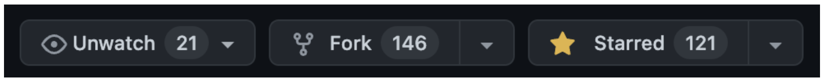
# Upwork Shortlist Test Task: Spec

Welcome to the test task. If you are here it's because we can imagine working with you long term.

Thank you for completing the quiz and especially to those of you who downloaded Minima and left your wallet address. Everyone who gave me their wallet address now has some MINIMA waiting for them in their wallet - this is yours to keep and play with.

## Some admin

### First steps:

1. FOLLOW and STAR the Minima repo on GitHub - https://github.com/minima-global/Minima
   

   - This means i can check that you own your github and also i can give you access to the private repo containing the minidapp we will be working on.
2. Request me (@monthrie) to be your friend on discord (so I know you have given me the correct handle and so we can communicate easily)

### Some comments:

Minima offers a unique and challenging developer environment. If you don't enjoy working with it or cannot see the point of Minima, this contract may not be a good fit for you. Please feel free to say that at any time and I will take you off our list.

Impressing us with this test could lead to multiple future job opportunities and bounties in the coming weeks and months - regardless of whether you are chosen for this job.

Successfully passing this test will put you at the front of the line for our online (or in-person) hackathons, which feature large prizes. We currently have plans to hold in-person hackathons in India, Pakistan, Vietnam, Ethiopia and Rwanda - and we are keeping our eyes open for potential partners and collaborators.

I'm available to answer questions, but it would be great to see evidence that you can work independently to solve problems using Minima.

Anyone who makes an attempt at this task will receive feedback and everyone will be able to keep any MINIMA tokens that I have already sent.

Thank you and good luck.

## Requirements:

- Download the minidapp I send you as a zip file on Upwork. (if i have missed you, please let me know).

- Open it up in your code editor (we typically use VS code but you are welcome to use whatever you like)

- Download a desktop instance of Minima. This should be mainnet rather than private node. As the instructions show, you can do this through Docker or from the command line. Ask me if you are really struggling with this for whatever reason.
    - You will need to add peers to actually join the network.
    - Go to terminal and type:
        - `peers action:addpeers peerslist:https://www.spartacusrex.com/minimapeers.txt`
        - `If that doesn't work you can also go into minima > settings > add connections > then just write in the URL above in full and that should also work.`
        - `Let me know if any issues.`
- `You should already have MINIMA in your wallet, if not, let me know and I will send you a few to play with.`

- Zip the minidapp and install it onto your node.

### Some notes on the current form of the Minidapp that you will be amending:

**Overview**

The "Shout Out" MiniDAPP is a web application which is supposed to function a bit like Bitcointalk. It doesn't really, but that's why we are here!

It is very simple - it was built in just a day or so by the inventor of Minima (spartacusrex). It is simple javascript and HTML and some styling. It is not a pretty application(!).

It leverages the Minima blockchain for decentralized operations and uses various JavaScript libraries for functionality and styling.

## Key Files and Their Roles

1. index.html
   - **Main interface for navigating categories, creating topics, and viewing recent messages.**
   - Components:
     - Header: Logo, title, and navigation icons for settings and home.
     - Category Section: Search and navigate categories and sub-categories.
     - Topic Section: Create new topics with title and description.
     - Topics List: Display list of topics with options to view details.
     - Navigation Buttons: "Back" and "Next" buttons for pagination.
     - Welcome Message: Introduction and usage instructions.
   - Interactions:
     - Navigation: Functions like `jumpRecent`, `jumpSettings`, and `jumpHome` for page redirection.
     - Category Management: Functions to validate, clean, and manage categories.
     - Topic Management: Functions to create and display topics.
     - User Interaction: Functions to handle user actions like blocking categories and showing notifications.

2. recent.html
   - **Display recent messages.**
   - Components:
     - Header: Logo, title, and navigation icons for settings and home.
     - Title: "All recent messages".
     - Messages Container: Div (`#recentmsg`) to display recent messages.
     - Navigation Buttons: "Back" and "Next" buttons for pagination.
   - Interactions:
     - Initialization: `MDS.init` to set up the MiniDAPP and call `writeRecent()`.
     - Navigation: Functions like `jumpSettings`, `jumpHome`, and `nextBatch` for page redirection.
     - Message Handling: `writeRecent` fetches and displays messages using `selectRecentMessages` from `sql.js`.

3. usermessages.html
   - **Display messages from a specific user.**
   - Components:
     - Header: Logo, title, and navigation icons for settings and home.
     - User Title: Div (`#usertitle`) to display user's name and ID.
     - Messages Container: Div (`#recentmsg`) to display user's messages.

