Next.js Design Jam 2024

Hackathon Day Instructions

Dear Q2 Students (Next.js/TypeScript)

We are thrilled to announce that our Hackathon will kick off on **Sunday, December 8th, at 12:00 AM (midnight, between Saturday and Sunday)**. This document outlines everything you need to participate successfully. Please read it carefully and follow the instructions to ensure a smooth experience.

Template Assignment

Each student will work on a specific Figma template assigned based on the last digit of their roll number. This ensures fairness and variety in the projects. If you haven't received the Figma template link, please watch Daniyal Nagori Page Last Facebook Live video. Check the table below for this template:

Last Digit of Roll Number	Template Number
0	Template 0
1	Template 1
2	Template 2
3	Template 3
4	Template 4
5	Template 5
6	Template 6
7	Template 7
8	Template 8
9	Template 9

Make sure you select the correct template according to the last digit of your roll number.

Examples to Avoid Confusion

Here are some examples to clarify how to select the appropriate template based on the last digit of your roll number:

1. **Roll Number: 00388418**
    - Last digit: **8**
    - **Template to use:** Template **8**
2. **Roll Number: 00388427**
    - Last digit: **7**
    - **Template to use:** Template **7**
3. **Roll Number: 00367029**
    - Last digit: **9**
    - **Template to use:** Template **9**
4. **Roll Number: 00367020**
    - Last digit: **0**
    - **Template to use:** Template **0**
5. **Roll Number: 00123456**
    - Last digit: **6**
    - **Template to use:** Template **6**
6. **Roll Number: 00123434**
    - Last digit: **4**
    - **Template to use:** Template **4**
7. **Roll Number: 00345012**
    - Last digit: **2**
    - **Template to use:** Template **2**
8. **Roll Number: 00345013**
    - Last digit: **3**
    - **Template to use:** Template **3**
9. **Roll Number: 00345005**
    - Last digit: **5**
    - **Template to use:** Template **5**
10. **Roll Number: 00367001**

    - Last digit: **1**
    - **Template to use:** Template **1**

Key Points to Remember

1. Always look at the **last digit** of your roll number.
2. Select the corresponding template number as per the last digit.
3. Double-check your roll number to avoid errors.

Hackathon Details

- **Start Time**: **Sunday, December 8th, 12:00 AM** (midnight, between Saturday and Sunday)
- **Duration**: 24 hours
- **Objective**: Create a pixel-perfect, responsive UI based on your assigned Figma template.

Requirements

1. **Framework**: Use Next.js with App Router (dynamic routes).
2. **Styling**: Use Custom CSS and Tailwind CSS.
3. **Backend**: Not required; focus solely on UI/UX.
4. **Libraries**: You may optionally use ShadCN or other libraries.
5. **Design Standard**: Achieve pixel-perfect and responsive design.

Permitted Resources

- **Copy-Paste**: Allowed.
- **AI Assistance**: Use tools like ChatGPT.

The goal is to simulate real-world challenges and help you deliver industry-standard designs.

Sunday Class Adjustments

- The Sunday Q2 class has been postponed to allow focus on the UI/UX Hackathon.

Submission Guidelines

1. **Deadline**: Submit your project within the allocated time (**Sunday, December 8th, 11:59 PM, GMT + 5**).
2. **Extended Deadline**: If you encounter issues, request an extension before the end of the hackathon for an additional 6 hours (until December 9th Monday, 6:00 AM).
3. **Submission Format**: Use the link https://forms.gle/vqEsQPaW8btsvnTC7 to submit:
   - Source code repository link (GitHub or similar).
   - Hosted link (Vercel, Netlify, etc.).
   - A brief description of your approach.
4. **Evaluation Criteria (Points 250)**:
   - Pixel-perfect implementation (75 points).
   - Responsiveness. (100 points)
   - Adherence to the Figma template (75 points).

Step-by-Step Guide

Step 1: Understand Your Assignment

- **Check Roll Number**: Match the last digit of your roll number to your assigned template.
- **Access the Figma Template**:
  - **Accessing Figma**: Make sure you have access to the Figma template. Use the shared link provided to open the design.
  - **Understand the Design**: Review all design elements such as spacing, fonts, colors, layout, and interactions.
  - **Measure and Inspect**: Use Figma's inspector tool to measure margins, paddings, and font sizes. Familiarize yourself with each component to accurately replicate the design.

