

AI in the SDLC - Full Service Development - Lovable Prompts

Executive Summary

Tool	Time (minutes)	Productivity Change (%)	Prompts Required	Successful prompts (%)
Manual	480	-	-	-
Lovable	100	+ 79%	10	70% (7)
Repl.it	60	+ 87%	15	73% (11)
V0	90	+ 81%	14	57% (8)

Introduction

Before conducting the practical activity of recreating the FaCT service using SaaS AI development products, an investigation into an ideal starting prompt was undertaken to be used as the initial prompt across all tools. Dependent on the output from the SaaS tools, subsequent prompts were tailored to either remedy errors from the outputs, steer the tools to use the technology stack in use by the current FaCT front end, or iterate on the output to implement new features and functionality.

Starting Prompt

“Please create a full-featured web application that exactly replicates the UK “Find a court or tribunal” service as seen on the GOV.UK website.

The application must meet the following requirements: Overall Purpose and Look & Feel: The website should mirror the official GOV.UK service for finding courts and tribunals.

It must use the UK Government Design System (GDS) for all visual elements, ensuring that the styling, fonts, and layout are identical to those used on GOV.UK.

The design must follow accessibility standards (WCAG 2.1), meaning that all text is clear, navigable, and easy to read, with proper contrast and keyboard accessibility.

Functionality: Homepage: The main page should feature a clear and prominent search bar where users can enter a town name or postcode. The page should include a header (with GOV.UK branding) and a footer with standard links (like Help, Contact, and Terms and Conditions).

Search Results Page: When a user enters a search term, the website should display a list of courts or tribunals that match the query. Each result should show the court's name and its address in a neat, easy-to-read list. If no results are found, the user should see a friendly message suggesting they try a different search.

Court Details Page: When a user clicks on a specific court from the results, the website should show a dedicated details page. This page must include: The court's full name and address. Opening hours (e.g., Monday to Friday, 9am to 5pm). Contact details, including a phone number and an email address. A list of the types of cases or areas of law that the court handles.

Mock Data: The website should operate with sample (mock) data. Although there is no need for a live data connection or database, the information shown must look realistic and mimic what users would see on the actual service.

Technology and Structure (Described Simply): The application should be built using a modern web server that uses TypeScript for clarity and safety. It should render web pages on the server before sending them to the user, ensuring the content is fully available when the page loads.

The pages should be created using Nunjucks templates (a tool for designing HTML pages) to keep the layout consistent with the GDS standards. The backend (server) will use Express, a framework that manages the website's navigation and pages, while the front-end will display the pages as traditional HTML styled according to GOV.UK guidelines.

Local Running: The finished website should be able to run on a local computer for testing purposes, meaning no deployment to a live server is necessary. It should be organized in a clear and logical structure so that someone can start it locally with minimal setup.

Exact Replication: The goal is to have the website behave and appear as close as possible to the actual GOV.UK court finder. This includes the same layout, branding, and even the same manner of displaying search results and court details.

