TimeBank App

Project summary

TimeBank is a peer-to-peer time credit exchange platform. Members earn credits by offering services and spend credits to request help from others. The current repository contains a full-stack scaffold with a functional frontend demo (hardcoded data) and a minimal backend connection layer.

Objectives

- Validate the core loop of the platform: offer, request, accept, complete, transfer credits.
- Provide a clean, modern UI prototype for feedback and iteration.
- Establish a backend foundation to connect to MongoDB and evolve toward a real API.

Core loop (product logic)

1. Registration gives each user a 3-credit bonus.
2. A provider lists an offer (e.g., guitar lessons, 1 credit/hour).
3. A requester with credits submits a service request.
4. The provider accepts and the service happens.
5. The requester marks the service as completed.
6. Credits transfer from requester to provider.

Current implementation

Frontend

- Multi-page UI built with React Router.
- Marketplace with search, credit filter, tag filter, clear-filters action, and empty state.
- Service detail page and request flow timeline.
- Profiles listing and profile detail pages with offers + activity.
- Hardcoded demo data stored in JSON at `client/src/data/demoData.json`.
- Cursor-following background glow and custom styling.

Backend

- Express server with CORS and JSON body parsing.
- MongoDB connection using Mongoose via `MONGO_URI`.
- Health/test route at `/`.

Pages and routes

- `/` Overview and core loop summary
- `/marketplace` Offers listing + filters
- `/service/:id` Service detail
- `/request/:id` Request flow timeline
- `/profiles` Community profiles
- `/profile/:id` Profile detail

- `/login` and `/register` (UI only)

Folder structure

- client/ React frontend (Vite)
- server/ Express API server

Environment variables

Create `server/.env` with:

- `MONGO_URI` (MongoDB connection string)
- `PORT` (server port; defaults to 5000)

Do not commit secrets or credentials to source control.

Run the project

Server

```
cd server
npm install
npm run dev
```

Client

```
cd client
npm install
npm run dev
```

The client runs on Vite's dev server, and the API runs on the configured `PORT`.

Deployment

GitHub Pages is configured to deploy the frontend on every push to the `master` branch. This deployment is frontend-only; the Node/Express API must be hosted separately.

Current limitations

- No authentication or authorization.
- No API routes beyond the test route.
- No database models or schemas beyond the MongoDB connection.
- All frontend data is hardcoded (no API integration).
- No persistence for requests, balances, or profiles.

Planned work

- Add Mongoose models (User, Service, Transaction).
- Build REST API routes and controllers.
- Add auth (JWT + bcrypt) and middleware.
- Add a client API layer and wire the UI to the backend.
- Replace hardcoded JSON data with API data.
- Add validation, error handling, and logging.