Software Engineering

(BSCS3001)

Submission by Team 15

Yash Agrawal (22f3003228)

Vidhan Mertiya (22f1001289)

Neelotpal Dutta (21f3002397)

Aviral Kaintura (21f2000172)

Mihir R Shreshti (22f2001525)

Samadhan Kashinath Patil (21f1006852)

IITM BS Degree Program

Indian Institute of Technology, Madras, Chennai

Tamil Nadu, India – 600036

Contents

Milestone 5

1 Introduction

This document outlines comprehensive test cases for the CoinCraft FastAPI backend application. The application supports multiple user roles (Parent, Teacher, Younger Child, Older Child) with role-specific functionalities including financial management, learning modules, task management, and redemption systems.

2 Test Environment Setup

2.1 Authentication Fixtures

These fixtures establish test clients and handle user authentication for different roles.

2.1.1 Parent Setup Fixture

Fixture Name: parent_setup_data
Scope: session
Description: Logs in as a Parent user and retrieves authentication token.

Test Case:

1. **Login as Parent User**
2. **Passed Inputs:**

```
{
  "email": "parent@test.com",
  "password": "Parent@123"
}
```

3. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
{
  "access_token": "valid_jwt_token",
  "token_type": "bearer",
  "user": {
    "id": "uuid",
    "email": "parent@test.com",
    "name": "Test Parent",
    "role": "parent",
    "avatar_url": null,
    "created_at": "2025-01-01T00:00:00Z",
    "updated_at": "2025-01-01T00:00:00Z",
    "is_active": true,
    "is_superuser": false,
    "is_verified": false
  }
```

```
}
```

4. **Result:** Passed

Pytest Code:

```
@pytest.fixture(scope='session')
def parent_setup_data(client):
    response = client.post('/api/auth/register', json={
        'email': 'parent@test.com',
        'password': 'Parent@123',
        'name': 'Test Parent',
        'role': 'parent'
    })
    assert response.status_code == 200
    data = response.json()
    token = data['access_token']
    yield token, client
```

2.1.2 Teacher Setup Fixture

Fixture Name: teacher_setup_data
Scope: session
Description: Logs in as a Teacher user and retrieves authentication token.

Test Case:

1. **Login as Teacher User**

2. **Passed Inputs:**

```
{
  "email": "teacher@test.com",
  "password": "Teacher@123"
}
```

3. **Expected Output:**

   - HTTP Status Code: 200
   - JSON Response:

```
{
  "access_token": "valid_jwt_token",
  "token_type": "bearer",
  "user": {
    "id": "uuid",
    "email": "teacher@test.com",
    "name": "Test Teacher",
    "role": "teacher",
    "avatar_url": null,
    "created_at": "2025-01-01T00:00:00Z",
    "updated_at": "2025-01-01T00:00:00Z",
    "is_active": true,
```

```
    "is_superuser": false,
    "is_verified": false
  }
}
```

4. **Result:** Passed

2.1.3 Child Setup Fixture

Fixture Name: child_setup_data
Scope: session
Description: Logs in as a Younger Child user and retrieves authentication token.

Test Case:

1. **Login as Child User**
2. **Passed Inputs:**

```
{
  "email": "child@test.com",
  "password": "Child@123"
}
```

3. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
{
  "access_token": "valid_jwt_token",
  "token_type": "bearer",
  "user": {
    "id": "uuid",
    "email": "child@test.com",
    "name": "Test Child",
    "role": "younger_child",
    "avatar_url": null,
    "created_at": "2025-01-01T00:00:00Z",
    "updated_at": "2025-01-01T00:00:00Z",
    "is_active": true,
    "is_superuser": false,
    "is_verified": false
  }
}
```

4. **Result:** Passed

3 Authentication Endpoints

3.1 User Registration

Endpoint: POST /api/auth/register
Description: Register a new user and immediately log them in.
Test Cases:

3.1.1 Valid Parent Registration

Test Case:

1. **Valid Parent Registration**
2. **Passed Inputs:**

```
{
  "email": "newparent@test.com",
  "password": "NewParent@123",
  "name": "New Parent",
  "role": "parent"
}
```

3. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
{
  "access_token": "valid_jwt_token",
  "token_type": "bearer",
  "user": {
    "id": "uuid",
    "email": "newparent@test.com",
    "name": "New Parent",
    "role": "parent",
    "avatar_url": null,
    "created_at": "2025-01-01T00:00:00Z",
    "updated_at": "2025-01-01T00:00:00Z",
    "is_active": true,
    "is_superuser": false,
    "is_verified": false
  }
}
```

4. **Result:** Passed

Pytest Code:

```
def test_valid_parent_registration(client):
    """Test valid parent registration."""
    response = client.post('/api/auth/register', json={
        'email': 'newparent@test.com',
```

```
        'password': 'NewParent@123',
        'name': 'New Parent',
        'role': 'parent'
    })

    assert response.status_code == 200
    data = response.json()
    assert 'access_token' in data
    assert data['token_type'] == 'bearer'
    assert data['user']['email'] == 'newparent@test.com'
    assert data['user']['role'] == 'parent'
    assert data['user']['is_active'] == True
```

3.1.2 Invalid Registration - Duplicate Email

Test Case:

1. **Duplicate Email Registration**
2. **Passed Inputs:**

```
{
  "email": "existing@test.com",
  "password": "Test@123",
  "name": "Test User",
  "role": "parent"
}
```

3. **Expected Output:**
   - HTTP Status Code: 400
   - JSON Response:

```
{
  "detail": "User with this email already exists"
}
```

4. **Result:** Passed

Pytest Code:

```
def test_duplicate_email_registration(client):
    """Test registration with duplicate email."""
    # First registration
    client.post('/api/auth/register', json={
        'email': 'existing@test.com',
        'password': 'Test@123',
        'name': 'Test User',
        'role': 'parent'
    })

    # Second registration with same email
```

```
    response = client.post('/api/auth/register', json={
        'email': 'existing@test.com',
        'password': 'Test@123',
        'name': 'Test User',
        'role': 'parent'
    })

    assert response.status_code == 400
    data = response.json()
    assert data['detail'] == 'User with this email already exists'
```

3.2 User Login

Endpoint: POST /api/auth/jwt/login
Description: Login with email and password using OAuth2 form data.

Test Cases:

3.2.1 Valid Login

Test Case:

1. **Valid User Login**
2. **Passed Inputs:**

```
{
  "username": "parent@test.com",
  "password": "Parent@123"
}
```

3. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
{
  "access_token": "valid_jwt_token",
  "token_type": "bearer"
}
```

4. **Result:** Passed

Pytest Code:

```
def test_valid_login(client):
    """Test valid user login."""
    # First register a user
    client.post('/api/auth/register', json={
        'email': 'parent@test.com',
        'password': 'Parent@123',
        'name': 'Test Parent',
```

```
        'role': 'parent'
    })

    # Login with the registered user
    response = client.post('/api/auth/jwt/login', data={
        'username': 'parent@test.com',
        'password': 'Parent@123'
    }, headers={'Content-Type': 'application/x-www-form-urlencoded'
       })

    assert response.status_code == 200
    data = response.json()
    assert 'access_token' in data
    assert data['token_type'] == 'bearer'
```

3.2.2 Invalid Login

Test Case:

1. **Invalid Credentials**
2. **Passed Inputs:**

```
{
  "username": "wrong@test.com",
  "password": "WrongPassword"
}
```

3. **Expected Output:**
   - HTTP Status Code: 401
   - JSON Response:

```
{
  "detail": "LOGIN_BAD_CREDENTIALS"
}
```

4. **Result:** Passed

Pytest Code:

```
def test_invalid_login(client):
    """Test login with invalid credentials."""
    response = client.post('/api/auth/jwt/login', data={
        'username': 'wrong@test.com',
        'password': 'WrongPassword'
    }, headers={'Content-Type': 'application/x-www-form-urlencoded'
       })

    assert response.status_code == 401
    data = response.json()
    assert data['detail'] == 'LOGIN_BAD_CREDENTIALS'
```

4 User Management Endpoints

4.1 Get Current User Profile

Endpoint: GET /api/users/me
Description: Get current user's profile information.
Test Cases:

4.1.1 Authenticated User Profile

Test Case:

1. **Get Current User Profile**
2. **Headers:**

```
{
  "Authorization": "Bearer valid_jwt_token"
}
```

3. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
{
  "id": "uuid",
  "email": "parent@test.com",
  "name": "Test Parent",
  "role": "parent",
  "avatar_url": null,
  "created_at": "2025-01-01T00:00:00Z",
  "updated_at": "2025-01-01T00:00:00Z",
  "is_active": true,
  "is_superuser": false,
  "is_verified": false
}
```

4. **Result:** Passed

Pytest Code:

```
def test_get_current_user_profile(client):
    """Test getting current user profile."""
    # Register and login to get token
    register_response = client.post('/api/auth/register', json={
        'email': 'parent@test.com',
        'password': 'Parent@123',
        'name': 'Test␣Parent',
        'role': 'parent'
    })
    token = register_response.json()['access_token']

```

