cprime

09-05-25 - Angular Project - React Color Palette Generator

JOSHNA ACSHA S

APPRENTICE - CPRIME

EMP ID: B2FFE7YKC

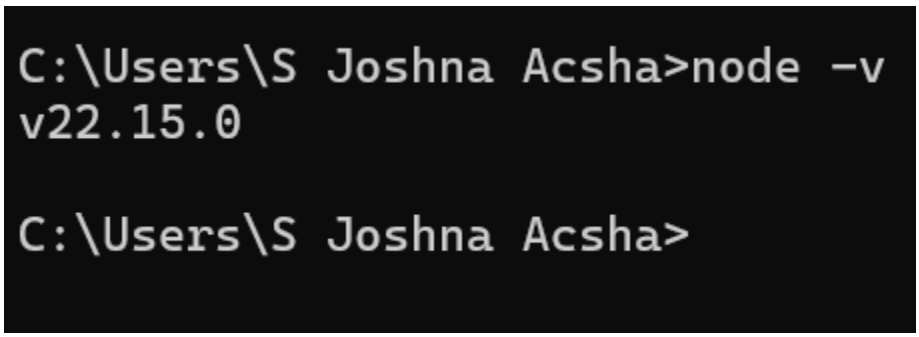
Project Overview

The **Angular Color Palette Generator** is a dynamic and responsive web application built with Angular 15+. It allows users to create, preview, and manage custom color palettes with ease. The app supports HEX color input, color picker integration, preset color selection, and full palette management, including saving and reloading previously created palettes.

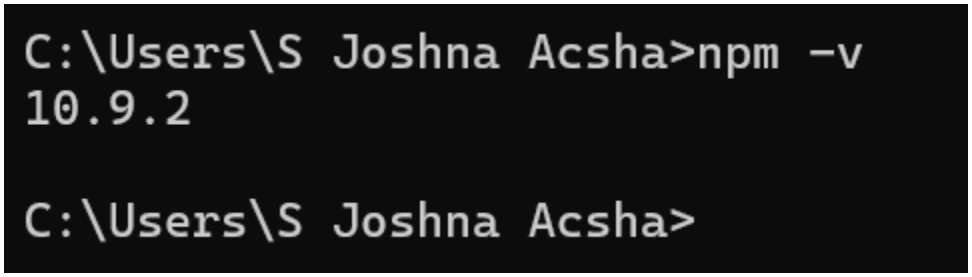
1. Prerequisites

Before you start, ensure that you have the following tools installed:

- **Node.js** (https://nodejs.org/)



- **npm** (comes with Node.js)



- **Angular CLI** (https://angular.io/cli)

  To install Angular CLI globally, run the following command in your terminal:

  ```
  npm install -g @angular/cli
  ```

```
C:\Users\S Joshna Acsha>npm install -g @angular/cli

added 274 packages in 34s

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

C:\Users\S Joshna Acsha>
```

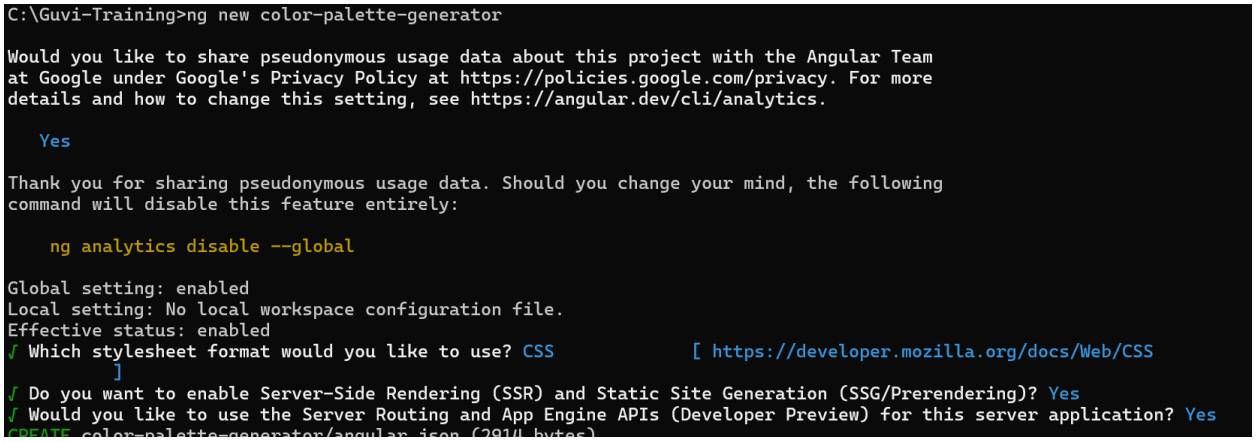
2. Setting Up the Angular Application

2.1. Create a New Angular Project

Open your terminal and run the following command to create a new Angular project:

```
ng new color-palette-generator
```

1. During the setup, Angular CLI will ask for a few configurations:
    - **Would you like to add Angular routing?**: No
    - **Which stylesheet format would you like to use?**: CSS (You can choose SCSS if needed)



```
CREATE color-palette-generator/src/app/app.routes.ts (80 bytes)
CREATE color-palette-generator/src/app/app.config.server.ts (509 bytes)
CREATE color-palette-generator/src/app/app.routes.server.ts (174 bytes)
CREATE color-palette-generator/public/favicon.ico (15086 bytes)
√ Packages installed successfully.
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
    Successfully initialized git.

C:\Guvi-Training>
```

2.2. Navigate to the Project Folder

After the project is created, navigate into your project directory:

```
cd color-palette-generator
```

```
C:\Guvi-Training>cd color-palette-generator

C:\Guvi-Training\color-palette-generator>ng serve
```

