Technical Test

Junior Backend Developer Position

Summit Global Teknologi

Description

Create a RESTful API for a library management system using Express.js with the specified project structure.

Project Structure

```
src/
├── config/
│   └── database.js
├── controllers/
│   ├── bookController.js
│   ├── memberController.js
│   └── borrowingController.js
├── models/
│   ├── book.js
│   ├── member.js
│   └── borrowing.js
├── services/
│   ├── bookService.js
│   ├── memberService.js
│   └── borrowingService.js
├── routes/
│   ├── bookRoutes.js
│   ├── memberRoutes.js
│   └── borrowingRoutes.js
└── app.js
```

Additional directories may be added as needed.

Requirements

1. Server & Database Setup

- Create web server using Express.js
- Implement error handling
- Setup PostgreSQL database with proper indexing
- Document application setup and running instructions in README.md

2. Database Design

Database schema is already on Sample SQL last page, these are 3 tables and their relationships that would be used on code:

Books Table

- id (Primary Key, UUID)
- title (VARCHAR(255), NOT NULL)
- author (VARCHAR(255), NOT NULL)
- published_year (INTEGER, NOT NULL)
- stock (INTEGER, NOT NULL, DEFAULT 0)
- created_at (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
- updated_at (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
- isbn (VARCHAR(13), UNIQUE, NOT NULL)

Members Table

- id (Primary Key, UUID)
- name (VARCHAR(255), NOT NULL)
- email (VARCHAR(255), UNIQUE, NOT NULL)
- phone (VARCHAR(15), NOT NULL)
- address (TEXT, NOT NULL)
- created_at (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
- updated_at (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)

Borrowings Table

- id (Primary Key, UUID)
- book_id (UUID, Foreign Key to Books, NOT NULL)
- member_id (UUID, Foreign Key to Members, NOT NULL)
- borrow_date (DATE, NOT NULL)
- return_date (DATE)
- status (ENUM('BORROWED', 'RETURNED'), DEFAULT 'BORROWED')
- created_at (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
- updated_at (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)

3. API Implementation

Implement these endpoints with appropriate validation:

1. GET /api/books
    - List all books with their current stock
    - Query Parameters:
        - title (string): Filter by title (case-insensitive)
        - author (string): Filter by author (case-insensitive)
        - page (integer, default: 1)
        - limit (integer, default: 10)

    Response Format:

```
{
  "data": [{
    "id": "uuid",
    "title": "string",
    "author": "string",
    "published_year": "integer",
    "stock": "integer",
    "isbn": "string",
    "available": "boolean"
  }],
  "pagination": {
    "total": "integer",
    "page": "integer",
    "limit": "integer",
    "totalPages": "integer"
  }
}
```

2. POST /api/members
    - Register new member

    Request Body:

    ```
    {
      "name": "string",
      "email": "string",
      "phone": "string",
      "address": "string"
    }
    ```

    - Validations:
        - Email must be unique and valid format
        - Phone must be valid format
        - All fields are required

3. POST /api/borrowings
    - Create new book borrowing

    Request Body:

    ```
    {
      "book_id": "uuid",
      "member_id": "uuid"
    }
    ```

    - Business Rules:
        - Check book stock (must be > 0)
        - Update book stock (-1)
        - Member cannot borrow more than 3 books
        - Record borrowing date as current date

4. PUT /api/borrowings/:id/return
    - Process book return
    - Path Parameters:
        - id: Borrowing ID
    - Updates:
        - Change status to 'RETURNED'
        - Update book stock (+1)
        - Set return_date to current date

5. GET /api/members/:id/borrowings
    - Get member's borrowing history
    - Path Parameters:
        - id: Member ID
    - Query Parameters:
        - status: Filter by status (BORROWED/RETURNED)
        - page (integer, default: 1)
        - limit (integer, default: 10)
    - Response includes book details

Business Logic Requirements

All business logic should be implemented in the **service** layer:

1. Borrowing Process:
    - Verify book availability (stock > 0)
    - Check member's current borrowing count (≤ 3)
    - Use database transaction for:
        - Decreasing book stock
        - Creating borrowing record

2. Return Process:
    - Verify borrowing record exists, if not return error
    - Verify book belongs to member
    - Use database transaction for:
        - Increasing book stock
        - Updating borrowing status and return date

Submission Requirements

1. GitHub repository with:
    - Complete source code
    - SQL schema file
    - Sample data SQL file
    - Comprehensive README.md

