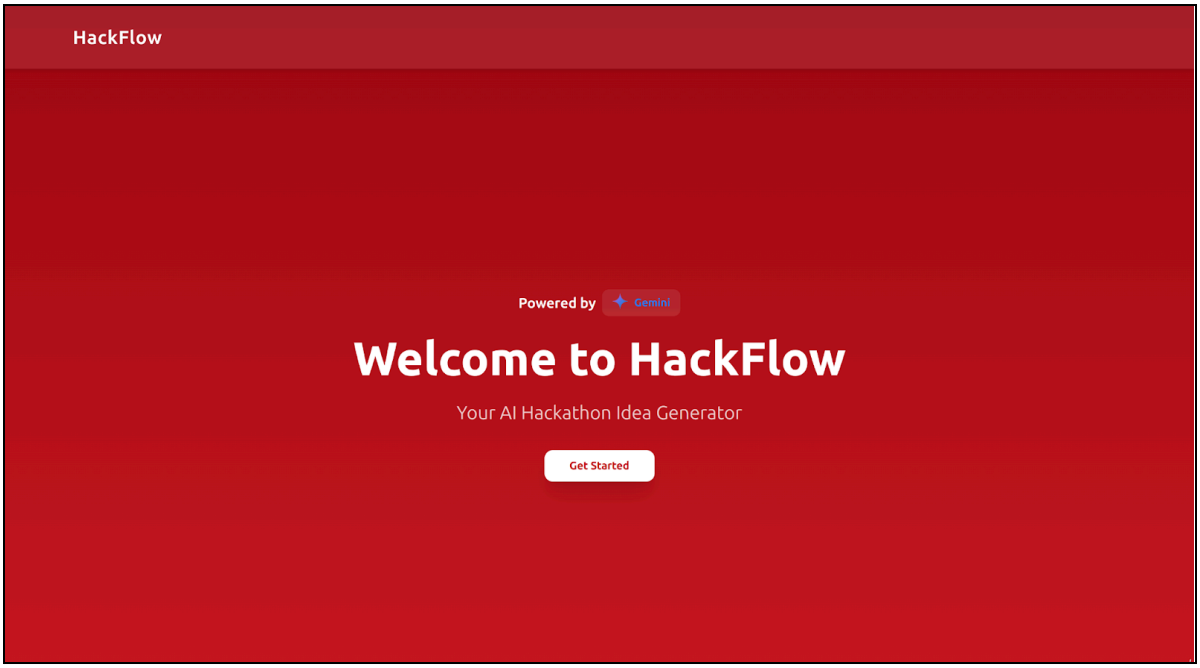
The Guide Handbook - Workshop 1

Table Of Contents



1.0 Pre-Event Software Installation Setup

1. Windows

Software	Installation Link	YouTube guide
GIT	Git - Install for Windows	How to install Git on …
Node Js v22	https://nodejs.org/dist/v22.21.1/node-v22.21.1-x64.msi	How to Install NodeJ…
Pnpm	Run the following in the terminal: “npm install pnpm -g”	-
VScode	https://code.visualstudio.com/docs/?dv=win64user	How Install Visual St…

2. Mac

Software	Installation Link	YouTube guide
GIT	Git - Install for macOS	How to Install Git on …
Node Js v22	https://nodejs.org/dist/v22.21.1/node-v22.21.1.pkg	How To Download A…
Pnpm	Run the following in the terminal: “npm install pnpm -g”	-
VScode	https://code.visualstudio.com/docs/?dv=osx	How to Install Visual …

3. Linux

Software	Installation Link	YouTube guide
GIT	Git - Install for Linux	Git - Installation on L…
Node Js v22	https://nodejs.org/dist/v22.21.1/node-v22.21.1-linux-x64.tar.xz	How to Install Node.j…

Pnpm	Run the following in the terminal: “npm install pnpm -g”	-
VScode	https://code.visualstudio.com/docs/?dv=linux64_deb	How to Correctly Inst…

1.1 Common Troubleshooting Solutions:

- **Git**

  Git command not recognised. Install Git and restart the terminal or system.
  Name and email not set. Run git config --global user.name and user.email.
  Cannot clone repository. Check the URL and ensure internet access.
  Permission denied. Use HTTPS instead of SSH for the first setup.