2.3. Serve the Application

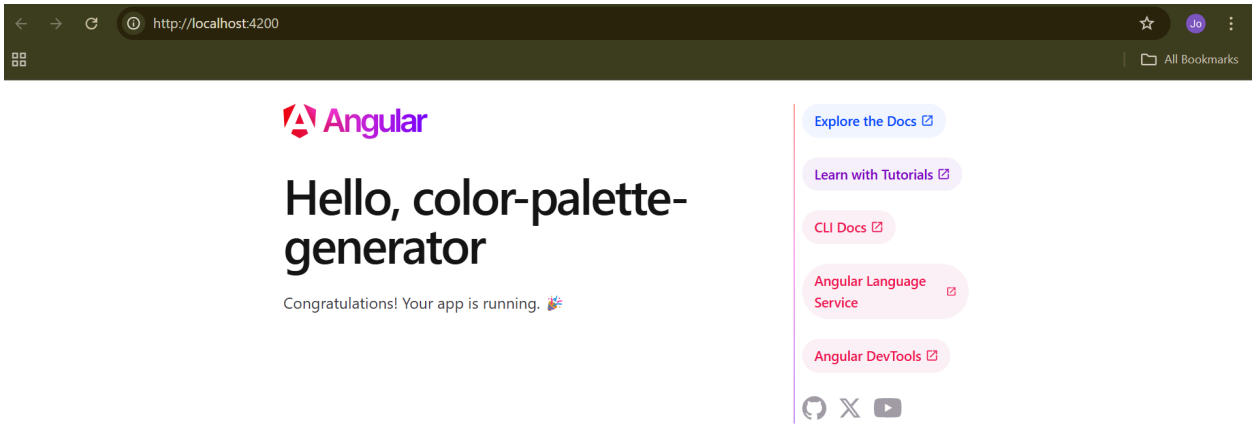
To run the application locally, use the following command:

```
ng serve
```

```
C:\Guvi-Training\color-palette-generator>ng serve

Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.dev/cli/analytics.

   Yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

    ng analytics disable
```

```
Application bundle generation complete. [7.567 seconds]

Watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
  → Local:   http://localhost:4200/
  → press h + enter to show help
```

Then, open your browser and go to http://localhost:4200 to see your app running.



3. Creating Angular Components

We will create three components for the Color Palette Generator:

1. **Color Picker Component**: Allows the user to pick a color.
2. **Palette Display Component**: Displays the selected colors in a palette.
3. **Saved Palettes Component**: Displays saved color palettes.

3.1. Generate the Components

Run the following commands to generate each of the components:

Generate `color-picker` Component

```
ng generate component components/color-picker
```

```
PS C:\Guvi-Training\color-palette-generator> ng generate component components/color-picker
CREATE src/app/components/color-picker/color-picker.component.html (28 bytes)
CREATE src/app/components/color-picker/color-picker.component.spec.ts (651 bytes)
CREATE src/app/components/color-picker/color-picker.component.ts (248 bytes)
CREATE src/app/components/color-picker/color-picker.component.css (0 bytes)
```

Generate `palette-display` Component

```
ng generate component components/palette-display
```

```
PS C:\Guvi-Training\color-palette-generator> ng generate component components/palette-display
CREATE src/app/components/palette-display/palette-display.component.html (31 bytes)
CREATE src/app/components/palette-display/palette-display.component.spec.ts (672 bytes)
CREATE src/app/components/palette-display/palette-display.component.ts (260 bytes)
CREATE src/app/components/palette-display/palette-display.component.css (0 bytes)
```

Generate `saved-palettes` Component

```
ng generate component components/saved-palettes
```

```
PS C:\Guvi-Training\color-palette-generator> ng generate component components/saved-palettes
CREATE src/app/components/saved-palettes/saved-palettes.component.html (30 bytes)
CREATE src/app/components/saved-palettes/saved-palettes.component.spec.ts (665 bytes)
CREATE src/app/components/saved-palettes/saved-palettes.component.ts (256 bytes)
CREATE src/app/components/saved-palettes/saved-palettes.component.css (0 bytes)
PS C:\Guvi-Training\color-palette-generator>
```

4. Modify the Components

We will now modify each component to add functionality.

4.1. `color-picker.component.ts`

This component will allow the user to select a color and emit it to the parent component.

```
// src/app/components/color-picker/color-picker.component.ts
import { Component, EventEmitter, Output } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';

@Component({
  selector: 'app-color-picker',
  standalone: true,
  imports: [CommonModule, FormsModule],
  template: `
    <div class="color-picker-container">
      <div class="color-input-group">
        <input
          type="color"
          [(ngModel)]="selectedColor"
          (change)="emitColor()"
          class="color-input"
        >
```

```
        <input
          type="text"
          [(ngModel)]="selectedColor"
          (change)="emitColor()"
          placeholder="#RRGGBB"
          pattern="^#([A-Fa-f0-9]{6})$"
          class="color-text-input"
        >
        <button (click)="emitColor()" class="add-button">Add
Color</button>
      </div>

      <div class="presets">
        <h3>Quick Colors</h3>
        <div class="preset-colors">
          <button
            *ngFor="let color of presetColors"
            class="preset-color"
            [style.background-color]="color"
            (click)="selectPreset(color)"
          ></button>
        </div>
      </div>
    </div>
  `,
  styles: [`
    .color-picker-container {
      background-color: #f5f5f5;
      padding: 20px;
      border-radius: 8px;
      margin-bottom: 20px;
    }

    .color-input-group {
      display: flex;
      gap: 10px;
      margin-bottom: 15px;
    }
```

```
.color-input {
  width: 60px;
  height: 40px;
  padding: 0;
  border: 1px solid #ccc;
  cursor: pointer;
}

.color-text-input {
  flex: 1;
  padding: 8px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.add-button {
  padding: 8px 15px;
  background-color: #4CAF50;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.add-button:hover {
  background-color: #45a049;
}

.presets {
  margin-top: 15px;
}

.presets h3 {
  margin-top: 0;
  margin-bottom: 10px;
}
```

```
    .preset-colors {
      display: flex;
      flex-wrap: wrap;
      gap: 10px;
    }

    .preset-color {
      width: 30px;
      height: 30px;
      border-radius: 50%;
      border: 1px solid #ddd;
      cursor: pointer;
    }

    .preset-color:hover {
      transform: scale(1.1);
    }
  `]
})
export class ColorPickerComponent {
  @Output() colorSelected = new EventEmitter<string>();

  selectedColor: string = '#3366FF';

  presetColors: string[] = [
    '#FF6633', '#FFB399', '#FF33FF', '#FFFF99', '#00B3E6',
    '#E6B333', '#3366E6', '#999966', '#99FF99', '#B34D4D',
    '#80B300', '#809900', '#E6B3B3', '#6680B3', '#66991A',
    '#FF99E6', '#CCFF1A', '#FF1A66', '#E6331A', '#33FFCC'
  ];

  emitColor(): void {
    if (this.isValidHexColor(this.selectedColor)) {
      this.colorSelected.emit(this.selectedColor);
    }
  }

  selectPreset(color: string): void {
```

```
    this.selectedColor = color;
    this.colorSelected.emit(color);
  }

  private isValidHexColor(color: string): boolean {
    const regex = /^#([A-Fa-f0-9]{6})$/;
    return regex.test(color);
  }
}
```