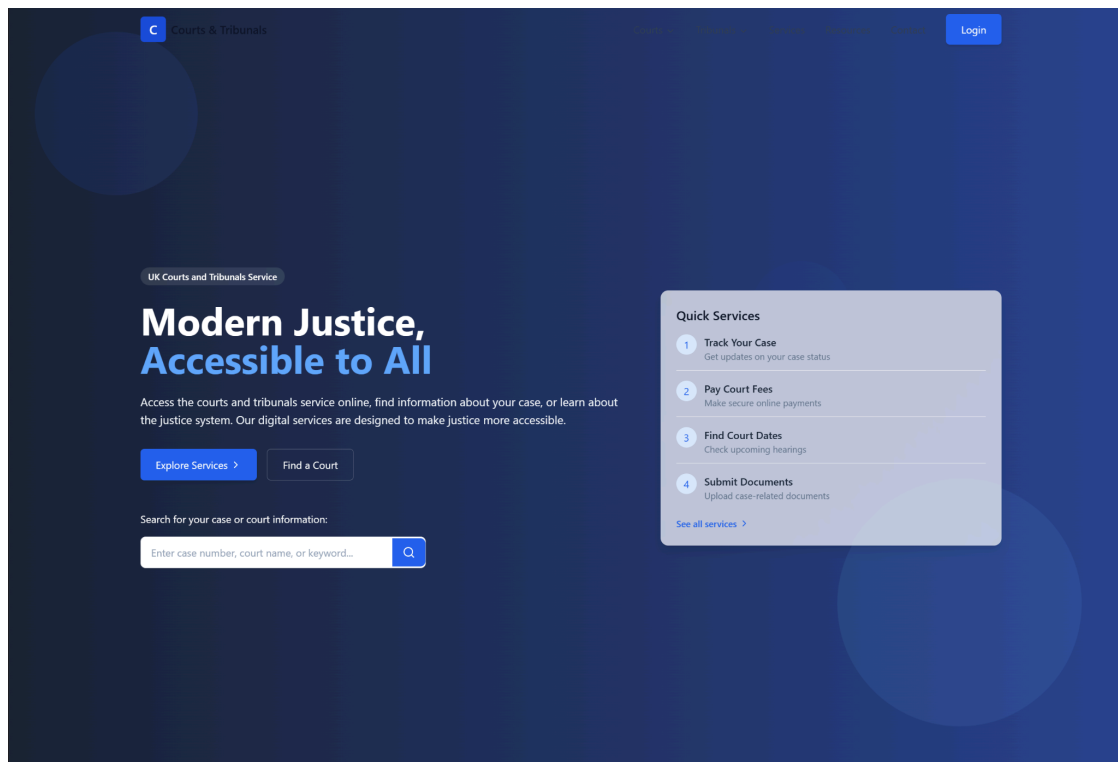
All interactive elements (like the search bar and navigation links) should work in a way that users expect based on their experience with GOV.UK. Please ensure that every aspect of the service—from the homepage and search functionality to the detailed court pages—is designed exactly like the official GOV.UK "Find a court or tribunal" service, using the UK Government Design System and following best practices for accessibility and usability. You must use the nunjucks components provided by the government design system for the front end"

Starting Prompt Breakdown

After experimenting with a variety of starting prompts ranging from The very basic and vague *“Please create a replica of the Find a Courts and Tribunal Service”* To the more complex prompt which detailed every page navigation, page content as an attempt to build the full end-to-end front end navigation in 1 prompt, the above prompt was settled upon. The basic prompt either loosely adhered to its own idea of the GDS, whereas the approach to build the full e2e in 1 prompt lost track of routing logic and page routing. This introduced a lot of bugs, compilation errors, type conflicts, and incorrect styling. Subsequent prompts for these approaches were focused on fixing the errors and flawed logic, or steering the output to fit within the Government Design System Standard.

