Bill Splitter Project Instructions

1. index.html

Create the HTML structure for a simple bill-splitting tool using the following specifications:

- The page must include these core HTML elements: `<html>`, `<head>`, `<title>`, `<link>`, `<body>`, `<div>`, `<button>`, `<input>`, `<select>`, `<label>`, `<script>`.
- The `<head>` should include:
  - A `<title>` tag with the text "Bill Splitter"
  - A link to the external CSS file: `style.css`
- Inside the `<body>`, add a top-level container `<div class="container">` to hold all elements.
- Include a **theme toggle button** (`<button id="toggle-theme">`) with the text ("Light" for light mode, "Dark" for dark mode).
- Add three main input fields:
  - `<input type="number" id="amount" value=”100.00”>` for Bill Amount
  - `<input type="number" id="tip" value=”10”>` for Tip percentage
  - `<select id="people">` to choose number of people (1–10)
- Below these, add a **Payment Mode** section using radio buttons inside a `<div class="payment-options">`.
- Add a **Calculate** button: `<button id="calculate">Calculate</button>`
- Below the button, display the results inside a `<div class="output">`:
  - One `<p>` to show Tip Amount

    - One `<p>` to show Amount Per Person
- Link the external JavaScript file: `script.js` at the end of the body.

---

2. style.css

Add styles for the layout and appearance of the Bill Splitter app:

- Use the universal selector `body` to reset and apply base styles (e.g., font-family, margin, padding).
- Center the `.container` and apply background color, padding, border-radius, and box-shadow.
- Style form elements:
    - Inputs (`<input type="number">`) and `<select>` should be full-width with padding and border styling.
- Style the **Calculate** button with:
    - Consistent size
    - Background color
    - Rounded corners
    - Hover effect
- Style the **output section** (`.output`) with bold text and spacing.
- Use CSS to handle **theme switching**:
    - Default is light mode
    - Add a `.dark-mode` class to the body for dark mode styling
- Theme toggle button (`#toggle-theme`) should be positioned top-right inside `.container`.

---

3. script.js

Write the JavaScript logic to calculate and display results, and handle theme switching:

- Use `document.getElementById()` to access elements.
- Declare a function `calculateBill()` to:
  - Read input values: bill amount, tip %, and number of people
  - Compute the tip amount and total amount per person
  - Update the result fields (`#tip-amount` and `#per-person`)
- Define a `toggleTheme()` function to:
  - Toggle the `dark-mode` class on the `body`
  - Update the text on the toggle button
- Add a `fillPeopleDropdown()` function
- Use event listeners:
  - On `#calculate` button to trigger calculation
  - On `#toggle-theme` to switch themes
- Use a DOMContentLoaded event to initialize everything when the page loads.

Detailed HTML Structure Guide for Bill Splitter App

1. Document Declaration and Root Elements

Defines the type and language of the document:

- Starts with a `<!DOCTYPE html>` declaration to indicate HTML5.
- Opens the `<html>` tag with a language attribute (`lang="en"`) to define the document language.

2. Head Section (`<head>`)

Contains metadata and external links:

- Includes a `<meta charset="UTF-8">` tag to support a wide range of characters.
- Adds a `<meta name="viewport" content="width=device-width, initial-scale=1.0">` tag for responsive scaling on mobile devices.
- Uses a `<title>` tag with the text "Bill Splitter" as the page title.
- Links to the external stylesheet using a `<link>` tag with `rel="stylesheet"` and `href="style.css"`.

3. Body Element (`<body>`)

Holds all visible page content:

- Assigns a default class of `light-mode` to the `<body>` to enable theme toggling via CSS.

4. Main Container (`<div class="container">`)

Acts as a wrapper for all interface elements:

- Uses a class name “container” to apply card-like layout and styling.

5. Heading (`<h1>`)

Displays the app title:

- Contains the text “Bill Splitter” to represent the main heading.

6. Theme Toggle Button (`<button id="toggle-theme">`)

Provides a button to toggle light/dark themes:

- Uses an ID of `toggle-theme` for JS and styling reference.
- Optionally adds a `title` attribute like “Toggle Theme” for tooltip text on hover.

7. Bill Amount Input Section

Accepts user input for the bill:

- Uses a `<label>` with a `for` attribute set to “amount” and text like “Bill Amount”.
- Follows with an `<input>` element:
  - Sets `type="number"` for numeric entry.
  - Uses `id="amount"` to link with the label and for JavaScript reference.
  - Sets a default value 100.00 for testing.

8. Tip Percentage Input Section

Accepts the desired tip percentage:

- Uses a `<label>` with `for="tip"` and the text “Tip (%)”.
- Follows with an `<input>`:

    - Sets `type="number"`.
    - Uses `id="tip"` for label binding and script access.
    - Sets a default value 10.

9. People Dropdown Section

Allows user to select how many people will split the bill:

- Adds a `<label>` with `for="people"` and text “Number of People”.
- Follows with a `<select>` element:
    - Sets `id="people"` for identification and event access.
    - Includes `<option>` elements for values 1 to 10.
    - Pre-selects “2” by default using the `selected` attribute.

