Skinstric

Project Specifications

Skinstric.ai

SKINSTRIC BUILD

Frontend Simplified X Skinstric.ai

In this real-world project, you will be tasked with contributing to the development of a technologically sophisticated website for Skinstric, an AI startup specializing in creating customized skincare products using artificial intelligence. This project presents an exceptional opportunity to validate your skills as a professional frontend developer.

The purpose of this task is to replace Skinstric's outdated and unmaintainable current codebase with modern coding best practices and technologies.

The Skinstric CEO wants the site to be built using:

- Next JS (for SEO, performance and ease of use reasons)
- Tailwind (for CSS)
- Lotties (for animations)
- GSAP (for animations)
- Three.js (for 3D objects)
- ESLint (for code quality)
- Lighthouse (for testing)
- Storybook (for testing)
- Playwright (for staging)

Ultimately, the goal of this project is to get real experience on your resume, and to also increase your proficiency in tech to help you pass high salary job interviews.

Next to each of the 3 Phases, you will see the estimated market value of a developer who can complete the phase, i.e:

- Phase 1 - $80,000/year
- Phase 2 - $100,000/year
- Phase 3 - $120,000/year

Initially, this project was unpaid. However, for the sake of telling recruiters that ask that your internship is paid (to increase your perceived value), you will be sent $100 upon completion.

Students that complete this project with a high score and minimal assistance will be considered for a remote frontend developer position.

Best regards,
David Bragg
FES

Estimated market value: $80,000/year

Phase 1

Skinstric.ai

SKINSTRIC BUILD

Phase 1 Demo

[Website](#)
[Demo Video](#)
[Figma File](#)
[Assets](#)

LEVEL 1 /

Level 1: Building Basic Form Fields To Capture Customer Data

Build the following form field flow (see Figma link on the left). This will simulate you capturing a new user's details to be used later in Levels 2 - 5. The form field flow has 2 pages to it {name} and {location}, both of which are necessary. Take a moment to visit the live demo website (left link) to note button hover states, the rotating form animation, typography and general CSS styling. Gather your font file assets from the left panel.

Fig 1 : Small details separate a quality website from a not quality one. While this form field is simple, try implementing the hover states and idle animations to take it to the next level.

Fig 2 : Remember, this is a flow and so there are multiple pages (2 pages) in this example that must work with each other seamlessly. Consider edge cases such as leaving on form 1 and not getting to form 2.

SKINSTRIC BUILD

Phase 1 Demo

[Website](#)
[Demo Video](#)
[Figma File](#)
[Assets](#)

Level 1 /

Expected Deliverables

To pass this level, we expect:

Pass

- Design matches Figma for 1920 / 1280 (either is fine depending on your screen size availability
- Website is performant (no unnecessary CPU usage
- Both form fields working for {name} and {location}. They store the user's information and send it to the backend to the API given.

4/5 Score

- Google Places API is implemented in the locations form with autocomplete as shown on the demo website / video. The form returns a longitude and latitude which is captured and sent to the same API above also
- User is not allowed to proceed without entering the expected value (form validation that Name and Location are strings and do not contain numbers or broken values)

5/5 Score

- Rotating diamond animation in the centre form
- Button "back" , "proceed" hover animation implemented
- Unit tests implemented for form and button animation logic
- Considers edge cases such as leaving form 1 and not getting to form 2, what happens then?

6/5 Score

- Mobile and Tablet responsive screens built too (they're on the Figma already)

Skinstric.ai

SKINSTRIC BUILD

Phase 1 Demo

[Website]
[Demo Video]
[Figma File]
[Assets]

LEVEL 1 /

API Details

Ping this API to submit your collected form data of user name and location.

