

HackPush: Browser Extension Technical Documentation

Automatically Sync Accepted HackerRank Solutions to GitHub

Executive Summary

HackPush is a Chrome/Firefox browser extension designed to automatically synchronize your accepted HackerRank coding solutions to a GitHub repository^{[1] [2] [3]}. Inspired by the popular LeetHub extension^{[1] [2]}, HackPush adapts the workflow specifically for HackerRank's DOM structure and submission process, using **Manifest V3** standards for modern browser compatibility^{[4] [5] [6]}.

The extension monitors HackerRank challenge pages, detects when a submission is accepted, extracts the code and metadata, and pushes it to a configured GitHub repository—all without requiring any manual intervention from the user^{[1] [2]}.

1. Project Architecture & File Structure

1.1 Directory Organization

```
HackPush/
├── manifest.json
├── icons/
│   ├── icon16.png
│   ├── icon48.png
│   └── icon128.png
├── src/
│   ├── js/
│   │   ├── background.js (service worker)
│   │   ├── content.js (main logic)
│   │   ├── dom-parser.js (DOM utilities)
│   │   ├── github-api.js (API wrapper)
│   │   ├── storage.js (data management)
│   │   └── oauth2.js (authentication)
│   ├── popup/
│   │   ├── popup.html
│   │   ├── popup.js
│   │   └── popup.css
│   └── options/
│       ├── options.html
│       ├── options.js
│       └── options.css
├── README.md
└── package.json
```

1.2 Core Components

manifest.json

The manifest file defines extension metadata, permissions, and component registration^{[7] [8] [9]}. Key configurations include:

- **Manifest Version:** 3 (latest standard)^{[4] [5] [10]}
- **Permissions:** storage, identity, scripting, activeTab^[7]
- **Host Permissions:** https://www.hackerrank.com/*, https://api.github.com/*^[7]
- **Background Service Worker:** Single file, ES6 module support^{[11] [10] [12]}
- **Content Scripts:** Injected into HackerRank challenge pages^{[7] [13]}

Content Script (content.js)

Runs directly on HackerRank pages to monitor submissions^[13]. Responsibilities:

- Detect submit button clicks
- Observe DOM changes using MutationObserver^{[14] [15] [16] [17]}
- Wait for submission result panel
- Extract code, language, and problem metadata
- Send accepted submissions to background script

Background Service Worker (background.js)

Handles GitHub API operations and OAuth authentication^{[11] [10] [12]}. Functions:

- Process messages from content script
- Manage GitHub API requests^{[18] [19] [20]}
- Create commits and push to repository^{[19] [20] [21]}
- Store configuration and submission history

DOM Parser (dom-parser.js)

Utility module for extracting information from HackerRank's dynamic DOM^{[22] [23] [14]}:

- Code extraction from CodeMirror^{[24] [25] [26] [27]} or Monaco editors^[28]
- Language detection from selectors
- Problem title and slug extraction
- Category identification from URL patterns
- Submission status detection

2. Technical Implementation Details

2.1 DOM Detection Strategy

HackerRank uses dynamic page rendering, making robust DOM detection critical^[22] ^[14]. The extension implements multiple fallback strategies:

Code Editor Detection

Primary Method: CodeMirror^[22] ^[24] ^[25] ^[26]

```
// Access CodeMirror instance
const cmElement = document.querySelector('.CodeMirror');
if (cmElement && cmElement.CodeMirror) {
  const code = cmElement.CodeMirror.getValue();
}
```

Fallback 1: Monaco Editor^[28]

```
const monacoElement = document.querySelector('.monaco-editor');
if (monacoElement && window.monaco) {
  const editor = monaco.editor.getModels()[^0];
  const code = editor.getValue();
}
```

Fallback 2: Textarea

```
const textarea = document.querySelector('textarea[name="code"]');
const code = textarea.value;
```

Language Detection

Multiple selector strategies ensure language extraction works across UI updates:

- `select[name="language"]` - Standard dropdown
- `[class*="language"] select` - Class-based selector
- Button text extraction with normalization map

The extension maintains a comprehensive language-to-extension mapping:

```
{
  'python3': 'py',
  'java': 'java',
  'javascript': 'js',
  'cpp': 'cpp',
  // ... 15+ languages supported
}
```

