Birthday Digital Scrapbook Template

Introduction

Welcome to the Birthday Digital Scrapbook Template! This comprehensive and customizable template is designed to help you create a memorable and interactive digital experience for any birthday celebration. Whether you're commemorating a milestone or simply want to share cherished memories, this template provides a beautiful and engaging platform. It's built with modern web technologies, ensuring a smooth and responsive experience across various devices. This guide will walk you through the features, structure, and customization options to help you make the most of this template for your personal or client projects.

Features

This template comes packed with a variety of features designed to make your digital scrapbook dynamic and engaging:

- **Interactive Entry Experience:** A personalized entry point that sets the tone for the celebration.
- **Dynamic Header:** A customizable header section that can display the birthday person's name and other key information.
- **Hero Section with Countdown:** A prominent hero area featuring a customizable image/video background and a real-time countdown to the birthday.
- **Personalized Music Playlist:** An integrated music player allowing you to curate a special playlist for the celebration, with support for both local files and external streaming services like Spotify.
- **Photo Gallery:** A beautiful and organized photo gallery to showcase cherished memories, complete with captions and categories.
- **Interactive Timeline:** A "Then & Now" timeline feature to highlight significant milestones and moments throughout the years.
- **Video Messages Section:** A dedicated area for heartfelt video messages from loved ones, adding a personal touch to the celebration.
- **Guest Book:** An interactive guest book where friends and family can leave birthday wishes and messages.
- **Virtual Gift Wall:** A unique feature allowing guests to "pin" virtual gifts, images, poems, or links to the birthday wall.

- **Wishlist Integration:** A section to display a birthday wishlist, making it easy for guests to contribute to desired gifts.
- **Responsive Design:** Optimized for various screen sizes, ensuring a seamless experience on desktops, tablets, and mobile devices.
- **Easy Customization:** Clearly structured code and comments make it simple to update content, images, videos, and messages.

Project Structure

The template is organized into a clear and intuitive directory structure to facilitate easy navigation and customization. Below is an overview of the main folders and files:

Plain Text

```
birthday-digital-scrapbook-template/
├── node_modules/
├── public/
│   ├── index.html
│   └── README.MD... (other static assets)
├── src/
│   ├── assets/
│   │   ├── audio/
│   │   │   └── HBD.mp3
│   │   ├── images/
│   │   │   └── ... (various image files)
│   │   └── videos/
│   │       └── vdo_name.mp4
│   ├── components/
│   │   ├── ui/
│   │   │   └── ... (UI-related components)
│   │   ├── EntryExperience.jsx
│   │   ├── GiftWall.jsx
│   │   ├── Guestbook.jsx
│   │   ├── Header.jsx
│   │   ├── Hero.jsx
│   │   ├── PhotoGallery.jsx
│   │   ├── Playlist.jsx
│   │   ├── Timeline.jsx
│   │   └── VideoMessages.jsx
│   ├── hooks/
│   │   └── useBirthdayCountdown.js
│   ├── lib/
│   ├── utils/
│   ├── App.css
│   ├── App.jsx
```

```
│   ├── index.css
│   └── main.jsx
├── components.json
├── eslint.config.js
├── index.html
├── jsconfig.json
├── package.json
├── package-lock.json
├── pnpm-lock.yaml
└── vite.config.js
```

Explanation of Key Directories and Files:

- `node_modules/`: Contains all the installed Node.js dependencies for the project.
- `public/` : Contains the main `index.html` file and other static assets that are served directly by the web server. This is where your compiled application will be rendered.
- `src/` : This is the core of your application, containing all the React components, assets, and logic.
  - `assets/`: Stores all media files used in the template.
    - `audio/` : For local audio files, such as the default `HBD.mp3`.
    - `images/`: For all static images used across the template, including user-specific photos and general UI elements.
    - `videos/`: For local video files, like `vdo_name.mp4` for video messages.
  - `components/`: Houses all the reusable React components that make up the different sections of the scrapbook.
    - `ui/` : Contains smaller, generic UI components that can be reused across different main components.
    - Individual `.jsx` files (e.g., `EntryExperience.jsx` , `Playlist.jsx`): These are the main building blocks of your digital scrapbook, each responsible for a specific feature or section.
  - `hooks/`: Contains custom React hooks, such as `useBirthdayCountdown.js` , to encapsulate reusable logic.
  - `lib/` and `utils/` : Likely contain utility functions and helper libraries.
  - `App.css`, `App.jsx`, `index.css`, `main.jsx`: Standard React application entry points and global styling.
- `components.json` , `eslint.config.js` , `index.html` , `jsconfig.json`, `package.json` , `package-lock.json` , `pnpm-lock.yaml` , `vite.config.js` : These are configuration files for the project, handling dependencies, linting, and build processes.

This guide explains how to run a React project locally, install missing dependencies, build for production, and deploy to Netlify.

1️⃣ Prerequisites
Before you begin, make sure you have:

- Node.js installed (Download: https://nodejs.org/)
- npm (comes with Node.js)
- A code editor like VS Code
- Git (optional but recommended)

2️⃣ Open the Project

1. Download or clone the repository.
2. Open the project folder in your terminal or VS Code.

Example:
cd your-project-folder

3️⃣ Install Dependencies
React projects use a node_modules folder to store dependencies. If it's missing, run:
npm install

This will read package.json and install all required packages.

4️⃣ If `npm install` Fails
Sometimes dependency conflicts cause errors. In that case, run:
```
npm install --legacy-peer-deps
```

This ignores strict peer dependency checks and installs the packages.

5️⃣ Run the Development Server
To start the project locally:
```
npm start
```

or (for Vite projects)

```
npm run dev
```

- `npm start` → Usually for Create React App projects.
- `npm run dev` → Usually for Vite or Next.js-based projects.

Your terminal will show a local development URL (e.g., http://localhost:3000 or http://localhost:5173).

6️⃣ Build for Production
When the app is ready to deploy, create an optimized build:
npm run build

- This creates a build/ folder (Create React App) or dist/ folder (Vite) containing the production files.

7️⃣ Deploy to Netlify

Method 1 — Using the Netlify Website

1. Go to https://app.netlify.com/
2. Create an account (or log in).
3. Click "Add new site" → "Deploy manually".
4. Drag and drop your build/ (or dist/) folder into the Netlify drop zone.

Method 2 — Using GitHub Integration

1. Push your project to GitHub.
2. In Netlify, choose "New site from Git".
3. Connect your GitHub account.
4. Select your repository.
5. Set Build Command to:
   npm run build
6. Set Publish Directory to:

- build (Create React App)
- dist (Vite)

7. **Click Deploy Site.**

8️ **After Deployment**

- Netlify will give you a live site URL like:
  https://yourprojectname.netlify.app
- You can customize the domain from the Netlify dashboard.

✅ Tip: Always test your `npm run build` locally before deploying to catch any build errors early.

Component Breakdown

This section provides a detailed look into each major component of the Birthday Digital Scrapbook Template, explaining its purpose and how to customize it.

index.html

This is the main entry point of your web application. It's a standard HTML file that serves as the container for your React application. You typically won't need to make extensive changes here, as most of the content is dynamically rendered by React. However, you might adjust the page title, meta descriptions, or link to external stylesheets/scripts if necessary.

EntryExperience.jsx

This component handles the initial entry animation or interaction for the user. It's designed to create an engaging first impression before the main content of the scrapbook is revealed. Customization typically involves changing the text, background, or animation sequence to match the birthday theme.

Header.jsx

This component represents the navigation bar or header section of the scrapbook. It usually contains the birthday person's name or a celebratory title, along with navigation links to different sections of the scrapbook (e.g., Photos, Music, Videos, Wishes, Gifts). You can customize the text, styling, and navigation links.

