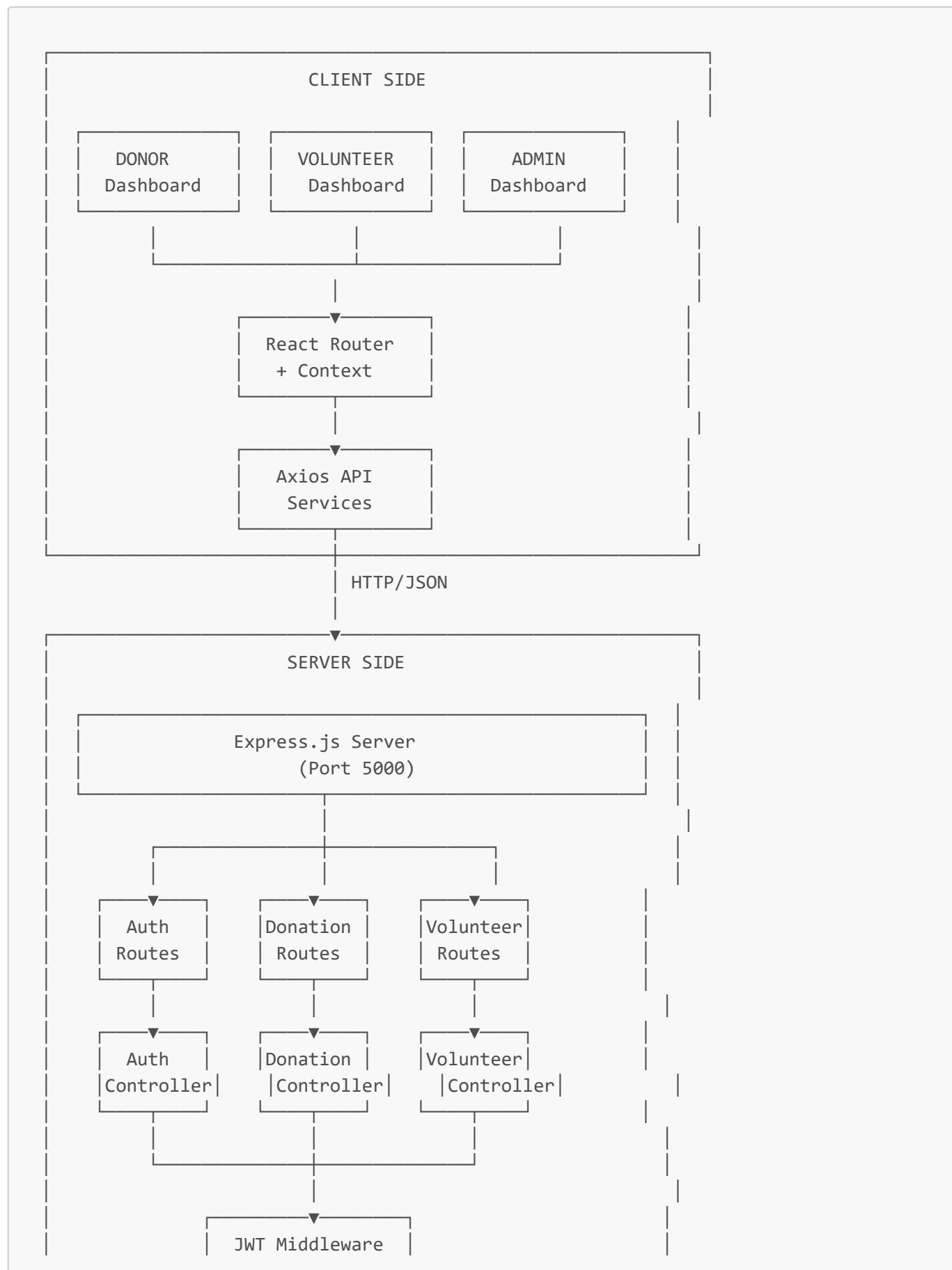
