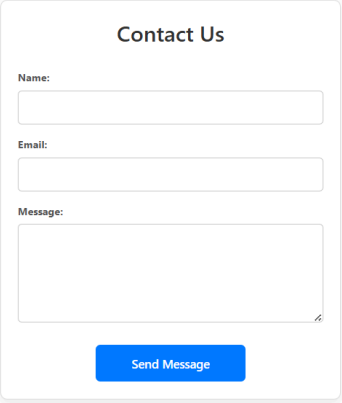
From Declarative DSL to Dynamic UI: Evaluating an LLM-Based Frontend Compiler - Prototype Evaluation

Simple Form Prompts

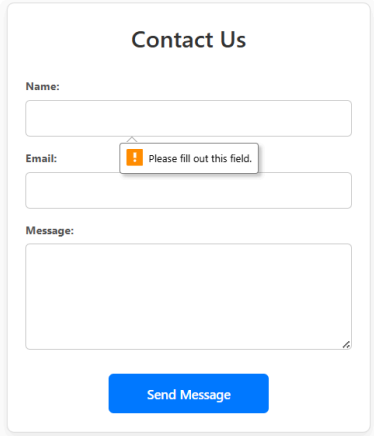
1. Basic Contact Form: This is a foundational test for basic text and email inputs.
DSL Prompt:
form(Contact Us): name, email, message -> /api/contact

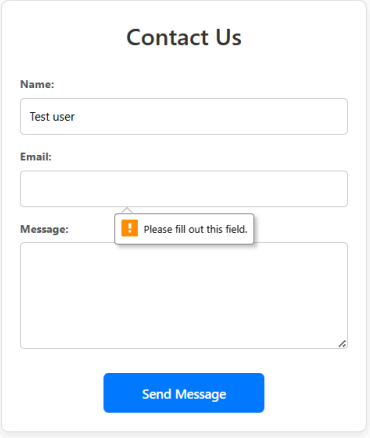
Run	Compilation Time
1	28.33s
2	29.32s
3	17.29s

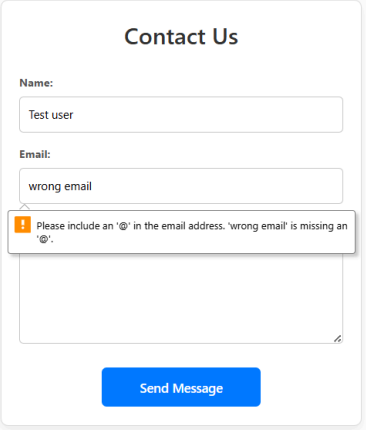
Generated ui:



Interacting with the form, we can see that basic validations are also applied







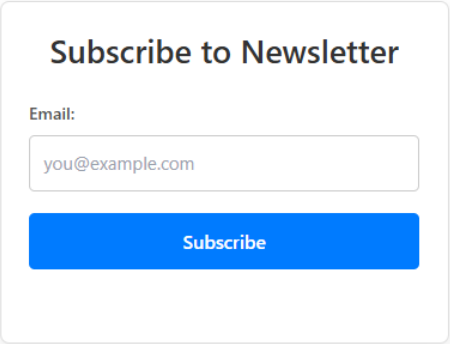
2. Newsletter Subscription: This tests the compiler's ability to handle a minimal form with a single input and a clear call to action.

DSL Prompt:

form(Subscribe to Newsletter): email -> /api/subscribe

Run	Compilation Time
1	56.19s
2	36.08s
3	61.60s

Generated ui:



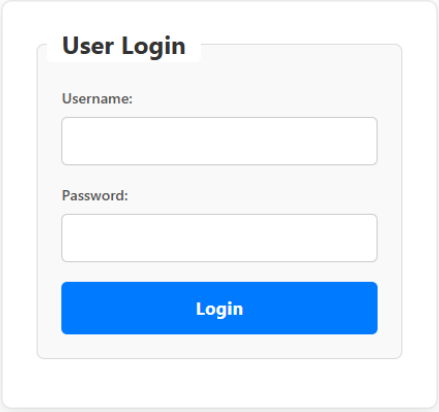
3. User Login: This tests a very common use case and the compiler's ability to handle different input types like password.

DSL Prompt:

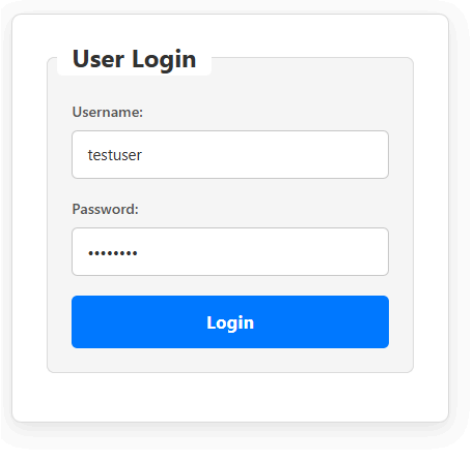
form(User Login): username, password -> /api/login

Run	Compilation Time
1	17.03s
2	29.74s
3	35.01s

Generated UI:



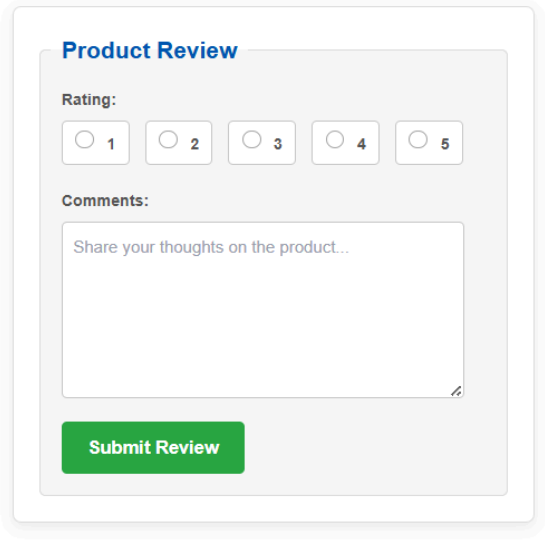
Manually filled form:
This image shows that the input field types are generated properly

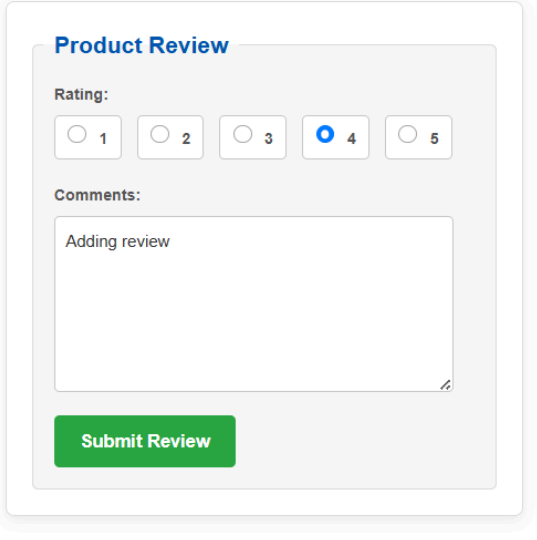


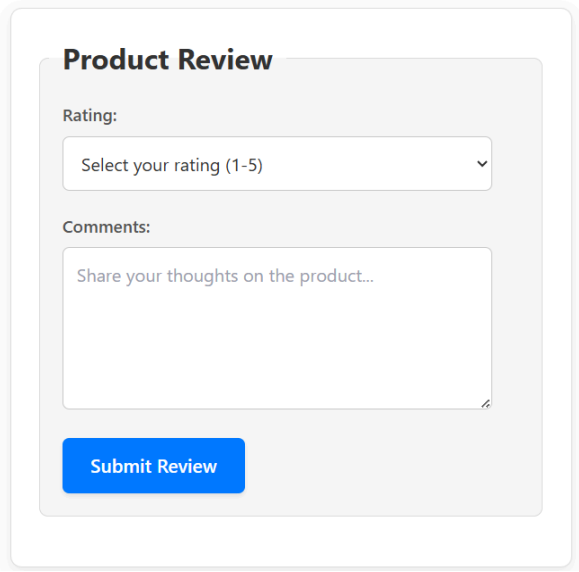
4. Product Review: This tests for more specific input types and instructions, requiring the compiler to infer a suitable input method (e.g., a number or star rating).
DSL Prompt:
form(Product Review): rating(1-5), comments -> /api/comments

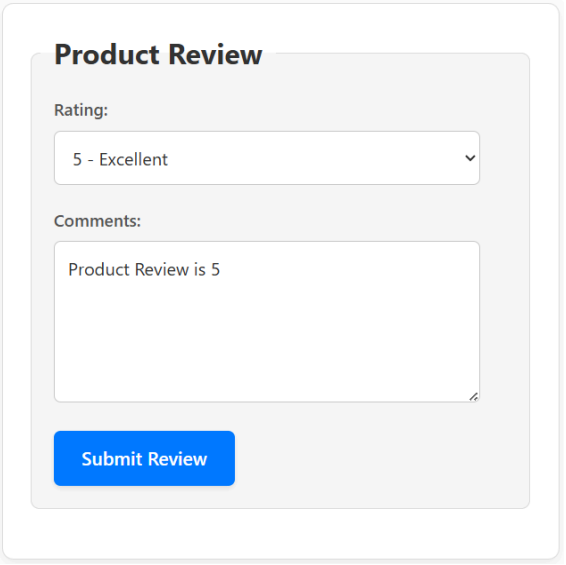
Run	Compilation Time
1	32.06s
2	50.74s
3	49.27s

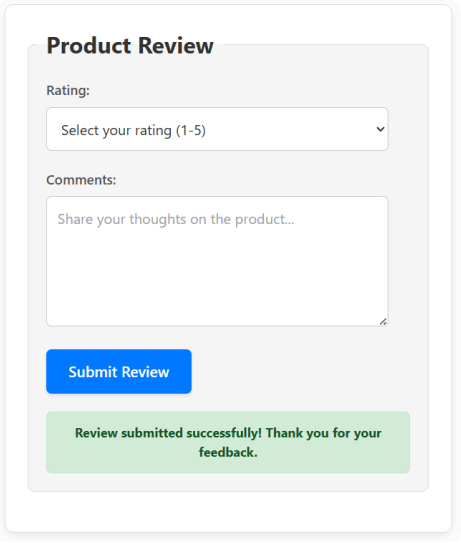
Generated UI: Each run yielded slightly different user experiences.

