

EMI Calculator Project Instructions

1 PROJECT ABSTRACT

The **EMI Calculator App** is a practical front-end project designed to strengthen skills in HTML, CSS, and JavaScript through a real-world financial application. This project allows users to calculate Equated Monthly Installment (EMI), total interest payable, and total repayment based on inputs like loan amount, tenure, and interest rate. It features loan type switching, tenure in years or months, and theme toggling between light and dark modes.

This project emphasizes:

- Handling dynamic inputs and real-time calculations with JavaScript.
- Creating a styled form interface using HTML & CSS.
- Implementing interactivity such as range-input syncing and unit toggles.

ScreenShots:

Light Mode

localhost 5500/src/index.html

EMI Calculator

Light Dark

☒ Home Loan ☐ Personal Loan

Loan Amount

5000000

Interest Rate (%)

9

Loan Tenure

20 Years Months

Start Date

mm/dd/yyyy

Calculate EMI

Clear

EMI:

Total Interest Payable:

Total Payment:

Dark Mode

localhost 5500/src/index.html

EMI Calculator

Light Dark

Home Loan Personal Loan

Loan Amount

5000000

Interest Rate (%)

9

Loan Tenure

20 Years Months

Start Date

mm/dd/yyyy

Calculate EMI

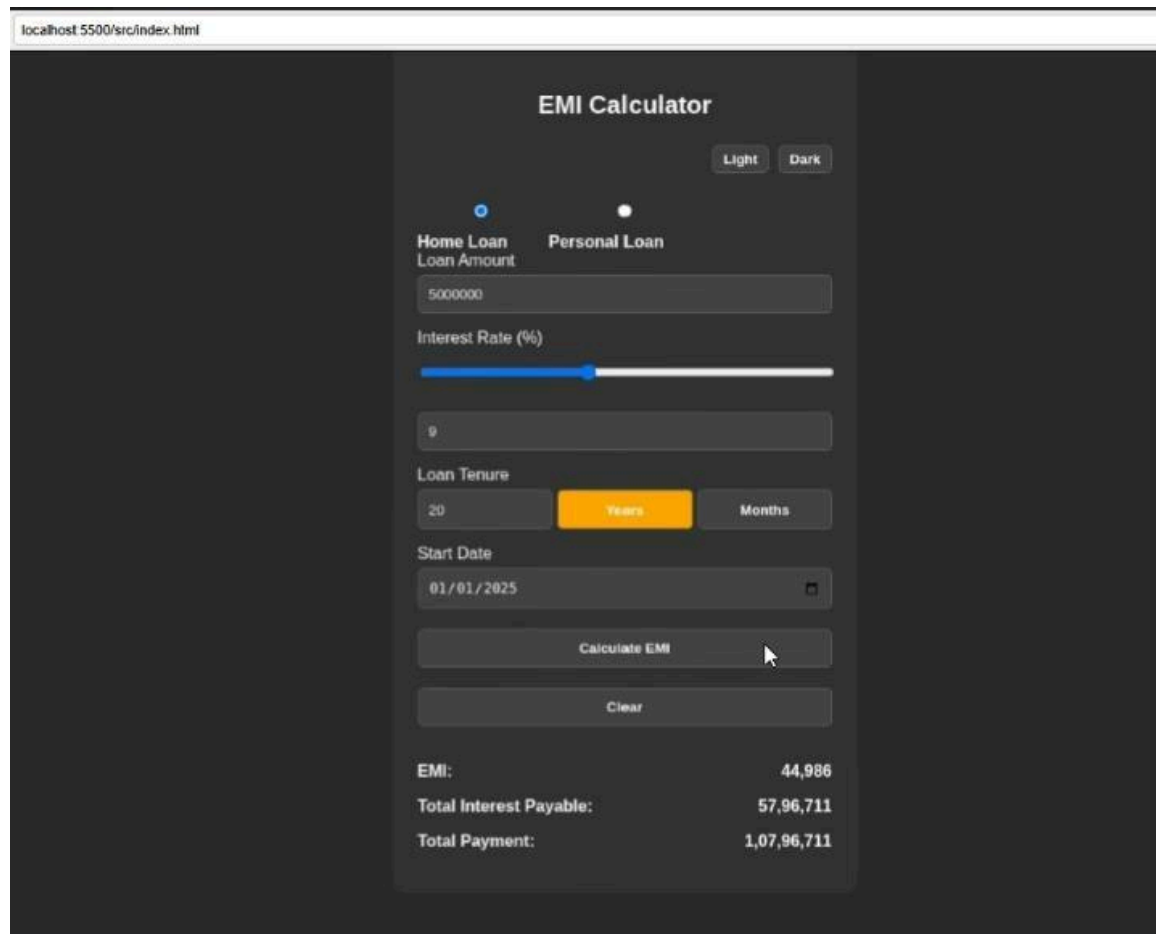
Clear

EMI:

Total Interest Payable:

Total Payment:

Calculation



localhost 5500/src/index.html

EMI Calculator

Light Dark

Home Loan Personal Loan

Loan Amount

5000000

Interest Rate (%)

9

Loan Tenure

20 Years Months

Start Date

01/01/2025

Calculate EMI

Clear

EMI:	44,986
Total Interest Payable:	57,96,711
Total Payment:	1,07,96,711

Instructions for Core Files

1. index.html

Create the HTML structure for a EMI calculator using the following specifications:

1. Start with HTML5 doctype and set the language to English.
2. Inside the `head`, set:
 - UTF-8 charset
 - Viewport for mobile responsiveness
 - Title as “EMI Calculator”
 - Link to external CSS
3. In the `body`, apply class `light-mode`.
4. Add a `div` with a class `container` to hold the entire layout.
5. Inside the container:
 - Add a heading with h2 with the app name as EMI Calculator
 - Add two buttons for Light and Dark theme switching
 - Add two radio buttons for Home and Personal Loan
 - Add input fields and labels for loan amount, interest rate, loan tenure, and start date
 - Add tenure toggle buttons ("Years", "Months")
 - Add a section for Calculate and Clear buttons
 - Add output paragraphs showing EMI, interest, and total amount
 - At the bottom, link the external JS file

2. style.css

Apply styles to design the EMI calculator layout and enable theme switching:

1. Set global font and remove default spacing.
2. Style `.container` with background, padding, shadow, and rounded corners.
3. Inputs and buttons should have uniform spacing, rounded corners, and responsive width.
4. Use orange color theme for primary buttons and interactions.
5. Style `.theme-toggle`, `.loan-type`, and `.tenure-toggle` for layout and interactivity.
6. Add `.output` styling for spacing and alignment.
7. For `.dark-mode`:
 - Use dark background colors and white/light text.
 - Update inputs and buttons to match the dark theme.

3. script.js

Write the JavaScript logic to handle all interactive behavior and EMI calculations:

1. Create a `calculateEMI()` function to:

- Read and convert inputs
 - Perform EMI formula calculation
 - Update result outputs
2. Create a `clearFields()` function to reset the form.
 3. Implement `toggleTheme(mode)` to switch between light and dark mode.
-

Detailed HTML Structure Guide for EMI Calculator App

1. Document Declaration and Root Elements

Start the document by declaring the HTML5 doctype. Then open the root HTML structure and set the language to English.

Inside the head section:

- Set the character encoding to UTF-8.
- Add a meta tag for responsive design by specifying the viewport width as device-width and the initial scale as 1.0.
- Set the title of the page to “EMI Calculator”.
- Link to an external stylesheet by referencing a file named “style.css”.

2. Body Setup

- Within the body of the page, apply a class called “light-mode” to allow toggling between light and dark themes.
- Create a main wrapper section using a division element. Assign it a class name called “container”. All remaining elements of the application should be placed inside this container.

3. Heading

- At the top of the container, add a second-level heading element. Set its visible text to “EMI Calculator”. This serves as the main title of the tool.

4. Theme Toggle Button

Add a section for theme switching using a division element with the class name “theme-toggle”. Inside this section:

- Add a button with the ID “lightBtn” and the text “Light”.
- Add another button with the ID “darkBtn” and the text “Dark”.

These buttons will allow users to manually switch between light and dark themes.

5. Loan Type Selection

Create a new section using a division with the class name “loan-type”. Inside this section:

- Add a radio button input labeled “Home Loan” with the value set to “home” and selected by default.
- Add another radio button input labeled “Personal Loan” with the value set to “personal”.

Both radio buttons should share the same name attribute to allow single selection.

6. Loan Amount Input

Add a label that reads “Loan Amount”. Below it:

- Insert a number input field.
- Assign it an ID of “loanAmount”.
- Set the default value to 5000000.

This input allows users to enter the total loan amount they wish to borrow.