2. README.md must include:
    - Project setup instructions
    - API documentation

Sample Data SQL

```
-- Create Books Table
CREATE TABLE books (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255) NOT NULL,
    published_year INTEGER NOT NULL,
    stock INTEGER NOT NULL DEFAULT 0,
    isbn VARCHAR(13) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Create Members Table
CREATE TABLE members (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    phone VARCHAR(15) NOT NULL,
    address TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Create Borrowings Table
CREATE TABLE borrowings (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    book_id UUID REFERENCES books(id),
    member_id UUID REFERENCES members(id),
    borrow_date DATE NOT NULL,
    return_date DATE,
    status VARCHAR(10) NOT NULL DEFAULT 'BORROWED',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
-- Insert Sample Data for Books
INSERT INTO books (title, author, published_year, stock, isbn) VALUES
('The Great Gatsby', 'F. Scott Fitzgerald', 1925, 5, '9780743273565'),
('To Kill a Mockingbird', 'Harper Lee', 1960, 3, '9780446310789'),
('1984', 'George Orwell', 1949, 4, '9780451524935'),
('Pride and Prejudice', 'Jane Austen', 1813, 6, '9780141439518'),
('The Catcher in the Rye', 'J.D. Salinger', 1951, 3, '9780316769488'),
('The Hobbit', 'J.R.R. Tolkien', 1937, 7, '9780547928227'),
('The Da Vinci Code', 'Dan Brown', 2003, 4, '9780307474278'),
('The Alchemist', 'Paulo Coelho', 1988, 5, '9780062315007'),
('The Little Prince', 'Antoine de Saint-Exupéry', 1943, 8, '9780156012195'),
('Brave New World', 'Aldous Huxley', 1932, 4, '9780060850524'),
('The Lord of the Rings', 'J.R.R. Tolkien', 1954, 6, '9780618640157'),
('Harry Potter and the Sorcerer''s Stone', 'J.K. Rowling', 1997, 7, '9780590353427'),
('The Chronicles of Narnia', 'C.S. Lewis', 1950, 5, '9780066238501'),
('One Hundred Years of Solitude', 'Gabriel García Márquez', 1967, 3, '9780060883287'),
('The Hunger Games', 'Suzanne Collins', 2008, 6, '9780439023481'),
('The Road', 'Cormac McCarthy', 2006, 4, '9780307387899'),
('The Kite Runner', 'Khaled Hosseini', 2003, 5, '9781594631931'),
('The Girl with the Dragon Tattoo', 'Stieg Larsson', 2005, 4, '9780307949486'),
('The Book Thief', 'Markus Zusak', 2005, 6, '9780375842207'),
('Life of Pi', 'Yann Martel', 2001, 5, '9780156027328');

-- Insert Sample Data for Members
INSERT INTO members (name, email, phone, address) VALUES
('John Doe', 'john.doe@email.com', '081234567890', '123 Main St, City'),
('Jane Smith', 'jane.smith@email.com', '081234567891', '456 Oak Ave, Town'),
('Robert Johnson', 'robert.j@email.com', '081234567892', '789 Pine Rd, Village'),
('Mary Williams', 'mary.w@email.com', '081234567893', '321 Elm St, Borough'),
('Michael Brown', 'michael.b@email.com', '081234567894', '654 Maple Dr, District'),
('Sarah Davis', 'sarah.d@email.com', '081234567895', '987 Cedar Ln, County'),
('James Wilson', 'james.w@email.com', '081234567896', '147 Birch Ave, State'),
('Emily Taylor', 'emily.t@email.com', '081234567897', '258 Spruce St, Province'),
('David Anderson', 'david.a@email.com', '081234567898', '369 Ash Rd, Territory'),
('Lisa Thomas', 'lisa.t@email.com', '081234567899', '741 Walnut Ct, Region'),
('Kevin Martin', 'kevin.m@email.com', '081234567800', '852 Cherry Ln, Area'),
('Jennifer White', 'jennifer.w@email.com', '081234567801', '963 Palm Ave, Zone'),
```

('Christopher Lee', 'chris.l@email.com', '081234567802', '159 Beach Rd, Sector'),
('Amanda Clark', 'amanda.c@email.com', '081234567803', '357 Coast St, District'),
('Daniel Martinez', 'daniel.m@email.com', '081234567804', '468 River Dr, County'),
('Michelle Garcia', 'michelle.g@email.com', '081234567805', '789 Lake Ave, State'),
('Andrew Robinson', 'andrew.r@email.com', '081234567806', '951 Ocean Blvd, Province'),
('Patricia Rodriguez', 'patricia.r@email.com', '081234567807', '753 Bay St, Territory'),
('Joseph Hall', 'joseph.h@email.com', '081234567808', '246 Harbor Rd, Region'),
('Nicole King', 'nicole.k@email.com', '081234567809', '135 Port Ave, Area');