POST DEMO API (not production)

https://wk7wmfz7x8.execute-api.us-east-2.amazonaws.com/live/FES_Virtual_Internship_1/level1

Input

```
{
    'Name': 'John Doe',
    'Location': 'New York'
}
```

Output

```
{
    SUCCESS: Added John Doe from New York
}
```

Skinstric.ai

SKINSTRIC BUILD

Phase 1 Demo

[Website]
[Demo Video]
[Figma File]
[Assets]

LEVEL 1 /

Tips

1. Animation Performance: GSAP is powerful, but make sure animations are performant. Utilize the will-change CSS property where necessary to hint to the browser which elements will animate.
2. Accessibility: Ensure that interactive elements are accessible. For example, make sure that the SVG icons used in buttons have appropriate ARIA labels.
3. Consistency in Code: Make sure to remove any commented-out code blocks that are no longer needed to keep the codebase clean.
4. Dynamic Imports: Use dynamic imports for large libraries or components that aren't immediately necessary. For example, GSAP or the PlacesInput component can be loaded dynamically only when needed to optimize the initial load time.
5. Error Handling: When using forms and inputs, implement proper error handling and form validation to guide the user and prevent submission of incorrect data.
6. Code Comments: Add comments to your code, especially in complex logic areas to describe what the code is doing, which is helpful for teams.
7. Server-Side Props: In getServerSideProps, you’re returning a bodyClass. Make sure this is utilized in the component to style the page correctly. Also, consider if getServerSideProps is necessary, or if you can statically generate the page with getStaticProps for better performance.
8. Naming Conventions: Ensure consistent and descriptive naming conventions for functions and variables. For example, submitNameForm and submitOriginForm are clear, but proceed is vague.
9. React Refs: You’re using refs for navigation links, which is good, but ensure you’re not manipulating the DOM directly, as React should handle all DOM updates.
10. CSS and Styling: Keep your CSS scalable and maintainable. Utilize CSS modules or styled-components for scoped and reusable styles.
11. State Management: You're using Recoil for state management, which is fine, but ensure the global state is necessary. For example, storing user input in global state makes sense if this data is required across multiple components.
12. Avoiding Memory Leaks: When using GSAP or any subscriptions within useEffect, always return a cleanup function to avoid memory leaks, especially when the component unmounts.
13. Polyfills: Load only necessary polyfills to avoid bloating the bundle size.
14. Server Response Handling: Make sure that server responses (e.g., from APIs) are handled correctly, including loading states and error states.
15. Testing: Implement unit tests, especially for the functions that handle form submissions and data processing.
16. Optimization: Consider using useMemo and useCallback to optimize the component's performance, preventing unnecessary renders.
17. Data Fetching: Handle loading states and errors in data fetching, and display appropriate feedback to the user.

Estimated market value: $100,000/year

Phase 2

Skinstric.ai

SKINSTRIC BUILD

Phase 2 Demo

[Website](#)
[Demo Video](#)
[Figma File](#)
[Assets](#)

LEVEL 1 /

Phase 2 : Building Basic Dashboard UI Elements

Build the following UI components and assemble them into the page as shown at the Figma link on the left. Use the API endpoint on the next page, POST an image in Base64, performing all of the necessary validation checks in the frontend and display the returned data in the UI dashboard components respectively. The API returned data is simulated, so do not expect it to be accurate!

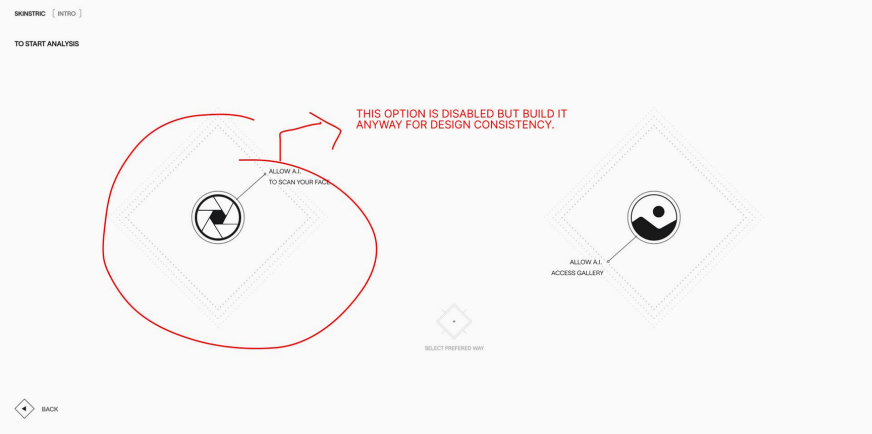


Fig 1 : First page to build in this flow is the upload image page. This is a simple attachment upload page. The button on the left is disabled and we are uploading by using the one on the right, though for consistency build it anyway.