4.2. palette-display.component.ts

This component will receive the colors from the parent and display them in a palette.

```
// src/app/components/palette-display/palette-display.component.ts
import { Component, EventEmitter, Input, Output } from
'@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';

@Component({
  selector: 'app-palette-display',
  standalone: true,
  imports: [CommonModule, FormsModule],
  template: `
    <div class="palette-display" *ngIf="colors.length > 0">
      <h2>Current Palette</h2>
```

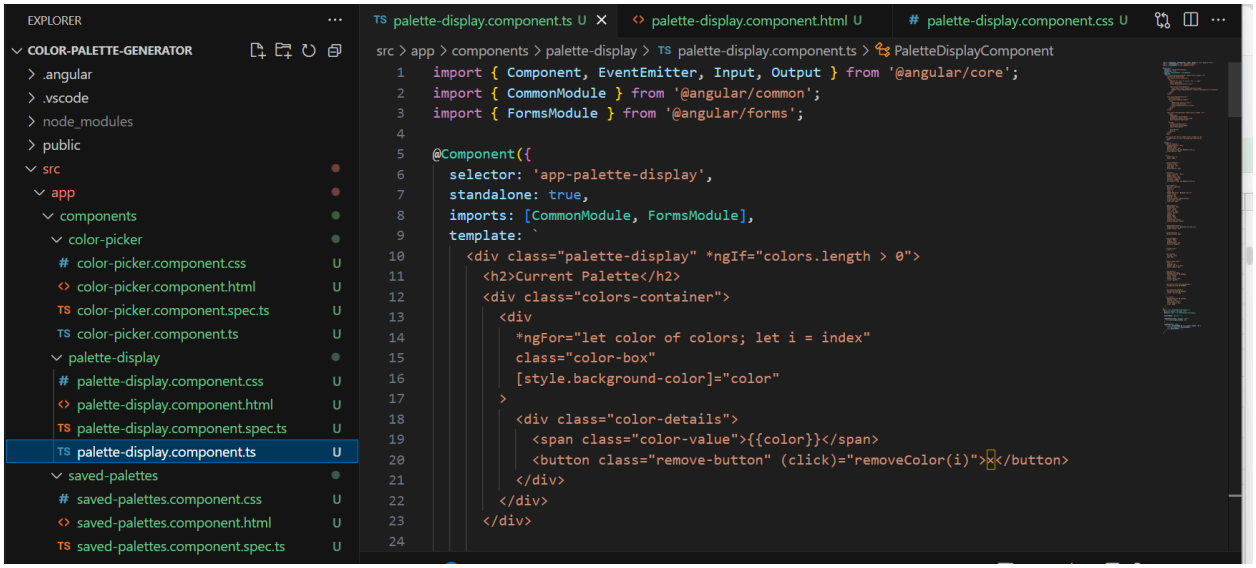
```
      <div class="colors-container">
        <div
          *ngFor="let color of colors; let i = index"
          class="color-box"
          [style.background-color]="color"
        >
          <div class="color-details">
            <span class="color-value">{{color}}</span>
            <button class="remove-button"
(click)="removeColor(i)">×</button>
          </div>
        </div>
      </div>

      <div class="palette-preview">
        <h3>Preview</h3>
        <div class="preview-sample">
          <div
            *ngFor="let color of colors"
            class="preview-color"
            [style.background-color]="color"
          ></div>
        </div>
      </div>

      <div class="save-palette" *ngIf="colors.length > 0">
        <input
          type="text"
          [(ngModel)]="paletteName"
          placeholder="Enter palette name"
          class="palette-name-input"
        >
        <button
          (click)="savePalette()"
          [disabled]="!paletteName"
          class="save-button"
        >
          Save Palette
```

```
      </button>
    </div>
  </div>

  <div class="no-colors" *ngIf="colors.length === 0">
    <p>Add colors to start building your palette</p>
  </div>
`,
styles: [`
  .palette-display {
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    margin-bottom: 20px;
  }

  h2 {
    margin-top: 0;
    color: #333;
  }

  .colors-container {
    display: flex;
    flex-wrap: wrap;
    gap: 10px;
    margin-bottom: 20px;
  }

  .color-box {
    width: calc(20% - 8px);
    aspect-ratio: 1;
    border-radius: 8px;
    position: relative;
    overflow: hidden;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  }
```

```
.color-details {
  position: absolute;
  bottom: 0;
  left: 0;
  right: 0;
  background-color: rgba(0,0,0,0.7);
  color: white;
  padding: 5px;
  display: flex;
  justify-content: space-between;
  align-items: center;
  font-size: 12px;
}

.remove-button {
  background: none;
  border: none;
  color: white;
  font-size: 16px;
  cursor: pointer;
  padding: 0;
  width: 20px;
  height: 20px;
  display: flex;
  align-items: center;
  justify-content: center;
}

.remove-button:hover {
  background-color: rgba(255,255,255,0.2);
  border-radius: 50%;
}

.palette-preview {
  margin-bottom: 20px;
}

.preview-sample {
```

```
  height: 40px;
  display: flex;
  border-radius: 4px;
  overflow: hidden;
}

.preview-color {
  flex: 1;
}

.save-palette {
  display: flex;
  gap: 10px;
}

.palette-name-input {
  flex: 1;
  padding: 8px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.save-button {
  padding: 8px 15px;
  background-color: #3498db;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.save-button:hover:not([disabled]) {
  background-color: #2980b9;
}

.save-button[disabled] {
  background-color: #95a5a6;
  cursor: not-allowed;
```

```
    }

    .no-colors {
      background-color: #f8f8f8;
      padding: 30px;
      text-align: center;
      border-radius: 8px;
      color: #666;
    }
  `]
})
export class PaletteDisplayComponent {
  @Input() colors: string[] = [];
  @Output() save = new EventEmitter<string>();

  paletteName: string = '';

  removeColor(index: number): void {
    this.colors.splice(index, 1);
  }

  savePalette(): void {
    if (this.paletteName && this.colors.length > 0) {
      this.save.emit(this.paletteName);
      this.paletteName = '';
    }
  }
}
```