Submission Status Detection

The extension uses `MutationObserver`^{[14] [15] [16] [17]} to watch for result panel updates:

```
const observer = new MutationObserver((mutations) => {
  const resultPanel = document.querySelector('[class*="result"]');
  const text = resultPanel.textContent.toLowerCase();

  const acceptedKeywords = [
    'accepted', 'success',
    'all test cases passed',
    'congratulations'
  ];

  const isAccepted = acceptedKeywords.some(kw => text.includes(kw));
});

observer.observe(document.documentElement, {
  childList: true,
  subtree: true,
  characterData: true,
  attributes: true
});
```

2.2 Submission Detection Workflow

Step 1: Monitor Submit Button

```
function monitorSubmitButton() {
  const button = DOMParser.findSubmitButton();
  button.addEventListener('click', handleSubmitClick);
}
```

Step 2: Wait for Result Panel

Uses promise-based waiting with timeout^[14]:

```
async function waitForElement(selector, timeout = 15000) {
  return new Promise((resolve, reject) => {
    // Check if element already exists
    const existing = document.querySelector(selector);
    if (existing) return resolve(existing);

    // Create observer
    const observer = new MutationObserver((mutations, obs) => {
      const element = document.querySelector(selector);
      if (element) {
        resolve(element);
        obs.disconnect();
      }
    });

    observer.observe(document.documentElement, {
```

```

        childList: true,
        subtree: true
    });

    // Timeout protection
    setTimeout(() => {
        observer.disconnect();
        reject(new Error('Timeout'));
    }, timeout);
    });
}

```

Step 3: Observe Status Changes

Implements debouncing to prevent multiple triggers ^[14] ^[15]:

```

const debouncedHandler = debounce(handleResultChange, 2000);
observer.observe(resultPanel, {
    childList: true,
    subtree: true,
    characterData: true,
    attributes: true,
    attributeFilter: ['class', 'data-status']
});

```

Step 4: Extract Data on Acceptance

```

async function processAcceptedSubmission() {
    const data = {
        code: DOMParser.extractCode(),
        language: DOMParser.extractLanguage(),
        problemTitle: DOMParser.extractProblemTitle(),
        problemSlug: DOMParser.extractProblemSlug(),
        category: DOMParser.extractCategory(),
        timestamp: new Date().toISOString(),
        url: window.location.href
    };

    chrome.runtime.sendMessage({
        action: 'pushToGitHub',
        data
    });
}

```

2.3 GitHub API Integration

The extension implements the full GitHub Contents API workflow ^[18] ^[19] ^[20] ^[21] ^[29] ^[30].

Authentication

Option 1: Personal Access Token (Recommended)^[31]

Users generate a token at <https://github.com/settings/tokens> with repo scope and enter it in extension options.

Option 2: OAuth Flow^{[4] [5] [32] [33]}

```
async function handleGitHubAuth() {
  const redirectUrl = chrome.identity.getRedirectURL();
  const clientId = 'YOUR_CLIENT_ID';

  const authUrl = `https://github.com/login/oauth/authorize?` +
    `client_id=${clientId}&` +
    `redirect_uri=${encodeURIComponent(redirectUrl)}&` +
    `scope=repo`;

  const responseUrl = await chrome.identity.launchWebAuthFlow({
    url: authUrl,
    interactive: true
  });

  const code = new URL(responseUrl).searchParams.get('code');
  // Exchange code for token via backend service
}
```

Security Note: OAuth requires a backend service to exchange the authorization code for an access token, as the `client_secret` cannot be exposed in the extension^{[33] [34] [35] [36]}.