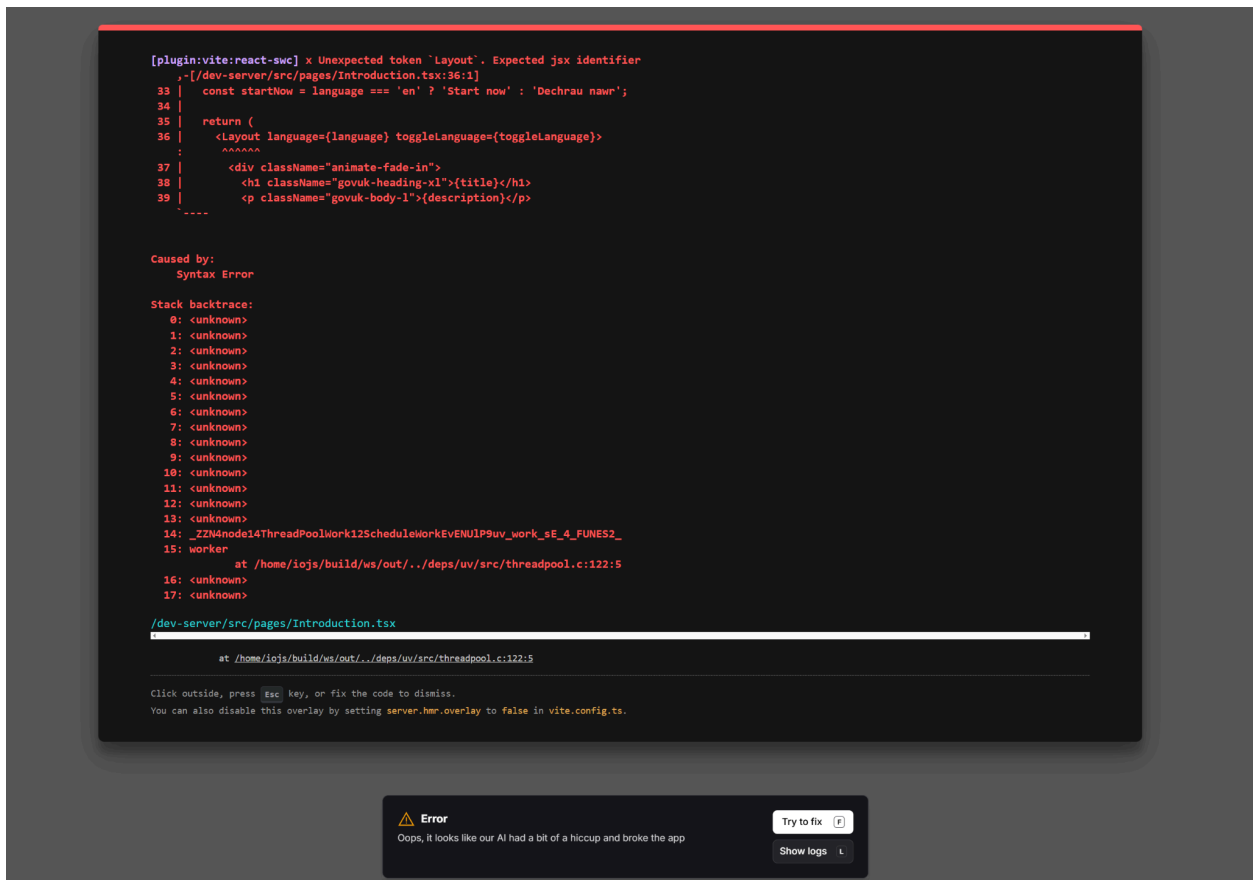
The above prompt produced more consistency on the output, making a nice balance between styling requirements, and functional requirements, allowing for subsequent prompts to be focused on iterating upon the initial output to extend functionality and implementing the full frontend e2e flow, opposed to fixing flaws, thus enhancing productivity. Below are example of the outputs of vague prompts vs the more complex prompts which attempted to build the full service in 1.

Vague Prompt Example



Vague Prompt Output From Lovable

Full e2e Prompt Example



Full e2e Prompt - Initial Output from Lovable

Lovable Prompt Transcript

Prompt & Response 1

User: *"Please create a full-featured web application that exactly replicates the UK "Find a court or tribunal" service as seen on the GOV.UK website.*

The application must meet the following requirements: Overall Purpose and Look & Feel: The website should mirror the official GOV.UK service for finding courts and tribunals.

It must use the UK Government Design System (GDS) for all visual elements, ensuring that the styling, fonts, and layout are identical to those used on GOV.UK.

The design must follow accessibility standards (WCAG 2.1), meaning that all text is clear, navigable, and easy to read, with proper contrast and keyboard accessibility.

Functionality: Homepage: The main page should feature a clear and prominent search bar where users can enter a town name or postcode. The page should include a header (with GOV.UK branding) and a footer with standard links (like Help, Contact, and Terms and Conditions).

Search Results Page: When a user enters a search term, the website should display a list of courts or tribunals that match the query. Each result should show the court's name and its address in a neat, easy-to-read list. If no results are found, the user should see a friendly message suggesting they try a different search.

Court Details Page: When a user clicks on a specific court from the results, the website should show a dedicated details page. This page must include: The court's full name and address. Opening hours (e.g., Monday to Friday, 9am to 5pm). Contact details, including a phone number and an email address. A list of the types of cases or areas of law that the court handles.