4.3. `saved-palettes.component.ts`

This component will display the list of saved color palettes.

```
// src/app/components/saved-palettes/saved-palettes.component.ts
import { Component, EventEmitter, Input, Output } from
'@angular/core';
import { CommonModule } from '@angular/common';

interface Palette {
  name: string;
  colors: string[];
}

@Component({
  selector: 'app-saved-palettes',
  standalone: true,
  imports: [CommonModule],
  template: `
    <div class="saved-palettes">
      <h2>Saved Palettes</h2>

      <div class="no-palettes" *ngIf="palettes.length === 0">
        <p>No saved palettes yet</p>
```

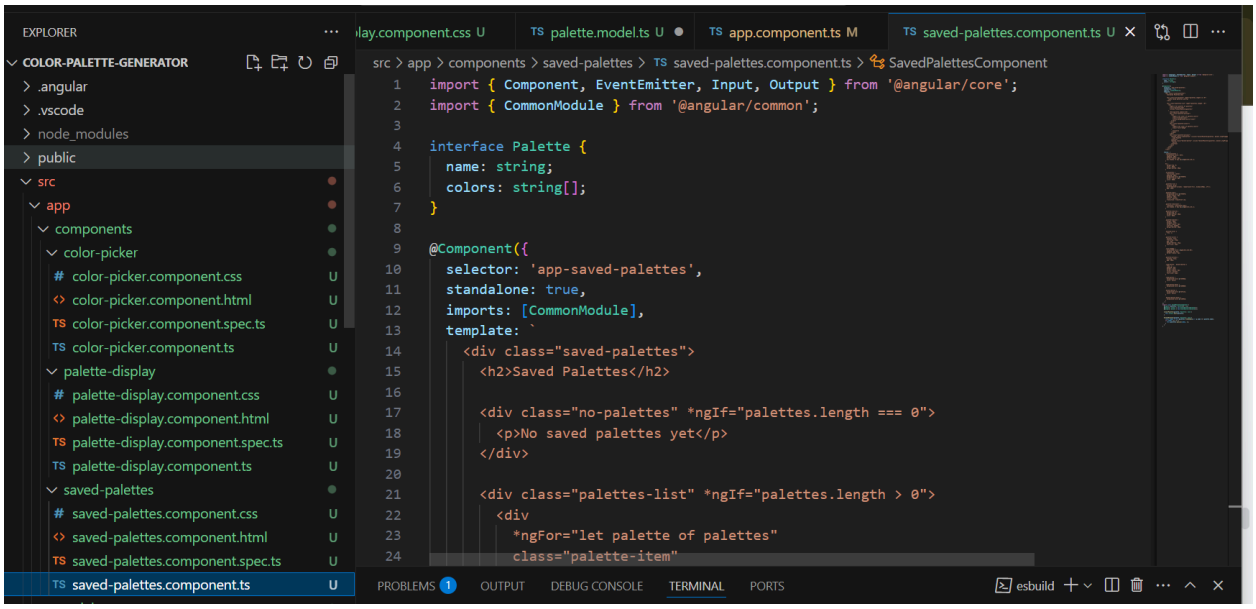
```
      </div>

      <div class="palettes-list" *ngIf="palettes.length > 0">
        <div
          *ngFor="let palette of palettes"
          class="palette-item"
          (click)="selectPalette(palette)"
        >
          <h3>{{palette.name}}</h3>
          <div class="palette-preview">
            <div
              *ngFor="let color of palette.colors"
              class="preview-color"
              [style.background-color]="color"
            ></div>
          </div>
          <div class="palette-colors">
            <span
              *ngFor="let color of palette.colors"
              class="color-badge"
            >
              {{color}}
            </span>
          </div>
          <div class="palette-actions">
            <button class="load-button"
(click)="selectPalette(palette); $event.stopPropagation();">
              Load Palette
            </button>
            <button class="delete-button"
(click)="deletePalette(palette); $event.stopPropagation();">
              Delete
            </button>
          </div>
        </div>
      </div>
    </div>
  `,
```

```
styles: [`
  .saved-palettes {
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
  }

  h2 {
    margin-top: 0;
    color: #333;
    margin-bottom: 20px;
  }

  .no-palettes {
    text-align: center;
    padding: 20px;
    background-color: #f8f8f8;
    border-radius: 6px;
    color: #666;
  }

  .palettes-list {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
    gap: 15px;
  }

  .palette-item {
    background-color: #f8f8f8;
    border-radius: 8px;
    padding: 15px;
    cursor: pointer;
    transition: transform 0.2s;
  }

  .palette-item:hover {
    transform: translateY(-3px);
```

```
  box-shadow: 0 4px 8px rgba(0,0,0,0.1);
}

.palette-item h3 {
  margin-top: 0;
  margin-bottom: 10px;
  color: #333;
}

.palette-preview {
  height: 30px;
  display: flex;
  border-radius: 4px;
  overflow: hidden;
  margin-bottom: 10px;
}

.preview-color {
  flex: 1;
}

.palette-colors {
  display: flex;
  flex-wrap: wrap;
  gap: 5px;
  margin-bottom: 15px;
  font-size: 12px;
}

.color-badge {
  background-color: rgba(0,0,0,0.05);
  padding: 2px 6px;
  border-radius: 4px;
}

.palette-actions {
  display: flex;
  gap: 10px;
```

```
    }

    .load-button, .delete-button {
      flex: 1;
      padding: 8px;
      border: none;
      border-radius: 4px;
      cursor: pointer;
      font-size: 12px;
    }

    .load-button {
      background-color: #3498db;
      color: white;
    }

    .load-button:hover {
      background-color: #2980b9;
    }

    .delete-button {
      background-color: #e74c3c;
      color: white;
    }

    .delete-button:hover {
      background-color: #c0392b;
    }
  `]
})
export class SavedPalettesComponent {
  @Input() palettes: Palette[] = [];
  @Output() select = new EventEmitter<Palette>();
  @Output() delete = new EventEmitter<Palette>();

  selectPalette(palette: Palette): void {
    this.select.emit(palette);
  }
```

```
  deletePalette(palette: Palette): void {
    const index = this.palettes.findIndex(p => p.name ===
palette.name);
    if (index !== -1) {
      this.palettes.splice(index, 1);
    }
  }
}
```



