Of course. For **Practical 2**, the focus is on documenting the responsive design and CSS layout techniques you used. Your goal is to explain *how* and *why* you made your website look good on different screen sizes.

Here is a guide on what to write in your practical file for Practical 2, using your "ByteBazaar" project as the example.

Practical 2: Responsive Layout with CSS

Aim: To design and implement a fully responsive layout for the ByteBazaar website's key pages (Home, About, Registration) using modern CSS techniques like Flexbox and Grid.

1. Key Questions & Analysis

1.1. How does the layout change with screen size?

The website layout adapts to different screen sizes to ensure optimal viewing and usability. This is achieved using CSS **media queries**, which apply different styles at specific screen width breakpoints.

- **Desktop (Large Screens > 1024px):** The layout is wide, often using multi-column grids for content like product listings and the footer. The main navigation links are displayed horizontally.
- **Tablet (Medium Screens < 1024px):** The layout starts to condense. For example, on the authentication pages, the two-column design collapses into a single column to better fit the screen.
- **Mobile (Small Screens < 768px):** This sees the most significant changes. The main navigation collapses into a "hamburger" menu. Grids reduce to a single column so users can scroll vertically. Font sizes and spacing are adjusted for readability on small screens.

Example Media Query from style.css:
This code snippet shows how the product grid is changed for smaller screens.

CSS

```css
/* From style.css */
@media (max-width: 768px) {
    .nav-search {
        order: 3;
        flex-basis: 100%;
        margin: 15px 0 0 0;
    }

    .nav-links {
        display: none; /* Hides the links to be replaced by a menu */
        flex-direction: column;
        position: absolute;
        top: 100%;
        left: 0;
    }

    .nav-toggle {
        display: flex; /* Shows the hamburger menu icon */
    }

    .hero-container {
        grid-template-columns: 1fr; /* Stacks hero content vertically */
        text-align: center;
    }
}
```

1.2. Which layout approach is used and why?

The project primarily uses a combination of **CSS Flexbox** and **CSS Grid** because they are modern, powerful, and efficient for creating complex and responsive layouts.

- **CSS Flexbox:** Used for one-dimensional layouts, like aligning items in the navigation bar, centering content within cards, and arranging action buttons. It is ideal for distributing space along a single axis.
- **CSS Grid:** Used for two-dimensional layouts, such as the main product grid (.products-grid), the team section on the "About Us" page, and the overall structure of the contact page. It excels at aligning content into rows and columns simultaneously.

Example of CSS Grid from style.css:
This rule creates a responsive grid for products that automatically adjusts the number of columns based on the available width.

CSS

```css
/* From style.css */
.products-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
    gap: 30px;
}
```

1.3. Are color schemes and fonts readable and user-friendly?

Yes. The design prioritizes readability and a clean aesthetic.

- **Color Scheme:** A consistent and accessible color palette is defined using CSS variables for easy management. The primary color (--primary-color: #ff6b35;) is used for key actions and branding, while neutral colors (--text-color: #333;, --background-color: #ffffff;) are used for text and backgrounds to ensure high contrast and readability.
- **Typography:** The 'Inter' font family is used throughout the site. It is a clean, sans-serif font known for its excellent readability on screens of all sizes. Font weights and sizes are used hierarchically to distinguish headings from body text.

2. Responsive Views (Post-Laboratory Work)

For this section, you should include screenshots of your website at different screen sizes to visually demonstrate its responsiveness.

Homepage - Desktop View
[Insert a full-page screenshot of your index.html on a desktop screen]
Homepage - Mobile View
[Insert a screenshot of your index.html on a mobile screen, showing the hamburger menu]
Registration Page - Desktop View
[Insert a screenshot of register.html on a desktop screen]
Registration Page - Mobile View
[Insert a screenshot of register.html on a mobile screen, showing the single-column layout]
By providing these code examples and visual proof, you effectively document your successful implementation of Practical 2.