Hero.jsx

The Hero component is the prominent top section of the scrapbook, often featuring a large image or video, a main title, and a countdown timer to the birthday. This is a key area for personalization. You can:

- **Change Background:** Replace the default image or video with your own.
- **Update Title/Message:** Customize the main heading and any accompanying text.
- **Set Countdown Date:** Configure the target date for the birthday countdown.

Playlist.jsx

This component manages the music playlist for the scrapbook. It allows you to curate a collection of songs that remind you of the birthday person or the special moments shared. The component supports both local audio files and integration with Spotify.

Customization:

- **Local Audio:** To use your own local audio files, place them in the `src/assets/audio/` directory. Update the `import` statements and the `audioURL` in the `songs` array within `Playlist.jsx` to point to your new files.
- **Spotify Integration:** The template includes a "Listen on Spotify" button. You can link this to a public Spotify playlist by updating the relevant URL in the component.
- **Song Details:** Modify the `title`, `artist`, `significance`, and `audioURL` for each song in the `songs` array to match your desired playlist.

PhotoGallery.jsx

This component displays a collection of photos, allowing you to showcase cherished memories in an organized and visually appealing manner. Each photo can have a caption, year, and category.

Customization:

- **Add Photos:** Place your image files in the `src/assets/images/` directory. Update the `import` statements at the top of `PhotoGallery.jsx` to import your new images.
- **Photo Data:** Modify the `staticPhotos` array to include details for each of your photos, such as `id`, `src` (referencing your imported image), `caption`, `year`, and `category`.

Timeline.jsx

The Timeline component provides a

visual representation of significant life events or milestones. It's a great way to tell a story through key moments.

Customization:

- **Timeline Events:** Update the data structure within `Timeline.jsx` to define your own events, including dates, descriptions, and associated images or icons.

VideoMessages.jsx

This component is dedicated to displaying video messages from friends and family. It provides a personal and heartfelt touch to the digital scrapbook.

Customization:

- **Add Videos:** Place your video files in the `src/assets/videos/` directory. Update the `import` statements in `VideoMessages.jsx` to import your new videos.
- **Video Data:** Modify the `videos` array to include details for each video, such as `id`, `title`, `thumbnail` (referencing an imported image for the video thumbnail), `videoUrl` (referencing your imported video), `sender`, and `message`.

GuestBook.jsx

The Guest Book component allows visitors to leave written birthday wishes and messages. These messages are preloaded as default messages and can be customized.

Customization:

- **Default Messages:** Modify the `defaultMessages` array in `Guestbook.jsx` to include your own preloaded birthday wishes, including `name`, `message`, `createdAt`, and `color`.

GiftWall.jsx

The Virtual Gift Wall is a unique and interactive feature where guests can

pin virtual gifts, which can be images, poems, or links. This adds a creative and personal touch to the celebration.

Customization:

- **Add Gifts:** The `staticGifts` array in `GiftWall.jsx` defines the types of gifts that can be displayed. You can add new entries to this array. Each entry can be of `type: "image"`, `type: "poem"`, or `type: "link"`.
  - For `type: "image"`, specify `title`, `content` (URL to the image), `sender`, and `message`.
  - For `type: "poem"`, specify `title`, `content` (the poem text, with `\n` for new lines), `sender`, and `message`.
  - For `type: "link"`, specify `title`, `content` (the URL), `sender`, and `message`.
- **Pin Colors:** The `pinColor` property allows you to customize the color of each pinned gift.

Wishlist

The Wishlist section provides a clear and organized way to display items the birthday person desires. This makes it convenient for guests who wish to contribute to gifts.

Customization:

- **Update Wishlist Items:** The wishlist items are likely defined within a component (e.g., `Wishlist.jsx` or similar, though not explicitly shown in code snippets for this component). You would typically modify an array of objects, each representing a wishlist item with properties like `name`, `description`, `link`, and `image`.

