Error Response (401 Unauthorized):

```
{
  "status": "error",
  "message": "Invalid credentials"
}
```

3. Get User ID by Username

Endpoint: `GET /api/auth/user-id/:username`

Description: Retrieves user ID for a given username.

Parameters:

- `username` (path): Target username

Success Response (200 OK):

```
{
  "status": "success",
  "message": "User ID retrieved for jane.doe",
  "data": {
    "user_id": 1
  }
}
```

Error Response (404 Not Found):

```
{
  "status": "error",
  "message": "User jane.doe not found"
}
```

4. Get User Profile

Endpoint: `GET /api/auth/:userId`

Description: Retrieves complete user information.

Authentication: Required

Success Response (200 OK):

```
{
  "status": "success",
  "data": {
    "user": {
      "user_id": 1,
      "username": "jane.doe",
      "full_name": "Jane Doe",
      "email": "jane@example.com",
      "two_factor_enabled": false,
      "created_at": "2025-06-24T14:30:00.000Z",
      "updated_at": "2025-06-24T14:30:00.000Z"
    }
  }
}
```

5. Update Comprehensive Settings

Endpoint: `PUT /api/auth/:userId/settings`

Description: Updates user profile, preferences, and notification settings in a single transaction. **Authentication:** Required

Request Body:

```
{
  "username": "jane.smith",
  "full_name": "Jane Smith",
  "theme": "dark",
  "avatar_id": 3,
  "inAppNotifications": true,
  "outOfAppEnabled": false,
  "twoFactorEnabled": true
}
```

Success Response (200 OK):

```
{
  "status": "success",
  "message": "User settings updated successfully."
}
```

6. Change Password

Endpoint: `PUT /api/auth/:userId/change-password`

Description: Changes user password with old password verification. **Authentication:** Required

Request Body:

```
{
  "oldPassword": "OldSecretPass@123",
  "newPassword": "NewSecretPass@456"
}
```

Success Response (200 OK):

```
{
  "status": "success",
  "message": "Password updated successfully."
}
```

Error Response (400 Bad Request):

```
{
  "status": "error",
  "message": "Incorrect old password"
}
```

---

7. Delete User Account

Endpoint: `DELETE /api/auth/:userId`

Description: Permanently deletes user account and all associated data. **Authentication:** Required

Success Response (200 OK):

```
{
  "status": "success",
  "message": "User account deleted successfully."
}
```

---

8. Get Profile Top Bar Data

Endpoint: `GET /api/auth/top-bar/:userId`

Description: Returns username, avatar, banner, and join date for the profile top bar display. **Authentication:** Required

Parameters: `userId` (number): The ID of the user profile to fetch

Request Body:

```
{
  "status": "success",
  "message": "Profile top bar data loaded.",
  "data": {
    "username": "john.doe",
    "created_at": "2025-01-15T08:30:00.000Z",
    "avatar_image_path": "/avatars/avatar_5.jpg",
    "banner_image_path": "/banners/banner_3.jpg"
  }
}
```

9.Get Sidebar Statistics

Endpoint: `GET /api/auth/sidebar/:userId`

Description: Returns sidebar stats including goals, achievements, accounts, transactions, communities, and lessons. **Authentication:** Required

Parameters: `userId` (number): The ID of the user to get stats for

Request Body:

```
{
  "status": "success",
  "message": "Sidebar statistics loaded.",
  "data": {
    "total_goals": 5,
    "achievement_percentage": 75,
    "total_accounts": 3,
    "recent_transactions": 12,
    "total_communities": 2,
    "lessons_completed_percentage": 60
  }
}
```

Goals Management Endpoints

The following endpoints should be added to your existing Authentication API specification to provide comprehensive financial goal management capabilities.