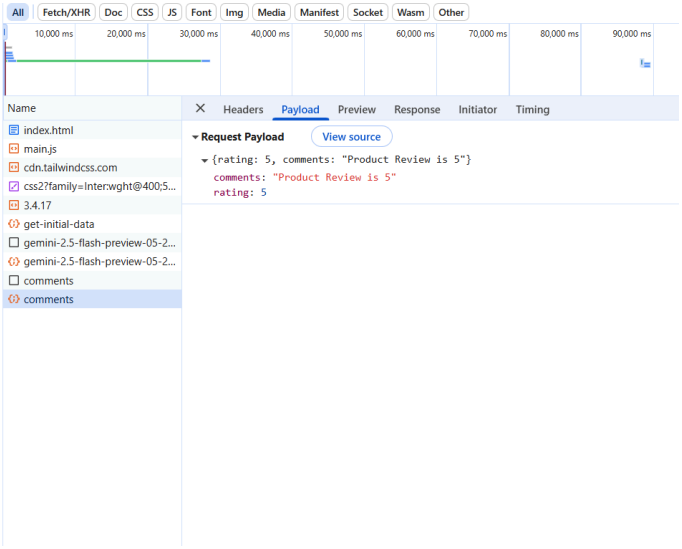












Complex Form Prompts

Here are some DSL prompts designed to test more advanced features of your AI compiler.

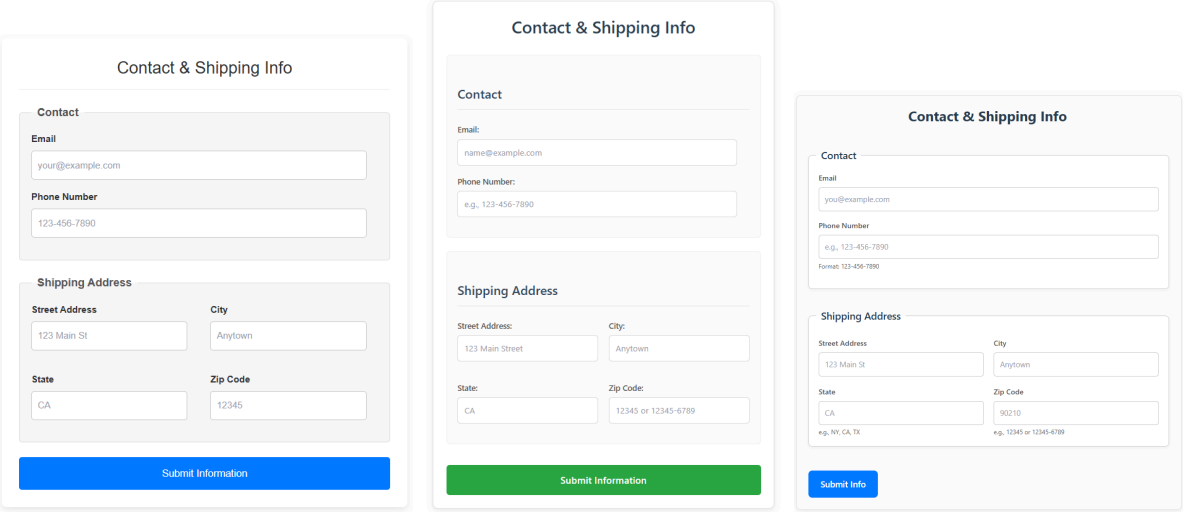
1. Nested Form with Two Columns: This prompt tests the ability to handle a more complex layout, requiring the compiler to create nested sections and organize fields into multiple columns.

DSL Prompt:

```
form(Contact & Shipping Info):
  section(Contact): email, phone_number
  section(Shipping Address | 2 columns): street_address, city, state, zip_code
  -> /api/submit-shipping
```

Run	Compilation Time
1	48.17s
2	43.82s
3	34.87s

Generated UI: Each run yielded slightly different user experiences.



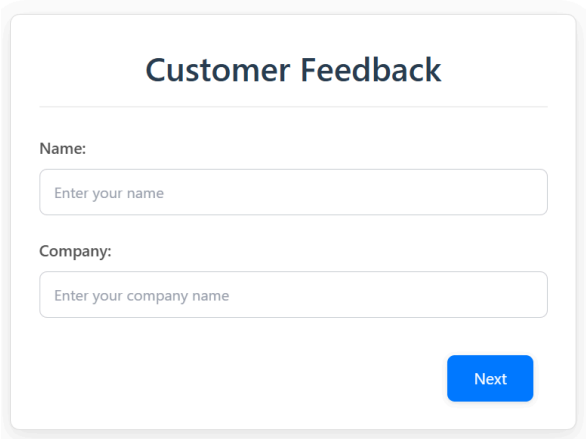
2. Multi-Page Survey: This prompt tests the compiler's ability to understand pagination and create a multi-step user experience. This requires generating navigation buttons and showing only a subset of fields at a time.

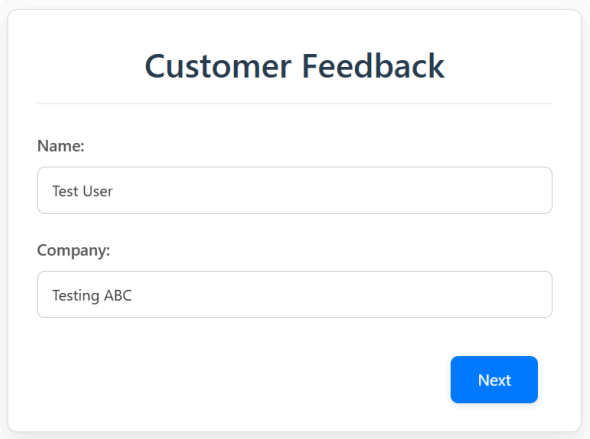
DSL Prompt:

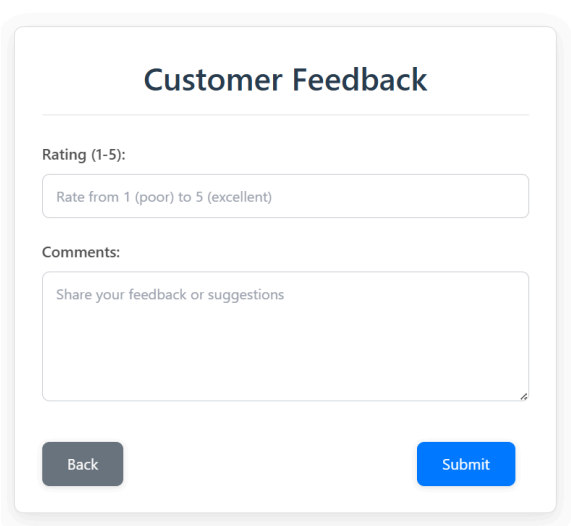
```
form(Customer Feedback):
  page1: name, company -> Next
  page2: rating, comments -> Submit -> /api/submit-survey
```

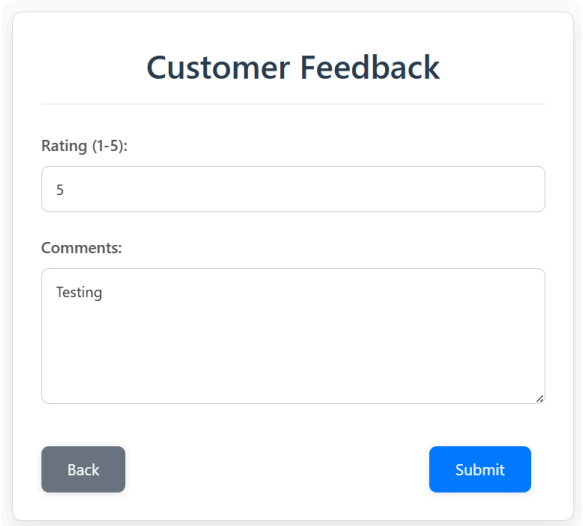
Run	Compilation Time
1	29.83s
2	38.81s
3	32.36s

Generated UI: The form handling was implemented correctly; upon submission, all entered data are accurately captured and sent in the payload to the specified endpoints for processing. This ensures reliable data transmission from the generated UI forms to backend services on







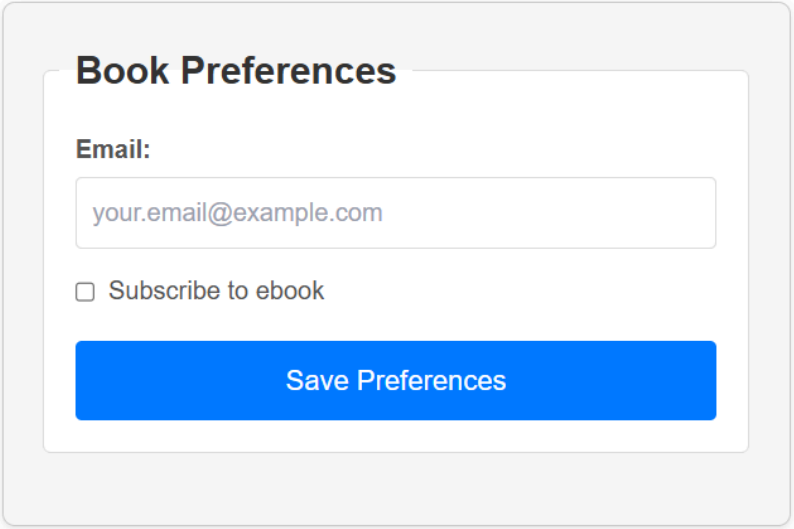


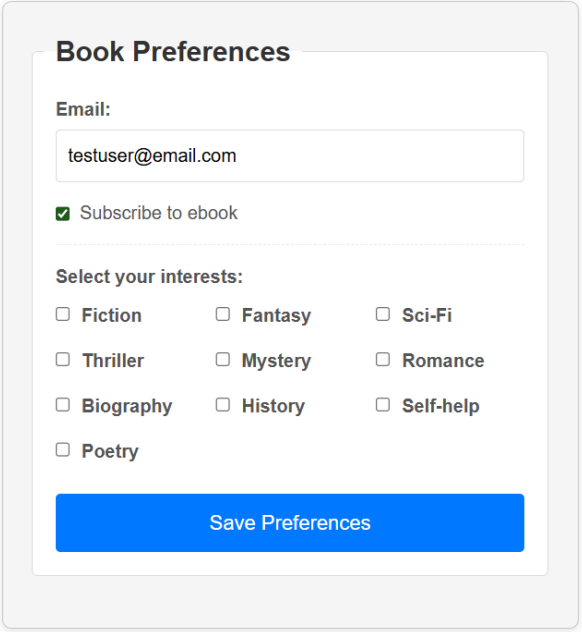
3. Form with Conditional Logic: This is a critical test for dynamic behavior. It requires the compiler to create an interactive form where one input (a checkbox) determines whether another field is visible.

