

Stock Check Tool - Owner Guide

Complete administrator guide for managing the Stock Check Tool system.

Table of Contents

1. [User Management via CLI](#)
 2. [Updating Ingredient List](#)
 3. [Viewing Submissions](#)
 4. [Deployment Guide](#)
 5. [Backend Management \(Render\)](#)
 6. [Troubleshooting](#)
-

User Management via CLI

Prerequisites

- Git Bash terminal (Windows) or Terminal (Mac/Linux)
- Navigate to project folder: `cd server`

List All Users

```
npm run list-users
```

This shows all registered users with their:

- Username
- Language preference
- Account creation date

Add New User

```
npm run add-user
```

Follow the interactive prompts:

1. Enter **username** (lowercase, no spaces)
2. Enter **password** (minimum 6 characters)
3. Select **language** preference:
 - `mm` = Myanmar (မြန်မာ)
 - `en` = English
 - `zh` = Chinese (中文)

Example:

```
$ npm run add-user

Enter username: john
Enter password: *****
Select language (mm/en/zh): en

☒ User 'john' created successfully!
```

Remove User

```
npm run remove-user
```

Follow the prompt:

1. Enter the **username** to remove
2. Confirm deletion

Example:

```
$ npm run remove-user

Enter username to remove: john

☒ User 'john' removed successfully!
```

Important Notes on User Management

- **Passwords are hashed** using bcrypt (secure, cannot be reversed)
- **Users stay logged in** on their devices until they logout
- **Removing a user** doesn't delete their submission history
- **Username is case-sensitive** during login but stored as lowercase

 **Updating Ingredient List****File Location**

```
client/src/data/items.js
```

How to Add/Edit Items

1. **Open the file** in your code editor

2. Find the items object:

```
export const items = {
  main: [
    { id: 1, name: { mm: 'တို့ဖူး', en: 'Tofu', zh: '豆腐' }, image:
'/webp/tofu.webp' },
    { id: 2, name: { mm: 'ဘဲကြွပ်', en: 'Cabbage', zh: '白菜' }, image:
'/webp/cabbage.webp' },
    // ... more items
  ],
}
```

Adding a New Item

```
{
  id: 20,                                // Must be unique!
  name: {
    mm: 'Myanmar name',                 // Myanmar language
    en: 'English name',                 // English language
    zh: 'Chinese name'                  // Chinese language
  },
  image: '/webp/item-name.webp'          // Image path or emoji
}
```

Steps:

1. Add the item to the array
2. Make sure the **id is unique**
3. Add translations for **all 3 languages**
4. (Optional) Add image to `client/public/webp/` folder
5. Save the file
6. Deploy (see [Deployment Guide](#))

Editing Existing Items

Just modify the name translations:

```
// Before:
{ id: 2, name: { mm: 'ဘဲကြွပ်', en: 'Cabbage', zh: '包菜' }, image:
'/webp/cabbage.webp' },

// After:
{ id: 2, name: { mm: 'ဘဲကြွပ်', en: 'Cabbage', zh: '白菜' }, image:
'/webp/cabbage.webp' },
```

Removing Items

Simply delete the entire line:

```
// Delete this:  
{ id: 5, name: { mm: 'ချဉ်သီး', en: 'Pickles', zh: '泡菜' }, image:  
  '/webp/pickles.webp' },
```

Adding Item Images

1. Prepare image:

- Format: WebP (for best performance)
- Size: 300x300 pixels
- File size: Under 100KB
- Name: lowercase-with-hyphens.webp

2. Place image in `client/public/webp/` folder

3. Reference in `items.js`:

```
{ id: 20, name: { ... }, image: '/webp/your-item.webp' }
```

4. Or use emoji (temporary):

```
{ id: 20, name: { ... }, image: '🥒' }
```

Viewing Submissions

Database Location

```
server/src/database/submissions.json
```

View Submission Data

Open the file in any text editor to see:

- User ID and username
- Submitted items with quantities
- Notes
- Submission date and time (Malaysia timezone)

Example submission:

```
{
  "id": 1,
  "userId": 1,
  "username": "john",
  "items": [
    {
      "id": 1,
      "name": { "mm": "豆腐", "en": "Tofu", "zh": "豆腐" },
      "quantity": 5
    }
  ],
  "notes": "Urgent - need by tomorrow",
  "date": "2025-12-07",
  "timestamp": "2025-12-07T10:30:00+08:00"
}
```

Database Files

Users Database:

```
server/src/database/users.json
```

Contains: All user accounts (username, hashed password, language)

Submissions Database:

```
server/src/database/submissions.json
```

Contains: All daily submissions (items, quantities, notes, timestamps)

Quantities Database:

```
server/src/database/quantities.json
```

Contains: Unit Check quantities (braised pork, kong bak) - ephemeral on Render

Deployment Guide

When to Deploy

Deploy whenever you make changes to:

- ☒ Item list (add/edit/remove items)
- ☒ UI/styling changes

- ☒ Bug fixes
- ☒ New features

Deployment Steps

See **DEPLOY.md** for complete guide. Quick summary:

1. Update Service Worker Version

Open `client/public/sw.js`:

```
const CACHE_VERSION = '5.4' // Increment this: 5.4 → 5.5
```

2. Commit Changes

```
git add .
git commit -m "Description of your changes"
git push
```

3. Deploy to Vercel (Frontend)

```
cd client
vercel --prod --yes --build-env VITE_API_URL=https://stockcheck-
api.onrender.com/api
```

4. Users Get Updates Automatically

- Users refresh → new version loads
- Changes appear within seconds
- No manual update needed!



