LocalSkill Backend API

Complete Node.js + Express + MongoDB backend for LocalSkill Marketplace platform.

📁 Project Structure

```
localskill-backend/
│
├── server.js              # Main server file
├── package.json           # Dependencies
├── .env                   # Environment variables (create this)
├── .env.example           # Example env file
│
├── models/
│   ├── User.js            # User schema
│   └── Service.js         # Service schema
│
├── routes/
│   ├── auth.js            # Authentication routes
│   ├── users.js           # User routes
│   └── services.js        # Service routes
│
└── middleware/
    └── auth.js            # JWT authentication middleware
```

🚀 Quick Start

1. Install Dependencies

```bash
npm install
```

2. Setup Environment Variables

Create a `.env` file in the root directory:

```bash
cp .env.example .env
```

Edit `.env` and add your configuration:

```env
```

```
PORT=5000
MONGODB_URI=mongodb://localhost:27017/localskill
JWT_SECRET=your-secret-key-here
NODE_ENV=development
```

3. Install MongoDB

Option A: Local MongoDB

- Download from mongodb.com
- Install and start MongoDB service

Option B: MongoDB Atlas (Cloud - Recommended)

- Create free account at mongodb.com/cloud/atlas
- Create cluster and get connection string
- Update `MONGODB_URI` in `.env`

4. Start the Server

Development mode (with auto-restart):

```bash
npm run dev
```

Production mode:

```bash
npm start
```

Server will run at: `http://localhost:5000`

📡 API Endpoints

Authentication Routes (`/api/auth`)

Register User

```http
```

```http
POST /api/auth/register
Content-Type: application/json

{
  "fullName": "John Doe",
  "email": "john@example.com",
  "password": "password123",
  "phone": "+91 9876543210",
  "city": "Mumbai",
  "userType": "freelancer",
  "skill": "Web Development",
  "experience": "5"
}
```

Login User

```http
POST /api/auth/login
Content-Type: application/json

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

Get Current User

```http
GET /api/auth/me
Authorization: Bearer YOUR_JWT_TOKEN
```

User Routes (`/api/users`)

Get All Freelancers

```http
GET /api/users?city=Mumbai&skill=Web Development
```

Get User by ID

```http
```

```
GET /api/users/:userId
```

Update Profile (Protected)

```http
PUT /api/users/profile
Authorization: Bearer YOUR_JWT_TOKEN
Content-Type: application/json


{
  "fullName": "John Updated",
  "phone": "+91 9876543211",
  "city": "Delhi"
}
```

Service Routes (`/api/services`)

Get All Services

```http
GET /api/services?category=Web Development&city=Mumbai
```

Get Service by ID

```http
GET /api/services/:serviceId
```

Create Service (Freelancer only)

```http
```

```
POST /api/services
Authorization: Bearer YOUR_JWT_TOKEN
Content-Type: application/json

{
  "title": "I will create a professional website",
  "category": "Web Development",
  "description": "Professional website with modern design",
  "price": 5000,
  "deliveryTime": "1 week",
  "features": "Responsive Design\nSEO Optimized\n3 Revisions"
}
```

Update Service (Owner only)

```http
PUT /api/services/:serviceId
Authorization: Bearer YOUR_JWT_TOKEN
Content-Type: application/json

{
  "price": 6000,
  "deliveryTime": "5 days"
}
```

Delete Service (Owner only)

```http
DELETE /api/services/:serviceId
Authorization: Bearer YOUR_JWT_TOKEN
```

Get User's Services

```http
GET /api/services/user/:userId
```

🔐 Authentication

The API uses JWT (JSON Web Tokens) for authentication.

1. Register or login to receive a token
2. Include token in `Authorization` header for protected routes:

```
Authorization: Bearer YOUR_JWT_TOKEN
```

📦 Database Schema

User Model

```javascript
{
  fullName: String,
  email: String (unique),
  password: String (hashed),
  phone: String,
  city: String,
  userType: 'client' | 'freelancer',
  skill: String,
  experience: String,
  description: String,
  rating: Number,
  totalProjects: Number,
  isActive: Boolean,
  createdAt: Date,
  updatedAt: Date
}
```

Service Model

```javascript
{
  userId: ObjectId (ref: User),
  title: String,
  category: String,
  description: String,
  price: Number,
  deliveryTime: String,
  features: [String],
  rating: Number,
  totalOrders: Number,
  isActive: Boolean,
  views: Number,
  createdAt: Date,
  updatedAt: Date
}
```

🛠️ Tech Stack

- **Node.js** - Runtime environment
- **Express.js** - Web framework
- **MongoDB** - Database
- **Mongoose** - ODM
- **JWT** - Authentication
- **bcryptjs** - Password hashing
- **CORS** - Cross-origin requests

📝 Testing the API

Using cURL

```bash
# Register
curl -X POST http://localhost:5000/api/auth/register \
  -H "Content-Type: application/json" \
  -d '{"fullName":"John Doe","email":"john@example.com","password":"password123","phone":"+91 9876543210","city":"Mu

# Login
curl -X POST http://localhost:5000/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"john@example.com","password":"password123"}'

# Get services
curl http://localhost:5000/api/services
```

Using Postman

1. Download Postman
2. Import endpoints from the API documentation above
3. Test all routes with proper headers and body

🔒 Security Features

- ✅ Password hashing with bcrypt
- ✅ JWT token authentication
- ✅ CORS protection

- ✅ Input validation
- ✅ MongoDB injection prevention
- ✅ Soft delete for accounts

🚀 Deployment

Deploy to Heroku

```bash
# Install Heroku CLI
heroku login
heroku create localskill-api
heroku config:set MONGODB_URI=your-mongodb-uri
heroku config:set JWT_SECRET=your-secret
git push heroku main
```

Deploy to Railway

1. Connect GitHub repo
2. Add environment variables
3. Deploy automatically

Deploy to Render

1. Create new Web Service
2. Connect repository
3. Add environment variables
4. Deploy

📄 License

MIT License - Feel free to use for your projects!

🧑‍💻 Developer

Built for LocalSkill Marketplace - Connecting local talent with local opportunities.

Need Help? Open an issue or contact support!