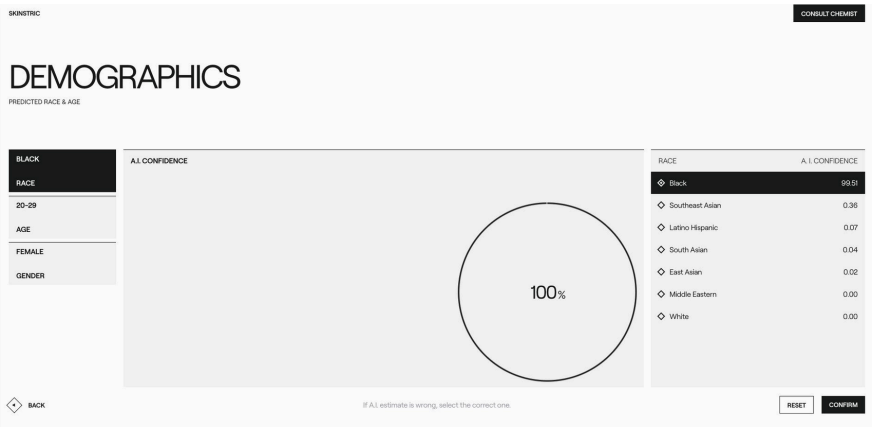


Fig 2 : The second page is using the data returned from the API endpoint on the next page (after you POST the image uploaded to it). You must parse this data clearly and display it as shown.

Skinstric.ai

SKINSTRIC BUILD

Phase 2 Demo

[Website](#)
[Demo Video](#)
[Figma File](#)
[Assets](#)

LEVEL 2 /

API Details

Upload your image in Base64 to this endpoint to be returned a randomized (simulated) list of demographic values as if the AI is scanning the face and giving you its predictions of what it thinks the face is.

POST DEMO API (not production)

https://wk7wmfz7x8.execute-api.us-east-2.amazonaws.com/live/FES_Virtual_Internship_1/level2

Input

```
data = {
    'Image': image_base64    # This is a base64 encoded string
}
```

Output


```
{
   "message":"SUCCESS: You hit the Level 2 API. Image
is a valid Base64 String.",
   "data":{
      "race":{
         "black":0.11956584717786628,
         "white":0.1280179046276461,
         "southeast asian":0.06297961651829671,
         "south asian":0.1425948353728242,
         "latino hispanic":0.0619650872094126,
         "east asian":0.2525825951799374,
         "middle eastern":0.23229411391401664
      },
      "age":{
         "20-29":0.031678993030692736,
         "30-39":0.14951751927400894,
         "40-49":0.21423285073736906,
         "10-19":0.060884420054723574,
         "50-59":0.14185781411091578,
         "3-9":0.11754071465957916,
         "60-69":0.0640062076182385,
         "70+":0.10014548458462194,
         "0-2":0.12013599592985022
      },
      "gender":{
         "male":0.520499217733165,
         "female":0.47950078226683496
      }
   }
}
```

Skinstric.ai

SKINSTRIC BUILD

Phase 2 Demo

[Website]
[Demo Video]
[Figma File]
[Assets]

LEVEL 2 /

Expected Deliverables

To pass this level we expect :

Pass

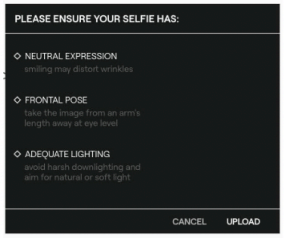
- Design matches Figma for 1920 / 1280 (either is fine depending on your screen size availability)
- Website is performant (no unnecessary CPU usage)
- User must send data to the API given. The results should be printed to console; hardcoding the results is not allowed as the API randomizes the returned values each call.

4/5 Score

- On the demographics page, scores (Race / Age / Gender) are listed in descending order and to 2 decimal places.
- Clicking on a score allows the user to update their ACTUAL attribute. E.g. if the AI says they are “Black” but they are really South Asian, clicking “South Asian” will auto update the left sidebar block.