Mock Data: The website should operate with sample (mock) data. Although there is no need for a live data connection or database, the information shown must look realistic and mimic what users would see on the actual service.

Technology and Structure (Described Simply): The application should be built using a modern web server that uses TypeScript for clarity and safety. It should render web pages on the server before sending them to the user, ensuring the content is fully available when the page loads.

The pages should be created using Nunjucks templates (a tool for designing HTML pages) to keep the layout consistent with the GDS standards. The backend (server) will use Express, a framework that manages the website's navigation and pages, while the front-end will display the pages as traditional HTML styled according to GOV.UK guidelines.

Local Running: The finished website should be able to run on a local computer for testing purposes, meaning no deployment to a live server is necessary. It should be organized in a clear and logical structure so that someone can start it locally with minimal setup.

Exact Replication: The goal is to have the website behave and appear as close as possible to the actual GOV.UK court finder. This includes the same layout, branding, and even the same manner of displaying search results and court details.

All interactive elements (like the search bar and navigation links) should work in a way that users expect based on their experience with GOV.UK. Please ensure that every aspect of the service—from the homepage and search functionality to the detailed court pages—is designed exactly like the official GOV.UK “Find a court or tribunal” service, using the UK Government Design System and following best practices for accessibility and usability. You must use the nunjucks components provided by the government design system for the front end”

Lovable: *“I’m excited to help you build a web application inspired by the UK’s “Find a court or tribunal” service. This project evokes the clean, accessible design principles of the GOV.UK website, which is known for its focus on usability, clarity, and accessibility.*

For this project, I’ll draw inspiration from the UK Government Design System (GDS), which features a distinct typography, color palette, and component styles that prioritize readability and accessibility. The design will be minimalist yet functional, with a focus on clear navigation and content presentation.

For the first version, I’ll implement these core features:

- *A homepage with a prominent search bar for finding courts by location*
- *A search results page displaying matching courts/tribunals*
- *A detailed court information page for each location*
- *Responsive design that works seamlessly on all devices*
- *Accessible design following WCAG 2.1 standards*

For styling, I’ll use:

- *The GDS color palette: primarily blue (#1d70b8) and black text on white backgrounds*
- *Clear typography with the GDS Transport font*
- *Consistent spacing and layout based on the GDS design principles*
- *Subtle animations for transitions between pages*

The UK “Find a court or tribunal” application has been created with responsive design, accurate GOV.UK styling, and smooth transitions between pages.”

What’s next?

- *Refine & Customize: Tweak the design, animations, and layouts via prompts or visual edits.*
- *Master Prompting: Use clear, detailed, and iterative prompts for better outcomes.*
- *Expand with Backend: Connect Supabase to add login, store data or advanced AI features.*
- *GitHub Sync: Transfer your project’s code to GitHub for two-way sync of edits.*
- *Debug with Ease: Activate our experimental “chat mode” to troubleshoot issues quickly.*
- *Add project knowledge: Set key context or custom instructions you want to include in every edit in this project.*

Prompt & Response 2

NOTE: When lovable detects an error, a button appears for the end user to click to make Lovable fix it, upon clicking this error, Lovable sent the following prompt on behalf of the user

User:

"For the code present, I get the error below. Please think step-by-step in order to resolve it.
...