        - Navigation Buttons: "Back" and "Next" buttons for pagination.
    - Interactions:
        - Initialization: Variables like `USER_PUBKEY` and `MSG_OFFSET` are retrieved from URL parameters.
        - Navigation: Functions like `jumpSettings`, `jumpHome`, and `nextBatch` for page redirection.
        - Message Handling: `writeRecent` fetches and displays user's messages using `selectUserMessages` from `sql.js`.
        - User Interaction: Functions like `tipUser` and `blockUser` for user-specific actions.

## 4. mds.js

**[very important in the Minima Ecosystem - it is the same in all minidapps and does not need to be changed!]**

- **Core library for MiniDAPPs, providing utility functions for interacting with the Minima blockchain.**
- Key Functions:
    - Initialization: `init(callback)` to set up the MiniDAPP.
    - Logging: `log(output)` for logging messages.
    - Notifications: `notify(output)` and `notifycancel()` for user notifications.
    - Commands: `cmd(command, callback)` and `sql(command, callback)` for executing Minima and SQL commands.
    - Network Requests: `net.GET(url, callback)` and `net.POST(url, data, callback)` for HTTP requests.
    - Key-Value Storage: `keypair.get(key, callback)` and `keypair.set(key, value, callback)` for persistent storage.
    - Communication: `comms.broadcast(msg, callback)` and `comms.solo(msg, callback)` for messaging.
    - File Operations: Methods for file management.
    - Utility Functions: Hex/Base64 conversions, state variable retrieval, etc.

## 5. sql.js

- Database interactions for the MiniDAPP.
- Key Functions:
    - Database Initialization: `createDB(callback)` to create necessary tables.
    - Message Handling: `insertMessage` and `selectRecentMessages` for inserting and fetching messages.

- Category and Topic Management: `selectCategories` and `selectTopics` for managing categories and topics.
- User and Message Filtering: `isUserBlocked`, `addBlockUsers`, `isTopicBlocked`, and `addBlockTopic` for filtering.
- Notification Management: `isNotify` and `newNotifyTopic` for managing notifications.

### Summary

The "Shout Out" MiniDAPP is a kind of microblogging site, leveraging the Minima blockchain for decentralized operations. It consists of several key files, each with specific roles and interactions. The `index.html` file serves as the main interface, while `recent.html` and `usermessages.html` display recent and user-specific messages, respectively. The `mds.js` library provides core functionalities, and `sql.js` handles database interactions. Together, these components create a robust and interactive MiniDAPP for users to communicate and manage messages, categories, and topics.

## Task:

Currently, the application opens to a confusing page with various categories, which isn't user-friendly.

Your task is to change this page so that it displays recent messages, similar to a (very) basic version of Twitter. There should also be a box for posting a message.

The result of your work should be: when the Minidapp opens, I see a list of all the recent messages with a box above it allowing me to post a new message.

All the necessary elements are already in the application; they just need to be rearranged. Ideally, we're removing categories since posts no longer need them. Then we are moving the ‘all recent messages’ view to the front page.

The key is that the entire Minidapp should still function properly in the most basic sense: there must be a way to post/broadcast a message and a way to display messages that others have sent.

The strange and beautiful thing about working with the Minidapp System (MDS) is that these messages are actually broadcast ‘on chain’ (and then saved locally on your device). Because we are using the chain, a tiny tiny fraction of a minima needs to be spent for each message (less than a millionth of a minima). This is already in built to the posting of a message, but something to be aware of.

It's acceptable if your minidapp only works in WRITE mode. If you need clarity on this, let me know and I will explain.

—

Please keep this in mind: You have been chosen because we respect your input so far and we like your profile and we can imagine working with you - if this task seems impossible or unrealistic, please let me know. I value your feedback.

If you come back and explain what you tried and what you are struggling with, it doesn't necessarily mean you will not get the job. I really want people that can communicate well with me and that I can trust to be honest and clear.

## Things worth considering / steps to take:

**Remove Categories:**

- index.html: Remove the category-related sections and functions.

**Move Post Message Section to Front Page:**

- index.html: Move the post message section to the top of the page.

**Make Front Page Display Recent Messages:**

- index.html: Integrate the functionality of `recent.html` to display recent messages on the front page.

**Update Navigation and Remove Unnecessary Elements:**

- index.html: Update navigation to remove category-related links and ensure settings and home links are functional.

**Adjust CSS for New Layout:**

- style.css: Update styles to match the new layout and give it a Twitter-like feel.

**Ensure Database Functions Support New Structure:**

- sql.js: Ensure `insertMessage` and `selectRecentMessages` functions are correctly handling the new message structure.

—

**Once you are done, Zip the application and test it on your node.**

**Send it to me on GitHub (username is monthrie), let me know that you have completed the task on Upwork.**

**DEADLINE = Wednesday 26 JUNE**