10. Create Goal

Endpoint: `POST /api/goal`
Description: Creates a new financial goal for a user.
Authentication: Required

Request Body:

json

```
{
  "user_id": 1,
  "goal_name": "Emergency Fund",
  "goal_type": "savings",
  "target_amount": 10000,
  "target_date": "2025-12-31",
  "goal_status": "in-progress"
}
```

Validation Rules:

- `user_id`: Required, must be a valid user ID
- `goal_name`: Required, string
- `goal_type`: Required, must be one of: "savings", "debt", "investment", "spending limit", "donation"
- `target_amount`: Required, positive number
- `target_date`: Required, date in YYYY-MM-DD format
- `goal_status`: Required, must be one of: "in-progress", "completed", "paused", "cancelled", "failed"

Success Response (201 Created):

```
json
{
  "status": "success",
  "message": "Goal created successfully.",
  "data": {
    "goal_id": 1
  }
}
```

Error Response (400 Bad Request):

```
json
{
  "status": "error",
  "message": "Missing required fields: user_id or community_id, goal_name, goal_type,
                        target_amount, target_date, goal_status"
}
```

11. Get Goal by ID

Endpoint: `GET /api/goal/:goalId`
Description: Retrieves a single goal by its ID.
Authentication: Required

Parameters:

- `goalId` (path): Goal ID to retrieve

Success Response (200 OK):

```
json
{
  "status": "success",
  "data": {
    "goal_id": 1,
    "user_id": 1,
    "goal_name": "Emergency Fund",
    "goal_type": "savings",
    "target_amount": 10000,
    "current_amount": 2500,
    "target_date": "2025-12-31",
    "goal_status": "in-progress",
    "created_at": "2025-06-24T14:30:00.000Z",
    "updated_at": "2025-06-24T14:30:00.000Z"
  }
}
```

Error Response (404 Not Found):

```
json
{
  "status": "error",
  "message": "Goal not found"
}
```

12. Get User Goals

Endpoint: `GET /api/goal/user/:userId`
Description: Retrieves all personal goals for a specific user.
Authentication: Required

Parameters:

- `userId` (path): User ID to fetch goals for

Success Response (200 OK):

json

```
{
  "status": "success",
  "data": [
    {
      "goal_id": 1,
      "user_id": 1,
      "goal_name": "Emergency Fund",
      "goal_type": "savings",
      "target_amount": 10000,
      "current_amount": 2500,
      "target_date": "2025-12-31",
      "goal_status": "in-progress",
      "created_at": "2025-06-24T14:30:00.000Z",
      "updated_at": "2025-06-24T14:30:00.000Z"
    }
  ]
}
```

13. Update Goal

Endpoint: `PUT /api/goal/:goalId`
Description: Updates a goal's details.
Authentication: Required

Parameters:

- `goalId` (path): Goal ID to update

Request Body:

json

```
{
  "goal_name": "Updated Emergency Fund",
  "target_amount": 15000,
  "target_date": "2026-06-30",
  "goal_status": "in-progress"
}
```

Success Response (200 OK):

json

```
{
  "status": "success",
  "message": "Goal updated successfully"
}
```

14. Delete Goal

Endpoint: `DELETE /api/goal/:goalId`
Description: Permanently deletes a goal by its ID.
Authentication: Required

Parameters:

- `goalId` (path): Goal ID to delete

Success Response (200 OK):

json

```
{
  "status": "success",
  "message": "Goal deleted successfully"
}
```

15. Add Goal Progress

Endpoint: `POST /api/goal/:goalId/progress`
Description: Adds progress/contribution to a specific goal.
Authentication: Required

Parameters:

- `goalId` (path): Goal ID to add progress to

Request Body:

```json
{
  "contributor_id": 1,
  "amount_added": 500
}
```

Validation Rules:

- `contributor_id`: Required, must be a valid user ID
- `amount_added`: Required, positive number

Success Response (201 Created):

```json
{
  "status": "success",
  "data": {
    "progress_id": 1
  }
}
```

Error Response (400 Bad Request):

```json
{
  "status": "error",
  "message": "contributor_id and amount_added are required"
}
```

17. Complete Goal

Endpoint: `POST /api/goal/:goalId/complete`
Description: Marks a goal as completed and awards points to the user.
Authentication: Required

Parameters:

- `goalId` (path): Goal ID to complete

Success Response (200 OK):

```
json
{
  "status": "success",
  "message": "Goal marked as completed"
}
```

18. Reduce Goal Progress

Endpoint: `POST /api/goal/:goalId/reduce`
Description: Reduces goal progress and deducts points from the user.
Authentication: Required

Parameters:

- `goalId` (path): Goal ID to reduce progress for

Request Body:

```
json
{
  "amount": 250
}
```

Validation Rules:

- `amount`: Required, positive number

Success Response (200 OK):

```
json
{
  "status": "success",
  "message": "Progress reduced"
```

}

Error Response (400 Bad Request):

json

```
{
  "status": "error",
  "message": "amount is required"
}
```

19. Get All Goals

Endpoint: `GET /api/goal`
Description: Retrieves all goals in the system.
Authentication: Required

Success Response (200 OK):

json

```
{
  "status": "success",
  "data": [
    {
      "goal_id": 1,
      "user_id": 1,
      "goal_name": "Emergency Fund",
      "goal_type": "savings",
      "target_amount": 10000,
      "current_amount": 2500,
      "target_date": "2025-12-31",
      "goal_status": "in-progress",
      "created_at": "2025-06-24T14:30:00.000Z",
      "updated_at": "2025-06-24T14:30:00.000Z"
    }
  ]
}
```

20. Get User Goal Statistics

Endpoint: `GET /api/goal/user/:user_id/stats`
Description: Retrieves goal statistics for a specific user.
Authentication: Required

Parameters:

- `user_id` (path): User ID to get statistics for

Success Response (200 OK):

json

```
{
  "status": "success",
  "data": {
    "total_goals": 5,
    "completed_goals": 2,
    "total_saved": 15750
  }
}
```

Learning Management Endpoints

21. Get All Learning Modules

Endpoint: `GET /api/learning`

Description: Retrieves all available learning modules with lesson counts.

Authentication: Required

Success Response (200 OK):

```
{
 "status": "success",
 "data": [
   {
     "module_id": 1,
     "title": "Introduction to Finance",
     "description": "Basic financial concepts",
     "difficulty": "beginner",
     "lesson_count": 5,
     "created_at": "2025-06-24T14:30:00.000Z"
   }
 ]
}
```

22. Get Learning Module by ID

Endpoint: `GET /api/learning/:moduleId`

Description: Retrieves a specific learning module by its ID.

Authentication: Required

Parameters:

`moduleId` (path): Module ID to retrieve

Success Response (200 OK):

```
{
 "status": "success",
 "data": {
   "module_id": 1,
   "title": "Introduction to Finance",
   "description": "Basic financial concepts",
   "difficulty": "beginner",
   "created_at": "2025-06-24T14:30:00.000Z"
 }
}
```

Error Response (404 Not Found):

```
{
 "status": "error",
 "message": "Module with ID 1 not found"
}
```

23. Get Completed Modules

Endpoint: `GET /api/learning/completed/:userId`

Description: Retrieves all completed learning modules for a specific user.

Authentication: Required

Parameters:

`userId` (path): User ID to fetch completed modules for

Success Response (200 OK):

```
{
 "status": "success",
 "data": [
   {
     "module_id": 1,
     "title": "Introduction to Finance",
     "description": "Basic financial concepts",
     "difficulty": "beginner",
     "lesson_count": 5
   }
 ]
}
```

24. Get Uncompleted Modules

Endpoint: `GET /api/learning/uncompleted/:userId`

Description: Retrieves all uncompleted learning modules for a specific user.

Authentication: Required

Parameters:

`userId` (path): User ID to fetch uncompleted modules for

Success Response (200 OK):

```
{
 "status": "success",
 "data": [
   {
     "module_id": 2,
     "title": "Advanced Investment Strategies",
     "description": "Complex investment concepts",
     "difficulty": "advanced",
     "lesson_count": 8
   }
 ]
}
```

25. Get Module Lessons

Endpoint: `GET /api/learning/:moduleId/lessons`

Description: Retrieves all lessons for a specific learning module.

Authentication: Required

Parameters:

`moduleId` (path): Module ID to fetch lessons for

Success Response (200 OK):

```
{
 "status": "success",
 "data": [
   {
     "lesson_id": 1,
     "module_id": 1,
     "title": "What is Money?",
     "content": "Lesson content here...",
     "lesson_number": 1,
     "created_at": "2025-06-24T14:30:00.000Z"
   }
 ]
}
```

26. Get Lesson by ID

Endpoint: `GET /api/learning/lessons/:lessonId`

Description: Retrieves a specific lesson by its ID.

