IA03 - User Registration API with React Frontend

Bảng điểm tự đánh giá

Thông tin sinh viên

Trường mục	Thông tin
Họ và tên	Lê Hoàng Việt
MSSV	22120430
Giáo viên hướng dẫn	Nguyễn Huy Khánh
Môn	Phát triển ứng dụng web nâng cao
Đề tài	IA03 - User Registration API with React Frontend
Backend URL	https://ia03-registration-api.onrender.com
Frontend URL	https://ia-03-registration-api.vercel.app

Bảng điểm chi tiết

Tổng điểm tối đa: 10
Tổng điểm đạt được: **10.0 / 10**

STT	Yêu cầu	Mô tả ngắn	Điểm tối đa	Điểm đạt	Bằng chứng / Ghi chú
1	Backend - API Endpoint	POST /user/register với validation và hashing	2.0	2.0	NestJS API với bcrypt hashing, email validation, unique check, error handling
2	Backend - Error Handling	Meaningful error messages cho validation và database	2.0	2.0	ValidationPipe global, class-validator decorators, MongoDB duplicate key handling
3	Frontend - Routing	Home, Login, Sign Up pages với React Router	1.0	1.0	React Router v6, 3 routes: `/`, `/login`, `/signup` với navigation
4	Frontend - Sign Up Page	Form với validation, API integration, React Query	2.0	2.0	React Hook Form validation, React Query mutation, success/error states, shadcn/ui components
5	Frontend - Login Page	Form với validation, UI với shadcn/ui	2.0	2.0	React Hook Form, email/password validation, mock login, shadcn/ui styled components
6	Deployment	Public host cho backend và frontend	1.0	1.0	Backend: https://ia03-registration-api.onrender.com, Frontend: https://ia-03-registration-api.vercel.app

Tổng cộng: 10.0 / 10

Chi tiết kỹ thuật

1. Backend - API Endpoint (2.0đ)

Database Setup

- **MongoDB Atlas:** Cloud database với connection string trong `.env`

- **User Schema:**

```
{
  email: String (required, unique, lowercase, trim)
  password: String (required, hashed with bcrypt)
  createdAt: Date (default: Date.now)
}
```

API Endpoint Implementation

- **Endpoint:** `POST /user/register`
- **Request Body:**

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

- **Validation:**
  - ✅ Email format validation (class-validator @IsEmail)
  - ✅ Email required (@IsNotEmpty)
  - ✅ Password min length 6 characters (@MinLength)
  - ✅ Password required (@IsNotEmpty)
- **Security Features:**
  - ✅ Password hashing với bcrypt (salt rounds: 10)
  - ✅ Unique email check trước khi create
  - ✅ Password không return trong response
  - ✅ Environment variables cho MongoDB URI

- **Response Examples:**
  - Success (201):

```
{
  "message": "User registered successfully",
  "user": {
    "_id": "...",
    "email": "user@example.com",
    "createdAt": "2025-10-28T..."
  }
}
```

  - Error (400 - Duplicate):

```
{
  "message": "Email already exists"
}
```

  - Error (400 - Validation):

```
{
  "message": ["email must be an email", "password must be longer than 6 characters"]
}
```

Files:

- `backend/src/user/user.controller.ts` - POST endpoint với @HttpCode(201)
- `backend/src/user/user.service.ts` - Business logic, bcrypt, duplicate check
- `backend/src/user/schemas/user.schema.ts` - Mongoose schema definition
- `backend/src/user/dto/register.dto.ts` - Validation với class-validator

2. Backend - Error Handling (2.0đ)

Global Validation Pipe

```
app.useGlobalPipes(
  new ValidationPipe({
    whitelist: true,           // Strip unknown properties
    forbidNonWhitelisted: true, // Throw error for unknown properties
    transform: true,            // Auto-transform to DTO types
  })
);
```

Error Handling Features

- ✅ **Validation Errors:** Automatic validation với class-validator
  - Invalid email format → "email must be an email"
  - Missing fields → "email should not be empty"
  - Short password → "password must be longer than 6 characters"
- ✅ **Database Errors:**
  - Duplicate email → Catch MongoDB duplicate key error (code 11000)
  - Connection errors → Meaningful messages
- ✅ **HTTP Status Codes:**
  - 201: Created (successful registration)
  - 400: Bad Request (validation errors)
  - 409: Conflict (duplicate email)
  - 500: Internal Server Error
- ✅ **User-Friendly Messages:**
  - Clear, actionable error messages

- No technical jargon exposed to frontend
- Consistent error response format

Files:

- `backend/src/main.ts` - Global ValidationPipe configuration
- `backend/src/user/user.service.ts` - try-catch error handling