7. Interest Rate Input

Add a label with the text “Interest Rate (%)”. Below that:

- A range slider input with values ranging from 5 to 15, stepping by 0.1.
- Assign this slider the ID “interestRateSlider” and set its default value to 9.
- Also include a number input field to reflect the slider value.
- Give this number field the ID “interestRate” and also set it to 9 by default.

These two inputs should remain in sync, allowing users to choose the interest rate using either control.

8. Loan Tenure Input and Toggle

Add a label with the text “Loan Tenure”.

Below the label, create a horizontal section using a division with the class name “tenure-toggle”. Inside this section:

- Add a number input field with the ID “loanTenure” and a default value of 20.
- Include two toggle buttons:
 - One with the ID “yearBtn”, the text “Years”, and marked as active by default.
 - One with the ID “monthBtn”, with the text “Months”.

This toggle allows the user to specify whether the loan tenure is in years or months.

9. Start Date Input

Add a label that reads “Start Date”. Below this label:

- Include a date input field with the ID “startDate”.

This field allows users to optionally choose the starting month or date for the EMI payments.

10. Buttons Section

Create a new division with the class name “buttons”. Inside this section:

- Add a button with the ID “calculateBtn” and the text “Calculate EMI”. This will trigger the EMI calculation.
- Add a second button with the ID “clearBtn” and the text “Clear”. This will reset all input fields.

11. Output Section

After the buttons, insert a division with the class name “output”. This section will display the calculation results. Inside this output area:

- Add a paragraph that reads “EMI:” followed by a placeholder span. Assign this span the ID “emiOutput” and initialize its text content with a dash.
- Add a second paragraph with the text “Total Interest Payable:” followed by a span with the ID “interestOutput”. Also initialize it with a dash.
- Add a third paragraph with the text “Total Payment:” followed by a span with the ID “paymentOutput”, again initialized with a dash.

These spans will be dynamically updated via JavaScript when the user calculates their EMI.

12. JavaScript File Link

- At the bottom of the body section, just before it closes, add a script reference to load an external JavaScript file named “script.js”. This will contain the logic for calculating EMI, switching themes, and updating the UI.

Detailed CSS Styling Guide for EMI Calculator App

This guide explains how the EMI Calculator’s CSS styles work and how they contribute to the visual design, accessibility, and usability of the app.

1. Body Styling

Sets foundational layout and appearance for the whole page:

- Uses a sans-serif font family like Arial for clean readability.
- Removes all default margin from the page.
- Applies a soft light-gray background color (#f0f0f0) to create a neutral canvas.

2. Main Container (class: container)

Defines the primary card-style box holding all form elements:

- Uses a white background to contrast against the light page background.
- Sets a maximum width of 400 pixels to ensure a compact, mobile-friendly layout.
- Centers the container horizontally using automatic left and right margins.
- Adds generous internal padding (25 pixels) for a comfortable layout.
- Applies rounded corners (12 pixels) for a soft, modern look.
- Adds a subtle drop shadow to give the container depth and elevation.

3. Heading (element: h2)

Styles the title of the app:

- Centers the text horizontally inside the container.

4. Inputs and Buttons (all input and button elements)

Applies a unified appearance to all interactive form controls:

- Makes elements span the full width of the container (100 percent).
- Adds padding inside the elements (10 pixels) to increase clickability and comfort.
- Adds margin above (5 pixels) and below (15 pixels) each element to separate them visually.
- Applies rounded corners (5 pixels) for smooth appearance.
- Adds a light gray border (#ccc) around each element for definition.
- Ensures padding and border are included in the total width using border-box sizing.

5. Default Button Styling

Sets the base design for buttons:

- Applies a bold orange background color (#ffa500).
- Sets text color to white for strong contrast.
- Removes the default border to maintain a clean design.
- Makes the cursor change to a pointer on hover to indicate interactivity.
- Uses bold font weight to emphasize button importance.

6. Button Hover Effect

Enhances button feedback on hover:

- Slightly darkens the orange background (#e69500) when the user hovers over the button.

7. Theme Toggle Section (class: theme-toggle)

Styles the theme switch buttons:

- Aligns buttons to the right side of the container.
- Adds spacing below this section (10 pixels) to separate it from the next block.

8. Theme Toggle Buttons (inside theme-toggle)

Specifies styling for light and dark toggle buttons:

- Makes buttons auto-width instead of full-width for compact appearance.
- Applies internal padding (5 pixels vertically and 10 pixels horizontally).
- Adds spacing between the two buttons by applying left margin.

- Sets a light gray background (#ddd) and dark gray text (#333) to distinguish them from other elements.