-- Insert Sample Data for Borrowings INSERT INTO borrowings (book_id, member_id, borrow_date, return_date, status) VALUES ('b1a2c3d4-e5f6-4321-a123-987654321abc', 'u1b2c3d4-e5f6-4321-a123-987654321def', '2024-11-21', '2024-11-28', 'RETURNED'), ('b2a3c4d5-e6f7-4321-a123-987654321abc', 'u2b3c4d5-e6f7-4321-a123-987654321def', '2024-11-22', '2024-11-29', 'RETURNED'), ('b3a4c5d6-e7f8-4321-a123-987654321abc', 'u3b4c5d6-e7f8-4321-a123-987654321def', '2024-11-23', '2024-11-30', 'RETURNED'), ('b4a5c6d7-e8f9-4321-a123-987654321abc', 'u4b5c6d7-e8f9-4321-a123-987654321def', '2024-11-24', NULL, 'BORROWED'), ('b5a6c7d8-e9f0-4321-a123-987654321abc', 'u5b6c7d8-e9f0-4321-a123-987654321def', '2024-11-25', NULL, 'BORROWED'), ('b6a7c8d9-e0f1-4321-a123-987654321abc', 'u6b7c8d9-e0f1-4321-a123-987654321def', '2024-11-26', '2024-12-03', 'RETURNED'), ('b7a8c9d0-e1f2-4321-a123-987654321abc', 'u7b8c9d0-e1f2-4321-a123-987654321def', '2024-11-27', NULL, 'BORROWED'), ('b8a9c0d1-e2f3-4321-a123-987654321abc', 'u8b9c0d1-e2f3-4321-a123-987654321def', '2024-11-28', '2024-12-05', 'RETURNED'), ('b9a0c1d2-e3f4-4321-a123-987654321abc', 'u9b0c1d2-e3f4-4321-a123-987654321def', '2024-11-29', NULL, 'BORROWED'), ('b0a1c2d3-e4f5-4321-a123-987654321abc', 'u0b1c2d3-e4f5-4321-a123-987654321def', '2024-11-30', '2024-12-07', 'RETURNED'), ('c1b2d3e4-f5a6-4321-a123-987654321abc', 'v1w2x3y4-z5a6-4321-a123-987654321def', '2024-12-01', NULL, 'BORROWED'), ('c2b3d4e5-f6a7-4321-a123-987654321abc', 'v2w3x4y5-z6a7-4321-a123-987654321def', '2024-12-02', '2024-12-09', 'RETURNED'), ('c3b4d5e6-f7a8-4321-a123-987654321abc', 'v3w4x5y6-z7a8-4321-a123-987654321def', '2024-12-03', NULL, 'BORROWED'), ('c4b5d6e7-f8a9-4321-a123-987654321abc', 'v4w5x6y7-z8a9-4321-a123-987654321def', '2024-12-04', '2024-12-11', 'RETURNED'), ('c5b6d7e8-f9a0-4321-a123-987654321abc', 'v5w6x7y8-z9a0-4321-a123-987654321def', '2024-12-05', NULL, 'BORROWED'), ('c6b7d8e9-f0a1-4321-a123-987654321abc', 'v6w7x8y9-z0a1-4321-a123-987654321def', '2024-12-06', '2024-12-13', 'RETURNED'), ('c7b8d9e0-f1a2-4321-a123-987654321abc', 'v7w8x9y0-z1a2-4321-a123-987654321def', '2024-12-07', NULL, 'BORROWED'),

('c8b9d0e1-f2a3-4321-a123-987654321abc', 'v8w9x0y1-z2a3-4321-a123-987654321def', '2024-12-08', NULL, 'BORROWED'), ('c9b0d1e2-f3a4-4321-a123-987654321abc', 'v9w0x1y2-z3a4-4321-a123-987654321def', '2024-12-09', NULL, 'BORROWED'), ('c0b1d2e3-f4a5-4321-a123-987654321abc', 'v0w1x2y3-z4a5-4321-a123-987654321def', '2024-12-10', '2024-12-17', 'RETURNED');