5/5 Score

- Rotating diamond animation for both upload boxes with correct hover animation.
- Fallback for API provided such as N/A or “-”.
- Skeleton states provided for slow connection speeds.
- “Please Ensure Your Selfie Has” popup included with easeIn and easeOut animations. The rough design for this can be found here.



6/5 Score

- Check that the uploaded image is:
  - A face using this ML model in JS browserside
  - Crop the facial region using the returned bounding box and upload only that, discarding unnecessary details
  - Check the bounding box has at least 500x500 worth of pixels (other wise the face is too low resolution)

Skinstric.ai

SKINSTRIC BUILD

Phase 2 Demo

[Website]
[Demo Video]
[Figma File]
[Assets]

Tips

1. Proper State Management: The useState hook is used to manage various states, including a complex state object for demographic data. Consider using useReducer when state logic becomes complex or when the next state depends on the previous one.
2. Immutability with Immer: You're importing Immer but not currently using it. Immer is excellent for handling complex state mutations in an immutable way, which is essential for predictable state updates, especially when dealing with nested objects.
3. Efficient State Updates: When updating state based on previous state, use functional updates to ensure you're not capturing stale state. For instance, setData(prevData => ...).
4. Side Effects Handling: It's unclear if side effects are managed appropriately since there's no useEffect shown handling them. Ensure that any side effects, like API calls or subscriptions, are managed in useEffect with proper cleanup.
5. Conditional Rendering: If certain elements or components should only render under specific conditions, make sure to implement conditional rendering logic to optimize UI updates and performance.
6. Error Handling for Asynchronous Actions: When making asynchronous requests with axios within the confirm function, ensure robust error handling to manage the UI state accordingly and provide user feedback.
7. Use of Local Storage: You're using local storage to save selected options, which is fine, but always consider the security implications of storing data in the browser and ensure it's not sensitive.
8. Code Redundancy and Optimization: There is redundant code within the setActiveKeyOptionById function, such as repeated object structure for updateDataz.Demographics. Look for ways to refactor and DRY (Don't Repeat Yourself) your code.
9. Data Fetching and Redirects: In getServerSideProps, you're checking for a cookie and redirecting based on its value. This is a good use of server-side logic to control user flow, but ensure that the logic is tight and the redirects are to appropriate paths.
10. A.I. Estimates Feedback Loop: There seems to be a feedback loop where the user corrects A.I. estimates. Ensure that this data is captured accurately and securely to improve the A.I. model.
11. Responsive Design: The mediaQueryDevice state is used to conditionally render button styles. Ensure that your design is responsive and provides a consistent user experience across all devices.
12. Consistent Styling and Theming: Ensure that the styling and theming are consistent throughout the application. This includes using a consistent naming convention for CSS classes and using theme providers if available.

Estimated market value: $120,000/year

Phase 3

Skinstric.ai

SKINSTRIC BUILD

Phase 3 Demo

[Website](#)
[Demo Video](#)
[Figma File](#)
[Assets](#)

LEVEL 3 /

Level 3: Building Complex Dashboard UI Elements

These elements are more complex than the previous. Pay attention to their animations of how they easeIn and easeOut on entering and exiting the page. This requires knowledge of GSAP or alternatively anime.js to orchestrate animation of DOM elements. Ping the API endpoint shown to get your Weather data. This call will also return some user Skintype data to be used in the 2nd page in the flow. The values are simulated and randomized so do not expect it to be accurate!

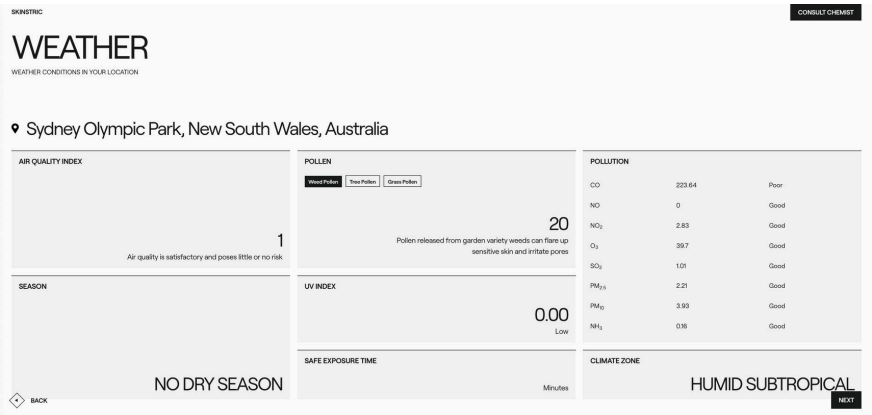


Fig 1 : This page tests your ability to work with animation orchestration and handling.