```python
    # Get current user profile
    response = client.get('/api/users/me', headers={
        'Authorization': f'Bearer {token}'
    })

    assert response.status_code == 200
    data = response.json()
    assert data['email'] == 'parent@test.com'
    assert data['name'] == 'Test Parent'
    assert data['role'] == 'parent'
    assert data['is_active'] == True
```

4.2 Update User Profile

Endpoint: PUT /api/users/user_id
Description: Update user profile information.
Test Cases:

4.2.1 Valid Profile Update

Test Case:

1. **Update User Profile**
2. **Passed Inputs:**

```json
{
  "name": "Updated Parent Name",
  "avatar_url": "https://example.com/avatar.jpg"
}
```

3. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```json
{
  "id": "uuid",
  "email": "parent@test.com",
  "name": "Updated Parent Name",
  "role": "parent",
  "avatar_url": "https://example.com/avatar.jpg",
  "created_at": "2025-01-01T00:00:00Z",
  "updated_at": "2025-01-01T00:00:00Z",
  "is_active": true,
  "is_superuser": false,
  "is_verified": false
}
```

4. **Result:** Passed

Pytest Code:

```python
def test_update_user_profile(client):
    """Test updating user profile."""
    # Register and login to get token
    register_response = client.post('/api/auth/register', json={
        'email': 'parent@test.com',
        'password': 'Parent@123',
        'name': 'Test Parent',
        'role': 'parent'
    })
    token = register_response.json()['access_token']
    user_id = register_response.json()['user']['id']

    # Update user profile
    response = client.put(f'/api/users/{user_id}', json={
        'name': 'Updated Parent Name',
        'avatar_url': 'https://example.com/avatar.jpg'
    }, headers={'Authorization': f'Bearer {token}'})

    assert response.status_code == 200
    data = response.json()
    assert data['name'] == 'Updated Parent Name'
    assert data['avatar_url'] == 'https://example.com/avatar.jpg'
```

5 Goals Management Endpoints

5.1 Get User Goals

Endpoint: GET /api/users/user_id/goals
Description: Get all goals for a user.
Test Cases:

5.1.1 Get Child Goals

Test Case:

1. **Get Child Goals**
2. **Headers:**

```
{
  "Authorization": "Bearer parent_token"
}
```

3. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
[
  {
    "id": "goal_uuid",
    "user_id": "child_uuid",
    "title": "Save for Toy",
    "description": "Save coins for a new toy",
    "target_amount": 100,
    "current_amount": 25,
    "icon": "toy",
    "color": "blue",
    "deadline": "2025-02-01T00:00:00Z",
    "is_completed": false,
    "created_at": "2025-01-01T00:00:00Z",
    "updated_at": "2025-01-01T00:00:00Z"
  }
]
```

4. **Result:** Passed

Pytest Code:

```
def test_get_child_goals(client):
    """Test getting child goals."""
    # Register parent and child
    parent_response = client.post('/api/auth/register', json={
        'email': 'parent@test.com',
        'password': 'Parent@123',
        'name': 'Test␣Parent',
        'role': 'parent'
    })
    parent_token = parent_response.json()['access_token']

    child_response = client.post('/api/auth/register', json={
        'email': 'child@test.com',
        'password': 'Child@123',
        'name': 'Test␣Child',
        'role': 'younger_child'
    })
    child_token = child_response.json()['access_token']
    child_id = child_response.json()['user']['id']

    # Create a goal for child
    client.post(f'/api/users/{child_id}/goals', json={
        'title': 'Save␣for␣Toy',
        'description': 'Save␣coins␣for␣a␣new␣toy',
        'target_amount': 100,
        'icon': 'toy',
        'color': 'blue',
        'deadline': '2025-02-01T00:00:00Z'
    }, headers={'Authorization': f'Bearer␣{child_token}'})

    # Get child goals as parent
    response = client.get(f'/api/users/{child_id}/goals', headers={
```

```
        'Authorization': f'Bearer {parent_token}'
    })

    assert response.status_code == 200
    data = response.json()
    assert len(data) > 0
    assert data[0]['title'] == 'Save for Toy'
    assert data[0]['target_amount'] == 100
```

5.2 Create Goal

Endpoint: POST /api/users/user_id/goals
Description: Create a new financial goal.
Test Cases:

5.2.1 Valid Goal Creation

Test Case:

1. **Create New Goal**
2. **Passed Inputs:**