Step 2: Set Up Your Development Environment

- **Install Tools**:
    - **Node.js**: Install the latest LTS version. Official Documentation
    - **Next.js**: Set up your Next.js project:

```
npx create-next-app@latest my-project –typescript
cd my-project
```

Next.js Documentation

-
    - **Tailwind CSS**: Install and set up Tailwind CSS by following the official documentation:Tailwind CSS Documentation
    - **Custom CSS**: You may also create custom styles by adding a globals.css file under the /styles directory.
- **Organize Your Project**:
- **Install Optional Libraries** (e.g., ShadCN):

```
npm install shadcn
```

Step 3: Start Coding

- **Layout & Structure**: Use the Next.js App Router for dynamic routes.
    - **Dynamic Routes**: Next.js allows for creating dynamic routes easily by using square brackets in the /apps directory. https://nextjs.org/docs/app/building-your-application/routing/dynamic-routes
    - **App Router**: In Next.js, the App Router is used to handle navigation between routes. By using next/link, you can easily create a seamless transition between routes. https://nextjs.org/docs/app/building-your-application/routing/linking-and-navigating
- **Styles**: Use Tailwind CSS for a consistent and responsive design.

```
<div className="max-w-sm rounded overflow-hidden shadow-lg">
  <img className="w-full" src="/img/product.jpg" alt="Product Image" />
  <div className="px-6 py-4">
    <div className="font-bold text-xl mb-2">Product Title</div>
    <p className="text-gray-700 text-base">Product description here.</p>
  </div>
</div>
```

-
    - **Custom CSS**: For custom styles that cannot be achieved with Tailwind, add them in globals.css and ensure consistency with the Figma template.
- **Responsiveness**: Tailwind makes it easy to build responsive designs by using utility classes.

- **Testing Responsiveness**:
  - Use Tailwind's responsive utilities like sm:, md:, lg:, xl:, and 2xl: to apply styles at different breakpoints.
  - Example:

```
<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4">
  <!-- Items here -->
</div>
```

- **Breakpoints**:
  - Tailwind's default breakpoints are:
    - sm: 640px (small screens)
    - md: 768px (medium screens)
    - lg: 1024px (large screens)
    - xl: 1280px (extra large screens)
    - 2xl: 1536px
- **Responsive Design Details**:
  - **Mobile (481px - 768px)**: Use a balanced layout with increased spacing between touch elements. Components should be stacked vertically, maintaining a clean hierarchy. Ensure interactive elements are touch-friendly with a minimum size of 48px x 48px and enough space to avoid accidental touches.
  - Test the design on actual mobile devices, not just emulators, to ensure the touch elements are user-friendly.
  - **Tablet (769px - 1024px)**: Transition from a single-column to a two-column layout where appropriate. Increase font sizes for better readability (16px to 18px for body text). Maintain ample spacing between elements, ensuring the layout feels less cramped compared to mobile.
  - **Laptops (1025px - 1440px)**: Utilize a three-column grid where applicable to maximize screen real estate. Increase spacing and padding for a clean, professional layout. Ensure navigation remains prominent and intuitive.
  - **Large Desktops (1441px and above)**: Use a four-column grid or other wide layouts to effectively utilize available space. Incorporate adaptive typography: increase font sizes proportionally

(e.g., 20px for body, 28px for headers). Introduce larger images and high-resolution assets to maintain visual quality on larger screens.

- **Breakpoints Consistency**:
  - Clearly define your breakpoints in CSS for consistency across projects:

```
@media (max-width: 480px) { /* Mobile Small */ }
@media (min-width: 481px) and (max-width: 768px) { /* Mobile Large */ }
@media (min-width: 769px) and (max-width: 1024px) { /* Tablet */ }
@media (min-width: 1025px) and (max-width: 1440px) { /* Small Laptop */ }
@media (min-width: 1441px) { /* Large Desktop */ }
```

- **White Space Utilization**:
  - Larger screens (e.g., desktops) benefit from generous white space to prevent designs from feeling too cluttered.
- **Navigation Adjustments**:
  - For tablets and larger screens, transition from collapsible menus (hamburgers) to visible horizontal navigation.
- **Testing**:
  - Emphasize the importance of testing on real devices to ensure responsiveness behaves as expected across all sizes.

Step 4: Test & Refine

- **Pixel-Perfect Design**: Compare your implementation with the Figma design. Consider using tools or browser extensions like PerfectPixel to help achieve pixel-perfect accuracy.
  - Use tools like Figma's Inspector and browser developer tools to measure spacing, font sizes, and colors.
- **Responsive Testing**: Test across various screen sizes using browser DevTools and real devices to verify consistent design quality.

Step 5: Submit Your Project

- **Prepare Repository**: Push to GitHub or similar with a detailed README.md file.
  - **README.md** should include:
    - Project purpose.
    - Key components implemented.
    - Steps to run the project locally.

- **Submit Link**: Use the Google form link provided above.