3. Frontend - Routing (1.0đ)

Routes Implementation

```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<Home />} />
    <Route path="/login" element={<Login />} />
    <Route path="/signup" element={<SignUp />} />
  </Routes>
</BrowserRouter>
```

Navigation Features

- ✅ React Router v6 với BrowserRouter
- ✅ Home page (`/`) - Landing page với navigation buttons
- ✅ Login page (`/login`) - Login form
- ✅ Sign Up page (`/signup`) - Registration form
- ✅ Link components cho seamless navigation
- ✅ Programmatic navigation với useNavigate (redirect after signup)

Files:

- `frontend/src/App.jsx` - Router configuration
- `frontend/src/pages/Home.jsx` - Landing page
- `frontend/src/pages/Login.jsx` - Login page
- `frontend/src/pages/SignUp.jsx` - Sign up page

4. Frontend - Sign Up Page (2.0đ)

Form Implementation

- **Library:** React Hook Form v7.49+
- **Fields:**
  - Email (required, valid email format)
  - Password (required, min 6 characters)
  - Confirm Password (required, must match password)

Validation Rules

```
{
  email: {
    required: 'Email is required',
    pattern: {
      value: /^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i,
      message: 'Invalid email address',
    },
  },
  password: {
    required: 'Password is required',
    minLength: {
      value: 6,
      message: 'Password must be at least 6 characters',
    },
  },
  confirmPassword: {
```

```
    required: 'Please confirm your password',
    validate: (value) => value === password || 'Passwords do not match',
  },
}
```

API Integration with React Query

```
const mutation = useMutation({
  mutationFn: (data) => authService.register(data),
  onSuccess: () => {
    // Show success message
    // Redirect to login after 2 seconds
    setTimeout(() => navigate('/login'), 2000);
  },
});
```

UX Features

- ✅ Real-time validation feedback
- ✅ Loading state during API call (button disabled, "Creating account..." text)
- ✅ Success alert với checkmark icon (green background)
- ✅ Error alert với X icon (red background)
- ✅ Auto-redirect to login sau khi success
- ✅ Responsive design (mobile-first)
- ✅ Accessibility (labels, aria-attributes)

Styling (shadcn/ui + Tailwind CSS)

- ✅ Card component với gradient background
- ✅ Input components với focus states
- ✅ Button với hover effects và loading state

- ✅ Alert components cho success/error messages
- ✅ Duolingo-inspired color scheme (#58CC02 green)

Files:

- `frontend/src/pages/SignUp.jsx` - Main component
- `frontend/src/components/ui/input.jsx` - Input component
- `frontend/src/components/ui/button.jsx` - Button component
- `frontend/src/components/ui/alert.jsx` - Alert component
- `frontend/src/lib/api.js` - API service with axios

5. Frontend - Login Page (2.0đ)

Form Implementation

- **Library:** React Hook Form
- **Fields:**
  - Email (required, valid format)
  - Password (required)

Validation

- ✅ Email format validation
- ✅ Required field validation
- ✅ Real-time error messages
- ✅ Error message styling (red text)

UI Features (Mock Login)

- ✅ Login form với email/password fields
- ✅ Submit button

- ✅ Mock success message (no backend integration required)
- ✅ "Don't have an account? Sign up" link
- ✅ Consistent styling với Sign Up page

shadcn/ui Styling

- ✅ Card component (dark background #2b3544)
- ✅ Input fields với dark theme
- ✅ Button với primary green color
- ✅ Typography với proper hierarchy
- ✅ Responsive layout
- ✅ Hover effects và transitions

Files:

- `frontend/src/pages/Login.jsx` - Login component
- `frontend/src/components/ui/card.jsx` - Card components
- Reused UI components from Sign Up page

6. Deployment (1.0đ)

Backend Deployment (Render)

- **Platform:** Render.com
- **URL:** https://ia03-registration-api.onrender.com
- **Configuration:**
  - ✅ Environment variables (MONGODB_URI, PORT)
  - ✅ Auto-deploy from GitHub
  - ✅ Build command: `npm install && npm run build`
  - ✅ Start command: `node dist/main.js`

- ✅ Health check endpoint

Frontend Deployment (Ready)

- **Prepared for:** Vercel / Netlify / Render
- **Configuration:**
  - ✅ Environment variables setup (`.env` , `.env.production`)
  - ✅ Build command: `npm run build`
  - ✅ Output directory: `dist`
  - ✅ CORS configured in backend for production domain
  - ✅ Vite configuration optimized for production

Environment Variables

- **Backend:**

```
MONGODB_URI=mongodb+srv://...
PORT=3000
NODE_ENV=production
```