9. Loan Type Section (class: loan-type)

Styles the radio buttons for selecting loan type:

- Adds spacing between the labels horizontally (10 pixels).
- Displays labels in a horizontal inline layout.
- Applies bold text to emphasize the importance of loan type.

10. Tenure Toggle Section (class: tenure-toggle)

Organizes and styles the toggle between years and months:

- Displays the number input and buttons in a row using a horizontal layout.
- Applies a gap between elements for breathing room (5 pixels).
- Vertically aligns elements using center alignment.

11. Tenure Toggle Inputs and Buttons

Styles the input and toggle buttons inside the tenure selector:

- Makes the input field take equal space with flexible width.
- Makes each button take equal space beside the input using Flexbox.
- Applies padding to make buttons easy to click (10 pixels).
- Uses a light gray background and dark text for default appearance.
- Removes the border and sets the cursor to pointer for interactivity.

12. Active Tenure Button (button with class active)

Highlights the currently selected tenure unit:

- Applies the same orange background as the primary button (#ffa500).
- Changes the text color to white for contrast.

13. Output Section (class: output)

Styles the result display area:

- Adds top margin (20 pixels) to separate it from the buttons above.

- Uses larger font size (1 rem) and bold text weight to highlight results.

14. Output Values (span elements inside output section)

Aligns result values:

- Floats the span to the right so the numeric output appears opposite the label.

15. Dark Mode (class applied to body element: dark-mode)

Defines the alternate styling for dark theme:

- Changes the page background to a dark gray (#2b2b2b).
- Sets the text color to white throughout the app for readability.

16. Dark Mode for Container

Updates the main container appearance in dark mode:

- Changes background to an even darker shade of gray (#333).

17. Dark Mode for Inputs and Buttons

Applies darker styles to form fields and buttons in dark mode:

- Changes backgrounds to a deeper gray (#444).
- Changes text color to white.
- Applies a medium-gray border color (#666) for visibility.

Detailed JavaScript Logic Guide for EMI Calculator App

This guide explains the JavaScript logic used in the EMI Calculator app. Each section describes the role of specific functions and code structure that drives the app's behavior.

1. Main Logic Function: calculateEMI()

Performs the EMI calculation based on user input:

- Reads the loan amount (P), annual interest rate, and loan tenure.
- Converts annual interest rate to monthly interest rate by dividing by 12 and 100.
- Converts the loan tenure into months depending on the selected unit (years or months).
- Applies the standard EMI formula to calculate the monthly installment.
- Calculates total payment and total interest over the loan duration.
- Formats the result values using Indian number formatting and updates the output fields.

2. Field Reset Function: `clearFields()`

Clears all user inputs and resets outputs to default:

- Empties the loan amount, interest rate, tenure, and start date fields.
- Resets the interest rate slider to its default value (9).
- Resets output fields to show dashes ("-") instead of values.

3. Theme Toggle Function: `toggleTheme(mode)`

Toggles between light and dark UI modes:

- Accepts a mode parameter (either "light" or "dark").
- Applies or removes the "dark-mode" class from the body accordingly.
- Updates the body's class name to reflect the chosen theme.