```
{
  "title": "Save for Game",
  "description": "Save coins for a new video game",
  "target_amount": 200,
  "icon": "game",
  "color": "green",
  "deadline": "2025-03-01T00:00:00Z"
}
```

3. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
{
  "id": "goal_uuid",
  "user_id": "child_uuid",
  "title": "Save for Game",
  "description": "Save coins for a new video game",
  "target_amount": 200,
  "current_amount": 0,
  "icon": "game",
  "color": "green",
  "deadline": "2025-03-01T00:00:00Z",
  "is_completed": false,
  "created_at": "2025-01-01T00:00:00Z",
  "updated_at": "2025-01-01T00:00:00Z"
}
```

4. **Result:** Passed

Pytest Code:

```python
def test_create_goal(client):
    """Test creating a new goal."""
    # Register child user
    child_response = client.post('/api/auth/register', json={
        'email': 'child@test.com',
        'password': 'Child@123',
        'name': 'Test Child',
        'role': 'younger_child'
    })
    child_token = child_response.json()['access_token']
    child_id = child_response.json()['user']['id']

    # Create goal
    response = client.post(f'/api/users/{child_id}/goals', json={
        'title': 'Save for Game',
        'description': 'Save coins for a new video game',
        'target_amount': 200,
        'icon': 'game',
        'color': 'green',
        'deadline': '2025-03-01T00:00:00Z'
    }, headers={'Authorization': f'Bearer {child_token}'})

    assert response.status_code == 200
    data = response.json()
    assert data['title'] == 'Save for Game'
    assert data['target_amount'] == 200
    assert data['current_amount'] == 0
    assert data['is_completed'] == False
```

5.3 Contribute to Goal

Endpoint: POST /api/users/user_id/goals/goal_id/contribute
Description: Add coins to a financial goal.

Test Cases:

5.3.1 Valid Goal Contribution

Test Case:

1. **Contribute to Goal**
2. **Passed Inputs:**

```json
{
  "amount": 10
}
```

3. **Expected Output:**

- HTTP Status Code: 200
- JSON Response:


```
{
  "goal": {
    "id": "goal_uuid",
    "user_id": "child_uuid",
    "title": "Save for Toy",
    "description": "Save coins for a new toy",
    "target_amount": 100,
    "current_amount": 35,
    "icon": "toy",
    "color": "blue",
    "deadline": "2025-02-01T00:00:00Z",
    "is_completed": false,
    "created_at": "2025-01-01T00:00:00Z",
    "updated_at": "2025-01-01T00:00:00Z"
  },
  "transaction": {
    "id": "transaction_uuid",
    "user_id": "child_uuid",
    "type": "save",
    "amount": 10,
    "description": "Contributed to goal: Save for Toy",
    "category": "goal",
    "reference_id": "goal_uuid",
    "reference_type": "goal",
    "created_at": "2025-01-01T00:00:00Z"
  },
  "new_coin_balance": 15
}
```


4. **Result:** Passed

Pytest Code:

```
def test_contribute_to_goal(client):
    """Test contributing to a goal."""
    # Register child user
    child_response = client.post('/api/auth/register', json={
        'email': 'child@test.com',
        'password': 'Child@123',
        'name': 'Test Child',
        'role': 'younger_child'
    })
    child_token = child_response.json()['access_token']
    child_id = child_response.json()['user']['id']

    # Create goal first
    goal_response = client.post(f'/api/users/{child_id}/goals', json
        ={
        'title': 'Save for Toy',
        'description': 'Save coins for a new toy',
        'target_amount': 100,
        'icon': 'toy',
```

```
        'color': 'blue',
        'deadline': '2025-02-01T00:00:00Z'
    }, headers={'Authorization': f'Bearer {child_token}'})
    goal_id = goal_response.json()['id']

    # Contribute to goal
    response = client.post(f'/api/users/{child_id}/goals/{goal_id}/
        contribute', json={
        'amount': 10
    }, headers={'Authorization': f'Bearer {child_token}'})

    assert response.status_code == 200
    data = response.json()
    assert 'goal' in data
    assert 'transaction' in data
    assert 'new_coin_balance' in data
    assert data['goal']['current_amount'] == 10
```

6 Dashboard Endpoints

6.1 Get Child Dashboard

Endpoint: GET /api/child/dashboard
Description: Get dashboard data for child users.
Test Cases:

6.1.1 Child Dashboard Data

Test Case:

1. **Get Child Dashboard**
2. **Headers:**

```
{
  "Authorization": "Bearer child_token"
}
```

3. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
{
  "user": {
    "id": "child_uuid",
    "email": "child@test.com",
    "name": "Test Child",
    "role": "younger_child",
    "avatar_url": null,
    "created_at": "2025-01-01T00:00:00Z",
```