5. Create a Service to Handle the Palettes

Now, let's create a service to manage the color palettes.

5.1. Generate the Service

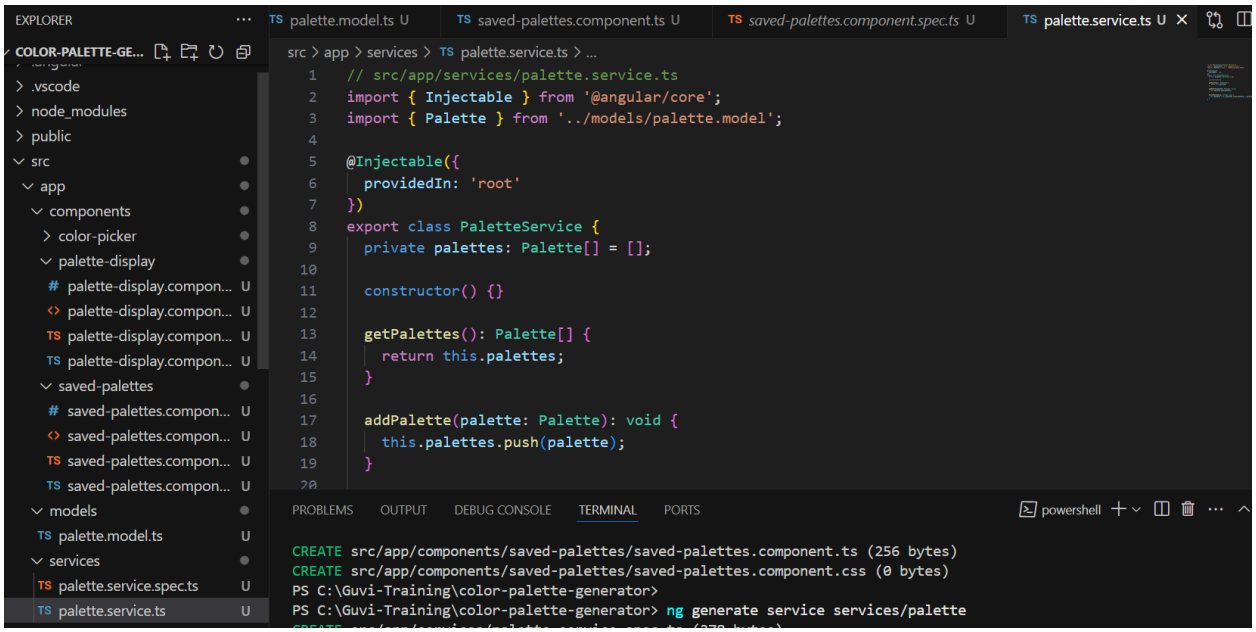
```
ng generate service services/palette
```

```
PS C:\Guvi-Training\color-palette-generator> ng generate service services/palette
CREATE src/app/services/palette.service.spec.ts (378 bytes)
CREATE src/app/services/palette.service.ts (145 bytes)
PS C:\Guvi-Training\color-palette-generator>
```

5.2. Service Code

```
// src/app/services/palette.service.ts
```

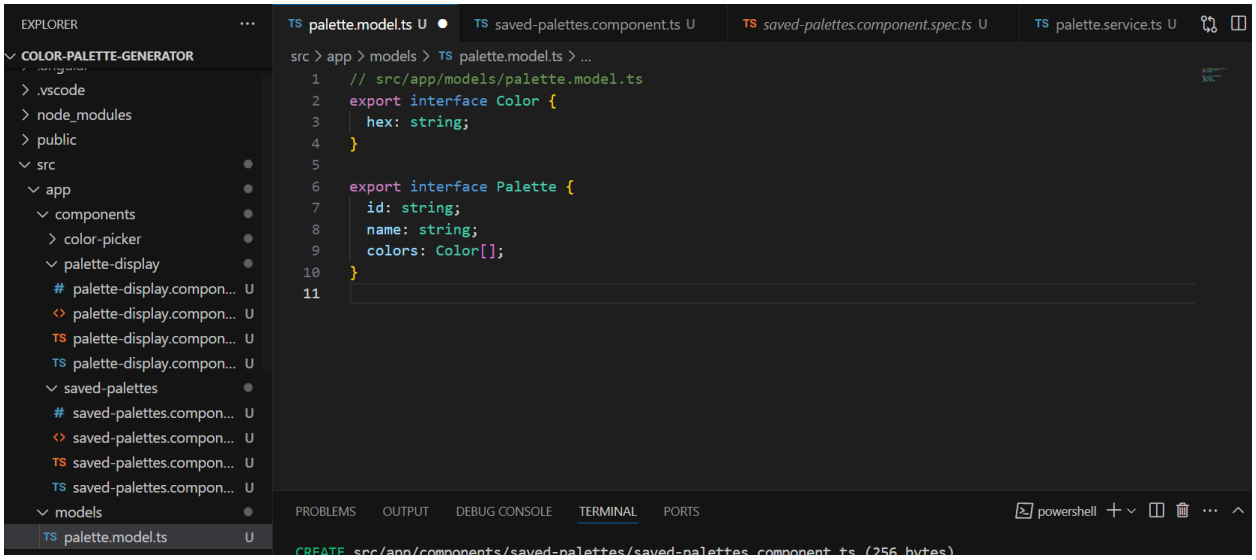
```
import { Injectable } from '@angular/core';
import { Palette } from '../models/palette.model';

@Injectable({
  providedIn: 'root'
})
export class PaletteService {
  private palettes: Palette[] = [];

  constructor() {}

  getPalettes(): Palette[] {
    return this.palettes;
  }

  addPalette(palette: Palette): void {
    this.palettes.push(palette);
  }

  deletePalette(id: string): void {
    this.palettes = this.palettes.filter(palette => palette.id !==
id);
  }
}
```