10. Payment Mode Selection

Provides radio buttons to choose how the payment will be made:

- Adds a `<label>` with just the text “Payment Mode” above the group.
- Wraps the radio buttons inside a `<div class="payment-options">`.
- Each option includes:
    - A nested `<label>` with an `<input type="radio">` inside.
    - Radio buttons share the same `name="payment"` so only one can be selected.
    - Each input has a unique `value` such as “Cash”, “Card”, or “UPI”.
    - “Cash” is selected by default using the `checked` attribute.

11. Calculate Button (`<button id="calculate">`)

Triggers the bill-splitting calculation:

- Uses a `button` with the ID `calculate` for JavaScript interaction.
- Button text is “Calculate”.

12. Output Section (`<div class="output">`)

Displays the results of the calculation:

- Uses a `div` with class “output” to group the result values.
- Includes two `<p>` elements:
    - First line: shows “Tip Amount” followed by a `<span>` with ID `tip-amount`.
    - Second line: shows “Amount Per Person” followed by a `<span>` with ID `per-person`.
- Both spans are initialized with placeholder values (e.g., ₹0.00) that will be updated by JavaScript.

13. JavaScript Link (`<script src="script.js">`)

Connects the external JavaScript file:

- Placed just before the closing `</body>` tag.
- Uses a `<script>` tag with a `src` attribute set to “script.js”.

14. Closing Tags

Ends the HTML structure:

- Closes the `</div>`, `</body>`, and `</html>` tags in proper order.

Detailed CSS Styling Guide for Bill Splitter App

This guide explains how to style the `style.css` file for the Bill Splitter project. Each section includes the **purpose** of the styles and **how they contribute** to the visual design and usability.

1. Body Styling

Applies base styling and centers the layout:

- Uses a sans-serif font like "Arial" for better readability.
- Adds outer padding around the page to avoid content sticking to the edges (2rem).
- Centers the app horizontally using Flexbox.
- Sets a light neutral background color for the page (#f5f5f5).
- Enables smooth transitions for color and background when switching themes (0.3s duration).

2. App Container (.container)

Defines the main card-style box that wraps all UI elements:

- Uses a solid white background for visual clarity.
- Adds internal padding for breathing space (2rem).
- Applies rounded corners for a soft, modern feel (12px).
- Includes a subtle shadow for depth and elevation (slight blur and transparency).
- Limits its width to a manageable size for mobile devices (maximum width of 300px).
- Stretches to full width inside its container (100% width).
- Uses relative positioning to support absolutely-positioned child elements.

3. Heading (h1)

Styles the title text at the top:

- Aligns the text to the center of the container.
- Adds bottom spacing to separate the heading from the next section (1.5rem).

4. Labels (label)

Styles labels above input fields for clarity:

- Displays labels as block-level elements so they appear on their own line.
- Adds vertical spacing above each label (1rem).
- Uses bold font weight to distinguish labels from other text.

5. Input Fields and Dropdowns (input[type="number"], select)

Ensures consistent design for number inputs and selection fields:

- Makes fields stretch across the full container width (100%).
- Adds internal padding for comfort (0.5rem).
- Separates the field slightly from its label with top margin (0.3rem).
- Applies light grey borders (#ccc) for subtle definition.
- Rounds the field edges slightly for a modern look (6px corner radius).

6. Calculate Button (#calculate)

Designs the primary action button clearly and accessibly:

- Makes the button as wide as its container (100% width).
- Adds padding to increase tap target area (0.7rem).

- Uses a clean sans-serif font with medium size (around 1rem).
- Applies a dark background color for visibility (#2c3e50).
- Sets the text color to white for high contrast.
- Removes default borders and applies rounded corners (6px).
- Changes the cursor to a pointer to signal interactivity.
- Adds spacing above the button to separate it from the fields (around 1.5rem).

7. Calculate Button Hover State

Adds feedback on user interaction:

- Slightly darkens the button background on hover to indicate it's clickable (#1a252f).

8. Payment Mode Section (.payment-options)

Styles the row of radio buttons:

- Adds top margin for spacing from the section above (0.5rem).
- Lays out labels in a row using horizontal spacing.
- Distributes space evenly between payment options using Flexbox.

9. Output Section (.output)

Displays calculated results clearly:

- Adds vertical spacing above the output area (around 1.5rem).
- Uses bold text to highlight the results like tip and per-person share.

10. Theme Toggle Button (#toggle-theme)

Styles the icon-based toggle in the top-right corner:

- Positions the button absolutely within the container.
- Places it near the top-right edge (around 1rem from top and right).
- Removes background and border for a clean icon-only appearance.
- Increases font size for emoji visibility (1.2rem).
- Sets the cursor to pointer to indicate clickability.

11. Dark Mode Styling (body.dark-mode)

Overrides default theme colors when dark mode is activated:

- Changes the page background to a dark tone (#1e1e1e).
- Updates the container background to a darker gray (#2b2b2b).
- Switches text color across the app to white for contrast.
- Changes form field backgrounds to dark gray shades (#444) and borders to slightly lighter grays (#555).
- Updates the calculate button to a warm orange color (#e67e22).
- Changes the button hover color to a slightly deeper orange (#cf711b).

Detailed JavaScript Logic Guide for Bill Splitter App

This guide explains the core JavaScript functionality implemented in `script.js`. Each section describes the role of specific functions and logic used to power the app's interactivity.

1. Initialization Wrapper

Encapsulates all functionality inside an immediately-invoked function:

- Starts with a self-executing function to avoid polluting the global scope.
- Groups all internal functions together for better organization and separation from other scripts.
- Ensures everything runs after the DOM is fully loaded using `DOMContentLoaded` or direct invocation if already ready.