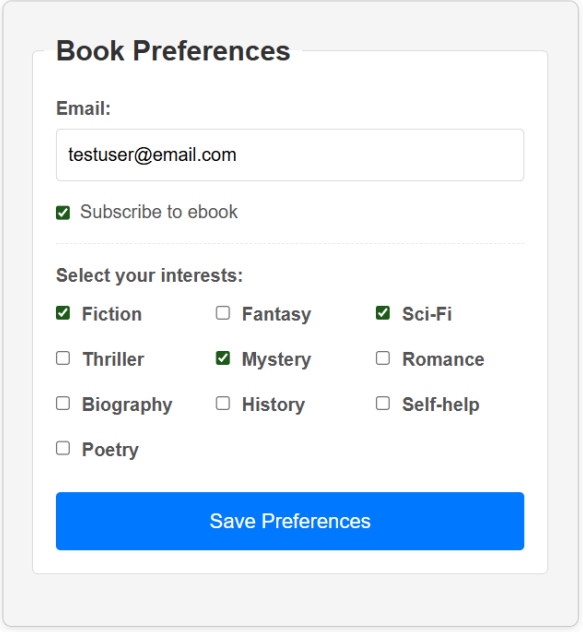
DSL Prompt:
form(Book Preferences): email, checkbox(Subscribe to ebook | id: ebook-toggle), on(ebook-toggle checked): show list(interests | genre: random | count: 10) -> /api/update-preferences

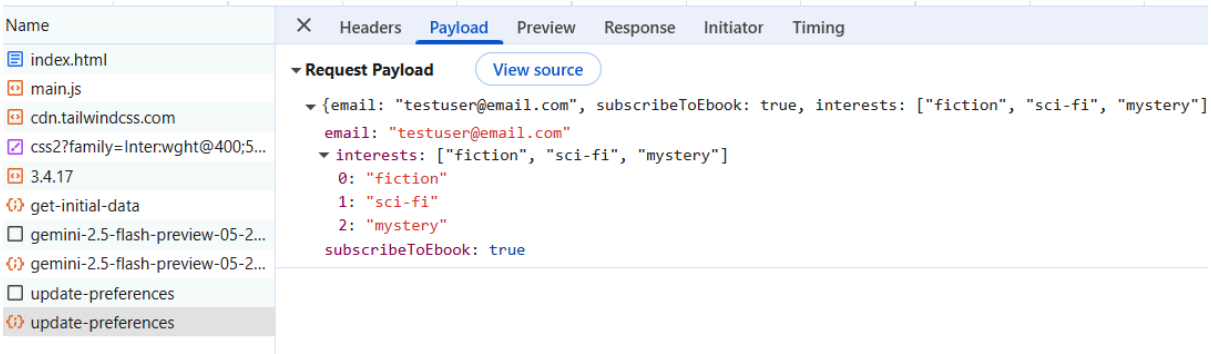
Run	Compilation Time
1	24.34s
2	47.44s
3	29.38s

Generated UI







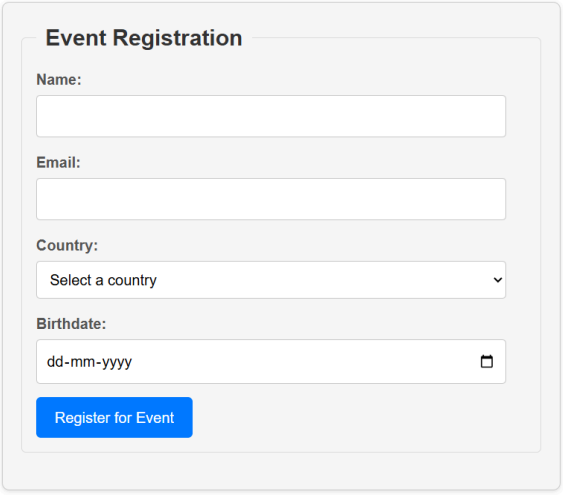


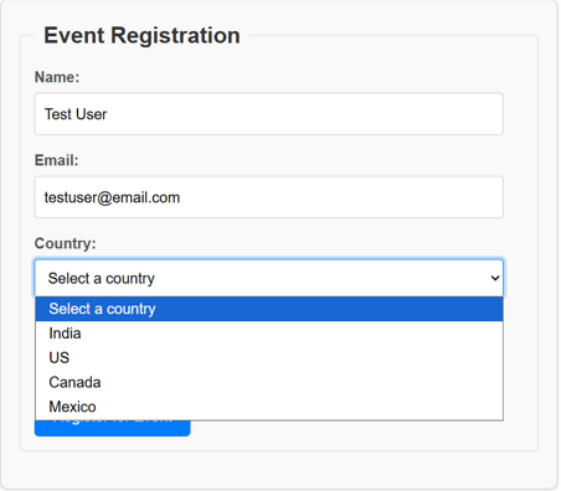
4. Form with Specific UI Components: This prompt tests the LLM's knowledge of common UI patterns and its ability to generate the appropriate HTML elements for non-text inputs.