How to Use and Customize

This template is designed for easy customization, even for those with basic knowledge of React and web development. Here's a general guide:

1. **Clone the Repository:**
   If you received this template as a repository, clone it to your local machine:
2. **Install Dependencies:**
   Navigate to the project directory and install the necessary Node.js packages:
3. **Run the Development Server:**
   Start the development server to view the template in your browser. Any changes you make to the code will automatically reload the page.
4. **Content Customization:**
   As detailed in the "Component Breakdown" section, most of the content (text, images, videos, audio, messages) can be customized by editing the `.jsx` files within the `src/components/` directory and updating the `src/assets/` folder.
   - **Text:** Look for `// TODO:` comments in the `.jsx` files. These indicate areas where you can easily change titles, descriptions, messages, and other text content. **Additionally, please note that the red boxes highlighted in the provided screenshots indicate specific areas within the code or UI that are intended for user modification and customization.**
   - **Images & Videos:** Replace placeholder images and videos in `src/assets/images/` and `src/assets/videos/` with your own. Remember to update the `import` paths and data arrays in the corresponding `.jsx` components (e.g., `PhotoGallery.jsx`, `VideoMessages.jsx`).
   - **Audio:** For the playlist, place your audio files in `src/assets/audio/` and update `Playlist.jsx` accordingly.
   - **Dates & Times:** For countdowns and timelines, adjust date and time values in the relevant components.
5. **Styling Customization:**
   The template uses CSS for styling. You can modify the appearance by editing the CSS files:
   - `App.css` and `index.css`: These files contain global styles. You can change fonts, colors, and general layout properties here.
   - Component-specific styling: Some components might have inline styles or use CSS modules/styled-components (depending on implementation details not fully visible in screenshots). Refer to the individual component files for specific styling adjustments.
6. **Deployment:**
   Once you are satisfied with your customizations, you can build the project for deployment. This will create an optimized production build of your application.

Installation

To get started with this template, follow these steps:

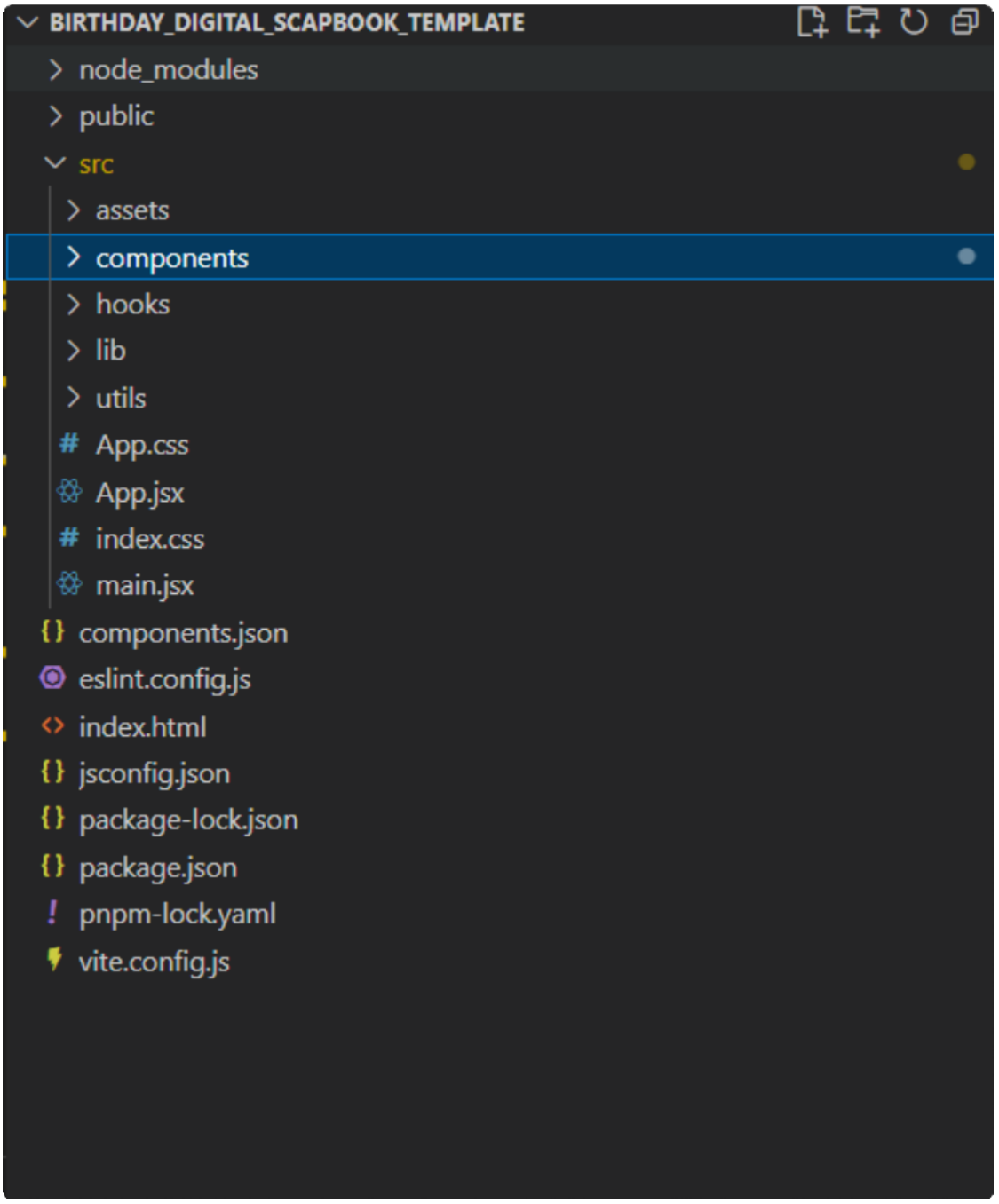
1. **Prerequisites:**
   Make sure you have Node.js (which includes npm) installed on your system. You can download it from nodejs.org. Yarn is also an option if you prefer.
2. **Download the Template:**
   Obtain the template files. If you received a `.zip` archive, extract its contents to your desired project directory.
3. **Navigate to Project Directory:**
   Open your terminal or command prompt and navigate to the root directory of the extracted template:
4. **Install Dependencies:**
   Run the following command to install all the required project dependencies:
5. **Start Development Server:**
   Once dependencies are installed, you can start the local development server:
6. **Begin Customization:**
   You are now ready to customize the template. Refer to the "How to Use and Customize" and "Component Breakdown" sections for detailed instructions on modifying content and styling.

Basic File Structure

Below is a visual representation of the template's basic file structure, providing a quick overview of how files are organized.