2. Utility Function: `getEl(id)`

Simplifies element access:

- Accepts an element ID as input.
- Returns the element using `document.getElementById`.

3. Main Logic Function: `calculateBill()`

Handles the core calculation logic:

- Gets values from the input fields:
  - Bill amount from the `amount` input.
  - Tip percentage from the `tip` input.
  - Number of people from the `people` dropdown.
- Retrieves output span elements where the results will be shown.
- Converts all values to numbers:
  - Uses `parseFloat` for amount and tip.
  - Uses `parseInt` for number of people.
  - Falls back to default zero if values are invalid.
- Calculates:

    - Tip amount as a percentage of the bill.
    - Total amount (bill + tip).
    - Amount per person (total divided by number of people).
- Updates the inner text of result spans with:
    - Tip amount formatted to two decimal places.
    - Per-person share formatted similarly.
    - Both values prefixed with the Rupee symbol (₹).

4. Theme Toggle Function: `toggleTheme()`

Switches between light and dark UI themes:

- Gets a reference to the toggle button using its ID.
- Adds or removes a `dark-mode` class on the `body` element.
- Updates the label text inside the button depending on current theme:
    - Uses the word "Dark" for dark mode.
    - Uses the word "Light" for light mode.
- Helps users easily toggle visual preferences.

5. Dropdown Setup Function: `fillPeopleDropdown()`

Fills the "Number of People" dropdown dynamically:

- Checks if the people dropdown already contains options to avoid duplicates.
- If empty, creates `<option>` elements for values from 1 to 10.
- Each option's value and display text is the same (e.g., "3").
- Sets the default selected value to "2" (a common case for two people splitting a bill).

6. Event Binding Function: `setupEvents()`

Links button actions to their corresponding functions:

- Adds a click event listener to the **Calculate** button:
    - When clicked, it triggers the bill calculation function.
- Adds a click event listener to the **Theme Toggle** button:
    - When clicked, it toggles between light and dark modes.

7. Initialization Function: `init()`

Coordinates app setup on page load:

- Calls `fillPeopleDropdown()` to populate the people dropdown.
- Calls `setupEvents()` to make the buttons interactive.

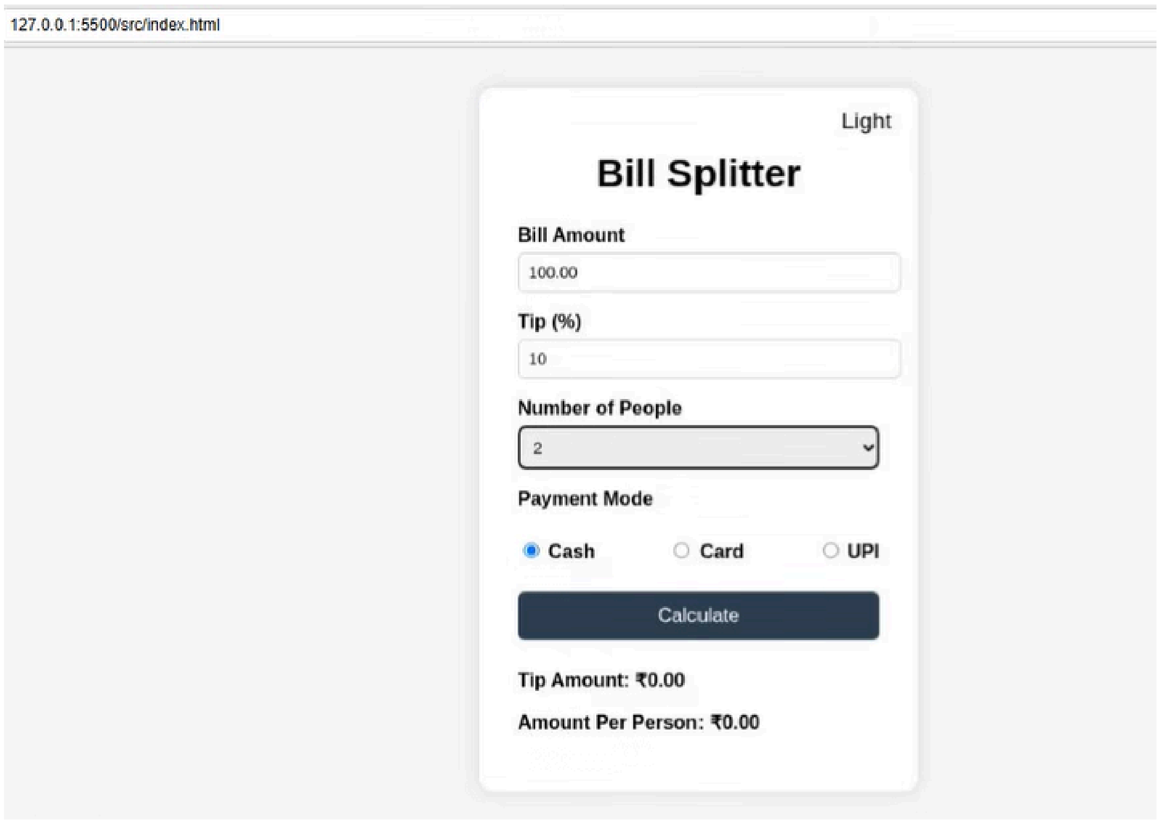
8. DOM Ready Check and Execution

Ensures the script runs only when the document is ready:

- Uses `document.readyState` to detect page load state.
- If still loading, waits for the `DOMContentLoaded` event.
- Otherwise, runs the `init()` function immediately.