[postcss] /dev-server/src/index.css:2:1: The `focus:outline-3` class does not exist. If `focus:outline-3` is a custom class, make sure it is defined within a `@layer` directive.

```
{
  "message": "[postcss] /dev-server/src/index.css:2:1: The `focus:outline-3` class does not exist. If `focus:outline-3` is a custom class, make sure it is defined within a `@layer` directive.",
  "plugin": "vite.css",
  "loc": {
    "line": 2,
    "file": "/dev-server/src/index.css",
    "column": 0
  },
  "frame": "1 | \n2 | @tailwind base;\n | ^\n3 | @tailwind components;\n4 | @tailwind utilities;",
  "pluginCode": "\n@tailwind base;\n@tailwind components;\n@tailwind utilities;\n\n@import url('https://fonts.googleapis.com/css2?family=Noto+Sans:wght@400;700&display=swap');\n\n@layer base {\n  :root {\n    --background: 0 0% 100%;\n    --foreground: 240 10% 3.9%;\n\n    --card: 0 0% 100%;\n    --card-foreground: 240 10% 3.9%;\n\n    --popover: 0 0% 100%;\n    --popover-foreground: 240 10% 3.9%;\n\n    --primary: 203 70% 46.1%;\n    --primary-foreground: 0 0% 100%;\n\n    --secondary: 240 4.8% 95.9%;\n    --secondary-foreground: 240 5.9% 10%;\n\n    --muted: 240 4.8% 95.9%;\n    --muted-foreground: 240 3.8% 46.1%;\n\n    --accent: 240 4.8% 95.9%;\n    --accent-foreground: 240 5.9% 10%;\n\n    --destructive: 0 84.2% 60.2%;\n    --destructive-foreground: 0 0% 98%;\n\n    --border: 240 5.9% 90%;\n    --input: 240 5.9% 90%;\n    --ring: 203 70% 46.1%;\n    --radius: 0px;\n  }\n\n  * {\n    @apply border-border;\n  }\n\n  body {\n    @apply bg-background text-govuk-black font-sans text-base antialiased;\n    font-family: 'Noto Sans', sans-serif;\n    line-height: 1.5;\n  }\n\n  h1, h2, h3, h4, h5, h6 {\n    @apply font-bold text-govuk-black;\n  }\n\n  /* GOV.UK specific styling *\n  .govuk-width-container {\n    @apply max-w-[960px] mx-auto px-4 sm:px-6;\n  }\n\n  .govuk-heading-xl {\n    @apply text-4xl leading-tight mb-6 font-bold;\n  }\n\n  .govuk-heading-l {\n    @apply text-3xl leading-tight mb-4 font-bold;\n  }\n\n  .govuk-heading-m {\n    @apply text-2xl leading-tight mb-3 font-bold;\n  }\n\n  .govuk-body {\n    @apply mb-5 text-base;\n  }\n\n  .govuk-link {\n    @apply text-govuk-blue underline
```



```

hover:text-black focus:outline-none focus:bg-govuk-yellow focus:text-govuk-black
transition-colors;\n }\n\n .govuk-button {\n  @apply inline-block px-8 py-2 text-base
bg-govuk-green text-white font-bold border-2 border-transparent \n  hover:bg-[#005a30]
focus:outline-3 focus:outline-govuk-yellow focus:bg-[#005a30] shadow-sm
transition-colors;\n }\n\n .govuk-button--secondary {\n  @apply bg-govuk-light-grey
text-govuk-black hover:bg-govuk-mid-grey;\n }\n\n .govuk-button--warning {\n  @apply
bg-govuk-red hover:bg-[#aa2a16];\n }\n\n .govuk-input {\n  @apply w-full p-2 border-2
border-govuk-black focus:outline-govuk-yellow focus:outline-3;\n }\n\n .govuk-label {\n
@apply block mb-1 text-base font-bold;\n }\n\n .govuk-error-summary {\n  @apply p-4
border-4 border-l-govuk-red mb-6;\n }\n\n /* Animation utilities *\n .page-transition-enter
{\n  @apply opacity-0;\n }\n\n .page-transition-enter-active {\n  @apply opacity-100
transition-opacity duration-300;\n }\n\n .page-transition-exit {\n  @apply opacity-100;\n
}\n\n .page-transition-exit-active {\n  @apply opacity-0 transition-opacity duration-300;\n
}\n}\n",
  "stack": "    at Input.error (/dev-server/node_modules/postcss/lib/input.js:106:16)\n    at
AtRule.error (/dev-server/node_modules/postcss/lib/node.js:115:32)\n    at processApply
(/dev-server/node_modules/tailwindcss/lib/lib/expandApplyAtRules.js:380:29)\n    at
/dev-server/node_modules/tailwindcss/lib/lib/expandApplyAtRules.js:551:9\n    at
/dev-server/node_modules/tailwindcss/lib/processTailwindFeatures.js:55:50\n    at async
plugins (/dev-server/node_modules/tailwindcss/lib/plugin.js:38:17)\n    at async
LazyResult.runAsync (/dev-server/node_modules/postcss/lib/lazy-result.js:261:11)\n    at
async compileCSS
(file:///dev-server/node_modules/vite/dist/node/chunks/dep-DyBnyoVI.js:36860:21)\n    at
async TransformPluginContext.transform
(file:///dev-server/node_modules/vite/dist/node/chunks/dep-DyBnyoVI.js:36141:11)\n    at
async PluginContainer.transform
(file:///dev-server/node_modules/vite/dist/node/chunks/dep-DyBnyoVI.js:49033:18)",
  "error_type": "immediate_runtime",
  "id": "/dev-server/src/index.css"
}