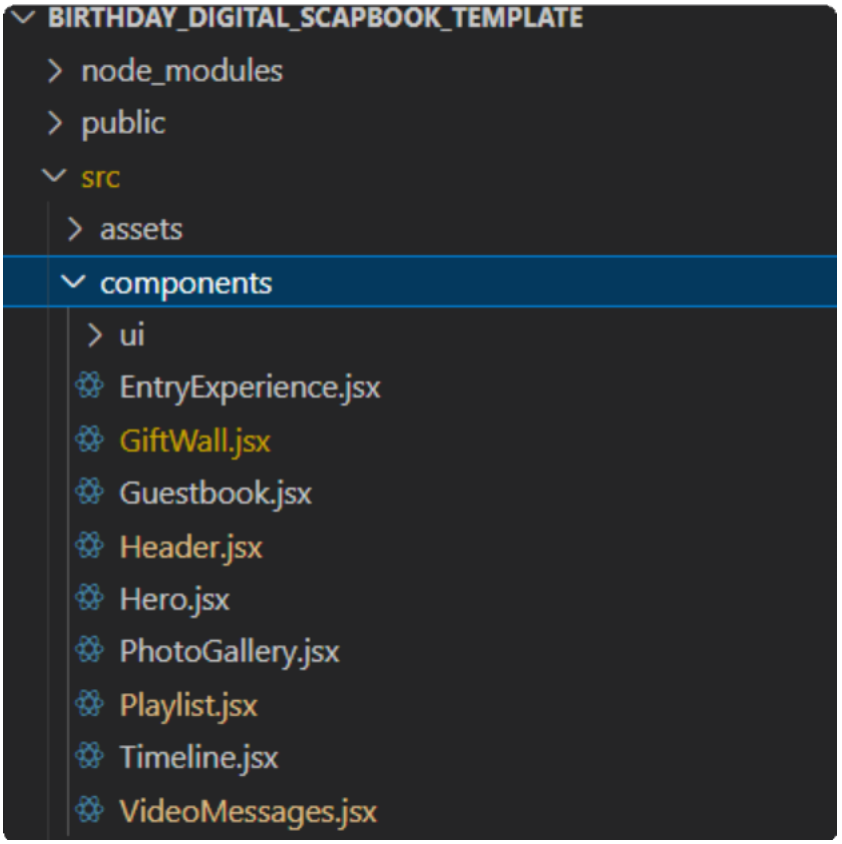
Basic File Structure

Below is a visual representation of the template's basic file structure, providing a quick overview of how files are organized.



Component Structure

This diagram illustrates the high-level component architecture of the template, showing how different parts of the application are modularized.