5.3. Palette Model

Create a new file **`palette.model.ts`** inside the **`models`** folder.

```
// src/app/models/palette.model.ts
export interface Color {
  hex: string;
}

export interface Palette {
  id: string;
  name: string;
  colors: Color[];
}
```



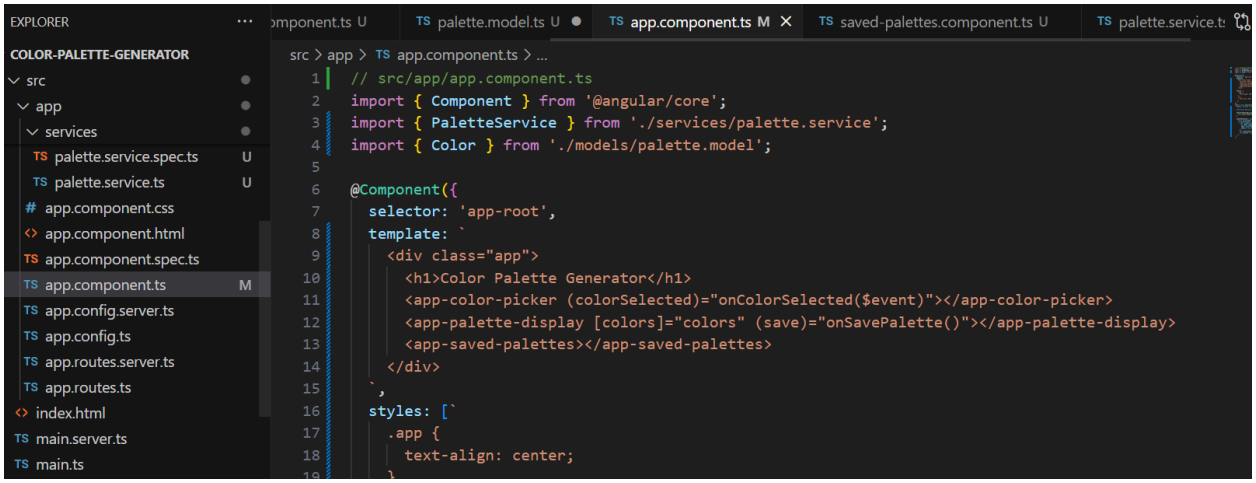
6. Modify the Root Component

Finally, modify the **`app.component.ts`** to connect the components together.

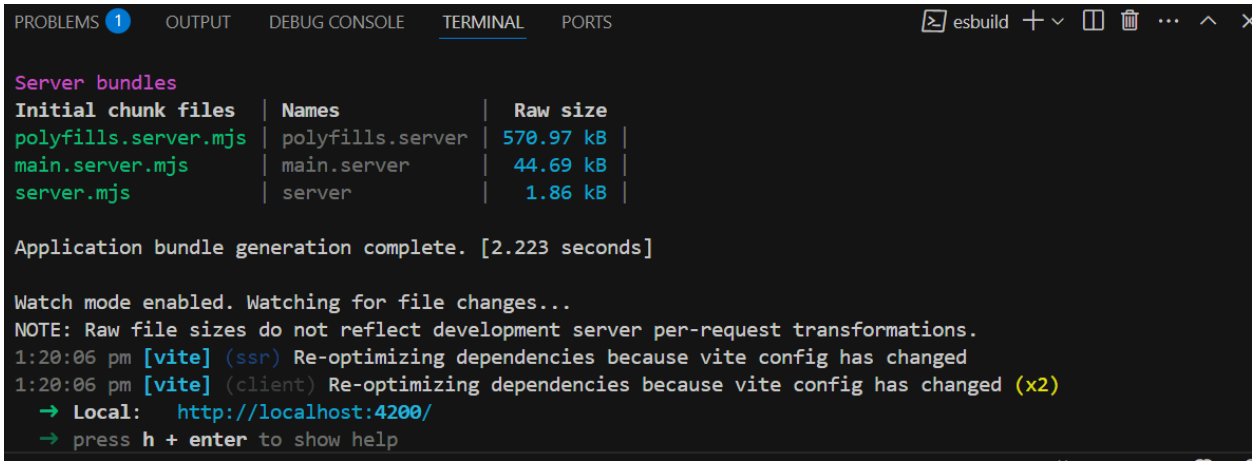
```
// src/app/app.component.ts
import { Component } from '@angular/core';
import { PaletteService } from './services/palette.service';
import { Color } from './models/palette.model';

@Component({
```