ScreenShots:

*****Light Mode*****



127.0.0.1:5500/src/index.html

Light

Bill Splitter

Bill Amount: 100.00

Tip (%): 10

Number of People: 2

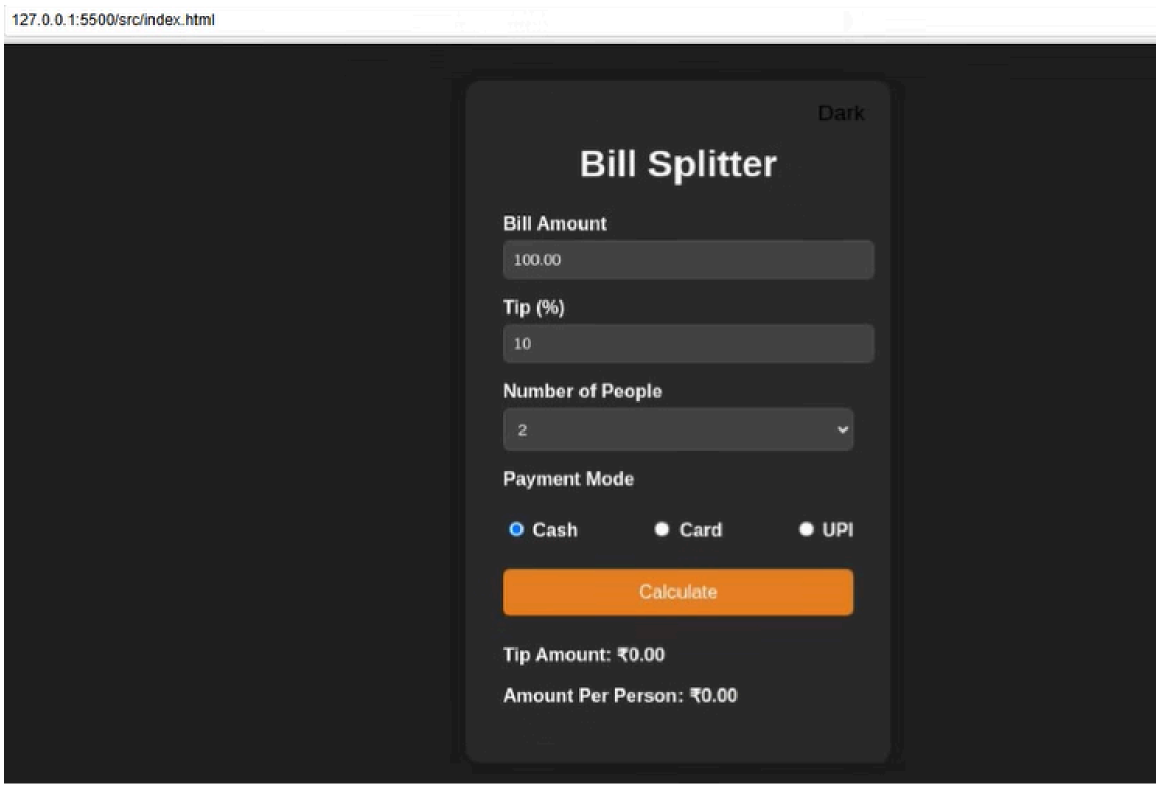
Payment Mode: [✓ Cash] Card UPI

Calculate

Tip Amount: ₹0.00

Amount Per Person: ₹0.00

Dark Mode


127.0.0.1:5500/src/index.html

Dark

Bill Splitter

Bill Amount: 100.00

Tip (%): 10

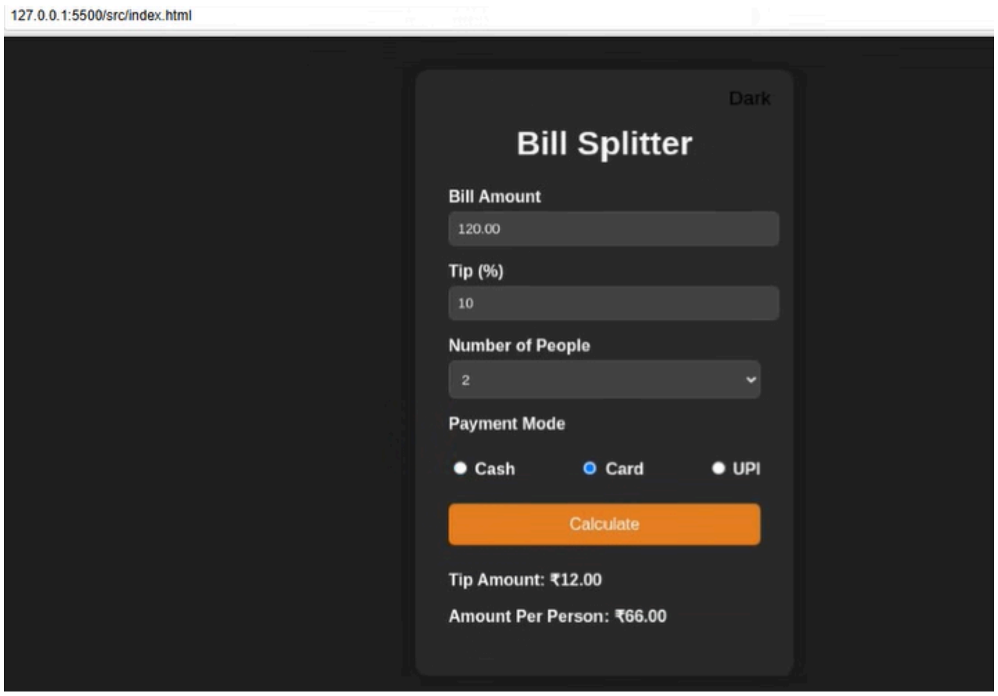
Number of People: 2