File Creation/Update Workflow

Simple Method: Contents API^[30]

```
async function createOrUpdateFile(owner, repo, path, content, message, sha = null) {
  const encodedContent = btoa(unescape(encodeURIComponent(content)));

  const body = {
    message,
    content: encodedContent,
    branch: 'main'
  };

  if (sha) {
    body.sha = sha; // Required for updates
  }

  const response = await fetch(
    `https://api.github.com/repos/${owner}/${repo}/contents/${path}`,
    {
      method: 'PUT',
      headers: {
        'Authorization': `token ${token}`,
        'Accept': 'application/vnd.github.v3+json'
      }
    }
  );
}
```

```

        },
        body: JSON.stringify(body)
    }
    );

    return response.json();
}

```

Advanced Method: Git Data API (for multiple files) [\[19\]](#) [\[20\]](#) [\[21\]](#) [\[37\]](#)

Seven-step process for atomic commits:

1. Get branch reference [\[19\]](#) [\[20\]](#)

```
GET /repos/:owner/:repo/git/refs/heads/:branch
```

2. Get commit

```
GET /repos/:owner/:repo/git/commits/:sha
```

3. Get tree

```
GET /repos/:owner/:repo/git/trees/:sha
```

4. Create blobs [\[19\]](#) [\[20\]](#)

```
POST /repos/:owner/:repo/git/blobs
{
  "content": "base64_encoded_content",
  "encoding": "base64"
}
```

5. Create tree [\[19\]](#) [\[20\]](#) [\[21\]](#)

```
POST /repos/:owner/:repo/git/trees
{
  "base_tree": "parent_tree_sha",
  "tree": [
    {
      "path": "file.py",
      "mode": "100644",
      "type": "blob",
      "sha": "blob_sha"
    }
  ]
}
```

6. Create commit [\[19\]](#) [\[20\]](#) [\[29\]](#)

```
POST /repos/:owner/:repo/git/commits
{
```

```

    "message": "Add solution for problem",
    "tree": "tree_sha",
    "parents": ["parent_commit_sha"]
}

```

7. Update reference ^[20] ^[21]

```

PATCH /repos/:owner/:repo/git/refs/heads/:branch
{
    "sha": "new_commit_sha"
}

```

2.4 File Organization Structure

Default organization follows LeetCode pattern ^[1] ^[2] ^[38]:

```

hackerrank/
├── algorithms/
│   ├── simple-array-sum.py
│   └── compare-triplets.java
├── data-structures/
│   ├── arrays-ds.cpp
│   └── 2d-array.js
└── sql/
    └── revising-select-query.sql

```

Customizable structure using templates:

```

const structure = 'hackerrank/{category}/{filename}';
// Variables: {category}, {filename}, {slug}, {language}

```

Each file includes metadata header:

```

"""
Problem: Simple Array Sum
Language: python3
Submitted: 2025-11-02 12:41:00
HackerRank URL: https://www.hackerrank.com/challenges/simple-array-sum/problem

Auto-synced by HackPush
"""

def simpleArraySum(ar):
    return sum(ar)

```

3. Manifest V3 Specific Considerations

3.1 Service Worker Lifecycle^{[11] [39] [10] [40] [12]}

Manifest V3 replaced persistent background pages with service workers that automatically terminate when idle^{[11] [10]}:

Challenges:

- Service workers sleep after ~30 seconds of inactivity
- Cannot use DOM APIs or window object^[10]
- Must use async patterns for all operations
- Storage is only via `chrome.storage` API

Solutions:

Message Passing^[10]

```
// content.js
chrome.runtime.sendMessage({action: 'pushToGitHub', data}, (response) => {
  console.log('Response:', response);
});

// background.js
chrome.runtime.onMessage.addListener((request, sender, sendResponse) => {
  if (request.action === 'pushToGitHub') {
    handlePush(request.data)
      .then(result => sendResponse({success: true, result}))
      .catch(error => sendResponse({success: false, error: error.message}));
    return true; // Required for async response
  }
});
```

ES Modules^[41]

```
{
  "background": {
    "service_worker": "background.js",
    "type": "module"
  }
}
```

```
// background.js
import { GitHubAPI } from './github-api.js';
import { StorageManager } from './storage.js';
```

Keep-Alive Pattern (Workaround)^[39]

```

let keepAliveInterval;

function keepAlive() {
  keepAliveInterval = setInterval(() => {
    chrome.runtime.getPlatformInfo(); // Lightweight API call
  }, 20000);
}

self.addEventListener('activate', keepAlive);