Backend Management (Render)

Accessing Render Dashboard

1. Go to: <https://dashboard.render.com/>
2. Login with your account
3. Click on **stockcheck-api** service

Important Environment Variables

Must be set in Render dashboard:

```
CLIENT_URL=https://client-scydom-chins-projects.vercel.app
PORT=5000
NODE_ENV=production
```

How to update:

1. Go to service → **Environment** tab
2. Click **Edit**
3. Add/modify variables
4. Click **Save Changes**
5. Render auto-redeploys

When Backend Changes Needed

You **ONLY** need to redeploy backend if:

- ✗ You modified files in **server/** folder
- ✗ You changed API routes
- ✗ You updated database logic
- ✗ You changed dependencies

You **DON'T** need to redeploy backend for:

- ☒ Frontend changes (items, UI, styles)
- ☒ Client-side code updates
- ☒ Adding new users via CLI

Redeploying Backend to Render

Backend auto-deploys when you push to GitHub:

```
# Make backend changes in server/ folder
git add .
git commit -m "Backend update: description"
git push
```

Render automatically:

1. Detects the push
2. Rebuilds the server
3. Redeploys (takes ~2-3 minutes)
4. Service restarts with new code

Checking Backend Status

Method 1: Quick Status Check

```
curl -s https://stockcheck-api.onrender.com/api/quantities/status
```

Expected response:

```
{"success":true,"date":"2025-12-09","current_period":"morning","can_submit":true,"morning":null,"evening":null}
```

☒ If you see JSON response → Server is **WORKING** ✗ If no response or error → Server is **DOWN**

Method 2: Check with Response Time (Recommended)

```
time curl -s https://stockcheck-api.onrender.com/api/quantities/status
```

What the response time tells you:

☒ **real 0m1.234s** (< 3 seconds) → Server was **AWAKE** 🟢 ⚠️ **real 0m25.678s** (20-50 seconds) → Server was **SLEEPING**, just woke up 🟡 ✗ **real 1m0.000s+** (timeout) → Server is **DOWN** 🛑

Example output:

```
{"success":true,"date":"2025-12-09",...}

real    0m1.456s   ← Fast response = Server awake!
user    0m0.031s
sys     0m0.000s
```

Why this matters: Render Free Tier sleeps after 15 minutes of inactivity. First request after sleep takes 20-50 seconds to wake up.

Method 3: Health Check (Legacy)

```
curl https://stockcheck-api.onrender.com/api/health
```

Expected response:

```
{"status":"ok","message":"Stock Check API is running"}
```

Backend Logs

View logs in Render dashboard:

1. Go to **stockcheck-api** service
 2. Click **Logs** tab
 3. See real-time server logs
-

Troubleshooting

Users Can't Login

Check:

1. Backend is running (health check)
2. **CLIENT_URL** on Render matches Vercel URL
3. User credentials are correct
4. Internet connection working

Fix:

```
# Check if backend is responding
curl https://stockcheck-api.onrender.com/api/health

# Verify user exists
cd server
npm run list-users
```

Items Not Showing

Check:

1. Items properly formatted in **items.js**
2. All 3 language translations present
3. No syntax errors in JSON
4. Service worker version incremented

Fix:

```
# Check for syntax errors
cd client
npm run build
```

Updates Not Appearing

Check:

1. Service worker version incremented?
2. Deployed to Vercel?

3. Users refreshed the page?

Fix:

1. Increment version in `sw.js`
2. Redeploy to Vercel
3. Tell users to refresh (Ctrl+Shift+R)

Backend Data Lost

Important:

- Render free tier has **ephemeral filesystem**
- Backend **restarts** can erase `quantities.json`
- `users.json` and `submissions.json` persist (stored in repo)

Solutions:

- User/submission data: Safe (committed to git)
- Quantity data: Lost on restart (this is expected)

CORS Errors

Symptom: Login fails with "Failed to fetch"

Fix:

1. Go to Render dashboard
2. Update `CLIENT_URL` environment variable
3. Match your Vercel deployment URL exactly
4. Save changes → Render redeploys

System Architecture

Frontend (Vercel)

- **URL:** `https://client-scydom-chins-projects.vercel.app`
- **Repository:** GitHub (auto-deploy on push)
- **Technology:** React + Vite + PWA
- **Deployment:** Manual via `vercel --prod`

Backend (Render)

- **URL:** `https://stockcheck-api.onrender.com/api`
- **Repository:** GitHub (auto-deploy on push)
- **Technology:** Node.js + Express
- **Database:** JSON files (local filesystem)

Database Files

- `users.json` - User accounts (persists)

- `submissions.json` - Daily submissions (persists)
 - `quantities.json` - Unit check data (ephemeral)
-

Security Best Practices

1. **Never commit** `.env` files to git
 2. **Change default** `JWT_SECRET` in production
 3. **Use strong passwords** for user accounts
 4. **Regularly backup** database files
 5. **Monitor Render logs** for suspicious activity
-

Additional Resources

- **User Guide:** See `USER_GUIDE.md`
 - **Deployment Guide:** See `DEPLOY.md`
 - **README:** See `README.md`
 - **GitHub Repo:** <https://github.com/Scydom8885/stockchecktool>
-

Emergency Contacts

If System is Down

1. **Check Render status:** Dashboard → stockcheck-api → Status
2. **Check Vercel status:** Dashboard → client project
3. **View logs:** Both platforms have log viewers
4. **Restart services:** Manual restart option in dashboards

Database Recovery

If `submissions.json` is corrupted:

1. Check GitHub history
2. Restore from previous commit
3. Use `git checkout` to recover file

If `users.json` is corrupted:

1. Restore from GitHub
 2. Re-run `npm run seed` to recreate default users
 3. Re-add users via CLI
-

System Version: 5.4 **Last Updated:** December 2025 **Maintained By:** Owner/Administrator