```
    "updated_at": "2025-01-01T00:00:00Z",
    "is_active": true,
    "is_superuser": false,
    "is_verified": false
  },
  "stats": {
    "total_coins": 25,
    "level": 1,
    "streak_days": 3,
    "goals_count": 2,
    "completed_tasks": 5
  },
  "active_goals": [
    {
      "id": "goal_uuid",
      "user_id": "child_uuid",
      "title": "Save for Toy",
      "description": "Save coins for a new toy",
      "target_amount": 100,
      "current_amount": 25,
      "icon": "toy",
      "color": "blue",
      "deadline": "2025-02-01T00:00:00Z",
      "is_completed": false,
      "created_at": "2025-01-01T00:00:00Z",
      "updated_at": "2025-01-01T00:00:00Z"
    }
  ],
  "pending_tasks": [
    {
      "id": "task_uuid",
      "title": "Clean Room",
      "description": "Clean your room",
      "coins_reward": 10,
      "assigned_by": "parent_uuid",
      "assigned_to": "child_uuid",
      "status": "pending",
      "due_date": "2025-01-02T00:00:00Z",
      "requires_approval": true,
      "created_at": "2025-01-01T00:00:00Z"
    }
  ],
  "recent_transactions": [
    {
      "id": "transaction_uuid",
      "user_id": "child_uuid",
      "type": "earn",
      "amount": 10,
      "description": "Completed task: Clean Room",
      "category": "task",
      "reference_id": "task_uuid",
      "reference_type": "task",
      "created_at": "2025-01-01T00:00:00Z"
    }
  ],
  "achievements": [
    {
      "id": "achievement_uuid",
```

```
        "title": "First Goal",
        "description": "Created your first goal",
        "icon": "star",
        "rarity": "common",
        "points_reward": 5,
        "earned_at": "2025-01-01T00:00:00Z"
      }
    ]
}
```

4. **Result:** Passed

Pytest Code:

```python
def test_get_child_dashboard(client):
    """Test getting child dashboard data."""
    # Register child user
    child_response = client.post('/api/auth/register', json={
        'email': 'child@test.com',
        'password': 'Child@123',
        'name': 'Test Child',
        'role': 'younger_child'
    })
    child_token = child_response.json()['access_token']

    # Get child dashboard
    response = client.get('/api/child/dashboard', headers={
        'Authorization': f'Bearer {child_token}'
    })

    assert response.status_code == 200
    data = response.json()
    assert 'user' in data
    assert 'stats' in data
    assert 'active_goals' in data
    assert 'pending_tasks' in data
    assert 'recent_transactions' in data
    assert 'achievements' in data
    assert data['user']['role'] == 'younger_child'
```

6.2 Get Parent Dashboard

Endpoint: GET /api/parent/dashboard
Description: Get dashboard data for parent users.

Test Cases:

6.2.1 Parent Dashboard Data

Test Case:

1. **Get Parent Dashboard**

2. **Headers:**

```
{
  "Authorization": "Bearer parent_token"
}
```

3. **Expected Output:**

   - HTTP Status Code: 200
   - JSON Response:

```
{
  "user": {
    "id": "parent_uuid",
    "email": "parent@test.com",
    "name": "Test Parent",
    "role": "parent",
    "avatar_url": null,
    "created_at": "2025-01-01T00:00:00Z",
    "updated_at": "2025-01-01T00:00:00Z",
    "is_active": true,
    "is_superuser": false,
    "is_verified": false
  },
  "stats": {
    "total_coins": 150,
    "level": 1,
    "streak_days": 0,
    "goals_count": 2,
    "completed_tasks": 8
  },
  "children": [
    {
      "id": "child_uuid",
      "email": "child@test.com",
      "name": "Test Child",
      "role": "younger_child",
      "avatar_url": null,
      "created_at": "2025-01-01T00:00:00Z",
      "updated_at": "2025-01-01T00:00:00Z",
      "is_active": true,
      "is_superuser": false,
      "is_verified": false
    }
  ],
  "recent_transactions": [
    {
      "id": "transaction_uuid",
      "user_id": "child_uuid",
      "type": "earn",
      "amount": 10,
      "description": "Completed task: Clean Room",
      "category": "task",
      "reference_id": "task_uuid",
      "reference_type": "task",
      "created_at": "2025-01-01T00:00:00Z"
```

```
    }
  ],
  "pending_redemptions": [
    {
      "id": "redemption_uuid",
      "user_id": "child_uuid",
      "coins_amount": 50,
      "cash_amount": 5.0,
      "description": "Convert coins to cash",
      "status": "pending",
      "created_at": "2025-01-01T00:00:00Z"
    }
  ]
}
```

4. **Result:** Passed

Pytest Code:

```python
def test_get_parent_dashboard(client):
    """Test getting parent dashboard data."""
    # Register parent user
    parent_response = client.post('/api/auth/register', json={
        'email': 'parent@test.com',
        'password': 'Parent@123',
        'name': 'Test Parent',
        'role': 'parent'
    })
    parent_token = parent_response.json()['access_token']

    # Get parent dashboard
    response = client.get('/api/parent/dashboard', headers={
        'Authorization': f'Bearer {parent_token}'
    })

    assert response.status_code == 200
    data = response.json()
    assert 'user' in data
    assert 'stats' in data
    assert 'children' in data
    assert 'recent_transactions' in data
    assert 'pending_redemptions' in data
    assert data['user']['role'] == 'parent'
```

7 Task Management Endpoints

7.1 Get User Tasks

Endpoint: GET /api/tasks
Description: Get tasks for current user.

Test Cases:

7.1.1 Get Child Tasks

Test Case:

1. **Get Child Tasks**
2. **Headers:**

```
{
  "Authorization": "Bearer child_token"
}
```

3. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
[
  {
    "id": "task_uuid",
    "title": "Clean Room",
    "description": "Clean your room thoroughly",
    "coins_reward": 10,
    "assigned_by": "parent_uuid",
    "assigned_to": "child_uuid",
    "status": "pending",
    "due_date": "2025-01-02T00:00:00Z",
    "requires_approval": true,
    "created_at": "2025-01-01T00:00:00Z"
  }
]
```

4. **Result:** Passed

Pytest Code:

```
def test_get_child_tasks(client):
    """Test getting child tasks."""
    # Register child user
    child_response = client.post('/api/auth/register', json={
        'email': 'child@test.com',
        'password': 'Child@123',
        'name': 'Test Child',
        'role': 'younger_child'
    })
    child_token = child_response.json()['access_token']

    # Get child tasks
    response = client.get('/api/tasks', headers={
        'Authorization': f'Bearer {child_token}'
    })

    assert response.status_code == 200
    data = response.json()
```

```
    assert isinstance(data, list)
```

7.2 Create Task

Endpoint: POST /api/tasks
Description: Create a new task.
Test Cases:

7.2.1 Valid Task Creation

Test Case:

1. **Create New Task**
2. **Passed Inputs:**

```
{
  "title": "Do Homework",
  "description": "Complete math homework",
  "coins_reward": 15,
  "assigned_to": "child_uuid",
  "due_date": "2025-01-03T00:00:00Z",
  "requires_approval": true
}
```

3. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
{
  "id": "task_uuid",
  "title": "Do Homework",
  "description": "Complete math homework",
  "coins_reward": 15,
  "assigned_by": "parent_uuid",
  "assigned_to": "child_uuid",
  "status": "pending",
  "due_date": "2025-01-03T00:00:00Z",
  "requires_approval": true,
  "created_at": "2025-01-01T00:00:00Z"
}
```

4. **Result:** Passed

Pytest Code:

```
def test_create_task(client):
    """Test creating a new task."""
    # Register parent and child
    parent_response = client.post('/api/auth/register', json={
        'email': 'parent@test.com',
```

```
        'password': 'Parent@123',
        'name': 'Test Parent',
        'role': 'parent'
    })
    parent_token = parent_response.json()['access_token']

    child_response = client.post('/api/auth/register', json={
        'email': 'child@test.com',
        'password': 'Child@123',
        'name': 'Test Child',
        'role': 'younger_child'
    })
    child_id = child_response.json()['user']['id']

    # Create task
    response = client.post('/api/tasks', json={
        'title': 'Do Homework',
        'description': 'Complete math homework',
        'coins_reward': 15,
        'assigned_to': child_id,
        'due_date': '2025-01-03T00:00:00Z',
        'requires_approval': True
    }, headers={'Authorization': f'Bearer {parent_token}'})

    assert response.status_code == 200
    data = response.json()
    assert data['title'] == 'Do Homework'
    assert data['coins_reward'] == 15
    assert data['status'] == 'pending'
```

8 Shop System Endpoints

8.1 Get Shop Items

Endpoint: GET /api/shop/items
Description: Get available shop items.

Test Cases:

8.1.1 Get Shop Items

Test Case:

1. **Get Shop Items**
2. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
[
  {
    "id": "item_uuid",
    "name": "Toy Car",
    "description": "A cool toy car",
    "price": 50,
    "category": "toys",
    "emoji": "[automobile]",
    "available": true
  },
  {
    "id": "item_uuid2",
    "name": "Video Game",
    "description": "Fun video game",
    "price": 200,
    "category": "games",
    "emoji": "[Video-Game]",
    "available": true
  }
]
```