```

3.2 Storage Management

Chrome Storage API [\[7\]](#) [\[34\]](#) [\[35\]](#)

```

// Save configuration
chrome.storage.local.set({
  github_token: 'ghp_...',
  github_repo: 'username/repo',
  branch: 'main',
  file_structure: 'hackerrank/{category}/{filename}'
});

// Retrieve configuration
chrome.storage.local.get(['github_token', 'github_repo'], (result) => {
  const {github_token, github_repo} = result;
});

// Save submission history
chrome.storage.local.get(['submissions'], (result) => {
  const submissions = result.submissions || [];
  submissions.push({
    problemSlug: 'two-sum',
    timestamp: Date.now(),
    githubUrl: 'https://github.com/...'
  });
  chrome.storage.local.set({submissions});
});

```

Security Considerations [\[34\]](#) [\[31\]](#) [\[35\]](#) [\[36\]](#)

⚠ **Tokens in `chrome.storage.local` are NOT encrypted**

Best practices:

1. Warn users about token security
2. Use short-lived tokens when possible
3. Implement token refresh mechanism
4. Clear tokens on logout
5. Never log tokens to console

```
// Token validation
async function validateToken(token) {
  try {
    const response = await fetch('https://api.github.com/user', {
      headers: {'Authorization': `token ${token}`}
    });
    return response.ok;
  } catch {
    return false;
  }
}
```

3.3 Content Security Policy

Manifest V3 enforces strict CSP ^[4] ^[5]:

- No inline scripts
- No `eval()` or `Function()` constructors
- No remote code loading
- All scripts must be local files

4. User Interface Components

4.1 Extension Popup

popup.html

```
<html>
<head>
  <link rel="stylesheet" href="popup.css">
</head>
<body>
  <div>
    <h1>HackPush</h1>
    <div></div>

    <div>
      <p>✔ Connected to <strong></strong></p>
      <p>Synced: <strong>0</strong> submissions</p>
      <button id="view-history">View History</button>
      <button id="disconnect">Disconnect</button>
    </div>

    <div>
      <p>Connect HackPush to your GitHub repository</p>
      <button id="setup">Get Started</button>
    </div>
  </div>
</body>
</html>
```

```
</body>
</html>
```

popup.js

```
document.addEventListener('DOMContentLoaded', async () => {
  const status = await checkConnection();

  if (status.connected) {
    showConnected(status);
  } else {
    showNotConnected();
  }
});

document.getElementById('setup').addEventListener('click', () => {
  chrome.runtime.openOptionsPage();
});

async function checkConnection() {
  return new Promise((resolve) => {
    chrome.runtime.sendMessage({action: 'testConnection'}, (response) => {
      resolve(response);
    });
  });
}
```

4.2 Options Page

options.html

```
<html>
<head>
  <link rel="stylesheet" href="options.css">
</head>
<body>
  <div>
    <h1>HackPush Configuration</h1>

    <section>
      <h2>GitHub Settings</h2>
      <form id="config-form">
        <label>
          Personal Access Token
          <input type="password" id="token" required>
          <small>
            Generate at
            <a href="https://github.com/settings/tokens">
              github.com/settings/tokens
            </a>
            with 'repo' scope
          </small>
        </label>
      </form>
    </section>
  </div>
</body>
</html>
```

```

        </label>

        <label>
            Repository (owner/repo)
            <input type="text" id="repo" placeholder="username/hackerrank-solutions" rec
        </label>

        <label>
            Branch
            <input type="text" id="branch" value="main">
        </label>

        <label>
            File Structure
            <input type="text" id="structure" value="hackerrank/{category}/{filename}">
            <small>Variables: {category}, {filename}, {slug}, {language}</small>
        </label>

        <button type="submit">Save Configuration</button>
        <button type="button" id="test-connection">Test Connection</button>
    </form>
</section>

<section>
    <h2>Submission History</h2>
    <div></div>
</section>
</div>
<script src="options.js"></script>
</body>
</html>

```

5. Testing & Debugging

5.1 Testing Checklist

DOM Detection Tests:

- ✓ Test on problems with CodeMirror editor
- ✓ Test on problems with Monaco editor
- ✓ Test with different programming languages
- ✓ Test with different problem categories
- ✓ Verify language selector detection across UI updates

Submission Tests:

- ✓ Submit correct solution (should sync)
- ✓ Submit incorrect solution (should not sync)
- ✓ Submit multiple times quickly (prevent duplicates)