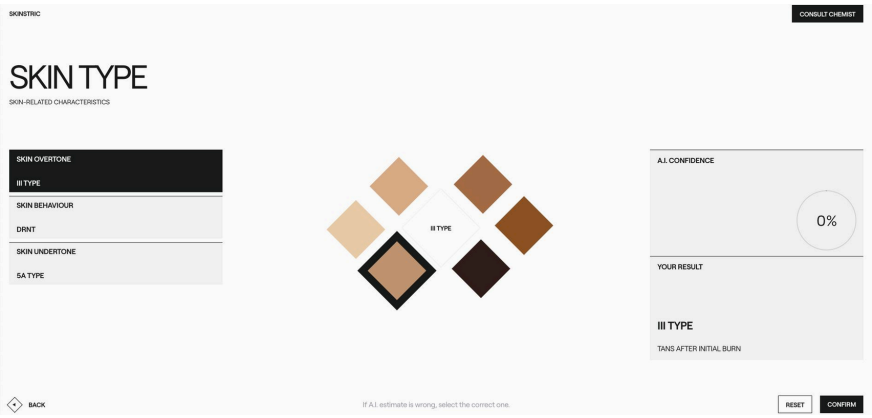


Fig 2 : Be warned this page has difficult component logic. Play around with the interactions at the demo link to fully understand how the components are expected to behave. The idea is that the AI makes a suggestion and the user is able to overwrite it if incorrect.

Skinstric.ai

SKINSTRIC BUILD

Phase 3 Demo

[Website](#)
[Demo Video](#)
[Figma File](#)
[Assets](#)

LEVEL 3 /

API Details

Upload your image in Base64 to this endpoint to be returned a randomized (simulated) list of demographic values as if the AI is scanning the face and giving you its predictions of what it thinks the face is.

POST DEMO API (not production)

https://wk7wmfz7x8.execute-api.us-east-2.amazonaws.com/live/FES_Virtual_Internship_1/level3

Input

```
data = {
    'Lat': "-33.868820",
    'Lon': "151.209290"
}
```

Output

```
NEXT PAGE
```

Skinstric.ai

SKINSTRIC BUILD

Phase 3 Demo

[Website](#)
[Demo Video](#)
[Figma File](#)
[Assets](#)

API Details


```
{
   "message":"SUCCESS: You hit the Level 3 API. Fake weather data for
Sydney, Australia",
   "data":{
      "Weather_Data":{
         "air_quality_index":229,
         "pollen":{
            "weed_pollen":"High",
            "tree_pollen":"Low",
            "grass_pollen":"High"
         },
         "pollution":{
            "CO":174.12,
            "NO":311.81,
            "NO2":330.75,
            "O3":345.05,
            "SO2":385.26,
            "PM2_5":97.73,
            "PM10":380.99,
            "NH3":381.48
         },
         "uv_index":5,
         "season":"Autumn",
         "climate_zone":"Continental",
         "safe_exposure_time":"30 minutes"
      },
      "Skintype_Data":{
         "skin_overtone":{
            "Type 1":0.31703552154900055,
            "Type 2":0.0195002386642915,
            "Type 3":0.1858322613376036,
            "Type 4":0.08820759766461125,
            "Type 5":0.07835059259920121,
            "Type 6":0.31107378818529186
         },
         "skin_behaviour":{
            "DSPW":0.03388259682692271,
            "DSPT":0.0025841177788123388,
            "DSNW":0.08953133002569298,
            "DSNT":0.08691087614314182,
            "DRPW":0.08812814795010636,
            "DRPT":0.08821745855182257,
            "DRNW":0.03601171979370639,
            "DRNT":0.07152909088016429,
            "OSPW":0.05520681172585418,
            "OSPT":0.018971547883948,
            "OSNW":0.09115673878567032,
            "OSNT":0.10725629606086917,
            "ORPW":0.05519926885807041,
            "ORPT":0.0571378174165911,
            "ORNW":0.007488937369197888,
            "ORNT":0.11078724394942953
         },
         "skin_undertone":{
            "Type 1A":0.19636607443963605,
            "Type 1B":0.37346970310429706,
            "Type 1C":0.43016422245606684,
            "Type 2A":0.5009505485239648,
            "Type 2B":0.41295433255009373,
            "Type 2C":0.08609511892594142,
            "Type 3A":0.2871645117182002,
            "Type 3B":0.5478603338354676,
            "Type 3C":0.1649751544463322,
            "Type 4A":0.580693180255659,
            "Type 4B":0.4053289166177068,
            "Type 4C":0.013977903126634196,
            "Type 5A":0.8689491057404404,
            "Type 5B":0.034734761443782396,
            "Type 5C":0.09631613281577728,
            "Type 6A":0.15290585337548873,
            "Type 6B":0.4965854008352898,
            "Type 6C":0.3505087457892215
         }
      }
   }
}
```