```

Lovable: "I'll fix this PostCSS error where the `focus:outline-3` class isn't recognized by Tailwind CSS. The error is fixed by replacing `focus:outline-3` with `focus:outline focus:outline-[3px]`. This uses Tailwind's standard outline utility with an arbitrary value of `[3px]` for the outline width, which is a valid approach in Tailwind CSS. The custom width is also added to the config file as a fallback, but the arbitrary value approach is more direct and should work reliably."

Prompt & Response 3

User: *"Please below the header, add clickable text entitled "Cymraeg" which when clicked, will update the the pages to use the Welsh language. This should feature on all pages.*

Also, the first page of the application should be an intro page with the header, "Find a court or tribunal". There should be a description below this which describes what this service is for with the following text...

"Find contact details and information on courts and tribunals in England and Wales, and some non-devolved tribunals in Scotland.

You can find the following details:

- * address*
- * contact details*
- * opening times*
- * how to get to the court or tribunal*
- * the areas of law it covers*
- * disabled access to the building.*

Below this there should be the green button from the GDS which says "Start now >" When pressed this should take you to another page.

This page will include a radio buttons. Above the radio buttons there should be a header with the text "Do you know the name of the court or tribunal?" and a sub header with the text "The name of the court or tribunal can be found on a letter, email or text from us." There should be 2 radio button options for this...

"I have the name"

"I do not have the name "

again, these radio button components must comply with the GDS standard.

If the user knows the name of the court or tribunal, take them to the page we have presently, which features the input box to type the town, city or postcode in.

If the user does not know the name of the court or tribunal, take them to another radio button page with the header text "What do you want to do?", with the following radio button options...

"Find the nearest court or tribunal"

"Find a court or tribunal to send documents to"

"Contact a court or tribunal to get an update on your application"

or

"It is not listed here"

Lovable: *I'll create the Welsh language toggle and implement the multi-step journey for the court finder application with the GDS design specifications. I've added a Welsh language toggle and created a multi-step journey for finding courts or tribunals. The journey now includes an introduction page, a question about knowing the court name, and further options based on the user's selection. All text supports both English and Welsh languages that can be toggled with the "Cymraeg" button in the header.*

Prompt & Response 4

User:

Changes Required:

Remove the unnecessary empty page after the first page. When pressing the button on the first page, users should directly see the page with these radio buttons:

What do you want to do?

Find the nearest court or tribunal

Find a court or tribunal to send documents to

Contact a court or tribunal to get an update on your application

It is not listed here

After selecting any of these options, take users directly to the next page with these radio buttons:

What do you want to know more about?

Money

Money claims, rent or mortgage disputes, bankruptcy, job disputes related to pay, appealing a tax or benefits decision.

Probate, divorce or ending civil partnerships

Probate application and disputes, divorce, ending a civil partnership.

Childcare and parenting

Arrangements for looking after your children if you separate from your partner or making an adoption legal.

Harm and abuse

Applying for an injunction against someone harassing or abusing you, being forced to marry against your will, or preventing Female Genital Mutilation (FGM).

Immigration and asylum

Seeking asylum, the right to live in the UK, and appealing deportation.

Crime

Single Justice Procedure and other criminal cases at a Crown or Magistrates' Court.

