- User authentication (register/login)
- Task management (CRUD operations)
- Task assignment and permissions
- Chat system for tasks
- File uploads (images and audio)
- SQLite database with Prisma ORM

Installation

1. Clone the repository:

```
git clone <repository-url>
cd livrableYallabaxna
```

2. Install dependencies:

```
npm install
```

3. Set up the database:

```
npx prisma migrate dev
npx prisma generate
```

4. Seed the database (optional):

```
npx prisma db seed
```

Development

```
npm run dev
```

This starts the server with hot reload on http://localhost:4000

Production

```
npm run build
npm run serve
```

API Endpoints

Authentication

- `POST /api/auth/register` - Register a new user
- `POST /api/auth/login` - Login user

Tasks

- `GET /api/tasks` - Get all tasks
- `POST /api/tasks` - Create a new task
- `GET /api/tasks/:id` - Get task by ID
- `PUT /api/tasks/:id` - Update task
- `DELETE /api/tasks/:id` - Delete task

Chat

- `GET /api/chat/:taskId` - Get chat messages for a task
- `POST /api/chat/:taskId` - Send a message in task chat

File Upload

- `POST /upload-vocal` - Upload audio file

Database Schema

The application uses SQLite with the following main models:

- `User` - Application users
- `taskFadil` - Tasks with status, dates, and assignments
- `Permission` - User permissions on tasks
- `Chat` - Chat messages for tasks

Technologies Used

- **Backend**: Node.js, Express.js, TypeScript
- **Database**: SQLite with Prisma ORM
- **Authentication**: JWT tokens
- **File Upload**: Multer
- **Validation**: Zod
- **Password Hashing**: bcrypt

Scripts

- `npm run dev` - Start development server
- `npm run build` - Build TypeScript to JavaScript
- `npm run serve` - Start production server
- `npm run start` - Start server with tsx

Environment Variables

Create a `.env` file if needed for additional configuration (currently using default settings).

Contributing

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Run tests
5. Submit a pull request

License

ISC