Authentication: Required

Parameters:

`lessonId` (path): Lesson ID to retrieve

Success Response (200 OK):

```
{
 "status": "success",
 "data": {
   "lesson_id": 1,
   "module_id": 1,
   "title": "What is Money?",
   "content": "Lesson content here...",
   "lesson_number": 1,
   "created_at": "2025-06-24T14:30:00.000Z"
 }
}
```

Error Response (404 Not Found):

```
{
 "status": "error",
 "message": "Lesson with ID 1 not found"
}
```

27. Get Module Quizzes

Endpoint: `GET /api/learning/quizzes/:moduleId`

Description: Retrieves all quizzes for a specific learning module.

Authentication: Required

Parameters:

`moduleId` (path): Module ID to fetch quizzes for

Success Response (200 OK):

```
{
 "status": "success",
 "data": [
   {
     "quiz_id": 1,
     "module_id": 1,
     "title": "Finance Basics Quiz",
     "questions": [...],
     "passing_score": 70,
     "created_at": "2025-06-24T14:30:00.000Z"
   }
 ]
}
```

Error Response (404 Not Found):

```
{
 "status": "error",
 "message": "No quizzes found for module ID 1"
}
```

28. Submit Quiz Attempt

Endpoint: `POST /api/learning/quizzes/:moduleId/submit`

Description: Submits a quiz attempt for a specific module and awards points based on performance.

Authentication: Required

Parameters:

`moduleId` (path): Module ID for the quiz

Request Body:

```
{
 "userId": 1,
 "answers": [
   {"question_id": 1, "selected_answer": "A", "correct": true},
   {"question_id": 2, "selected_answer": "B", "correct": false}
 ],
 "attempt_score": 85,
 "passed": true
}
```

Validation Rules:

userId: Required, must be a valid user ID

answers: Required, array of answer objects

attempt_score: Required, number between 0-100

passed: Required, boolean

Success Response (200 OK):

```
{
 "status": "success",
 "message": "Quiz for module 1 submitted successfully",
 "data": {
   "userId": 1,
   "answers": [...],
   "attempt_score": 85,
   "passed": true
 }
}
```

Error Response (400 Bad Request):

```
{
 "status": "error",
 "message": "User ID and answers are required"
}
```

29. Get Learning Summary

Endpoint: `GET /api/learning/summary/:userId`

Description: Retrieves comprehensive learning summary and statistics for a user.

Authentication: Required

Parameters:

`userId` (path): User ID to fetch learning summary for

Success Response (200 OK):

```
{
 "status": "success",
 "data": {
   "modules": 10,
   "points": 150,
   "total_attempts": 8,
   "total_quizzes_left": 5,
   "total_views": 1,
   "score": 642,
   "percent": "50.00"
 }
}
```

Error Response (404 Not Found):

```
{
 "status": "error",
 "message": "No learning summary found for user ID 1"
}
```

30. Get Learning Performance Score

Endpoint: GET /api/learning/score/:userId

Description: Retrieves the calculated learning performance score for a user (scaled 300-850).

Authentication: Required

Parameters:

`userId` (path): User ID to fetch performance score for

Success Response (200 OK):

```
{
 "status": "success",
 "data": 642
}
```

Error Response (404 Not Found):

```
{
 "status": "error",
 "message": "No learning performance found for user ID 1"
}
```

Account Management Endpoints

31. Create Account

Endpoint: `POST /api/accounts`
Description: Creates a new financial account for a user.
Authentication: Required

Request Body:

```
{
  "user_id": 1,
  "bank_name": "First National Bank",
  "account_name": "Main Checking",
  "account_type": "checking",
  "currency": "USD",
  "account_balance": 1500.00
}
```

Validation Rules:

- user_id: Required, must be a valid user ID
- bank_name: Required, string
- account_name: Required, string
- account_type: Required, string
- currency: Required, string
- account_balance: Optional, defaults to 0 if not provided

Success Response (201 Created):

```
{
  "status": "success",
  "message": "Account created successfully",
  "data": {
    "account_id": 1
  }
}
```

