GigProof – Stage 1 MVP

Chrome Extension Income Summary Generator

0. PURPOSE OF THIS MVP (READ FIRST)

What problem this MVP solves

Gig workers struggle to prove income because:

earnings are spread across platforms

screenshots look unprofessional

lenders/landlords don’t know what they’re looking at

What this MVP does

This MVP helps gig workers:

take the earnings numbers they already see

turn them into a clean, professional PDF

download and share it themselves

What this MVP does NOT do

It does NOT verify income

It does NOT contact lenders

It does NOT connect to banks

It does NOT log into accounts

It does NOT make decisions

This is presentation, not authority.

1. OVERALL SYSTEM (BIG PICTURE)

There are three parts only:

1. Chrome Extension → reads visible earnings data
2. Backend API → receives data, creates PDF
3. Website → login, payment, downloads

Nothing else.

2. HOW THE CHROME EXTENSION WORKS (CORE MECHANIC)

Critical rule

The extension never logs in.

The user logs in manually.

The extension only reads what is already visible on screen.

This is non-negotiable.

What “scraping” means here

In this MVP:

“Scraping” = reading text from the webpage DOM after the user is logged in.

It is equivalent to:

copying numbers by hand

pasting them into a document

But automated and formatted.

3. EXTENSION FLOW (USER EXPERIENCE)

Step-by-step user flow

User installs Chrome extension

User logs into a gig platform normally (Uber, DoorDash, etc.)

User navigates to the Earnings page

User clicks the extension icon

User clicks “Generate Income Report”

Extension reads earnings shown on page

Data is sent to backend

Backend generates PDF

User downloads PDF

Nothing happens automatically.

4. EXTENSION FILE STRUCTURE (EXACT)

Create a folder called:

gigproof-extension/

Inside it, create exactly these files:

manifest.json
contentScript.js
popup.html
popup.js
background.js

Do NOT add more files at this stage.

5. manifest.json (WHERE THE EXTENSION IS ALLOWED TO RUN)

This file controls permissions and safety.

```
{
  "manifest_version": 3,
  "name": "GigProof",
  "version": "0.1",
  "description": "Generate income summaries from gig platforms",
  "permissions": ["activeTab", "scripting"],
```

```
  "host_permissions": [
    "https://drivers.uber.com/*",
    "https://www.doordash.com/*"
  ],
  "action": {
    "default_popup": "popup.html"
  },
  "content_scripts": [
    {
      "matches": [
        "https://drivers.uber.com/*",
        "https://www.doordash.com/*"
      ],
      "js": ["contentScript.js"]
    }
  ]
}
```

What this means

The extension ONLY runs on gig sites

Chrome shows users these permissions

You are not spying on other pages

6. contentScript.js (THIS IS WHERE DATA IS READ)

This script runs inside the earnings page.

6.1 Mandatory Mock Mode (DO THIS FIRST)

You must build Mock Mode so you can test without real earnings.

```
const MOCK_MODE = true;

function extractEarnings() {
  if (MOCK_MODE) {
    return {
      platform: "Uber",
      dateRange: "Jan 1 – Jan 31, 2026",
      grossEarnings: "$4,213.87",
      currency: "USD"
    };
  }

  // real extraction later
}
```

Mock Mode allows:

full testing

PDF generation

payments

downloads

Before real scraping.

6.2 Real DOM extraction (after Mock Mode works)

Example for Uber:

```
function extractEarnings() {
  const gross = document.querySelector(
    '[data-testid="total-earnings"]'
  )?.innerText;

  const dateRange = document.querySelector(
    '[data-testid="date-range"]'
  )?.innerText;

  if (!gross || !dateRange) {
    throw new Error("Earnings not found. Please open the earnings page.");
  }

  return {
    platform: "Uber",
    dateRange,
    grossEarnings: gross,
    currency: "USD"
  };
}
```

This:

reads visible elements

does not access cookies

does not intercept network calls

7. popup.html (EXTENSION UI)

```
<!DOCTYPE html>
<html>
  <body>
    <h3>GigProof</h3>
    <button id="generate">Generate Income Report</button>
    <p id="status"></p>
    <script src="popup.js"></script>
  </body>
</html>
```

Simple is correct.

8. popup.js (USER ACTION → DATA EXTRACTION)

```
document.getElementById("generate").addEventListener("click", () => {
  document.getElementById("status").innerText = "Generating…";

  chrome.tabs.query({ active: true, currentWindow: true }, (tabs) => {
    chrome.tabs.sendMessage(tabs[0].id, { action: "EXTRACT_EARNINGS" });
  });
});
```

9. background.js (SEND DATA TO BACKEND)

```
chrome.runtime.onMessage.addListener((message, sender, sendResponse) => {
  if (message.type === "EARNINGS_DATA") {
    fetch("https://your-backend.com/api/report", {
      method: "POST",
```

```
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(message.payload)
  })
    .then(res => res.json())
    .then(data => {
      chrome.runtime.sendMessage({
        type: "REPORT_READY",
        url: data.url
      });
    });
 }
});
```

10. BACKEND (MINIMAL)

What backend does

receives JSON

validates fields

generates PDF

stores PDF

returns URL

What backend does NOT do

no scraping

no bank access

no verification

no lender logic

Example endpoint

```
app.post("/api/report", async (req, res) => {
  const { platform, dateRange, grossEarnings } = req.body;

  if (!platform || !grossEarnings) {
    return res.status(400).send("Invalid data");
  }

  const pdfUrl = await generatePDF(req.body);

  res.json({ url: pdfUrl });
});
```

11. PDF CONTENT (VERY IMPORTANT)

Allowed language

“Income Summary”

“User-generated report”

“Based on account information visible to the user”

Forbidden language

❌ Verified

❌ Official

❌ Certified

❌ Guaranteed

❌ Bank-verified

Mandatory disclaimer

Add this to every PDF:

This document is a user-generated income summary created at the request of the account holder.

GigProof does not verify, certify, or guarantee the accuracy of this information.

12. WEBSITE (MINIMUM)

Pages required

Landing page

Login/signup

Dashboard

Pricing

Download page

Pricing example

1 PDF → $9.99

10 PDFs → $19.99

Payment must happen before download.

13. TESTING WITHOUT GIG ACCOUNTS

You test using:

Mock Mode (primary)

Local fake earnings HTML pages

2–3 real testers later

You do NOT:

create fake gig accounts

bypass ToS

automate logins

14. DEFINITION OF DONE (STAGE 1)

You are finished when:

User opens earnings page

User clicks extension button

Earnings are extracted

Backend receives JSON

PDF is generated

Payment unlocks download

User can delete report

15. WHAT YOU DO NOT BUILD IN STAGE 1

Do NOT build:

Plaid

Bank integrations

Lender dashboards

Employer verification

APIs for third parties

Continuous syncing

Those are future stages only.

FINAL SUMMARY (READ THIS TWICE)

You are not hacking accounts

You are not scraping banks

You are not verifying income

You are formatting user-visible data

You are aligned with the original validation

This MVP is buildable and sellable