Backend API Documentation - Authentication System

Overview

This document outlines the required API endpoints for the PAU Bookit authentication system. All endpoints should be prefixed with the base URL from `REACT_APP_API_URL` environment variable.

Base Configuration

- **Base URL**: `${REACT_APP_API_URL}` (e.g., `http://localhost:3001/api`)
- **Content-Type**: `application/json`
- **Authentication**: JWT Bearer tokens

Authentication Endpoints

1. User Login

Endpoint: `POST /auth/login`

Purpose: Authenticate user credentials and return JWT token

Request Body:

```
json
```

```json
{
  "email": "student@pau.edu.ng",
  "password": "userPassword"
}
```

Success Response (200):

json

```
{
  "success": true,
  "message": "Login successful",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": "STU12345",
    "firstName": "John",
    "lastName": "Student",
    "email": "john.student@pau.edu.ng",
    "role": "student",
    "isVerified": true,
    "createdAt": "2024-01-15T10:00:00Z"
  }
}
```

Error Responses:

- **401 Unauthorized**: Invalid credentials

json

```
{
  "success": false,
  "message": "Invalid email or password"
}
```

- **403 Forbidden**: Account not verified

json

```
{
  "success": false,
  "message": "Please verify your email address before logging in"
}
```

- **404 Not Found**: User doesn't exist

```json
{
  "success": false,
  "message": "No account found with this email address"
}
```

---

2. User Registration

Endpoint: `POST /auth/signup`

Purpose: Create new user account

Request Body:

```json
{
  "firstName": "John",
  "lastName": "Student",
  "email": "john.student@pau.edu.ng",
  "password": "SecurePass123",
  "studentId": "STU12345",
  "role": "student"
}
```

Validation Rules:

- Email must end with `@pau.edu.ng`
- Password: minimum 8 characters, must contain uppercase, lowercase, and numbers
- Student ID: 3 letters + 5 numbers format (e.g., STU12345)
- All fields are required

Success Response (201):

```json
{
  "success": true,
  "message": "Account created successfully. Please check your email for verification instructic
  "user": {
    "id": "STU12345",
    "firstName": "John",
    "lastName": "Student",
    "email": "john.student@pau.edu.ng",
    "role": "student",
    "isVerified": false,
    "createdAt": "2024-01-15T10:00:00Z"
  }
}
```

Error Responses:

- **400 Bad Request**: Validation errors

```json
{
  "success": false,
  "message": "Validation failed",
  "errors": {
    "email": "Please use your university email address (@pau.edu.ng)",
    "password": "Password must contain uppercase, lowercase, and numbers",
    "studentId": "Student ID must be in format: ABC12345"
  }
}
```

- **409 Conflict**: User already exists

```json
{
  "success": false,
  "message": "An account with this email or student ID already exists"
}
```

---

3. Token Verification

Endpoint: `GET /auth/verify`

Purpose: Verify JWT token validity and get user info

Headers:

```
Authorization: Bearer {token}
```

Success Response (200):

```
json
```

```json
{
  "success": true,
  "isValid": true,
  "user": {
    "id": "STU12345",
    "email": "john.student@pau.edu.ng",
    "role": "student"
  }
}
```

Error Response (401):

```
json
```

```json
{
  "success": false,
  "isValid": false,
  "message": "Invalid or expired token"
}
```

---

4. Token Refresh

Endpoint: `POST /auth/refresh`

Purpose: Refresh JWT token before expiration

Headers:

```
Authorization: Bearer {current_token}
```

Success Response (200):

```json
{
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "expiresIn": 3600
}
```

Error Response (401):

```json
{
  "success": false,
  "message": "Unable to refresh token"
}
```

---

5. User Profile

Endpoint: `GET /auth/profile`

Purpose: Get current user profile information

Headers:

```
Authorization: Bearer {token}
```

Success Response (200):

```json
{
  "success": true,
  "user": {
    "id": "STU12345",
    "firstName": "John",
    "lastName": "Student",
    "email": "john.student@pau.edu.ng",
    "role": "student",
    "studentId": "STU12345",
    "isVerified": true,
    "createdAt": "2024-01-15T10:00:00Z",
    "lastLogin": "2024-01-20T14:30:00Z"
  }
}
```