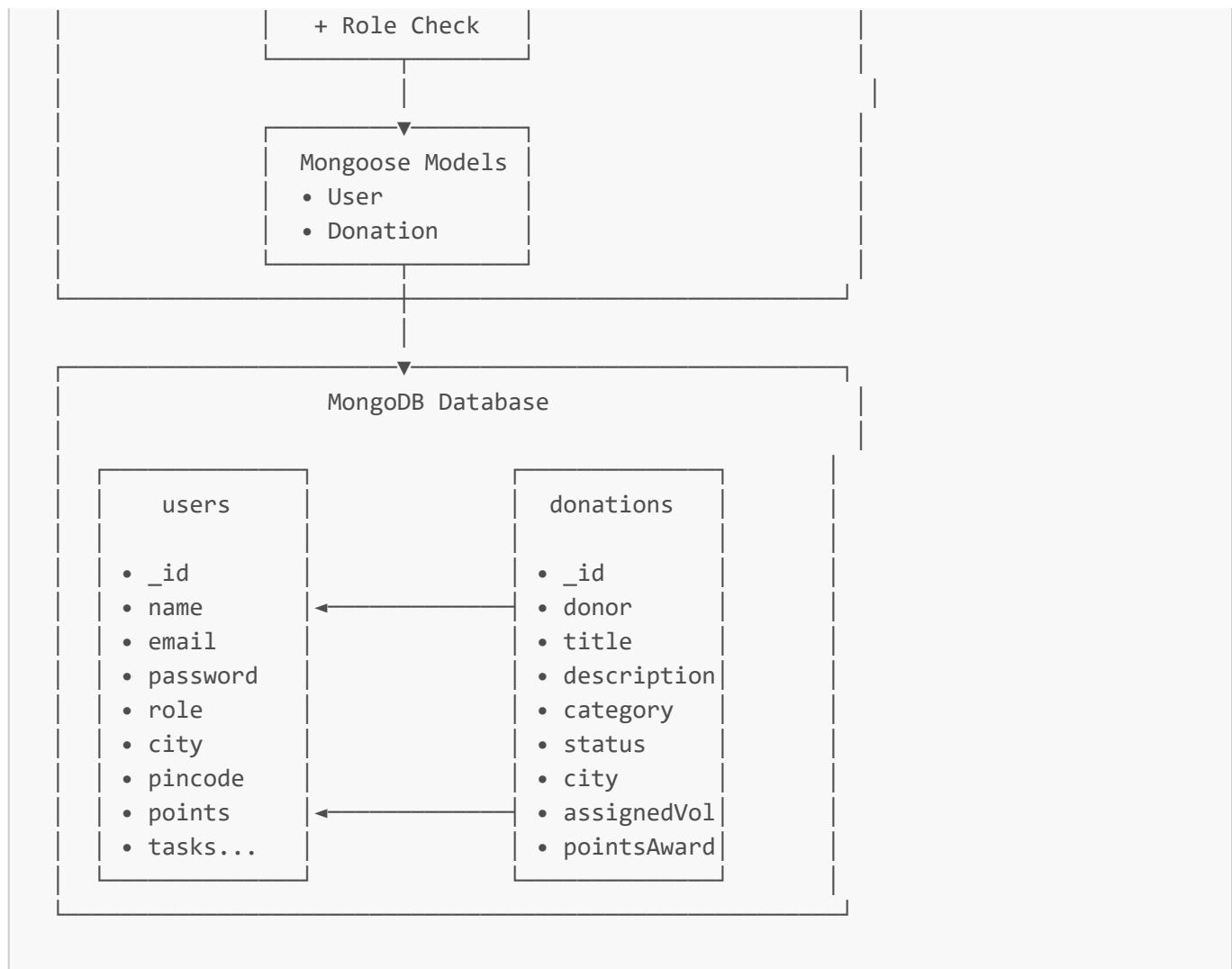