Payment Mode: [✓ Cash] Card UPI

Tip Amount: ₹0.00

Amount Per Person: ₹0.00

Calculation


127.0.0.1:5500/src/index.html

Dark

Bill Splitter

Bill Amount: 120.00

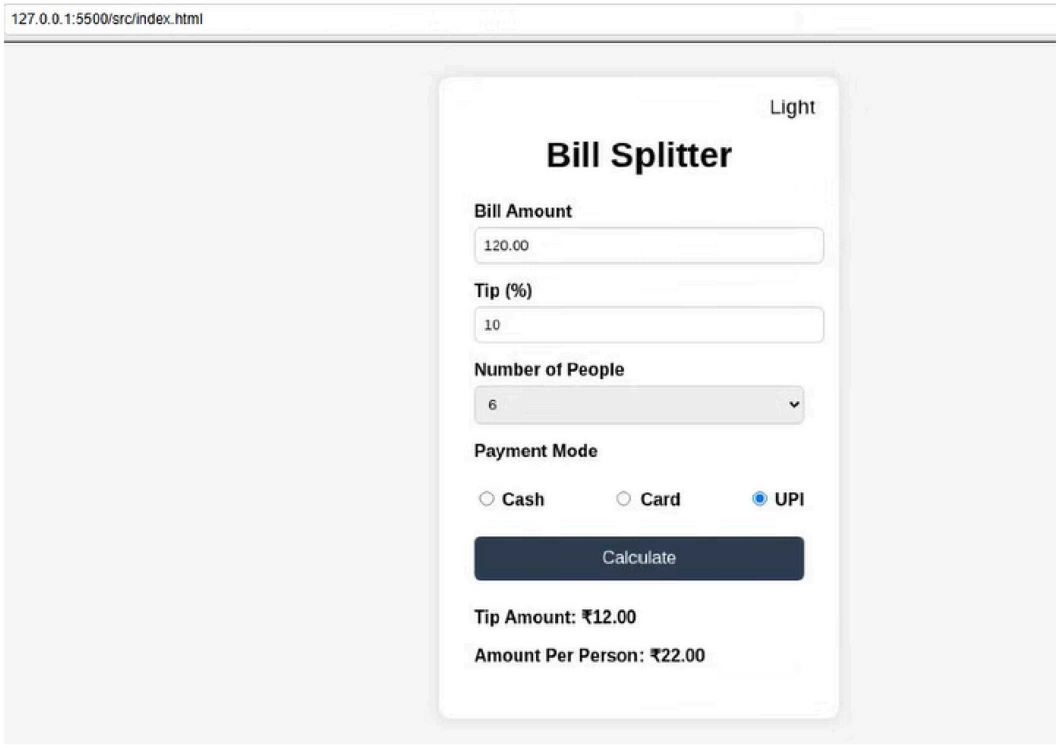
Tip (%): 10

Number of People: 2

Payment Mode: Cash [✓ Card] UPI

Tip Amount: ₹12.00

Amount Per Person: ₹66.00


127.0.0.1:5500/src/index.html

Light

Bill Splitter

Bill Amount: 120.00

Tip (%): 10

Number of People: 6

Payment Mode: Cash Card [✓ UPI]