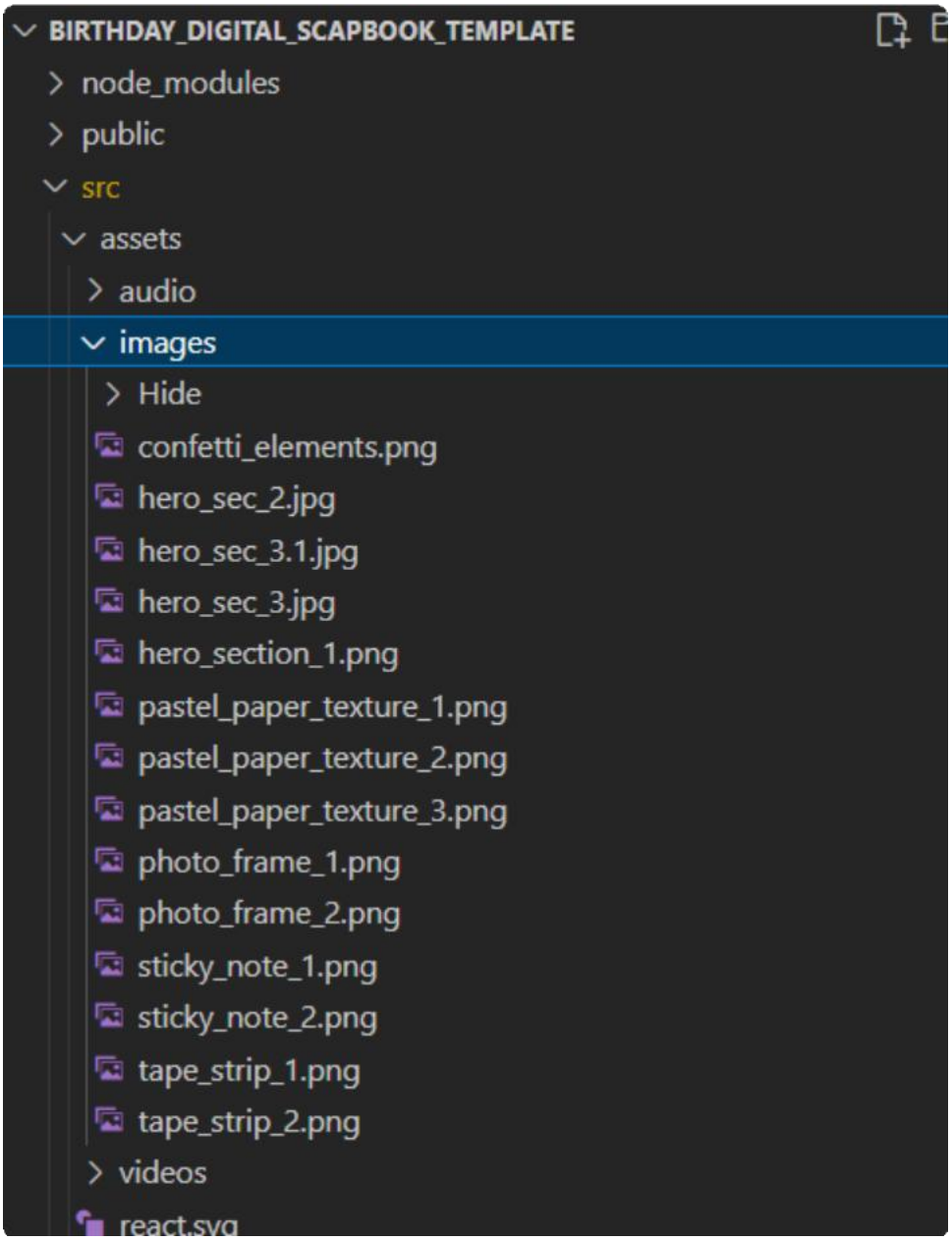


Resource Structure

This image details the organization of assets (audio, images, videos) within the `src/assets` directory, crucial for managing your media files.

Resource Structure

This image details the organization of assets (audio, images, videos) within the `src/assets` directory, crucial for managing your media files.



`index.html` Overview

This screenshot shows the basic structure of the `index.html` file, which serves as the entry point for the application.

index.html

```
index.html > ...
1   <!doctype html>
2   <html lang="en">
3     <head>
4       <meta charset="UTF-8" />
5       <link rel="icon" type="image/x-icon" href="/favicon.ico" />
6       <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7       <title>Happy Birthday { Name } - Digital Scrapbook</title>
8     </head>
9     <body>
10      <div id="root"></div>
11      <script type="module" src="/src/main.jsx"></script>
12    </body>
13  </html>
```