Social Mentor - Architecture Overview

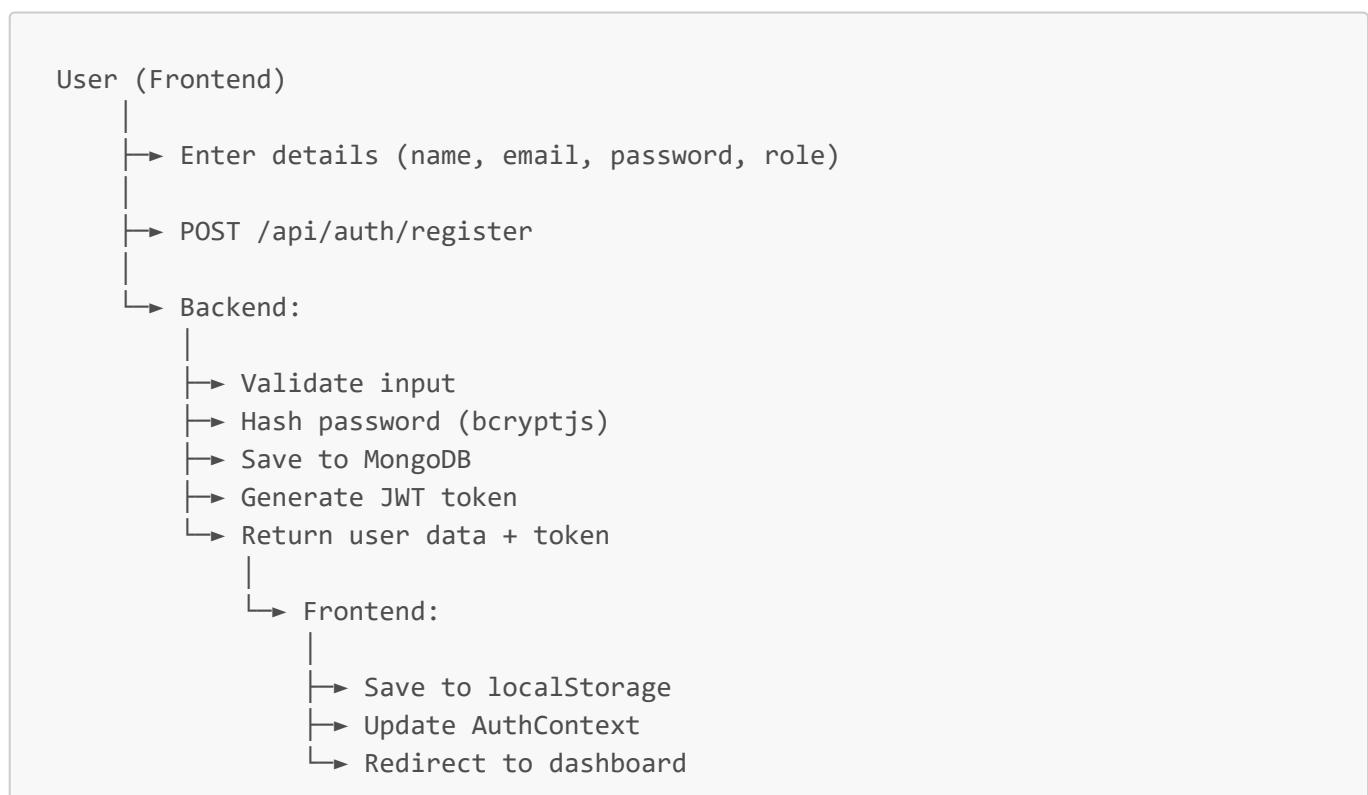
System Architecture





Data Flow Examples

1. User Registration Flow



2. Donation Creation Flow (Donor)

```
Donor Dashboard
  └─> Fill donation form
  └─> POST /api/donations (with JWT)
    └─> Backend:
      └─> Verify JWT token
      └─> Check role = 'donor'
      └─> Create donation document
        • donor: user._id
        • status: 'CREATED'
        • location: city/pincode
      └─> Increment donor's donationsCreated
      └─> Return donation data
        └─> Frontend:
          └─> Refresh donations list
          └─> Show success message
```