Error Response (400 Bad Request):

```
{
  "status": "error",
  "message": "Missing required fields: user_id, account_name, account_type"
}
```

32. Get User Accounts

Endpoint: `GET /api/accounts/user/:user_id`
Description: Retrieves all accounts belonging to a specific user.
Authentication: Required

Parameters:

- `user_id` (path): User ID to fetch accounts for

Success Response (200 OK):

```
{
  "status": "success",
  "data": [
    {
      "account_id": 1,
      "user_id": 1,
      "bank_name": "First National Bank",
      "account_name": "Main Checking",
      "account_type": "checking",
      "currency": "USD",
      "account_balance": 1500.00,
      "created_at": "2025-06-24T14:30:00.000Z",
      "updated_at": "2025-06-24T14:30:00.000Z"
    }
  ]
}
```

Error Response (400 Bad Request):

```
{
  "status": "error",
  "message": "Invalid user ID"
}
```

33. Get Account by ID

Endpoint: `GET /api/accounts/:account_id`
Description: Retrieves a specific account by its ID.
Authentication: Required

Parameters:

- account_id (path): Account ID to retrieve

Success Response (200 OK):

```
{
  "status": "success",
  "data": {
    "account_id": 1,
    "user_id": 1,
    "bank_name": "First National Bank",
    "account_name": "Main Checking",
    "account_type": "checking",
    "currency": "USD",
    "account_balance": 1500.00,
    "created_at": "2025-06-24T14:30:00.000Z",
    "updated_at": "2025-06-24T14:30:00.000Z"
  }
}
```

Error Response (404 Not Found):

```
{
  "status": "error",
  "message": "Account not found"
}
```

34. Update Account Name

Endpoint: `PUT /api/accounts/:account_id`
Description: Updates an account's name (only account name can be changed).
Authentication: Required

Parameters:

- account_id (path): Account ID to update

Request Body:

```
{
  "account_name": "Updated Account Name"
}
```

Success Response (200 OK):

```
{
  "status": "success",
  "message": "Account name updated successfully"
}
```

Error Response (400 Bad Request):

```
{
  "status": "error",
  "message": "account_name is required and must be a string"
}
```

35. Delete Account

Endpoint: `DELETE /api/accounts/:account_id`
Description: Permanently deletes an account by its ID and user ID.
Authentication: Required

Parameters:

- account_id (path): Account ID to delete

Request Body:

```
{
  "user_id": 1
}
```

Validation Rules:

- user_id: Required in request body for authorization

Success Response (200 OK):

```
{
  "status": "success",
  "message": "Account deleted successfully"
}
```

Error Response (400 Bad Request):

```
{
  "status": "error",
  "message": "Invalid or missing account_id or user_id"
}
```

Transaction Management Endpoints

36. Create Transaction

Endpoint: `POST /api/transactions`
Description: Creates a new financial transaction and updates account balance.
Authentication: Required

Request Body:

```
{
  "account_id": 1,
  "category_id": 5,
  "custom_category_id": null,
  "transaction_amount": 250.00,
  "transaction_type": "expense",
  "transaction_name": "Grocery Shopping",
  "transaction_date": "2025-06-24",
  "is_recurring": false,
  "linked_goal_id": 2,
  "linked_challenge_id": null,
  "budget_id": 1,
  "points_awarded": 0
}
```

Validation Rules:

- account_id: Required, must be a valid account ID
- transaction_amount: Required, positive number
- transaction_type: Required, must be one of: "expense", "income", "transfer", "fee", "withdrawal", "deposit"
- transaction_name: Required, string
- transaction_date: Required, ISO date format (YYYY-MM-DD)
- category_id OR custom_category_id: One must be provided
- is_recurring: Optional, boolean (defaults to false)

Success Response (201 Created):

```
{
  "status": "success",
  "message": "Transaction created successfully",
  "data": {
    "transaction_id": 1,
    "updated_balance": 1250.00
  }
}
```

Error Response (400 Bad Request):

```
{
  "status": "error",
  "message": "Insufficient funds for this transaction."
}
```

37. Get User Transactions

Endpoint: `GET /api/transactions/user/:user_id`
Description: Retrieves all transactions for a specific user across all accounts.
Authentication: Required