Tip Amount: ₹12.00

Amount Per Person: ₹22.00

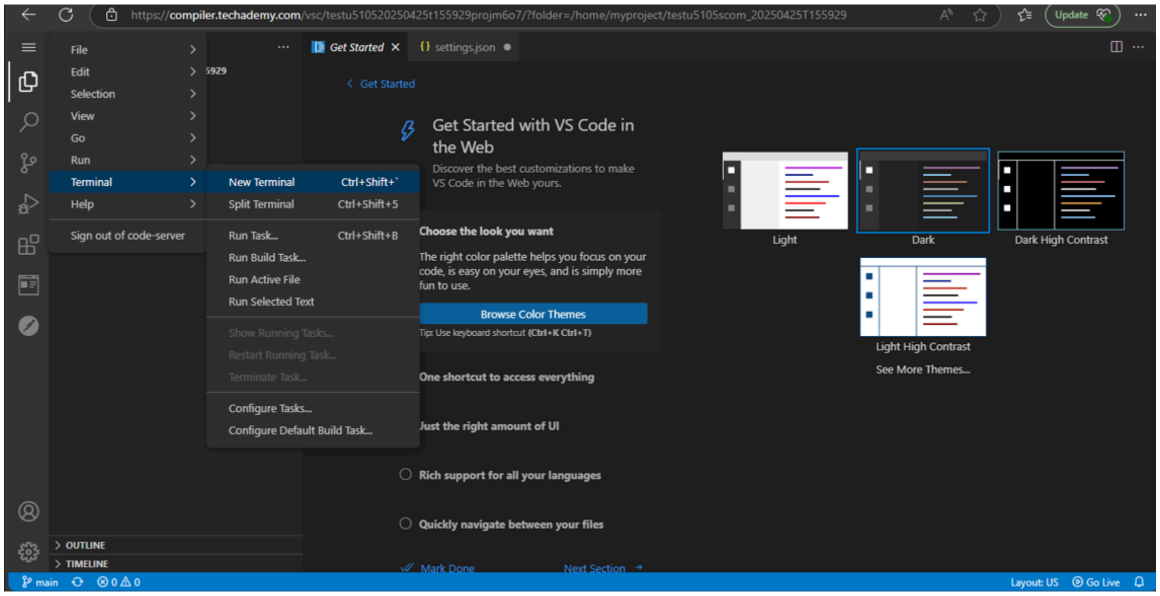
Assessment Guidelines

Step 1:

- Once the VS Code interface loads in the browser, wait until you see the workspace and left sidebar.
- To open the command terminal the test takers, need to go to

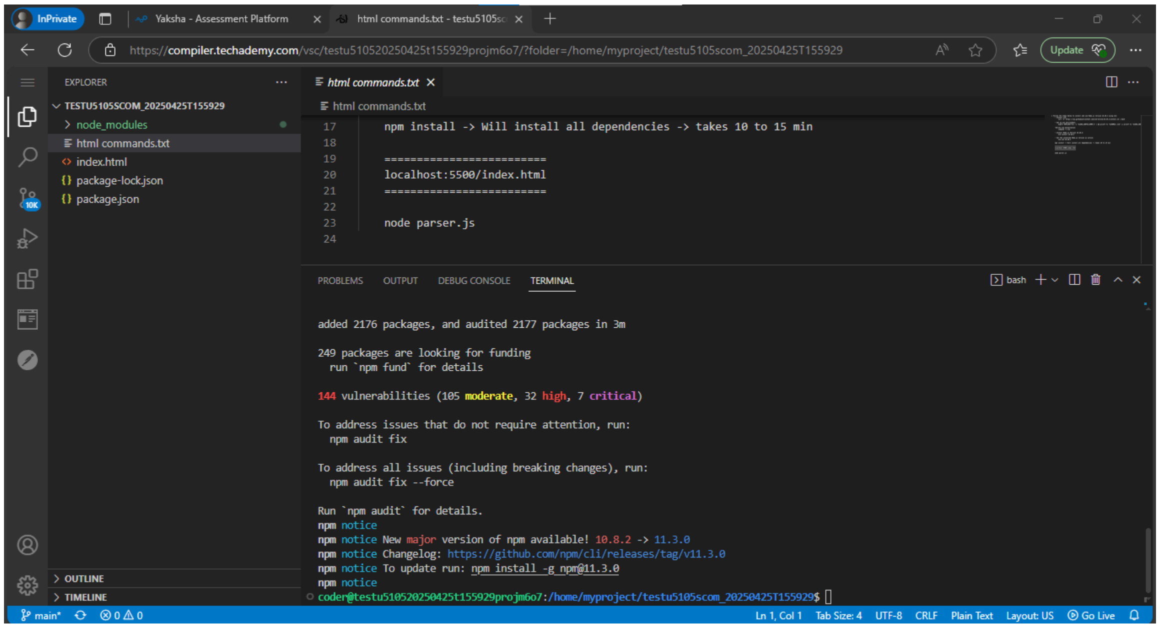
  Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.

Now in the terminal you need to install all dependencies using the **"npm install"** command.


https://compiler.techademy.com/vsc/testu510520250425t155929projm6o7/?folder=/home/myproject/testu5105scom_20250425T155929
Update
File
Edit
Selection
View
Go
Run
Terminal
Help
Sign out of code-server
New Terminal Ctrl+Shift+`
Split Terminal Ctrl+Shift+5
Run Task... Ctrl+Shift+B
Run Build Task...
Run Active File
Run Selected Text
Show Running Tasks...
Restart Running Task...
Terminate Task...
Configure Tasks...
Configure Default Build Task...
Get Started
settings.json
Get Started
Get Started with VS Code in the Web
Discover the best customizations to make VS Code in the Web yours.
Choose the look you want
The right color palette helps you focus on your code, is easy on your eyes, and is simply more fun to use.
Browse Color Themes
Tip: Use keyboard shortcut (Ctrl+K Ctrl+T)
One shortcut to access everything
Just the right amount of UI
Rich support for all your languages
Quickly navigate between your files
Mark Done
Next Section
Light
Dark
Dark High Contrast
Light High Contrast
See More Themes...
OUTLINE
TIMELINE
main
0 0
Layout: US
Go Live