3. Volunteer Task Acceptance Flow

```
Volunteer Dashboard
  └─> View available tasks (GET /api/volunteers/available-tasks)
    └─> Backend filters by location:
      • Same city OR pincode
      • Status = 'CREATED'

  └─> Click "Accept Task"

  └─> POST /api/volunteers/accept/:donationId
    └─> Backend:
      └─> Verify JWT token
      └─> Check role = 'volunteer'
      └─> Update donation:
        • status: 'ASSIGNED'
        • assignedVolunteer: volunteer._id
        • assignedAt: Date.now()
      └─> Return updated donation
```

```

    ↳ Frontend:
      ↳ Move to "My Tasks"
      ↳ Show donor contact info
      ↳ Enable status update buttons
  
```

4. Status Update & Points Award Flow

```

Volunteer (My Tasks)
  ↳ Click "Mark Picked Up"
    ↳ PUT /api/volunteers/update-status/:id
      • body: { status: 'PICKED_UP' }

  ↳ Backend updates: status = 'PICKED_UP'

  ↳ Click "Mark Delivered"
    ↳ PUT /api/volunteers/update-status/:id
      • body: { status: 'DELIVERED' }

  ↳ Backend:
    ↳ Update donation status = 'DELIVERED'
    ↳ Award points to volunteer:
      • volunteer.points += donation.pointsAwarded
      • volunteer.tasksCompleted += 1
    ↳ Return success
      ↳ Frontend:
        ↳ Update points display
        ↳ Show completion message
        ↳ Task marked as completed
  
```

API Endpoints Summary

Authentication ([/api/auth](#))

POST	/register	→ Create new user
POST	/login	→ Authenticate user
GET	/profile	→ Get user data [Protected]

Donations ([/api/donations](#))

POST /	→ Create donation [Donor]
GET /my-donations	→ Get donor's donations [Donor]
GET /:id	→ Get single donation [Protected]
PUT /:id	→ Update donation [Donor]
DELETE /:id	→ Delete donation [Donor]

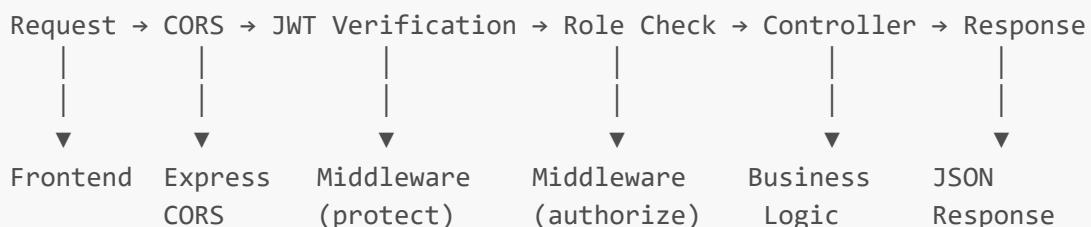
Volunteers ([/api/volunteers](#))

GET /available-tasks	→ Get location-matched tasks [Volunteer]
POST /accept/:donationId	→ Accept task [Volunteer]
GET /my-tasks	→ Get assigned tasks [Volunteer]
PUT /update-status/:donationId	→ Update task status [Volunteer]
GET /leaderboard	→ Get top volunteers [Public]

Admin ([/api/admin](#))

GET /donations	→ Get all donations [Admin]
GET /volunteers	→ Get all volunteers [Admin]
GET /donors	→ Get all donors [Admin]
GET /stats	→ Get platform stats [Admin]