- ✓ Submit with slow network (wait for result)
- ✓ Submit and refresh page immediately

GitHub Integration Tests:

- ✓ Create new file
- ✓ Update existing file
- ✓ Test with invalid token
- ✓ Test with wrong repository
- ✓ Test with rate limits
- ✓ Verify commit messages format
- ✓ Check file path generation

5.2 Debugging Tools

Chrome DevTools

View service worker console:

1. Go to `chrome://extensions`
2. Enable "Developer mode"
3. Find HackPush extension
4. Click "service worker" link

View content script console:

1. Open DevTools on HackerRank page (F12)
2. Check Console tab for [HackPush] logs

Storage Inspector

```
// View stored data
chrome.storage.local.get(null, (items) => {
  console.log('All storage:', items);
});

// Clear storage
chrome.storage.local.clear();
```

Message Logging

```
// Log all messages
chrome.runtime.onMessage.addListener((request, sender, sendResponse) => {
  console.log('[DEBUG] Message received:', {
    action: request.action,
    sender: sender.tab?.url,
    data: request.data
  });
});
```

```
});  
});
```

5.3 Common Issues & Solutions

Issue 1: Service worker goes to sleep

- **Solution:** Implement keep-alive or redesign for stateless operation

Issue 2: CodeMirror not detected

- **Solution:** Add `waitForElement` with timeout, implement fallbacks

Issue 3: Multiple submissions synced

- **Solution:** Track processed submissions with Set, use unique IDs

Issue 4: GitHub rate limiting

- **Solution:** Implement exponential backoff, queue submissions

Issue 5: Token expiration

- **Solution:** Validate token before each request, prompt for renewal

6. Deployment & Distribution

6.1 Chrome Web Store Submission

Prerequisites:

1. Developer account (\$5 one-time fee)
2. Privacy policy (required for extensions requesting OAuth)
3. Detailed description and screenshots
4. App icons (16x16, 48x48, 128x128)

Submission Steps:

1. Create ZIP of extension directory
2. Upload to Chrome Web Store Developer Dashboard
3. Fill in store listing details
4. Submit for review (1-3 days)

Review Guidelines:

- Clear description of functionality
- Justification for all permissions
- Privacy policy for data handling
- No obfuscated code

6.2 Firefox Add-ons

Manifest Compatibility:

Firefox supports Manifest V2 and V3 differently^[11]:

- Use `browser` namespace instead of `chrome`
- Background scripts supported, service workers optional
- Some APIs differ slightly

Polyfill:

```
if (typeof browser === "undefined") {  
  globalThis.browser = chrome;  
}
```

6.3 Local Development

Load Unpacked Extension:

1. Navigate to `chrome://extensions`
2. Enable "Developer mode"
3. Click "Load unpacked"
4. Select HackPush directory

Hot Reload:

- Manifest changes require reload
- Content scripts reload on page refresh
- Service worker changes require extension reload

7. Future Enhancements

7.1 Planned Features

1. Multi-Platform Support

- LeetCode integration
- CodeForces sync
- AtCoder support

2. Advanced Git Operations

- Branch selection per category
- Tag releases for milestones
- Pull request creation

3. Analytics Dashboard

- Problems solved by language
- Daily/weekly streaks
- Difficulty distribution charts

4. Social Features

- Share solutions (with permission)
- Compare with friends
- Leaderboard integration

5. Code Quality

- Linting before push
- Format with Prettier
- Add test cases to README

7.2 Architecture Improvements

1. Robust Token Management

- Implement token refresh flow
- Encrypted storage option
- Multi-account support

2. Offline Support

- Queue submissions when offline
- Sync when connection restored
- Local SQLite database

3. Performance Optimization

- Batch multiple submissions
- Compress large code files
- Cache API responses

8. Contributing & Open Source

8.1 Repository Structure

```
HackPush/  
├── .github/  
│   ├── workflows/  
│   │   └── ci.yml (automated testing)  
│   └── ISSUE_TEMPLATE/  
├── docs/  
│   ├── ARCHITECTURE.md  
│   ├── API.md  
│   └── CONTRIBUTING.md
```

```
├── tests/
│   ├── unit/
│   └── integration/
├── scripts/
│   ├── build.js
│   └── package.js
```