Parameters:

- user_id (path): User ID to fetch transactions for

Success Response (200 OK):

```
{
  "status": "success",
  "data": [
    {
      "transaction_id": 1,
      "transaction_amount": 250.00,
      "transaction_type": "expense",
      "account_id": 1,
      "account_name": "Main Checking",
      "transaction_name": "Grocery Shopping",
      "transaction_date": "2025-06-24",
      "category_name": "Food & Dining"
    }
  ]
}
```

38. Get Transaction by ID

Endpoint: `GET /api/transactions/:transaction_id`
Description: Retrieves a specific transaction by its ID.
Authentication: Required

Parameters:

- transaction_id (path): Transaction ID to retrieve

Success Response (200 OK):

```
{
  "status": "success",
  "data": {
    "transaction_id": 1,
    "account_id": 1,
    "category_id": 5,
    "custom_category_id": null,
    "transaction_amount": 250.00,
    "transaction_type": "expense",
    "transaction_name": "Grocery Shopping",
    "transaction_date": "2025-06-24",
    "is_recurring": false,
    "linked_goal_id": 2,
    "linked_challenge_id": null,
    "budget_id": 1,
    "points_awarded": 0,
    "created_at": "2025-06-24T14:30:00.000Z"
  }
}
```

39. Get Account Transactions

Endpoint: `GET /api/transactions/account/:account_id`
Description: Retrieves all transactions for a specific account.
Authentication: Required

Parameters:

- account_id (path): Account ID to fetch transactions for

Success Response (200 OK):

```
{
  "status": "success",
  "data": [
    {
      "transaction_id": 1,
      "account_id": 1,
      "transaction_amount": 250.00,
      "transaction_type": "expense",
      "transaction_name": "Grocery Shopping",
      "transaction_date": "2025-06-24",
      "category_id": 5
    }
  ]
}
```

40. Get Transactions by Category

Endpoint: `GET /api/transactions/category/:category_id`
Description: Retrieves all transactions for a specific category.
Authentication: Required

Parameters:

- category_id (path): Category ID to fetch transactions for

Success Response (200 OK):

```
{
  "status": "success",
  "data": [
    {
      "transaction_id": 1,
      "account_id": 1,
      "transaction_amount": 250.00,
      "transaction_type": "expense",
      "transaction_name": "Grocery Shopping",
      "transaction_date": "2025-06-24",
      "category_id": 5
    }
  ]
}
```

41. Get Transactions by Category

Endpoint: `GET /api/transactions/category/:category_id`
Description: Retrieves all transactions for a specific category.
Authentication: Required

Parameters:

- category_id (path): Category ID to fetch transactions for

Success Response (200 OK):

```
{
  "status": "success",
  "data": [
    {
      "transaction_id": 1,
      "account_id": 1,
      "transaction_amount": 250.00,
      "transaction_type": "expense",
      "transaction_name": "Grocery Shopping",
      "transaction_date": "2025-06-24",
      "category_id": 5
    }
  ]
}
```

42. Update Transaction Details

Endpoint: `PUT /api/transactions/:transaction_id`
Description: Updates transaction details (name, date, amount, category).
Authentication: Required

Parameters:

- transaction_id (path): Transaction ID to update

Request Body:

```
{
  "transaction_name": "Updated Transaction Name",
  "transaction_date": "2025-06-25",
  "transaction_amount": 300.00,
  "category_id": 6
}
```

Success Response (200 OK):

```
{
  "status": "success",
  "message": "Transaction updated successfully",
  "data": {
    "transaction_id": 1,
    "transaction_name": "Updated Transaction Name",
    "transaction_date": "2025-06-25",
    "transaction_amount": 300.00,
    "category_id": 6
  }
}
```

43. Delete Transaction

Endpoint: `DELETE /api/transactions/:transaction_id`
Description: Permanently deletes a transaction and restores account balance.
Authentication: Required

Parameters:

- transaction_id (path): Transaction ID to delete

Success Response (200 OK):

```
{
  "status": "success",
  "message": "Transaction deleted successfully"
}
```

44. Get Categories

Endpoint: `GET /api/categories`
Description: Retrieves all available transaction categories.
Authentication: Required