Skinstric.ai

SKINSTRIC BUILD

Phase 3 Demo

[Website]
[Demo Video]
[Figma File]
[Assets]

Expected Deliverables

To pass this level we expect :

Pass

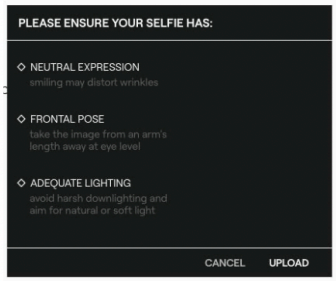
- Design matches Figma for 1920 / 1280 (either is fine depending on your screen size availability)
- Website is performant (no unnecessary CPU usage)
- User must send data to the API given. The results must be printed to console.
- Animation of intro and outro of “Weather” page must be functional with an appropriate tweening chosen.
- Skintype page should be editable and display the API response values correctly n the right places. The level 1 and level 2 of the “Skin Behaviour” and “Skin Undertone” components should be accessible correctly.

4/5 Score

- All microanimations completed :
  - Previous / Forward buttons
  - Pie charts
  - Hoverstates

5/5 Score

- Fallback for Weather / Skintype API provided such as N/A or “-”.
- Skeleton states provided for slow connection speeds.



6/5 Score

- Mobile and Tablet responsiveness built (provided in the Figma).

Skinstric.ai

SKINSTRIC BUILD

Phase 3 Demo

[Website]
[Demo Video]
[Figma File]
[Assets]

Tips

1. State Management: With various nested state structures, consider using useReducer or context for more complex state logic to make state transitions more predictable and organized.
2. Immutability Helpers: When updating state based on previous state, leverage immutability helpers like the immer library used here, which simplifies complex state updates and ensures immutability.
3. Data Transformation: Place data transformation logic outside of the component. Utilize custom hooks or utility functions to handle operations like getAccuracyArray and updateDummyData.
4. Effect Optimization: Optimize your useEffect calls with proper dependencies. Ensure effects are not running more often than needed, and each effect should have a single responsibility.
5. Animation Performance: Utilize GSAP's performance best practices, such as using transforms over top/left properties and leveraging hardware acceleration for smoother animations.
6. Server-Side Redirection: The logic in getServerSideProps for redirection should be clear and well-documented, as server-side logic can be critical for page flow and needs to be understood by all developers working on the project.
7. Accessibility and Semantics: Ensure that all interactive elements, especially custom-built components like SkinType, are accessible. This includes proper keyboard navigation and ARIA attributes.
8. UI Feedback: Provide immediate and clear feedback to the user when actions are taken. For instance, when "Reset" is clicked, there should be an indication that the action was successful.
9. Error Handling: Implement robust error handling for your API calls with axios. Ensure that failures are caught and handled gracefully, providing feedback to the user where appropriate.
10. Performance Considerations: Be mindful of the performance implications of large JSON objects and deep object structures, especially when they trigger re-renders. Use memoization techniques with useMemo to avoid unnecessary calculations and re-renders.