High Court district registries

Courts that deal with the most serious and high-profile cases in criminal and civil law.

I can't find what I'm looking for

Followed by a continue button.

Next Step Based on User Selection:

*If **Money** is selected, show these radio buttons:*

What kind of help do you need with money?

Money claims (Claims for when you are owed money or responding to claims)

Probate (Applying for probate, storing a will, settlement disputes)

Housing (Tenant evictions, rent or mortgage disputes)

Bankruptcy (Opposing a bankruptcy petition, support against violence order, cancelling bankruptcy)

Benefits (Appealing benefits such as PIP, ESA, Universal Credit)

Claims against employers (Employment disputes on pay, unfair dismissal, discrimination)

Tax (Appealing a tax decision)

Single Justice Procedure (TV Licensing, minor traffic offences)

I can't find what I'm looking for

If Probate, divorce or ending civil partnerships is selected, show:

What kind of help do you need with probate, divorce or ending civil partnerships?

Probate (Applying for probate, storing wills, disputes)

Divorce (Ending marriage, legal separation, finance/property/children help)

Civil partnership (Ending a civil partnership)

Financial remedy (Settling finances after divorce)

Forced marriage (Being forced to marry)

I can't find what I'm looking for

If Childcare and parenting is selected, show:

What kind of help do you need with childcare and parenting?

Childcare arrangements if separated (Making arrangements post-separation)

Adoption (Support for child adoption application)

I can't find what I'm looking for

If Harm and abuse is selected, show:

What kind of help do you need with harm and abuse?

Female Genital Mutilation Protection Orders (Protection for FGM victims)

Domestic abuse (Support for domestic abuse injunctions)

Forced marriage (Being forced to marry)

I can't find what I'm looking for

If Immigration and asylum is selected, directly show information page:

Court or tribunal search results

We manage immigration and asylum applications at our central service centre. Find where to send your documents or inquire about your application's progress.

Immigration and Asylum Appeals Service Centre (Serving all of England and Wales)

*If **Crime** is selected, show:*

What kind of help do you need with crime?

Single Justice Procedure (TV Licensing, minor traffic offences)

Other criminal offences (Criminal cases at Crown or Magistrates' Court)

I can't find what I'm looking for

*If **High Court district registries** is selected, display an input page:*

What is your postcode?

We will use your postcode to show you the closest court or tribunal handling high court district registry cases.

Enter your postcode.

Selecting any specific option or the "I can't find what I'm looking for" option on these pages should prompt the postcode input page described above.

Lovable: *I'll implement a complete user journey with the described radio button pages and navigation flow.*

I've created a complete flow for the court finder application with all the requested pages. The user journey now starts with the intro page and flows through each of the radio button selection pages as specified. The "Start" page has been removed, and clicking "Start now" takes users directly to the "Do you know the name of the court or tribunal?" page. All pages support both English and Welsh languages and use GDS design standards.

Prompt & Response 5

User: *For the code present, I get the error below. Please think step-by-step in order to resolve it.*

`src/pages/ChildcareOptions.tsx(79,150): error TS1005: '}' expected.`

`src/pages/ChildcareOptions.tsx(79,183): error TS1381: Unexpected token. Did you mean '{ }' or '}'?`

`src/pages/ChildcareOptions.tsx(84,73): error TS1005: '}' expected.`

`src/pages/ChildcareOptions.tsx(84,97): error TS1381: Unexpected token. Did you mean '{ }' or '}'?`

Lovable: *I'll fix the TypeScript errors in the ChildcareOptions.tsx file. The errors are related to special characters in the Welsh language text that need proper escaping. The errors have been fixed by properly escaping single quotes in the Welsh language text using backslashes. The issue was with apostrophes in phrases like "chi'n" and "rwy'n" that needed to be escaped as "chi'n" and "rwy'n" to avoid breaking the JSX syntax.*

Prompt & Response 6

User: *For the code present, I get the error below.*

Please think step-by-step in order to resolve it.

...

`src/pages/Index.tsx(23,32): error TS2322: Type '{ children: string; className: string; }' is not assignable to type 'IntrinsicAttributes & Readonly<Options>'.`

Property 'className' does not exist on type 'IntrinsicAttributes & Readonly<Options>'.