8.2 Development Scripts

package.json

```
{
  "scripts": {
    "build": "webpack --config webpack.config.js",
    "watch": "webpack --watch",
    "test": "jest",
    "lint": "eslint src/",
    "format": "prettier --write src/",
    "package": "node scripts/package.js"
  },
  "devDependencies": {
    "webpack": "^5.0.0",
    "eslint": "^8.0.0",
    "prettier": "^2.0.0",
    "jest": "^29.0.0"
  }
}
```

8.3 Contribution Guidelines

1. **Fork & Clone**
2. **Create Feature Branch:** `git checkout -b feature/new-feature`
3. **Write Tests:** Cover new functionality
4. **Lint & Format:** Run `npm run lint && npm run format`
5. **Submit PR:** With clear description

9. Security & Privacy

9.1 Data Collection

What HackPush Collects:

- GitHub token (stored locally)
- Repository configuration (stored locally)
- Submission metadata (problem name, timestamp)
- Code content (only when submitting to GitHub)

What HackPush Does NOT Collect:

- Personal information
- Browsing history
- Usage analytics (unless opted-in)
- Third-party tracking

9.2 Permissions Justification

Permission	Purpose
storage	Save GitHub token and configuration
identity	OAuth authentication flow
activeTab	Read code from HackerRank page
scripting	Inject content scripts dynamically
https://www.hackerrank.com/*	Monitor submissions
https://api.github.com/*	Push code to GitHub

9.3 Security Best Practices

For Users:

1. Use token with minimal scope (repo only)
2. Regularly rotate tokens
3. Review synced code before sharing repository
4. Use private repositories for practice code

For Developers:

1. Never log tokens or sensitive data
2. Validate all user inputs
3. Use HTTPS for all API calls
4. Implement rate limiting
5. Follow OWASP guidelines

10. Troubleshooting Guide

10.1 Common Errors

Error: "GitHub not configured"

- **Cause:** Missing token or repository in settings
- **Solution:** Go to options page and complete setup

Error: "Could not extract code from editor"

- **Cause:** Editor not detected or unsupported type
- **Solution:** Check browser console for DOM errors, report issue with page URL

Error: "GitHub API error: 401"

- **Cause:** Invalid or expired token
- **Solution:** Generate new token and update in settings

Error: "Service worker inactive"

- **Cause:** Service worker went to sleep
- **Solution:** Click extension icon to wake it up, or reload extension

10.2 FAQ

Q: Does HackPush work on other coding platforms?

A: Currently only HackerRank. LeetCode support planned for future release.

Q: Can I customize the file structure?

A: Yes, in options page you can define custom templates using variables like `{category}`, `{language}`, etc.

Q: What happens if I solve the same problem twice?

A: The extension will update the existing file with the new solution.

Q: Is my GitHub token secure?

A: Tokens are stored in browser's local storage. While not encrypted, they're only accessible to the extension and not sent to any third party.

Q: Can I sync past submissions?

A: No, the extension only syncs new submissions made after installation. Manual upload required for past solutions.

11. References & Resources

Technical Documentation

- [42] HackerRank Platform - <https://www.hackerrank.com>
- [4] Chrome Manifest V3 Guide - <https://developer.chrome.com/docs/extensions/mv3/intro/>
- [1] LeetCode Extension - <https://github.com/QasimWani/LeetCode>
- [18] GitHub REST API - <https://docs.github.com/en/rest>
- [14] MutationObserver API - <https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver>
- [7] Chrome Extension Manifest Format - <https://developer.chrome.com/docs/extensions/reference/manifest>
- [24] CodeMirror Documentation - <https://codemirror.net/docs/>

Community Resources

- Chrome Extensions Discord
- Stack Overflow: [google-chrome-extension] tag
- GitHub Discussions: Browser Extension Development

Appendix: Complete Code Examples

All complete code examples, including `storage.js`, `popup.js`, and `options.js`, are available in the project repository.