3. **Result:** Passed

Pytest Code:

```
def test_get_shop_items(client):
    """Test getting shop items."""
    response = client.get('/api/shop/items')

    assert response.status_code == 200
    data = response.json()
    assert isinstance(data, list)
    if len(data) > 0:
        assert 'id' in data[0]
        assert 'name' in data[0]
        assert 'price' in data[0]
        assert 'available' in data[0]
```

8.2 Purchase Item

Endpoint: POST /api/shop/purchase
Description: Purchase an item from the shop.

Test Cases:

8.2.1 Valid Purchase

Test Case:

1. **Purchase Item**
2. **Headers:**

```
{
  "Authorization": "Bearer child_token"
}
```

3. **Passed Inputs:**

```
{
  "item_id": "item_uuid",
  "quantity": 1
}
```

4. **Expected Output:**

   - HTTP Status Code: 200
   - JSON Response:

```
{
  "success": true,
  "message": "Item purchased successfully",
  "item": {
    "id": "item_uuid",
    "name": "Toy Car",
    "description": "A cool toy car",
    "price": 50,
    "category": "toys",
    "emoji": "[automobile]",
    "available": true
  },
  "coins_spent": 50,
  "new_balance": 25
}
```

5. **Result:** Passed

Pytest Code:

```
def test_purchase_item(client):
    """Test purchasing an item from shop."""
    # Register child user
    child_response = client.post('/api/auth/register', json={
        'email': 'child@test.com',
        'password': 'Child@123',
        'name': 'Test Child',
        'role': 'younger_child'
    })
    child_token = child_response.json()['access_token']

    # Get shop items first
    shop_response = client.get('/api/shop/items')
    shop_items = shop_response.json()

    if len(shop_items) > 0:
```

```
        item_id = shop_items[0]['id']

        # Purchase item
        response = client.post('/api/shop/purchase', json={
            'item_id': item_id,
            'quantity': 1
        }, headers={'Authorization': f'Bearer {child_token}'})

        assert response.status_code == 200
        data = response.json()
        assert 'success' in data
        assert 'coins_spent' in data
        assert 'new_balance' in data
```

9 Redemption System Endpoints

9.1 Get Redemption Requests

Endpoint: GET /api/users/user_id/conversion-requests
Description: Get user's redemption requests.
Test Cases:

9.1.1 Get Child Redemption Requests

Test Case:

1. **Get Child Redemption Requests**
2. **Headers:**

```
{
  "Authorization": "Bearer child_token"
}
```

3. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
[
  {
    "id": "redemption_uuid",
    "user_id": "child_uuid",
    "coins_amount": 50,
    "cash_amount": 5.0,
    "description": "Convert coins to cash",
    "status": "pending",
    "created_at": "2025-01-01T00:00:00Z"
  }
]
```

4. **Result:** Passed

Pytest Code:

```python
def test_get_child_redemption_requests(client):
    """Test getting child redemption requests."""
    # Register child user
    child_response = client.post('/api/auth/register', json={
        'email': 'child@test.com',
        'password': 'Child@123',
        'name': 'Test Child',
        'role': 'younger_child'
    })
    child_token = child_response.json()['access_token']
    child_id = child_response.json()['user']['id']

    # Get redemption requests
    response = client.get(f'/api/users/{child_id}/conversion-
       requests', headers={
        'Authorization': f'Bearer {child_token}'
    })

    assert response.status_code == 200
    data = response.json()
    assert isinstance(data, list)
```

9.2 Create Redemption Request

Endpoint: POST /api/users/user_id/conversion-requests
Description: Create a new redemption request.

Test Cases:

9.2.1 Valid Redemption Request

Test Case:

1. **Create Redemption Request**
2. **Headers:**

```
{
  "Authorization": "Bearer child_token"
}
```

3. **Passed Inputs:**

```
{
  "coins_amount": 100,
  "description": "Convert coins to cash for allowance"
}
```

4. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
{
  "id": "redemption_uuid",
  "user_id": "child_uuid",
  "coins_amount": 100,
  "cash_amount": 10.0,
  "description": "Convert coins to cash for allowance",
  "status": "pending",
  "created_at": "2025-01-01T00:00:00Z"
}
```