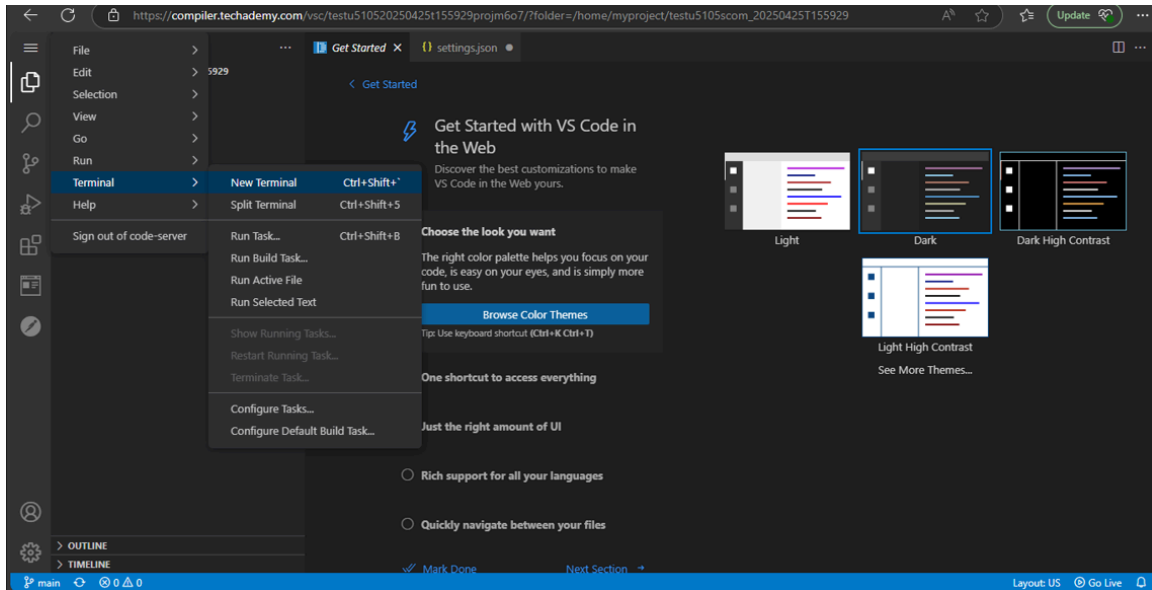
Assessment Guidelines

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.



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/>)

The screenshot shows the VS Code interface with the following details:

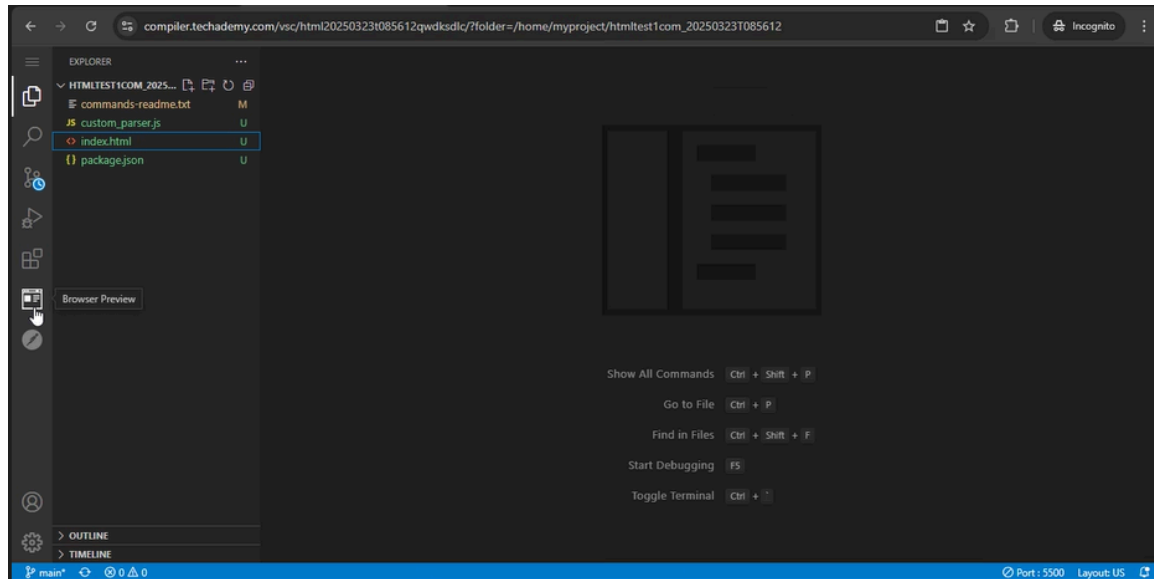
- Explorer Panel:** Displays the file structure of the project. The files listed are `node_modules`, `html commands.txt`, `index.html`, `package-lock.json`, and `package.json`.
- Editor Panel:** Shows the content of `html commands.txt`. The text includes instructions for installing dependencies and running the application.
- Terminal Panel:** Displays the output of the `npm install` command. The output shows that 2176 packages were added and 2177 packages were audited. It also lists 144 vulnerabilities (105 moderate, 32 high, 7 critical) and provides instructions for addressing them.

This screenshot is identical to the one above, showing the VS Code interface with the Explorer, Editor, and Terminal panels. The Explorer panel shows the file structure of the project. The Editor panel shows the content of `html commands.txt`. The Terminal panel shows the output of the `npm install` command.

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**.

localhost 5500/src/index.html

EMI Calculator

Light Dark

☒ Home Loan ☐ Personal Loan

Loan Amount

5000000

Interest Rate (%)

9

Loan Tenure

20 Years Months

Start Date

mm/dd/yyyy

Calculate EMI

Clear

EMI:

Total Interest Payable:

Total Payment:

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.