Success Response (200 OK):

```
{
  "status": "success",
  "data": [
    {
      "category_id": 1,
      "category_name": "Food & Dining"
    },
    {
      "category_id": 2,
      "category_name": "Transportation"
    },
    {
      "category_id": 3,
      "category_name": "Shopping"
    }
  ]
}
```

45. Get Account Summary

Endpoint: `GET /api/transactions/user/:user_id/summary`
Description: Returns total spent by category for a specific user

Parameters:

- account_id (path): Account ID to fetch transactions for

Success Response (200 OK):

```
{
   "status": "success",
   "data": [
      {
         "category": "transport",
         "total_spent": "8251.40"
      },
      {
         "category": "groceries",
         "total_spent": "1000.57"
      },
   ]
}
```

46. Get Category Name by ID

Endpoint: `GET /api/categories/:category_id`
Description: Retrieves the name of a specific category by its ID.
Authentication: Required

Parameters:

- category_id (path): Category ID to retrieve name for

Success Response (200 OK):

```
{
  "status": "success",
  "data": {
    "category_name": "Food & Dining"
  }
}
```

Error Response (404 Not Found):

```
{
  "status": "error",
  "message": "Category not found"
}
```

Budget Management Endpoints

47. Create Budget

Endpoint: `POST /api/budgets`
Description: Creates a new budget with category allocations for a user.
Authentication: Required

Request Body:

```
{
  "user_id": 1,
  "budget_name": "Monthly Budget",
  "period_start": "2025-06-01",
  "period_end": "2025-06-30",
  "allocations": [
    {
      "category_id": 1,
      "target_amount": 500.00
    },
    {
      "category_id": 2,
      "target_amount": 300.00
    }
  ]
}
```

Success Response (201 Created):

```
{
  "status": "success",
  "message": "Budget created successfully",
  "data": {
    "budget_id": 1
  }
}
```

48. Get User Budgets

Endpoint: `GET /api/budgets/user/:user_id`
Description: Retrieves all budgets for a specific user.
Authentication: Required

Parameters:

- user_id (path): User ID to fetch budgets for

Success Response (200 OK):

```
{
  "status": "success",
  "data": [
    {
      "budget_id": 1,
      "budget_name": "Monthly Budget"
    },
    {
      "budget_id": 2,
      "budget_name": "Food & Dining"
    }
  ]
}
```

49. Update Budget Name

Endpoint: `PUT /api/budgets/:budget_id/name`
Description: Updates a budget's name.
Authentication: Required

Parameters:

- budget_id (path): Budget ID to update

Request Body:

```
{
  "budget_name": "Updated Budget Name",
  "user_id": 1
}
```

Success Response (200 OK):

```
{
  "status": "success",
  "message": "Budget name updated successfully"
}
```

50. Update Budget Details

Endpoint: `PUT /api/budgets/:budget_id`
Description: Updates budget details (name, period dates).
Authentication: Required

Parameters:

- budget_id (path): Budget ID to update

Request Body:

```
{
  "budget_name": "Updated Budget",
  "period_start": "2025-07-01",
  "period_end": "2025-07-31"
}
```

Success Response (200 OK):

```
{
  "status": "success",
  "message": "Budget updated successfully"
}
```

},

51. Delete Budget

Endpoint: `DELETE /api/budgets/:budget_id`
Description: Permanently deletes a budget and all associated category allocations.
Authentication: Required

Parameters:

- budget_id (path): Budget ID to delete

Request Body:

```
{
  "user_id": 1
}
```

Success Response (200 OK):

```
{
  "status": "success",
  "message": "Budget deleted successfully"
}
```

52. Get Budget Categories

Endpoint: `GET /api/budget/categories`
Description: Retrieves all categories for budgets.
Authentication: Required

Success Response (200 OK):

```
{
  "status": "success",
  "data": [
          "category_id": 27,
          "category_name": "accommodation"
      },
      {
          "category_id": 30,
          "category_name": "bonus"
```

```
        },
        {
            "category_id": 37,
            "category_name": "business expense"
        }
            …
  ]
}
```