- **VS Code**

  Code not saving. Enable auto save or press Ctrl S.
  Terminal missing. Open it from View then Terminal.
  The theme or layout looks broken. Reset layout from the Command Palette.
  File icons missing. Install a file icon theme extension.

- **pnpm**

  pnpm not recognised. Install Node first, then enable pnpm with corepack enabled.
  Install command fails. Run pnpm install inside the project folder.
  Permission errors. Avoid sudo and reinstall Node correctly.
  Package not found. Check package name spelling.

- **Node.js**

  node or npm not recognised. Restart terminal after install.
  Version too old or too new. Install an LTS version of Node.
  Cannot run scripts. Ensure you are in the project directory.
  Unexpected errors on start. Reinstall Node and try again.

1.2 Additional Resources

- MDN Web Docs
- The Modern JavaScript Tutorial
- Node.js
- Svelte
- Simple collaboration from your desktop · GitHub

2.0 Workshop 1: Intro To SvelteKit

The following guide contains snippets of code that you can copy/paste or refer to if at any point you get lost during the workshop. Organised in somewhat chronological order. Enjoy :)

Summary

We'll provide a demo of what we'll accomplish at the end of the four-part series. In this first episode, we will learn what a modern web app is at a high level and where SvelteKit fits. We'll set up our local environment, clone the start project, and understand the minimum file structure needed to work productively. We will write our first Svelte components, use basic reactivity, and see immediate browser feedback. We'll also learn how file-based routing maps folders to URLs and how multiple pages are created. To put these things into practice, we will create the landing page and a nice footer.

---

1. Environment Setup

Before we begin, let's make sure we have the correct tools installed.

- **Node.js**: Open your terminal and verify your Node.js version. We recommend version 18 or higher.

  node -v

- **NPM**: Verify your npm version.

  npm -v

- **VS Code Extensions**: For the best development experience, we recommend installing the following extensions in VS Code:

  - Svelte for VS Code
  - Prettier - Code formatter
  - ESLint

- **Clone Project**: Let's clone the repository and navigate to the starting point for workshop 1.

  git clone <repo-url>

cd web_dev_series/workshop_1/start

- **Install Dependencies**:

  npm install

- **Run the App**:

  npm run dev

We should now see our application running at `http://localhost:5173`.

---

2. Project Structure Walkthrough

SvelteKit uses a file-based routing system. The `src` directory is the heart of our project.

- `src/routes/`: This is the most important folder. Every Svelte file in here becomes a page or a route in our application.
- `src/routes/+page.svelte`: This is the file for our home page (`/`).
- `src/routes/+layout.svelte`: This is a special file that defines the layout for all pages in the same folder and below. It allows us to have shared UI elements like a navigation bar or a footer.
- `src/lib/`: This folder is for our own components, utilities, and assets. Code in here can be easily imported into our routes.
- `static/`: This folder is for static assets that don't need to be processed, like `robots.txt` or a favicon.

---

3. $Runes and State

Svelte 5 introduces **Runes**, which are special symbols that control reactivity. Let's explore them together.

For this exercise, we can create a new file `src/routes/state/+page.svelte` to test this out.

`src/routes/state/+page.svelte`

```
<script lang="ts">
```

```
	// `$state` creates a reactive variable.

	// When its value changes, the UI will automatically update.

	let count = $state(0);

	let name = $state('Svelte');

	// `$derived` creates a new reactive value that depends on other
state variables.

	// `doubled` will automatically update whenever `count` changes.

	let doubled = $derived(count * 2);

	// `$effect` runs a piece of code whenever one of its dependencies
changes.

	// It's useful for side effects like logging or making API calls.

	$effect(() => {

		console.log(`Count changed to ${count}`);

	});

</script>

<div class="max-w-2xl mx-auto p-8 text-white">

	<h1 class="text-2xl font-bold mb-4">Runes Example</h1>

	<!-- Simple state binding -->

	<p class="mb-2">Hello {name}!</p>

	<input type="text" bind:value={name} class="p-2 rounded bg-gray-800
text-white w-full mb-4" />

	<!-- Derived state -->

	<p class="mb-2">Count: {count}</p>
```

```
      <p class="mb-4">Count doubled is: {doubled}</p>

      <!-- Event handler updating state -->

      <button

            on:click={() => count++}

            class="bg-blue-600 hover:bg-blue-700 text-white font-bold py-2
px-4 rounded"

      >

            Clicked {count} {count === 1 ? 'time' : 'times'}

      </button>

</div>
```

---

4. Routing

Creating new pages is as simple as creating new folders inside `src/routes`.