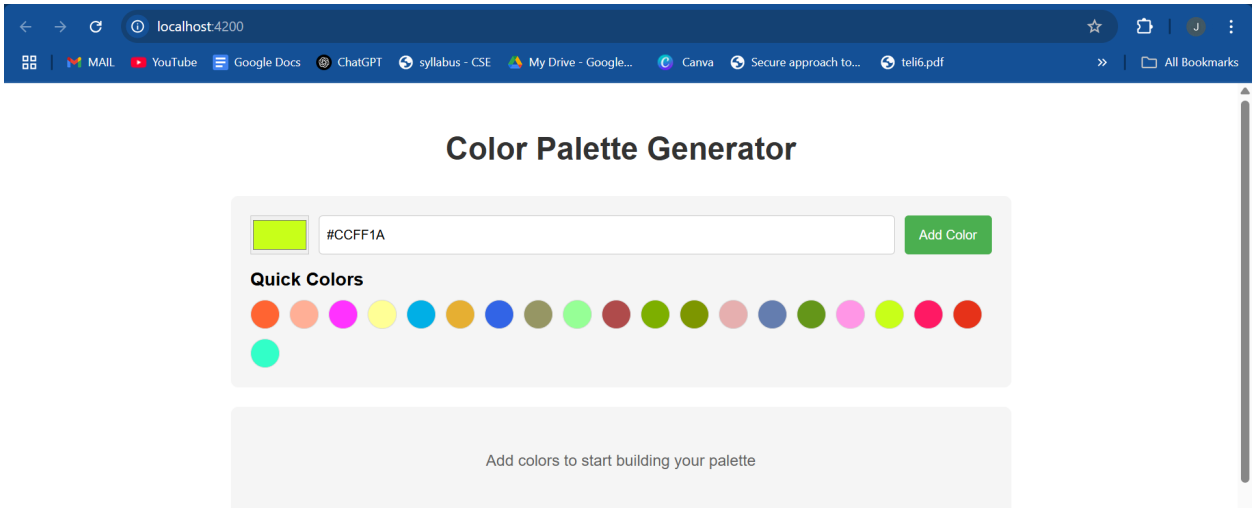
```
  selector: 'app-root',
  template: `
    <div class="app">
      <h1>Color Palette Generator</h1>
      <app-color-picker
(colorSelected)="onColorSelected($event)"></app-color-picker>
      <app-palette-display [colors]="colors"
(save)="onSavePalette()"></app-palette-display>
      <app-saved-palettes></app-saved-palettes>
    </div>
  `,
  styles: [`
    .app {
      text-align: center;
    }
    h1 {
      font-size: 2em;
    }
  `]
})
export class AppComponent {
  colors: Color[] = [];

  constructor(private paletteService: PaletteService) {}

  onColorSelected(color: string): void {
    this.colors.push({ hex: color });
  }

  onSavePalette(): void {
    const paletteName = prompt('Enter a name for this palette');
    if (paletteName) {
      const newPalette = {
        id: Math.random().toString(36).substring(2),
        name: paletteName,
        colors: this.colors
      };
      this.paletteService.addPalette(newPalette);
```

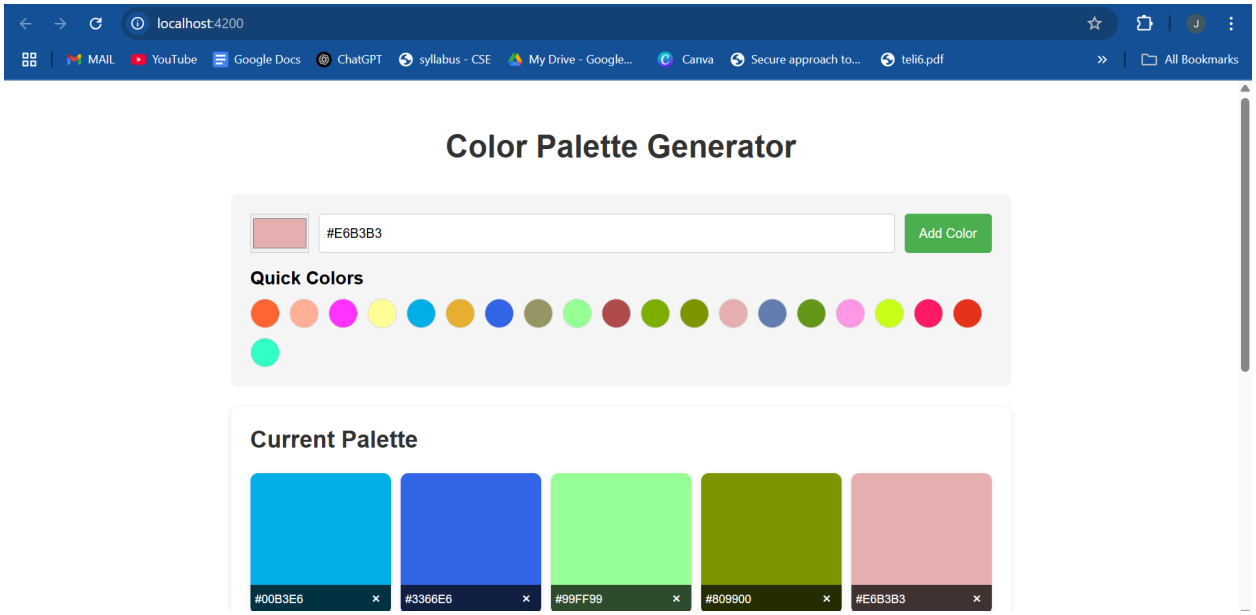
```
      this.colors = [];
    }
  }
}
```