Profile Endpoints

53. Get Current Goals

Endpoint: `GET /api/auth/profile/current-goals/:userId`
Description: Returns current in-progress goals for the user with progress tracking.
Authentication: Required
Parameters: `userId` (number): The ID of the user to get goals for

Response Body
```
{
  "status": "success",
  "message": "Current goals fetched successfully.",
  "data": [
    {
      "goal_id": 1,
      "goal_name": "Emergency Fund",
      "goal_type": "savings",
      "current_amount": 2500.00,
      "target_amount": 5000.00,
      "progress_percentage": 50,
      "xp_reward": 250
    }
  ]
}
```

54. Get Performance Statistics

Endpoint: `GET /api/auth/profile/performance-stats/:userId`

Description: Returns performance stats including accuracy, leaderboard rank, challenges, and goals completion.
Authentication: Required
Parameters: `userId` (number): The ID of the user to get performance stats for

Request Body
```
{
  "status": "success",
  "message": "Performance stats fetched successfully.",
  "data": {
    "accuracy": 85,
    "leaderboard_rank": 15,
    "challenges_joined": 8,
    "goals_completed": 3,
    "goals_total": 5
  }
}
```

55. Get Recent Achievements

Endpoint: `GET /api/auth/profile/recent-achievements/:userId`
Description:Returns the top 3 most recent achievements earned by the user.
Authentication: Required
Parameters: `userId` (number): The ID of the user to get achievements for

Request Body
```
{
  "status": "success",
  "message": "Recent achievements fetched successfully.",
  "data": [
    {
      "achievement_id": 1,
      "achievement_title": "First Goal Completed",
      "xp_reward": 100,
      "icon_image_path": "/badges/first_goal.png",
      "awarded_at": "2025-06-20T14:30:00.000Z"
    }
  ]
}
```

56. Get User Communities Endpoint

Endpoint: `GET /api/auth/profile/communities/:userId`
Description:Returns all communities the user is a member of with stats and preview data.
Authentication: Required

Parameters: `userId` (number): The ID of the user to get communities for

Request Body
```
{
  "status": "success",
  "message": "User communities fetched successfully.",
  "data": [
    {
      "community_id": 1,
      "community_name": "Savings Champions",
      "banner": "/banners/community_1.jpg",
      "xp_total": 1250,
      "member_count": 45,
      "challenge_count": 12,
      "preview_avatars": ["/avatars/avatar_1.jpg", "/avatars/avatar_2.jpg"]
    }
  ]
}
```

57. Get Performance Summary

Endpoint: `GET /api/auth/profile/performance-summary/:userId`
Description:Returns performance score, label, avatar, level, and tier information.
Authentication: Required
Parameters: `userId` (number): The ID of the user to get performance summary for

Request Body
```
{
  "status": "success",
  "message": "Performance summary retrieved successfully.",
  "data": {
    "avatar_image_path": "/avatars/avatar_5.jpg",
    "tier_level": "Silver",
    "level_number": 3,
    "performance_score": 750,
    "performance_label": "Good"
  }
}
```

58. Get Level Progress

Endpoint: `GET /api/auth/profile/level-progress/:userId`
Description:Returns performance score, label, avatar, level, and tier information.
Authentication: Required
Parameters: `userId` (number): The ID of the user to get level progress for

Request Body

```
{
  "status": "success",
  "message": "Level progress retrieved successfully.",
  "data": {
    "level_number": 3,
    "tier_status": "Silver",
    "next_level": 4,
    "current_tier_xp": 500,
    "tier_xp_required": 3000,
    "points_to_next_tier": 2500
  }
}
```

Error Responses

Common Error Status Codes

- `400 Bad Request`: Invalid request data or validation errors
- `401 Unauthorized`: Invalid credentials or missing authentication
- `403 Forbidden`: Insufficient permissions
- `404 Not Found`: Resource not found
- `409 Conflict`: Resource already exists (duplicate email/username)
- `500 Internal Server Error`: Server-side error

Security Considerations

- All passwords are hashed using Argon2id
- PASETO v3.local tokens provide secure, stateless authentication
- Input validation prevents injection attacks
- Rate limiting should be implemented on authentication endpoints
- HTTPS required for all authentication operations
- Sensitive operations require token verification