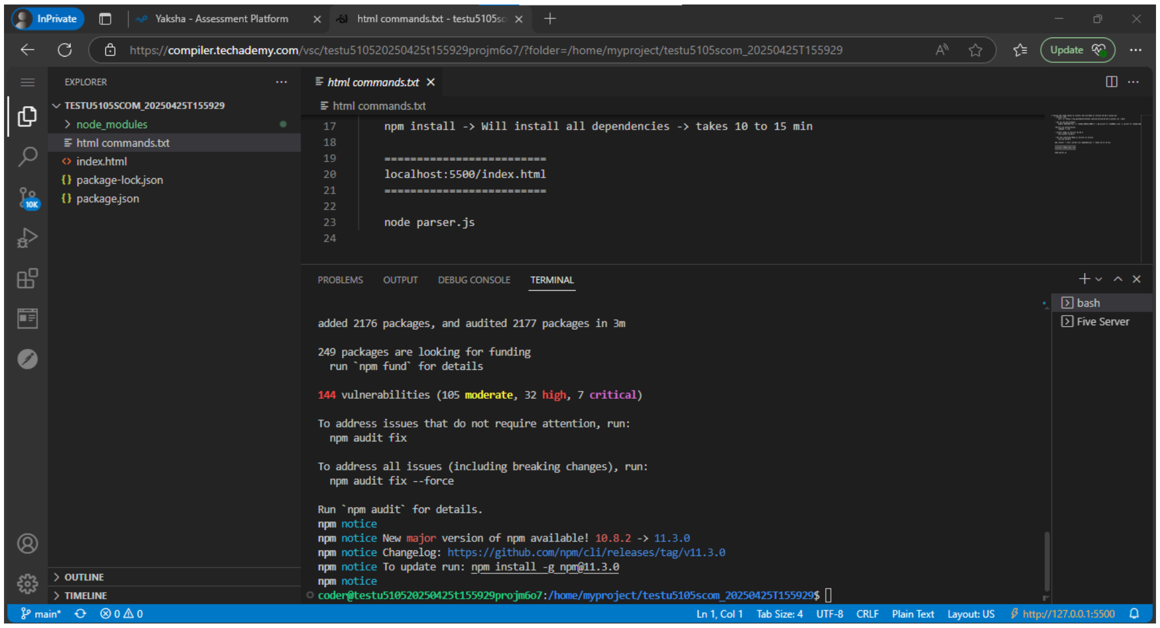
Step 2:

- Once installation completes, go to the **bottom right corner** of the VS Code screen.
- Click the **"Go Live"** button – This will start a **live server**, The server will run at port `5500` (e.g., `http://localhost:5500/`)


InPrivate
Yaksha - Assessment Platform
html commands.txt - testu5105sc
https://compiler.techademy.com/vsc/testu510520250425t155929projm6o7/?folder=/home/myproject/testu5105scom_20250425T155929
Update
EXPLORER
TESTU5105SCOM_20250425T155929
node_modules
html commands.txt
index.html
package-lock.json
package.json
OUTLINE
TIMELINE
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
bash
main*
Ln 1, Col 1
Tab Size: 4
UTF-8
CRLF
Plain Text
Layout: US
Go Live

```
npm install -> Will install all dependencies -> takes 10 to 15 min

========================
localhost:5500/index.html
========================

node parser.js
```

```
added 2176 packages, and audited 2177 packages in 3m

249 packages are looking for funding
  run `npm fund` for details

144 vulnerabilities (105 moderate, 32 high, 7 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.3.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.3.0
npm notice To update run: npm install -g npm@11.3.0
npm notice
coder@testu510520250425t155929projm6o7:/home/myproject/testu5105scom_20250425T155929$
```

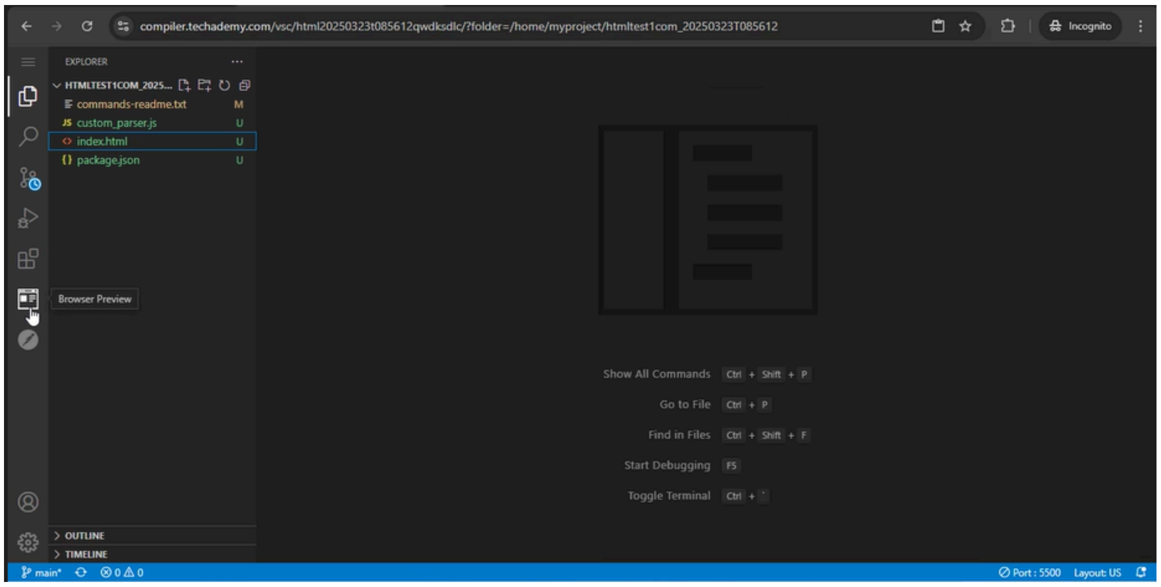


```
npm install -> Will install all dependencies -> takes 10 to 15 min

========================
localhost:5500/index.html
========================

node parser.js
```

```
added 2176 packages, and audited 2177 packages in 3m

249 packages are looking for funding
  run `npm fund` for details

144 vulnerabilities (105 moderate, 32 high, 7 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.3.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.3.0
npm notice To update run: npm install -g npm@11.3.0
npm notice
coder@testu510520250425t155929projm6o7:/home/myproject/testu5105scom_20250425T155929$
```

