TechSolutions Intranet

Flowcharts, Pseudocode and Prototype Workbook

Total time: 5 hours

- Part 1: 2 hours – Flowcharts and pseudocode (main focus)
- Part 2: 3 hours – HTML, CSS and JavaScript prototype

Assessment focus:

- Pseudocode carries more marks than code
- Clear logic and validation are more important than complexity

Scenario overview

You work for a software development company that has secured a contract to build an **intranet system** for **TechSolutions Manufacturing**.

The company specialises in precision manufacturing equipment and needs an internal system to help maintenance teams work more efficiently.

The intranet should help staff to:

- log maintenance activities
- plan and schedule maintenance
- receive clear alerts about upcoming work

Key features you are designing for

You are **not building a full system**.

You are building a **prototype** that demonstrates logic and structure.

The key features are:

- Maintenance Log
- Maintenance Scheduler
- Notification or status messages

What you will produce by the end

By the end of this workbook, you will have:

- flowcharts for two features
- clear pseudocode written using the approved style
- a working intranet prototype page
- JavaScript that follows your pseudocode
- testing evidence including performance and stress checks
- a commit history that shows progress

Files and folders you must use

Create this folder structure in Visual Studio Code **exactly**:

TechSolutions_Intranet

- index.html
- readme.md
- docs
    - flowcharts.md
    - pseudocode.md
    - testing.md
- static
    - style.css
    - app.js
- Pseudocode Guide.pdf

If your folder structure is incorrect, fix it before continuing.

Important rules for this workbook

- You must **not copy answers from classmates**
- You must **not skip the pseudocode**
- You must **commit your work regularly**
- If your JavaScript does not match your pseudocode, the pseudocode is treated as incorrect
- Clear logic matters more than advanced features

Version control expectations (important)

You must commit your work to your repository regularly.

Minimum commits required:

- After completing flowcharts
- After completing pseudocode
- After building the HTML structure
- After adding JavaScript logic
- After completing testing

In `readme.md`, add a heading called **Commit Evidence** and list your commit messages in order.

PART 1: Flowcharts and Pseudocode

Time: 2 hours

This is the most important part of the workbook

What pseudocode is (simple explanation)

Pseudocode:

- describes what a program should do

- uses plain English and structured steps
- is not written in a real programming language
- helps plan logic before coding

You must follow the **Pseudocode Guide** provided.

Feature A: Maintenance Log

This feature allows a technician to record maintenance work.

Inputs required

- Technician name
- Machine ID
- Type of service
- Maintenance notes

Rules

- Technician name must not be blank
- Machine ID must not be blank
- A service type must be chosen
- Notes must be no longer than 200 characters

Outputs

- Clear error messages if something is wrong
- A confirmation message if the log is valid

Task 1: Flowchart for Maintenance Log

In `docs/flowcharts.md`:

Create a flowchart that shows the full process for the Maintenance Log feature.

Your flowchart must include:

- a clear Start and End
- input steps
- at least two decision points
- at least one error path
- one success path

You may draw this digitally or describe each step clearly in text.

Task 2: Pseudocode for Maintenance Log

In `docs/pseudocode.md`:

Write pseudocode for the Maintenance Log feature.

You must include:

- input statements for each field
- validation checks
- output statements for errors
- a success output that summarises the entry

Your pseudocode must be readable by another student.

Feature B: Maintenance Scheduler

This feature checks whether maintenance is due.

To keep things simple, you will not calculate real dates.

Instead, you will use:

- Days since last service (number)
- Service frequency (weekly, monthly, quarterly)

Rules

- Days must be 0 or more

- Frequency must be chosen
- Weekly is due at 7 days
- Monthly is due at 30 days
- Quarterly is due at 90 days

You should also include a **“due soon”** state just before each threshold.

Task 3: Flowchart for Maintenance Scheduler

In `docs/flowcharts.md`:

Create a flowchart that shows:

- input of days and frequency
- validation of the number
- decision logic for due now, due soon, or not due

Your flowchart must clearly show only **one final output** per run.

Task 4: Pseudocode for Maintenance Scheduler

In `docs/pseudocode.md`:

Write pseudocode that:

- reads both inputs
- validates them
- applies the correct rules
- outputs a clear status message

Your logic must match your flowchart.

Reflection checkpoint

Answer these in `docs/pseudocode.md`:

1. Which feature was easier to plan, and why?
2. What is one mistake a user might make?
3. Which validation rule prevents that mistake?

PART 2: Build the Intranet Prototype

Time: 3 hours

Step 1: Build the HTML structure

In `index.html`, create a page with:

- header
- main
- footer

You must use correct heading order:

- one h1
- h2 for each feature section

Section 1: Maintenance Log (HTML only)

Your page must include:

- a form
- inputs for all Maintenance Log fields
- labels that match inputs
- a button to save the log
- an area for messages
- an area for a summary

Section 2: Maintenance Scheduler (HTML only)

Your page must include:

- inputs for days since last service
- a dropdown for frequency
- a button to check the schedule
- an area for status messages

Step 2: Add basic CSS

In `static/style.css`:

- improve readability
- add spacing
- use borders to separate sections

Do not over-style the page.

Step 3: Add JavaScript using your pseudocode

In `static/app.js`:

- add event listeners
- read form inputs
- validate inputs exactly as written in your pseudocode
- show clear error messages
- show clear success messages

Rule:

You must **not add logic that is not in your pseudocode**.

Testing and Evidence

All testing evidence goes in `docs/testing.md`.

Functional testing

Include:

- 2 normal tests
- 2 invalid tests
- 2 boundary tests

For each test, record:

- input data
- expected result
- actual result
- fix made
- re-test result

6.5.2 Simple Performance Check (Page Load)

You must test the hosted version of your site.

What to do:

- Open your GitHub Pages link
- Go to https://webpagetest.org/
- Run a Quick Test on Desktop
- Run a Quick Test on Mobile

Record:

- load time
- largest contentful paint
- any warnings

Add screenshots and a short written note.

6.5.3 Browser Console Stress-Check

You must check your page does not break under repeated actions.

What to do:

- Open your hosted site
- Open Developer Tools
- Run a console loop that clicks your buttons repeatedly

Record:

- what you tested
- what you expected
- what happened
- what you fixed

Only use test data.

Final checklist

Before you submit, check:

- flowcharts exist
- pseudocode is complete and readable
- JavaScript matches pseudocode
- validation works
- testing evidence is complete
- commits show progress

Final reminder

This workbook is not about being clever.

It is about:

- planning clearly
- writing logic carefully
- proving your solution works

Strong pseudocode leads to strong marks.