Storage Manager (`storage.js`):

```
export class StorageManager {
  static async getConfig() {
    return new Promise((resolve) => {
      chrome.storage.local.get(
        ['github_token', 'github_repo', 'branch', 'file_structure'],
        (result) => { resolve(result) }
      );
    });
  }

  static async saveConfig(config) {
    return new Promise((resolve) => {
      chrome.storage.local.set(config, resolve);
    });
  }

  static async addSubmissionRecord(record) {
    const {submissions = []} = await new Promise((resolve) => {
      chrome.storage.local.get(['submissions'], resolve);
    });

    submissions.push(record);

    return new Promise((resolve) => {
      chrome.storage.local.set({submissions}, resolve);
    });
  }
}
```

```

static async getSubmissions() {
  return new Promise((resolve) => {
    chrome.storage.local.get(['submissions'], (result) => {
      resolve(result.submissions || []);
    });
  });
}

static async clearAll() {
  return new Promise((resolve) => {
    chrome.storage.local.clear(resolve);
  });
}
}

```

Conclusion

HackPush provides a seamless, automated solution for syncing HackerRank solutions to GitHub, leveraging modern browser extension APIs and robust DOM parsing techniques. By following this comprehensive guide, developers can build, customize, and deploy HackPush to enhance their coding practice workflow.

Key Takeaways:

- ✓ Manifest V3 requires service workers and async patterns
- ✓ MutationObserver is essential for dynamic page monitoring
- ✓ GitHub API integration enables automated commits
- ✓ Proper error handling and security practices are critical
- ✓ User feedback through notifications improves UX

Next Steps:

1. Clone the repository and explore the code
2. Test on various HackerRank problems
3. Customize for your workflow
4. Contribute enhancements back to the community

Happy coding! 🚀

[43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79]

✴✴

1. <https://chromewebstore.google.com/detail/leethub-v2/mhanfgfagplhgemhjfeolkdidbakocm?hl=en>
2. <https://subramanyarao.hashnode.dev/effortless-problem-solving-save-your-leetcode-submissions-with-one-click>
3. <https://addons.mozilla.org/en-US/firefox/addon/leethub-2-0-for-firefox/>
4. https://dev.to/jaymalli_programmer/unlocking-g-suite-sso-in-your-chrome-extension-the-definitive-manifest-v3-guide-2ghi
5. <https://stackoverflow.com/questions/72514608/chrome-extension-manifest-v3-mv3-authentication>

6. <http://developerlife.com/2023/08/11/chrome-extension-shortlink/>
7. <https://www.freecodecamp.org/news/how-to-build-a-chrome-extension-using-javascript-and-manifest-v3/>
8. <https://developer.chrome.com/docs/extensions/reference/manifest>
9. <https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json>
10. <https://www.sendeco2.com/base/resources/javascript/CodeMirror2/manual.html>
11. <https://stackoverflow.com/questions/72982051/how-to-get-the-text-value-of-a-codemirror-6-editor>
12. <https://codemirror.net/3/doc/manual.html>
13. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Anatomy_of_a_WebExtension
14. <https://substack.com/home/post/p-115346370>
15. <https://stackoverflow.com/questions/73615527/javascript-mutation-observer-not-working-in-google-chrome-extension>
16. <https://developer.chrome.com/blog/detect-dom-changes-with-mutation-observers>
17. <https://developer.mozilla.org/en-US/docs/Web/API/MutationObserver>
18. <https://docs.github.com/rest/commits/statuses>
19. <https://docs.github.com/en/rest/git/commits>
20. <https://stackoverflow.com/questions/11801983/how-to-create-a-commit-and-push-into-repo-with-github-api-v3>
21. <https://github.com/orgs/community/discussions/166611>
22. <https://stackoverflow.com/questions/28008405/select-and-replace-code-from-hackerranks-codemirror>
23. <https://gist.github.com/0xdevalias/2fc3d66875dcc76d5408ce324824deab>
24. <https://www.youtube.com/watch?v=piO2qVn10wY>
25. <https://www.sivasoft.in/online-full-stack-ui-mern-classes-in-rajanna-sircilla.html>
26. https://www.reddit.com/r/chrome_extensions/comments/z67al0/how_to_have_backgroundpersistent_in_v3/
27. <https://www.youtube.com/watch?v=kuKfv-M3KfK>
28. <https://github.com/codewars/codewars.com/issues/1870>
29. <https://docs.github.com/en/rest/commits/commits>
30. <https://docs.github.com/rest/repos/contents>
31. <https://stackoverflow.com/questions/46645843/where-to-store-my-git-personal-access-token>
32. <https://xiegers.com/post/chrome-extension-oauth-web-auth-flow-firebase-google/>
33. <https://community.auth0.com/t/chrome-extension-manifest-v3-using-auth0-in-a-secure-manner/125433>
34. <https://www.answeroverflow.com/m/1223328118278983680>
35. <https://community.auth0.com/t/chrome-extension-advice/38887>
36. <https://github.com/please-openit/token-leak-extension>
37. <https://www.youtube.com/watch?v=nwHqXtk6LHA>
38. <https://github.com/raphaelheinz/LeetHub-3.0>
39. <https://docs.vertigisstudio.com/essentials/gyh/latest/SamplesViewer/Libraries/CodeMirror/manual.html>
40. <https://discuss.codemirror.net/t/get-new-codemirror-text-in-javascript/2398>
41. <https://discuss.yjs.dev/t/how-to-initialize-value-of-codemirror-binding-to-yjs/26>
42. <https://www.hackerrank.com>
43. https://www.youtube.com/watch?v=txFuucg_Tf0