...

Lovable: *Looking at the error message, there's an issue with the className property being passed to the ReactDOM component in the Index.tsx file. The error indicates that the ReactDOM component doesn't accept a className prop directly.*

Let's fix this error by properly applying the styling to the ReactDOM component:

The issue was with passing className directly to the ReactMarkdown component, which doesn't accept this prop. I've fixed it by moving the space-y-4 class to the parent div element that contains the ReactMarkdown component, while keeping the existing govuk-body mb-8 classes. This approach ensures proper styling while resolving the TypeScript error.

Prompt & Response 7

User: *When I get to What do you want to do? page, it doesnt take me to the radio button page which features the Money*

Money claims, rent or mortgage disputes, bankruptcy, job disputes related to pay, appealing a tax or benefits decision.

Probate, divorce or ending civil partnerships

Probate application and disputes, divorce, ending a civil partnership.

Childcare and parenting

Arrangements for looking after your children if you separate from your partner or making an adoption legal.

Harm and abuse

Applying for an injunction against someone who is harassing or abusing you, being made to marry against your will or preventing Female Genital Mutilation (FGM).

Immigration and asylum

Seeking asylum, right to live in the UK, and appealing deportation.

Crime

Single Justice Procedure and other criminal cases at a Crown or Magistrates' Court

High Court district registries

Courts that deal with the most serious and high profile cases in criminal and civil law.

or

I can't find what I'm looking for

Continue

fix this, and also make sure those options then lead to the aforementioned radio button pages respectively.

Lovable: Looking at the issue, I need to modify the navigation logic in the SearchPurpose.tsx file to direct users to the SearchCategory.tsx page after selecting an option, rather than directly to the search-by-name page.

I've fixed the navigation in the SearchPurpose page to correctly take users to the SearchCategory page after selecting an option. Now when a user selects an option on the "What do you want to do?" page and clicks Continue, they will be taken to the page with options for Money, Probate, Childcare, etc. The SearchCategory page already has the correct navigation logic to take users to the appropriate specialized options pages (MoneyOptions, ProbateDivorceOptions, etc.) based on their selection.

Prompt & Response 8

User: Cool, can you now implement the backend for this, using <https://github.com/hmcts/fact-api> as an example to build your work off of

Lovable: I'll help you understand how we can approach this, but we need to take a step back and consider the architecture of your application before implementing a backend.

Looking at the GitHub repository you linked (hmcts/fact-api), it's a Spring Boot Java application with PostgreSQL, which is significantly different from the React frontend we've been working on.

To implement a backend for your court finder application, you have a few options:

Use a Supabase integration, which is natively supported by Lovable and would give you database functionality without writing a separate backend.

Create mock API services that simulate backend functionality (similar to what we already have with the courts data).

Build a real backend service, but this would need to be done outside Lovable as it doesn't support Java Spring Boot backend development.

Before I write any code, could you clarify which approach you'd prefer? Specifically:

Would you like to enhance the current mocked data approach?

Would you like to integrate with Supabase for real database functionality?

Or do you have a different approach in mind?

This way, I can provide the most appropriate solution for your needs.

Prompt & Response 9

User: *use supabase*

Lovable: *To integrate Supabase into your court finder application, I'll help you set up the necessary database structure and services for your application.*

I've set up the integration with Supabase for your court finder application. The implementation includes a dedicated courts service, TypeScript types for Supabase tables, and utility functions to seed your database with initial data. To finish the setup, you'll need to connect your project to Supabase and add the environment variables for the Supabase URL and anonymous key

Prompt & Response 10

User: *"ok revert to the original mock api integration"*

<NOTE: Reverting as a non-technical user would struggle with the Supabase set up if Lovable is unable to do this itself.>

Lovable: *I'll revert back to using the original mock API integration instead of Supabase.*

I've now reverted all the Supabase integration changes and returned to using the original mock API implementation. The app will now use the local data from src/data/courts.ts for court information instead of connecting to Supabase.