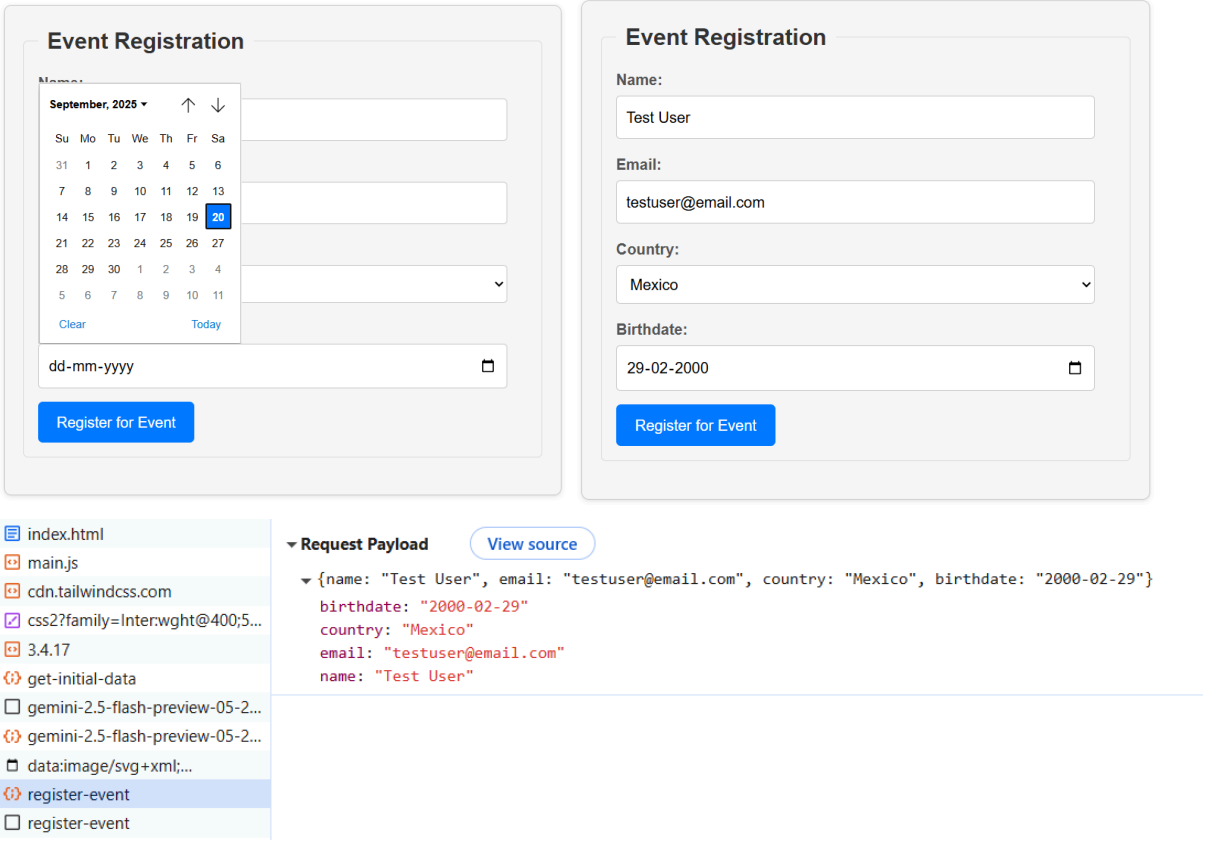
DSL Prompt:
form(Event Registration): name, email select(Country): options(India, US, Canada, Mexico) date_picker(Birthdate) -> /api/register-event

Run	Compilation Time
1	23.05s
2	17.59s
3	26.53s

Generated UI:



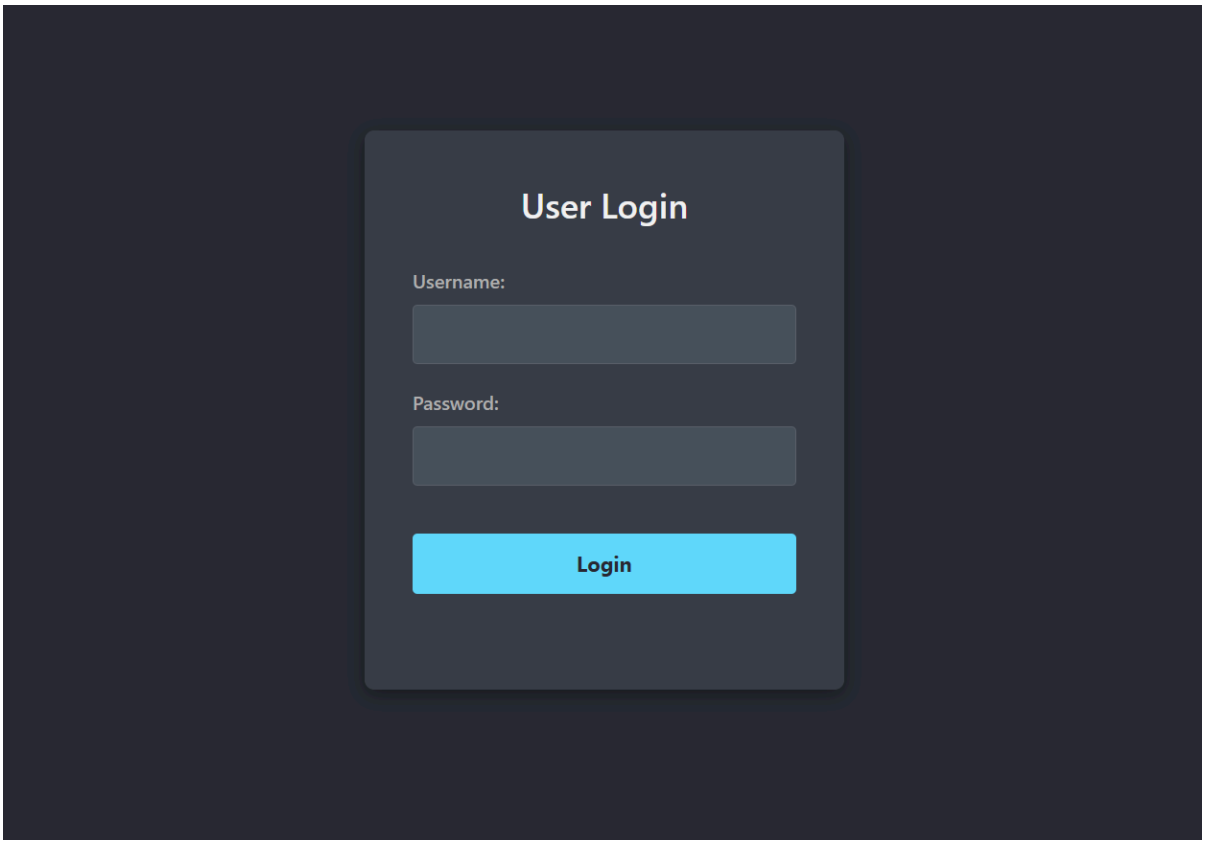


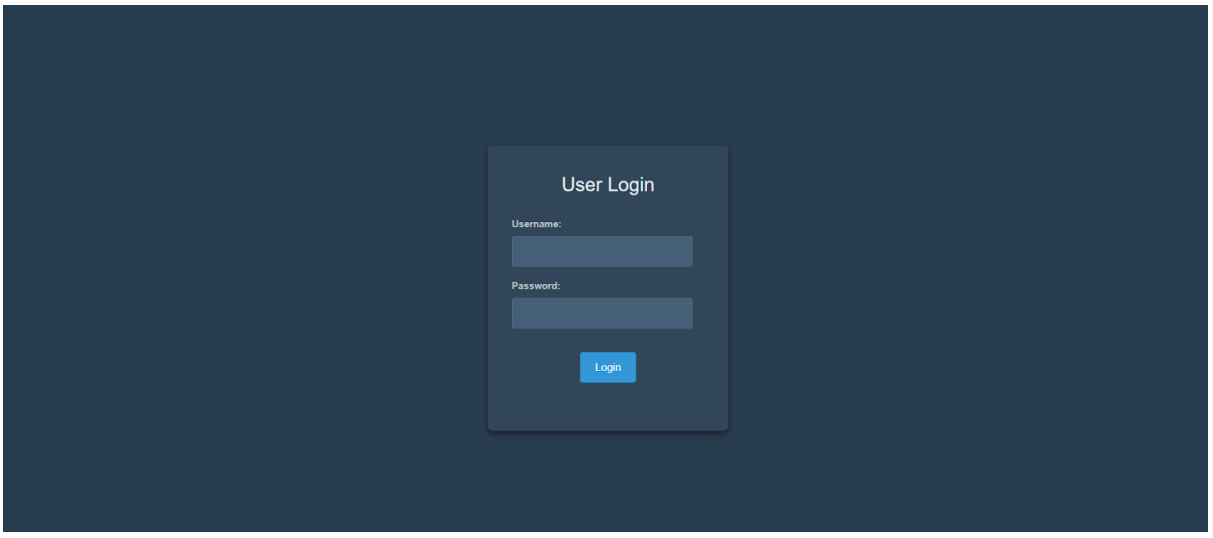


Theme/Style Form Prompts

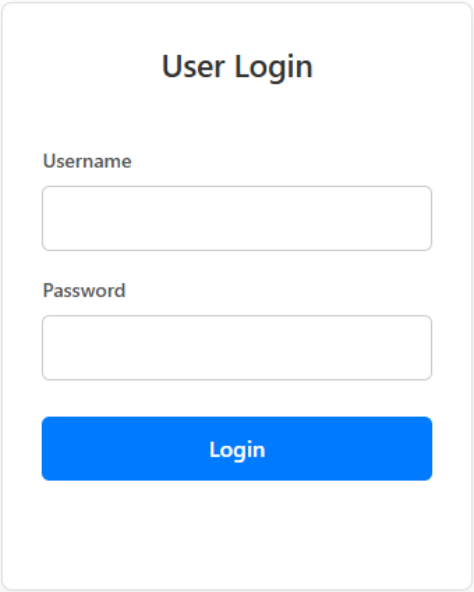
Tried different theme styles for simple forms and observed behaviour - for reach reload able to see different styles for the same theme, which was refreshing

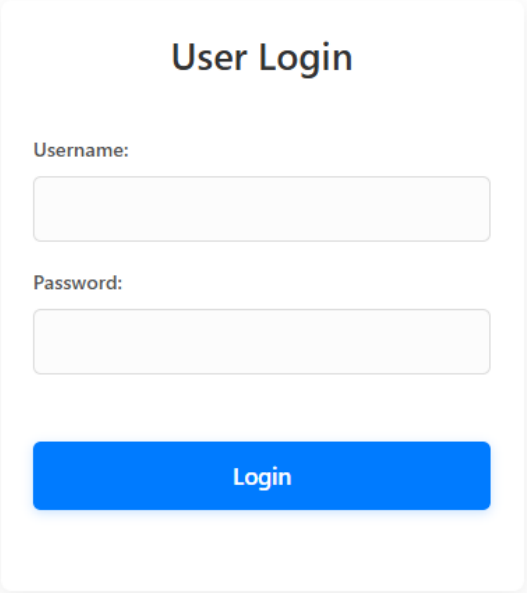
- form(User Login | Style: dark theme): username, password -> /api/login