Guidelines & Best Practices

1. **Code Quality**: Write clean, modular, and well-commented code.
   - Example:

```
// components/Header.tsx
export default function Header() {
  return (
    <header className="bg-blue-500 text-white py-4">
      <h1 className="text-center text-2xl">My E-commerce App</h1>
    </header>
  );
}
```

2. **Design Accuracy**: Align your design pixel-for-pixel with the Figma template.
3. **Responsiveness**: Ensure compatibility across mobile, tablet, and desktop screens.
4. **No Collaboration**: This is an individual project.
5. **Permitted Tools**: AI tools and code snippets are allowed as long as requirements are met.
6. **Creativity**: Enhance the design where possible, but meet all requirements first.

FAQs

1. **What if I don't understand my template?** Read document carefully.
2. **Can I collaborate with another student?** No, collaboration is not allowed.
3. **Is using external libraries mandatory?** No, it's optional.
4. **What if I miss the deadline?** Late submissions are accepted only if you have prior approval.
5. **Can I add extra features?** Yes, but ensure required elements are perfect first.
6. **Can I use ChatGPT or copy code?** Yes, as long as requirements are fulfilled.
7. **How can I ensure my design is pixel-perfect?** Follow the Figma template and use browser tools to test responsiveness.
8. **What if I encounter technical issues?** Reach out to the discussion forum.

9. **How do I ensure my project is responsive?** Use Tailwind's responsive utilities and test on multiple devices and screen sizes.
10. **Can I use other frameworks like React or Angular?** No, you must use Next.js as specified.
11. **Do I need to deploy the project?** Yes, use platforms like Vercel or Netlify.
12. **Can I use pre-built components?** Yes, as long as they match the Figma design.
13. **What is the primary goal of this hackathon?** To create a pixel-perfect, responsive UI.
14. **How do I measure padding and spacing in Figma?** Use Figma's Inspect tool to get exact values.
15. **Can I ask for design feedback during the hackathon?** Yes, reach out to faculty for clarification but not for full reviews.
16. **Is there a penalty for late submissions?** Yes, unless prior approval is obtained.
17. **How detailed should the README file be?** Include project purpose, key components, and steps to run it.
18. **What if I don't finish all features?** Submit what you have completed by the deadline.
19. **How can I make my design stand out?** Follow best practices, focus on details, and add subtle enhancements.
20. **Do I need to write tests?** It's not mandatory.
21. **Can I use paid assets?** No, use only free or self-created assets.
22. **How do I deploy using Vercel?** Follow the official Vercel deployment guide for Next.js.
23. **Can I use CSS frameworks other than Tailwind?** No, Tailwind is required.
24. **What if my GitHub repository is private?** Repo should be public.
25. **Can I change the assigned Figma template?** No, each student must use their assigned template.
26. **What should I do if I encounter build errors?** Debug using browser console and Next.js error messages.
27. **How do I create dynamic routes in Next.js?** Use [param].tsx syntax in the /app directory.

28. **What if I don't know how to use Tailwind CSS?** Refer to the Tailwind documentation and examples provided.
29. **Can I use other CSS preprocessors like SASS?** No, stick to Tailwind and custom CSS.
30. **What should I include in the project description?** Mention your approach, libraries used, and any customizations.
31. **Is it okay to use animations?** Yes, as long as they align with the design and enhance UX.
32. **Can I add more pages than the Figma template?** Yes, but prioritize completing the required pages first.
33. **What if I don't understand Tailwind's utility classes?** Use the Tailwind documentation or online cheat sheets.
34. **Can I work offline?** Yes, but ensure you commit and push your code.
35. **Do I need to host the project if it's not complete?** Yes, deploy whatever you have completed.
36. **How do I handle images in the project?** Use optimized images and add alt attributes for accessibility.
37. **Should I consider accessibility (a11y)?** Yes, follow basic accessibility guidelines for better usability.
38. **Can I use TypeScript features extensively?** Yes, TypeScript is encouraged, especially for type safety.
39. **How do I test the responsiveness of my design?** Use browser DevTools, real devices, and online testing tools.
40. **What should I do if I need more time?** Request an extension before the deadline.
41. **Can I use form libraries for contact forms?** Yes, if forms are part of your template, you may use form libraries.
42. **Should I use absolute or relative units for CSS?** Prefer relative units like rem, em, % for better responsiveness. You can use px.
43. **How do I use Next.js Link for navigation?** Import Link from next/link and use it for client-side routing.
44. **Do I need to use ShadCN?** No, it is optional, use it only if you find it helpful.

45. **Can I refactor my code after submission?** No, only the code submitted before the deadline will be evaluated.
46. **Will I lose marks for using AI tools?** No, as long as requirements are met, AI tools are allowed.
47. **How do I add SEO to my Next.js project?** Use the next/head component to add meta tags.
48. **How do I handle errors in dynamic routing?** Use conditional checks to ensure the data is available before rendering.

Closing Note

This hackathon is a great chance to show off your skills and creativity while building industry-standard designs. We encourage you to take full advantage of the available resources, focus on quality, and have fun!

Best of luck,

Dean of Faculty Governor IT Initiative