5. **Result:** Passed

Pytest Code:

```
def test_create_redemption_request(client):
    """Test creating a redemption request."""
    # Register child user
    child_response = client.post('/api/auth/register', json={
        'email': 'child@test.com',
        'password': 'Child@123',
        'name': 'Test Child',
        'role': 'younger_child'
    })
    child_token = child_response.json()['access_token']
    child_id = child_response.json()['user']['id']

    # Create redemption request
    response = client.post(f'/api/users/{child_id}/conversion-
       requests', json={
        'coins_amount': 100,
        'description': 'Convert coins to cash for allowance'
    }, headers={'Authorization': f'Bearer {child_token}'})

    assert response.status_code == 200
    data = response.json()
    assert data['coins_amount'] == 100
    assert data['status'] == 'pending'
    assert 'cash_amount' in data
```

10 Health Check Endpoints

10.1 Health Check

Endpoint: GET /health
Description: Perform a health check on the application.

Test Cases:

10.1.1 Health Check

Test Case:

1. **Health Check**
2. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
{
  "status": "OK"
}
```

3. **Result:** Passed

Pytest Code:

```
def test_health_check(client):
    """Test health check endpoint."""
    response = client.get('/health')

    assert response.status_code == 200
    data = response.json()
    assert data['status'] == 'OK'
```

10.2 API Status

Endpoint: GET /api/status
Description: Get backend status and environment configuration.
Test Cases:

10.2.1 API Status

Test Case:

1. **Get API Status**
2. **Expected Output:**
   - HTTP Status Code: 200
   - JSON Response:

```
{
  "status": "OK",
  "environment": "development",
  "database_type": "SQLite",
  "secret_key_configured": true,
  "algorithm": "HS256"
}
```

3. **Result:** Passed

Pytest Code:

```python
def test_api_status(client):
    """Test API status endpoint."""
    response = client.get('/api/status')

    assert response.status_code == 200
    data = response.json()
    assert data['status'] == 'OK'
    assert 'environment' in data
    assert 'database_type' in data
    assert 'secret_key_configured' in data
    assert 'algorithm' in data
```

11 Error Handling Test Cases

11.1 Unauthorized Access

Test Case:

1. **Access Protected Endpoint Without Token**
2. **Headers:** None
3. **Expected Output:**
   - HTTP Status Code: 401
   - JSON Response:

```
{
  "detail": "Not authenticated"
}
```

4. **Result:** Passed

Pytest Code:

```python
def test_unauthorized_access(client):
    """Test accessing protected endpoint without token."""
    response = client.get('/api/users/me')

    assert response.status_code == 401
    data = response.json()
    assert data['detail'] == 'Not␣authenticated'
```

11.2 Invalid Token

Test Case:

1. **Access Protected Endpoint With Invalid Token**
2. **Headers:**

```
{
  "Authorization": "Bearer invalid_token"
}
```

3. **Expected Output:**
   - HTTP Status Code: 401
   - JSON Response:

```
{
  "detail": "Could not validate credentials"
}
```

4. **Result:** Passed

Pytest Code:

```
def test_invalid_token(client):
    """Test accessing protected endpoint with invalid token."""
    response = client.get('/api/users/me', headers={
        'Authorization': 'Bearer invalid_token'
    })

    assert response.status_code == 401
    data = response.json()
    assert 'Could not validate credentials' in data['detail']
```

11.3 Resource Not Found

Test Case:

1. **Access Non-existent Resource**
2. **Headers:**

```
{
  "Authorization": "Bearer valid_token"
}
```

3. **Expected Output:**
   - HTTP Status Code: 404
   - JSON Response:

```
{
  "detail": "Not found"
}
```

4. **Result:** Passed

Pytest Code:

```
def test_resource_not_found(client):
    """Test accessing non-existent resource."""
    # Register user to get valid token
    register_response = client.post('/api/auth/register', json={
        'email': 'test@test.com',
        'password': 'Test@123',
        'name': 'Test User',
        'role': 'parent'
    })
    token = register_response.json()['access_token']

    # Try to access non-existent resource
    response = client.get('/api/non-existent-endpoint', headers={
        'Authorization': f'Bearer {token}'
    })

    assert response.status_code == 404
    data = response.json()
    assert data['detail'] == 'Not found'
```

12 Conclusion

This document provides comprehensive test cases for the CoinCraft FastAPI backend application. The test cases cover authentication, user management, goals, tasks, dashboard functionality, shop system, redemption system, and error handling. Each test case includes the expected inputs, outputs, and HTTP status codes to ensure proper API functionality.

The test cases are designed to validate:

- Proper authentication and authorization
- CRUD operations for all entities
- Role-based access control
- Data validation and error handling
- Integration between different system components