---

6. User Logout

Endpoint: `POST /auth/logout`

Purpose: Invalidate current session/token

Headers:

```
Authorization: Bearer {token}
```

Success Response (200):

```json
{
  "success": true,
  "message": "Logged out successfully"
}
```

---

7. Forgot Password

Endpoint: `POST /auth/forgot-password`

Purpose: Initiate password reset process

Request Body:

```json
{
  "email": "john.student@pau.edu.ng"
}
```

Success Response (200):

```json
{
  "success": true,
  "message": "Password reset instructions have been sent to your email"
}
```

Error Response (404):

```json
{
  "success": false,
  "message": "No account found with this email address"
}
```

---

8. Reset Password

Endpoint: `POST /auth/reset-password`

Purpose: Reset password using token from email

Request Body:

```json
{
  "token": "password_reset_token_from_email",
  "password": "NewSecurePass123"
}
```

Success Response (200):

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

Error Responses:

- **400 Bad Request**: Invalid/expired token

```json
{
  "success": false,
  "message": "Invalid or expired reset token"
}
```

- **422 Unprocessable Entity**: Password validation failed

```json
{
  "success": false,
  "message": "Password must be at least 8 characters and contain uppercase, lowercase, and numb
}
```

JWT Token Structure

Payload:

```json
{
  "sub": "STU12345",
  "email": "john.student@pau.edu.ng",
  "role": "student",
  "iat": 1642766400,
  "exp": 1642770000
}
```

Token Expiration: 1 hour (3600 seconds)

Security Requirements

Request Headers

All authenticated endpoints should accept:

```
Content-Type: application/json
Authorization: Bearer {jwt_token}
```

CORS Configuration

Allow requests from frontend domain with credentials:

```javascript
{
  origin: process.env.FRONTEND_URL,
  credentials: true,
  methods: ['GET', 'POST', 'PUT', 'DELETE'],
  allowedHeaders: ['Content-Type', 'Authorization']
}
```

Rate Limiting

Implement rate limiting for authentication endpoints:

- Login: 5 attempts per 15 minutes per IP
- Signup: 3 attempts per hour per IP
- Password reset: 3 attempts per hour per email

Password Security

- Hash passwords using bcrypt with salt rounds ≥ 12
- Enforce password policy on backend
- Store password reset tokens with expiration (15 minutes)

Database Schema Requirements

Users Table

```sql
CREATE TABLE users (
  id VARCHAR(10) PRIMARY KEY,
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  student_id VARCHAR(8) UNIQUE NOT NULL,
  role ENUM('student', 'admin') DEFAULT 'student',
  is_verified BOOLEAN DEFAULT FALSE,
  email_verification_token VARCHAR(255),
  password_reset_token VARCHAR(255),
  password_reset_expires DATETIME,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  last_login TIMESTAMP
);
```

Indexes

```sql
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_users_student_id ON users(student_id);
CREATE INDEX idx_users_verification_token ON users(email_verification_token);
CREATE INDEX idx_users_reset_token ON users(password_reset_token);
```

Error Handling Standards

All endpoints should return consistent error responses:

```json
{
  "success": false,
  "message": "Human-readable error message",
  "errors": {
    "field_name": "Specific field error message"
  },
  "code": "ERROR_CODE",
  "timestamp": "2024-01-20T14:30:00Z"
}
```

Environment Variables Required

Backend should use these environment variables:

env

```
JWT_SECRET=your_jwt_secret_key_here
JWT_EXPIRES_IN=3600
DB_HOST=localhost
DB_PORT=3306
DB_NAME=pau_bookit
DB_USER=your_db_user
DB_PASS=your_db_password
EMAIL_SERVICE=gmail
EMAIL_USER=your_email@gmail.com
EMAIL_PASS=your_app_password
FRONTEND_URL=http://localhost:3000
BCRYPT_ROUNDS=12
```

Testing Endpoints

Use tools like Postman or curl to test endpoints:

bash

```
# Login Test
curl -X POST http://localhost:3001/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "student@pau.edu.ng",
    "password": "TestPass123"
  }'

# Protected Route Test
curl -X GET http://localhost:3001/api/auth/profile \
  -H "Authorization: Bearer YOUR_JWT_TOKEN"
```

This documentation provides a complete specification for implementing the authentication system backend that will work seamlessly with the provided React frontend components.