- **Frontend:**

```
VITE_API_URL=https://ia03-registration-api.onrender.com
```

Files:

- `backend/.env` - Backend environment variables
- `frontend/.env` - Frontend development variables
- `frontend/.env.production` - Frontend production variables

Security Features

Backend Security

1. **Password Security:**
   - ✅ Bcrypt hashing với salt rounds 10
   - ✅ Password không bao giờ return trong response
   - ✅ No plaintext passwords stored
2. **CORS Configuration:**
   - ✅ Whitelist specific origins (localhost + production)
   - ✅ Proper preflight handling
   - ✅ Credentials support
   - ✅ Environment-based origin list
3. **Validation:**
   - ✅ Input sanitization (whitelist, forbidNonWhitelisted)
   - ✅ Type transformation
   - ✅ Email format validation
   - ✅ Password length requirements
4. **Environment Variables:**
   - ✅ Sensitive data trong .env
   - ✅ .env trong .gitignore
   - ✅ ConfigModule global

Frontend Security

1. **Input Validation:**
    - ✅ Client-side validation trước khi submit
    - ✅ XSS prevention (React auto-escaping)
    - ✅ CSRF protection (same-origin policy)
2. **API Communication:**
    - ✅ HTTPS trong production
    - ✅ Environment-based API URLs
    - ✅ Error message sanitization

Code Quality

Architecture

- ✅ **Backend:** Modular NestJS architecture
    - Controllers → Services → Repositories pattern
    - DTOs cho validation
    - Schemas cho database models
    - Separation of concerns
- ✅ **Frontend:** Component-based React architecture
    - Pages → Components → UI Components
    - Custom hooks (useForm, useMutation, useNavigate)
    - Utility functions (cn, api service)
    - Reusable UI components

Best Practices

- ✅ TypeScript trong backend (type safety)
- ✅ ESLint + Prettier configuration
- ✅ Async/await error handling
- ✅ Proper HTTP status codes
- ✅ RESTful API design
- ✅ Responsive design (mobile-first)
- ✅ Accessibility (semantic HTML, labels)
- ✅ Performance optimization (React Query caching)

Project Structure

```
IA03/
├── backend/
│   ├── src/
│   │   ├── user/
│   │   │   ├── dto/
│   │   │   │   └── register.dto.ts
│   │   │   ├── schemas/
│   │   │   │   └── user.schema.ts
│   │   │   ├── user.controller.ts
│   │   │   ├── user.service.ts
│   │   │   └── user.module.ts
│   │   ├── app.module.ts
│   │   └── main.ts
│   ├── .env
│   └── package.json
└── frontend/
    ├── src/
    │   ├── components/
    │   │   └── ui/
    │   │       ├── button.jsx
```

```
│   │       ├── input.jsx
│   │       ├── card.jsx
│   │       └── alert.jsx
│   ├── lib/
│   │   ├── api.js
│   │   └── utils.js
│   ├── pages/
│   │   ├── Home.jsx
│   │   ├── Login.jsx
│   │   └── SignUp.jsx
│   ├── App.jsx
│   └── main.jsx
├── .env
├── .env.production
└── package.json
```

Testing Guide

Backend Testing

1. Test API Endpoint với Postman/Thunder Client:

POST `https://ia03-registration-api.onrender.com/user/register`

Headers:

```
Content-Type: application/json
```

Body (Success):

```
{
  "email": "test@example.com",
  "password": "password123"
}
```

Expected Response (201):

```
{
  "message": "User registered successfully",
  "user": {
    "_id": "67200abc123...",
    "email": "test@example.com",
    "createdAt": "2025-10-28T15:30:00.000Z"
  }
}
```

Body (Duplicate Email):

```
{
  "email": "test@example.com",
  "password": "password123"
}
```

Expected Response (400):

```
{
  "message": "Email already exists"
}
```

Body (Invalid Email):

```
{
  "email": "invalid-email",
  "password": "password123"
}
```

Expected Response (400):

```
{
  "message": ["email must be an email"],
  "error": "Bad Request",
  "statusCode": 400
}
```

2. Test CORS:

```
curl -X OPTIONS https://ia03-registration-api.onrender.com/user/register \
  -H "Origin: https://your-frontend-url.vercel.app" \
  -H "Access-Control-Request-Method: POST"
```

Expected: Status 204 với Access-Control-Allow-Origin header

Frontend Testing

1. Local Development:

```
cd frontend
npm install
npm run dev
```

Truy cập: http://localhost:5173

2. Test Sign Up Flow:

- [ ] Navigate to `/signup`
- [ ] Enter invalid email → See error message
- [ ] Enter short password (< 6 chars) → See error message
- [ ] Enter mismatched passwords → See error message
- [ ] Enter valid credentials → See loading state
- [ ] Wait for success → See success alert
- [ ] Auto-redirect to `/login` after 2 seconds