44. <https://www.youtube.com/watch?v=s6m4H10B8rc>
45. <https://www.hackerrank.com/domains/tutorials/30-days-of-code>
46. <https://github.com/pocketbase/pocketbase/discussions/1899>
47. <https://www.youtube.com/watch?v=mRUhVnWdnI4>
48. <https://www.hackerrank.com/challenges/simple-text-editor/problem>
49. <https://www.youtube.com/watch?v=f2CuEMGvjT0>
50. <https://froala.com/?p=t&q=excel>
51. <https://www.hackerrank.com/domains/data-structures>
52. <https://building.lang.ai/hacking-with-dom-mutationobservers-348a50231580>
53. <https://groups.google.com/a/chromium.org/g/chromium-extensions/c/-ORVEejde44>
54. <https://docs.github.com/rest/using-the-rest-api/getting-started-with-the-rest-api>
55. <https://github.com/crxjs/chrome-extension-tools/issues/811>
56. <https://www.hackerrank.com/challenges/challenges/problem>
57. <https://developer.chrome.com/docs/extensions/get-started/tutorial/hello-world>
58. https://dev.to/artem_turlenko/simplifying-chrome-extension-development-with-github-oauth-55i6
59. <https://stackoverflow.com/questions/64085531/solution-logic-to-organizing-containers-of-balls-question-on-hackerrank>
60. <https://stackoverflow.com/questions/75911091/how-to-make-a-chrome-extension-pop-up-automatically-when-i-enter-a-certain-website>
61. <https://www.youtube.com/watch?v=Cv9C9wUIKks>
62. <https://gist.github.com/dhammikamare/96bb77d242d0d67f12fa>
63. <https://dev.to/kjdowns/chrome-extensions-manifest-file-47he>
64. <https://www.hackerrank.com/challenges/document-classification/problem>
65. <https://www.callstack.com/blog/how-to-build-a-chrome-extension>
66. <https://prepinsta.com/hackerrank/coding-questions-and-answers/>
67. <https://www.hackerrank.com/challenges/organizing-containers-of-balls/problem>
68. <https://github.com/QasimWani/LeetHub>
69. <https://www.hackerrank.com/blog/css-interview-questions-every-developer-should-know/>
70. <https://github.com/Ruslan-Shevyrev/HackerRankCSS>
71. <https://stackoverflow.com/questions/60591712/css-inheritance-problems>
72. <https://www.hackerrank.com/skills-verification/css>
73. <https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/background>
74. <https://codemirror.net/2/doc/manual.html>
75. <https://developer.chrome.com/docs/extensions/develop/migrate/to-service-workers>
76. <https://github.com/GoogleChrome/chrome-extensions-samples/issues/528>
77. <https://stackoverflow.com/questions/70715942/chrome-extension-manifest-v3-libraries-in-background>
78. <https://developer.chrome.com/docs/extensions/reference/manifest/background>
79. <https://stackoverflow.com/questions/43528075/how-to-use-javascript-in-hackerrank-and-hackerearth>