1. Let's create a new folder: `src/routes/about`
2. Now, let's create a new file inside it: `src/routes/about/+page.svelte`

`src/routes/about/+page.svelte`

```
<div class="p-8 text-white">

    <h1 class="text-3xl font-bold">About Us</h1>

    <p class="mt-4">This is the about page for the HackFlow
application.</p>

    <a href="/" class="text-blue-400 hover:underline mt-4 inline-block">Go
Home</a>

</div>
```

Now, if we navigate to `/about` in our browser, we will see this new page.

---

5. Shared Layouts

To avoid repeating UI elements like a navigation bar and footer on every page, we'll use a layout.

Let's update the main layout file to include our `Nav` and `Footer` components.

`src/routes/+layout.svelte`

```
<script lang="ts">

	import './layout.css';

	import favicon from '$lib/assets/favicon.svg';

	import Nav from '$lib/components/Nav.svelte';

	import Footer from '$lib/components/Footer.svelte';

	import { Container } from '$lib/components';

	let { children } = $props();

</script>

<svelte:head>

	<link rel="icon" href={favicon} />

	<title>HackFlow</title>

</svelte:head>

<Container>

	<Nav />

	<main>
```

```
            {@render children()}

      </main>

      <Footer />

</Container>
```

---

6. Landing Page and Footer Practice

Now let's build out the components for our landing page.

src/lib/components/Nav.svelte

This component will be our top navigation bar.

```
<nav class="fixed top-0 z-50 w-full bg-white/10 p-6 shadow-md
backdrop-blur-md">

      <div class="flex w-full items-center justify-between px-4 md:px-16">

            <p class="text-2xl font-bold tracking-wide
text-white">HackFlow</p>

      </div>

</nav>
```

src/lib/components/Landing.svelte

This is the main hero section for our home page.

```
<script>

      import Gemini from '$lib/assets/gemini.png';

      import Badge from './ui/badge/badge.svelte';

      import Button from './ui/button/button.svelte';
```

```
</script>

<section

      class="relative flex h-screen flex-col items-center justify-center
px-4 pt-24 text-center md:pt-32"

>

      <span class="mb-4 flex items-center gap-2 text-lg font-bold
text-white">

            Powered by

            <Badge class="flex items-center gap-2 rounded-md bg-white/10
px-3 py-1 backdrop-blur-sm">

                  <img src={Gemini} alt="Gemini icon" class="mb-1 h-5 w-5"
/>

                  <span class="text-sm font-bold
text-blue-500">Gemini</span>

            </Badge>

      </span>

      <h1 class="mb-4 text-5xl leading-tight font-extrabold text-white
md:text-6xl">

            Welcome to HackFlow

      </h1>

      <p class="mb-8 max-w-2xl text-xl text-white/80 md:text-2xl">Your AI
Hackathon Idea Generator</p>

      <a href="#chat">

            <Button

                  size="lg"
```

```
                  class="rounded-lg bg-white px-8 py-4 font-semibold
text-red-700 shadow-xl transition hover:scale-105 hover:bg-white/90"

            >

                  Get Started

            </Button>

      </a>

</section>
```

src/lib/components/Footer.svelte

And our site-wide footer.

```
<footer class="bg-red-600 p-8">

      <div class="mx-auto flex max-w-5xl items-center justify-between">

            <p class="font-semibold text-white">MERAH | HackFlow</p>

            <p class="text-white">Built by DONEWITHWORK</p>

      </div>

</footer>
```

src/routes/+page.svelte

Finally, let's assemble our landing page component on the home page.

```
<script lang="ts">

      import { Landing } from '$lib/components';

</script>

<Landing />

<section id="chat" class="flex w-full flex-col items-center px-4 py-24">
```

```
      <p class="mb-4 text-4xl font-semibold text-white">Generate Your
Magnum Opus</p>

      <p class="mb-12 max-w-xl text-center text-white/80">

            Fill in your hackathon details and let AI craft a project idea
tailored for your team.

      </p>

</section>
```

With these changes, our home page should now display a complete landing page with a navigation bar and footer. In the next workshop we will be working on forms and best practices to work with lots of data.