3. Test Login UI:

- [ ] Navigate to `/login`
- [ ] Enter email and password
- [ ] Click login button → See mock success message
- [ ] Click "Sign up" link → Navigate to `/signup`

4. Test Routing:

- [ ] Visit `/` → See Home page
- [ ] Click "Sign Up" button → Navigate to `/signup`
- [ ] Click "Login" button → Navigate to `/login`
- [ ] Use browser back button → Navigate back correctly

5. Test Responsive Design:

- [ ] Resize browser window
- [ ] Test on mobile (< 640px): Single column layout
- [ ] Test on tablet (640-1024px): Centered card
- [ ] Test on desktop (> 1024px): Centered card with max-width

Công tác kiểm tra / Hướng dẫn xác minh

Backend Checklist

- [ ] MongoDB Atlas connection successful
- [ ] POST /user/register endpoint accessible
- [ ] Email validation working
- [ ] Password hashing working (check database - no plaintext)
- [ ] Duplicate email check working
- [ ] Error messages clear and meaningful
- [ ] CORS enabled for frontend domain
- [ ] Environment variables configured
- [ ] Deployed và accessible publicly

Frontend Checklist

- [ ] Home page displays correctly
- [ ] Routing between pages works
- [ ] Sign Up form validation working
- [ ] API integration successful (can register users)
- [ ] Success/error messages display correctly
- [ ] Login page UI complete
- [ ] Responsive design on all screen sizes
- [ ] shadcn/ui components styled correctly
- [ ] Duolingo-inspired color theme applied
- [ ] Environment variables configured for production

Installation & Running Locally

Prerequisites

- Node.js v18+
- npm hoặc yarn
- MongoDB Atlas account (hoặc local MongoDB)

Backend Setup

```
cd backend
npm install

# Tạo file .env
echo "MONGODB_URI=your_mongodb_connection_string" > .env
echo "PORT=3000" >> .env

# Run development server
npm run start:dev
```

Backend sẽ chạy tại: http://localhost:3000

Frontend Setup

```
cd frontend
npm install

# Tạo file .env
echo "VITE_API_URL=http://localhost:3000" > .env
```

```
# Run development server
npm run dev
```

Frontend sẽ chạy tại: http://localhost:5173

Build for Production

```
# Backend
cd backend
npm run build
npm run start:prod

# Frontend
cd frontend
npm run build
# Output trong folder dist/
```

Deployment Instructions

Backend (Render)

1. Push code to GitHub
2. Tạo new Web Service trên Render
3. Connect GitHub repository
4. Configure:
   - Build Command: `npm install && npm run build`
   - Start Command: `node dist/main.js`
   - Environment Variables: MONGODB_URI, PORT, NODE_ENV
5. Deploy

Frontend (Vercel)

1. Push code to GitHub
2. Import project trên Vercel
3. Configure:
   - Framework Preset: Vite
   - Build Command: `npm run build`
   - Output Directory: `dist`
   - Environment Variables: `VITE_API_URL`
4. Deploy

Screenshots

1. Home Page (Desktop)

Landing page với Welcome message, Sign Up và Login buttons, dark gradient background

2. Sign Up Page

Registration form với email, password, confirm password fields, validation messages, green submit button

3. Sign Up Success

Success alert với checkmark, "User registered successfully" message, auto-redirect notification

4. Sign Up Error (Duplicate Email)

Error alert với X icon, "Email already exists" message

5. Sign Up Error (Validation)

Form validation errors: invalid email format, password too short, passwords don't match

6. Login Page

Login form với email và password fields, mock success message, "Sign up" link

7. Mobile View (Sign Up)

Responsive design: single column, full-width card, touch-friendly inputs

8. API Testing (Postman)

POST request to /user/register, request body, 201 response với user data

9. MongoDB Atlas

Users collection với hashed passwords, createdAt timestamps

10. Render Deployment

Backend deployed successfully, logs showing "Application is running on port 3000"

Conclusion

Dự án IA03 đã hoàn thành đầy đủ các yêu cầu:

- ✅ Backend API hoàn chỉnh với NestJS, MongoDB, validation, error handling
- ✅ Frontend React với routing, form validation, React Query, shadcn/ui
- ✅ Security features: password hashing, CORS, input validation

- ✅ Professional UI/UX với Duolingo-inspired design
- ✅ Deployment-ready với environment variables configuration
- ✅ Clean code architecture và best practices

Tổng điểm tự đánh giá: 10.0 / 10