Security Layers



Technology Stack Details

Frontend Stack

React 18.2	→ UI Library
Vite 5.0	→ Build Tool (faster than webpack)
Tailwind CSS 3.3	→ Utility-first CSS
React Router 6.20	→ Client-side routing
Axios 1.6	→ HTTP client
Context API	→ State management

Backend Stack

Node.js	→ Runtime
Express.js 4.18	→ Web framework
MongoDB	→ Database
Mongoose 8.0	→ ODM for MongoDB
JWT 9.0	→ Authentication tokens
bcryptjs 2.4	→ Password hashing
dotenv 16.3	→ Environment variables
CORS 2.8	→ Cross-origin requests

File Structure with Purpose

```

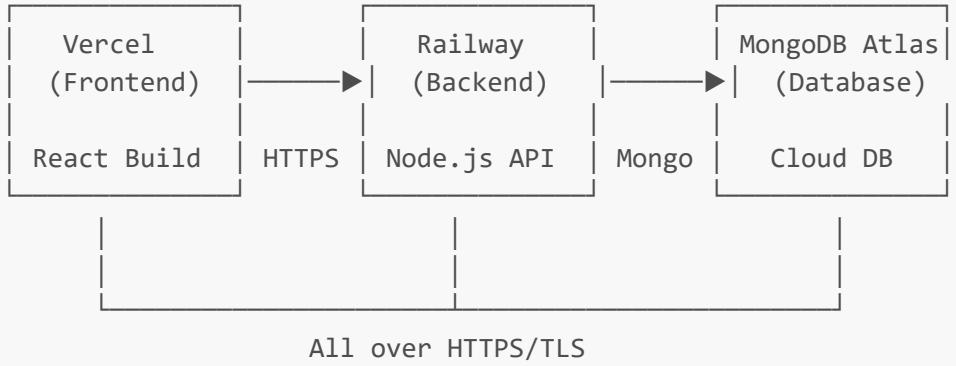
backend/
├── config/
│   └── db.js           → MongoDB connection setup
├── controllers/
│   ├── authController.js    → Registration, login, profile
│   ├── donationController.js → Donation CRUD operations
│   ├── volunteerController.js → Task management, points
│   └── adminController.js   → Admin operations, stats
├── models/
│   ├── User.js          → User schema (donor/volunteer/admin)
│   └── Donation.js       → Donation schema + workflow
├── routes/
│   ├── authRoutes.js     → Auth endpoints
│   ├── donationRoutes.js → Donation endpoints
│   ├── volunteerRoutes.js → Volunteer endpoints
│   └── adminRoutes.js    → Admin endpoints
├── middleware/
│   └── authMiddleware.js → JWT verify + role check
└── server.js           → Express app setup

frontend/
├── src/
│   ├── components/
│   │   ├── Navbar.jsx      → Top navigation bar
│   │   └── ProtectedRoute.jsx → Route guard component
│   ├── context/
│   │   └── AuthContext.jsx → Global auth state
│   ├── pages/
│   │   ├── Home.jsx        → Landing page
│   │   ├── Login.jsx       → Login form
│   │   ├── Register.jsx    → Registration form
│   │   ├── DonorDashboard.jsx → Donor interface
│   │   ├── VolunteerDashboard.jsx → Volunteer interface
│   │   └── AdminDashboard.jsx → Admin interface
│   │   └── Leaderboard.jsx   → Public leaderboard
│   ├── services/
│   │   └── api.js          → Axios configuration + API calls
└── App.jsx              → Main app + routing

```

```
|- main.jsx          → React entry point  
|- index.css        → Global styles + Tailwind
```

Deployment Architecture (Future)



This architecture is designed for:

- **Scalability**: Easy to add features
- **Maintainability**: Clear separation of concerns
- **Security**: JWT + role-based access
- **Performance**: Optimized queries, lean responses
- **Developer Experience**: Clean, readable code

Perfect for a hackathon MVP! 🎉