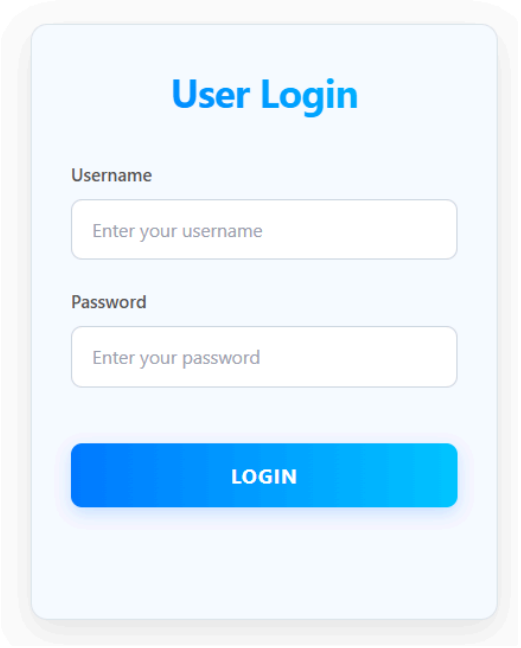


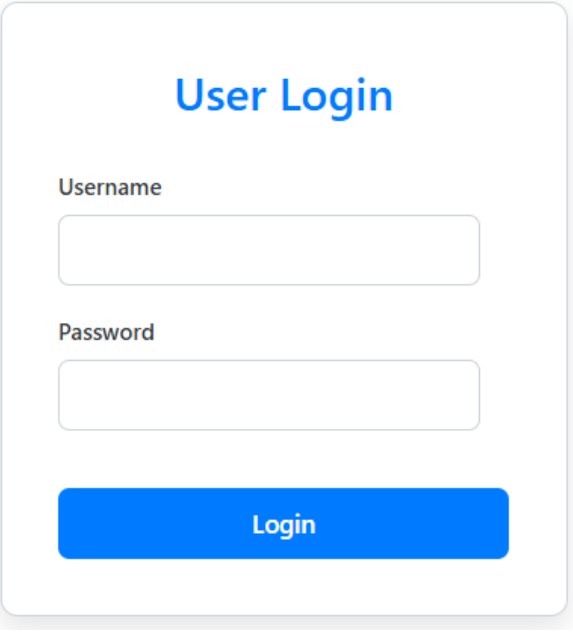
- form(User Login | Style: minimalist): username, password -> /api/login



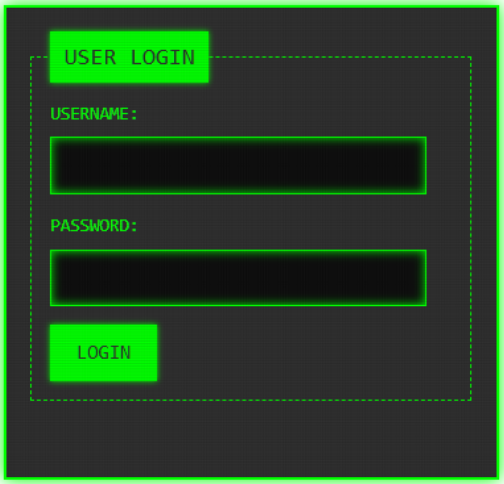


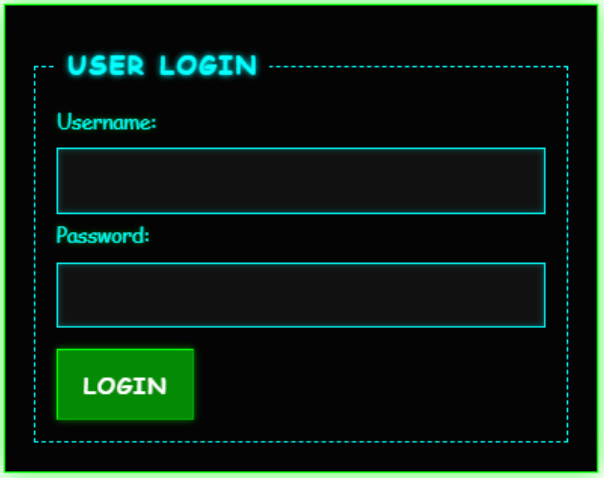
- form(User Login | Style: vibrant, modern) : username, password -> /api/login





- form(User Login | Style: retro): username, password -> /api/login





Compilation Time Calculation

The compilation time is the time taken for the request to hit the queue, wait for the server response, and download time

Example:
Compilation Time: 32.36s

Queued at 342.20 ms

Started at 764.76 ms

Resource Scheduling	DURATION
Queueing	422.55 ms
Connection Start	**DURATION**
Stalled	0.50 ms
Request/Response	**DURATION**
Request sent	0.51 ms
Waiting for server response	31.94 s
Content Download	13.18 ms
Explanation	**32.37 s**

Server Timing	TIME
gfet4t7	31.89 s

Evaluation Summary

The LLM-Based Frontend Compiler consistently produced functional UI elements for a wide range of DSL prompts, covering basic to advanced use cases.

- **Speed:** Compilation times ranged from 17s to 61s, with most runs under 35s, which is acceptable for prototype settings and complex code generation.

- **Accuracy:** The generated forms correctly reflected the intent of the DSL, including field types, required validations, layout structure, and submission logic.

- **Complexity Handling:** The compiler managed multi-page, multi-section, and conditionally rendered interfaces smoothly, demonstrating ability to infer and implement advanced UI patterns.

- **Theming:** Style instructions in the DSL were respected, showing strong adaptability to visual requirements.

- **User Experience:** Each run delivered a usable, interactive UI with expected behavior and dynamic validation where specified. Minor layout variability between compilations was observed, reflecting LLM-driven creativity.

- **Reliability:** Across varied prompts, the system reliably mapped DSL intent to HTML/CSS/JS output, validating its utility for declarative UI generation.

Overall, the LLM frontend compiler prototype shows robust real-world application potential for rapid UI generation from high-level specifications, with responsive compilation and faithful prompt interpretation.