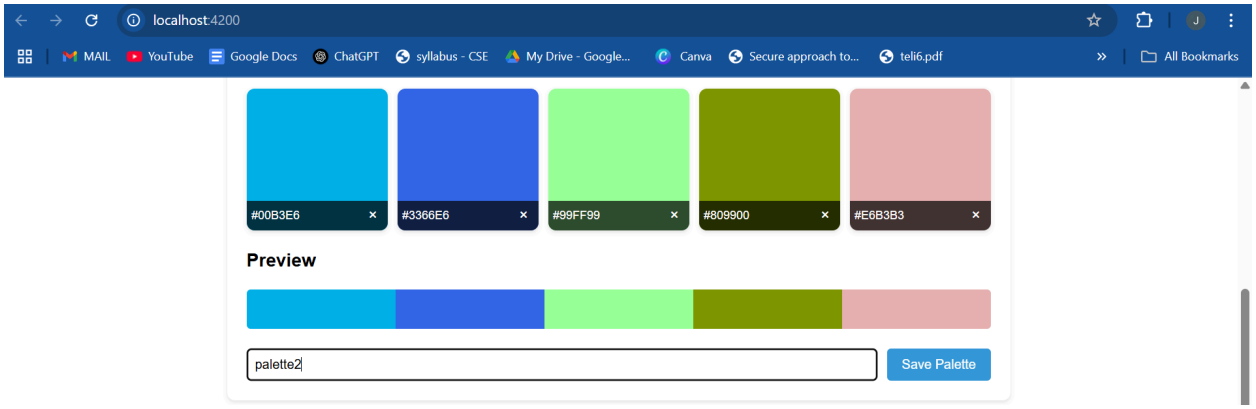


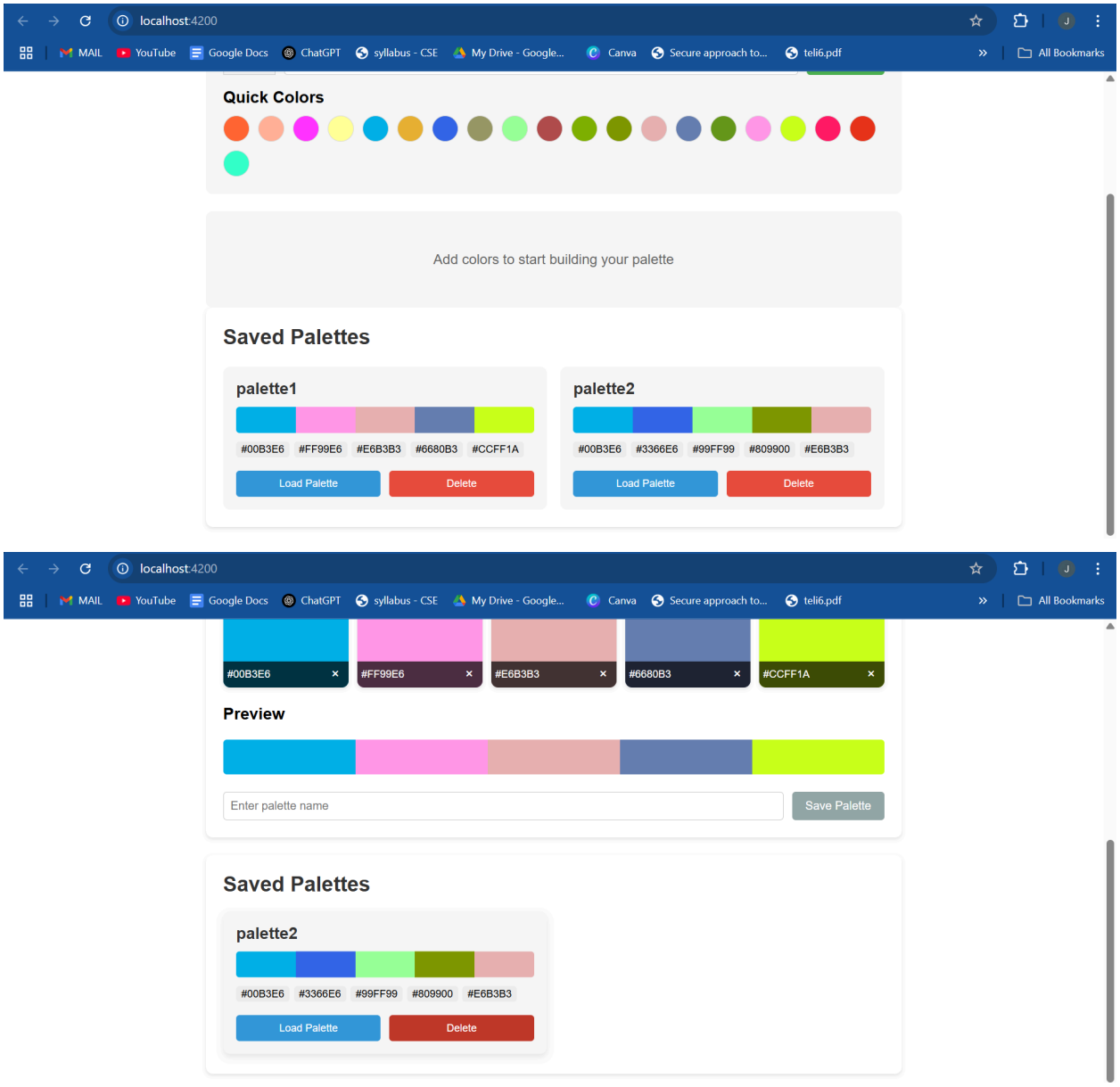
7. Output




localhost:4200
MAIL
YouTube
Google Docs
ChatGPT
syllabus - CSE
My Drive - Google...
Canva
Secure approach to...
teli6.pdf
All Bookmarks
Color Palette Generator
#CCFF1A
Add Color
Quick Colors
Add colors to start building your palette


localhost:4200
MAIL
YouTube
Google Docs
ChatGPT
syllabus - CSE
My Drive - Google...
Canva
Secure approach to...
teli6.pdf
All Bookmarks
Color Palette Generator
#E6B3B3
Add Color
Quick Colors
Current Palette
#00B3E6
#3366E6
#99FF99
#809900
#E6B3B3


localhost:4200
MAIL
YouTube
Google Docs
ChatGPT
syllabus - CSE
My Drive - Google...
Canva
Secure approach to...
teli6.pdf
All Bookmarks
#00B3E6
#3366E6
#99FF99
#809900
#E6B3B3
Preview
palette2
Save Palette



8. Conclusion

You've now created a **Color Palette Generator** app using Angular, which allows users to:

- Select colors.
- Display a palette of selected colors.
- Save and delete color palettes.

This app can be extended with more features such as color theme generation, exporting palettes, or saving them to a backend server for persistence.