Step 3: Preview Output in Browser

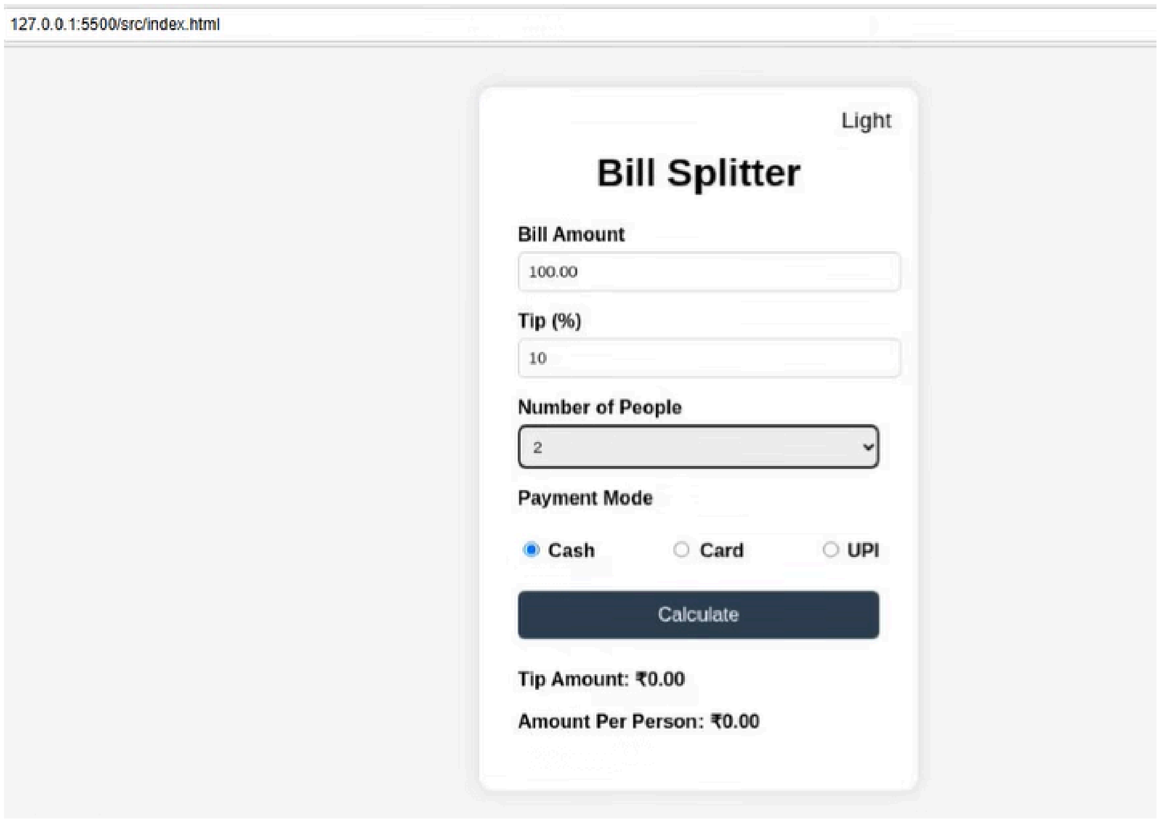
- This is a **web-based application**, so to view it in a browser, use the **internal browser inside the workspace.**
- Click on the **second last icon on the left panel** (the one labeled **"Browser Preview"**). This will open a tab within VS Code where you can **launch and view your application.**
- Note: The application **will not open in your system's local browser** — it must be viewed using the internal browser.



In the **Browser Preview tab**, type the following URL in the address bar and press **Enter**:

Your file is being served on: `localhost:5500/src/index.html`

This will load your HTML file and display the output of your web page **inside the internal browser**.


127.0.0.1:5500/src/index.html

Light

Bill Splitter

Bill Amount

100.00

Tip (%)

10

Number of People

2

Payment Mode

Cash [✓] Card UPI

Calculate

Tip Amount: ₹0.00

Amount Per Person: ₹0.00

Step 4:

- Go back to the **terminal** and type the following command, then press **Enter**:

  node src/test/custom-parser.js

- This command will **execute the validation script** and display the test results for your HTML file in the terminal.

Mandatory Assessment Guidelines:

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.**
3. **This editor Auto Saves the code.**
4. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
5. **This is a web-based application, to run the application on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**

   Note: The application will not run in the local browser
6. **You can follow series of command to setup environment once you are in your project-name folder:**
   a. **npm install -> Will install all dependencies -> takes 10 to 15 min.**
   b. **node src/test/custom-parser.js -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min.**
7. **Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on "Submit Assessment" after you are done with code.**