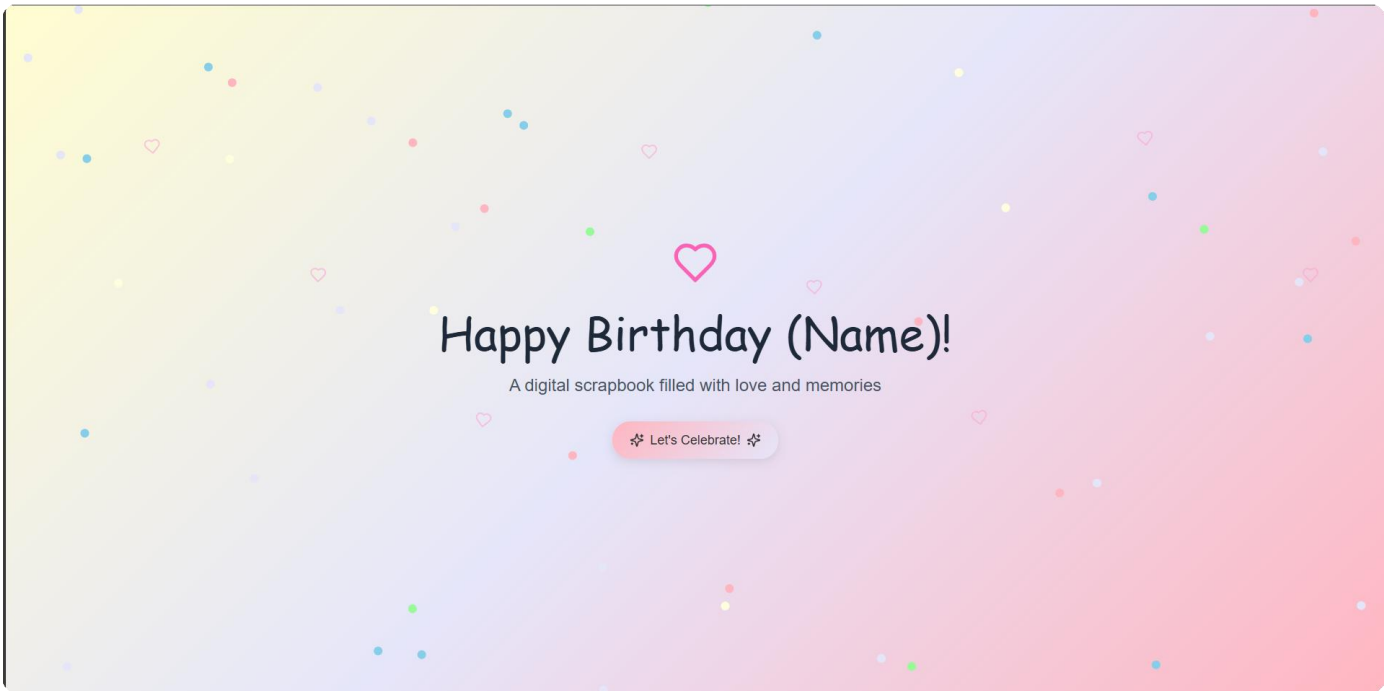
Entry Experience Component

These images show the code and the visual output of the `EntryExperience.jsx` component, highlighting its role in creating an engaging initial user interaction.

EntryExperience.jsx 9

src > components > EntryExperience.jsx > EntryExperience

```
6   const EntryExperience = ({ onEnter }) => {
79            <motion.div
80              initial={{ scale: 0 }}
81              animate={{ scale: 1 }}
82              transition={{ duration: 0.8, type: 'spring', bounce: 0.4 }}
83              className="mb-8"
84            >
85              {/* 💔 Icon above title */}
86              <Heart className="w-16 h-16 mx-auto text-pink-400 mb-4" />
87
88              {/* **TODO: Change the name inside title** */}
89              <h1 className="handwritten text-responsive-xl text-gray-800 mb-2">
90                Happy Birthday (Name)!
91              </h1>
92
93              {/* **TODO: Change tagline/subtitle here** */}
94              <p className="text-responsive-md text-gray-600">
95                A digital scrapbook filled with love and memories
96              </p>
97            </motion.div>
98
99            {/* Countdown display */}
100           {countdown > 0 && (
101             <motion.div
102               key={countdown}
103               initial={{ scale: 0, opacity: 0 }}
104               animate={{ scale: 1, opacity: 1 }}
105               exit={{ scale: 0, opacity: 0 }}
```



Header Component

This screenshot displays the code for the `Header.jsx` component, which manages the top navigation and branding of the scrapbook.

```
src > components > Header.jsx > Header
15   const Header = () => {
62       >
63         <div className="container mx-auto px-4 py-4">
64           <div className="flex items-center justify-between">
65             {/* **🎉 Site title** */}
66             // TODO: Replace "(Name)" with the actual birthday person's name
67             <motion.div
68               className="handwritten text-2xl md:text-3xl text-pink-600 font-bold"
69               whileHover={{ scale: 1.05 }}
70               onClick={() => scrollToSection('hero')}
71             >
72               (Name)'s Birthday
73             </motion.div>
74
75             {/* **💻 Desktop Navigation (visible on large screens)** */}
76             {/* **TODO: Update icons or labels if needed** */}
77             <nav className="hidden lg:flex items-center space-x-6">
78               {navItems.map(({ id, label, icon: Icon }) => (
79                 <motion.button
80                   key={id}
81                   onClick={() => scrollToSection(id)}
82                   className="flex items-center gap-2 px-3 py-2 rounded-full text-gray-700 hover:text-pink-600 tra
83                   whileHover={{ scale: 1.05 }}
84                   whileTap={{ scale: 0.95 }}
85                 >
```

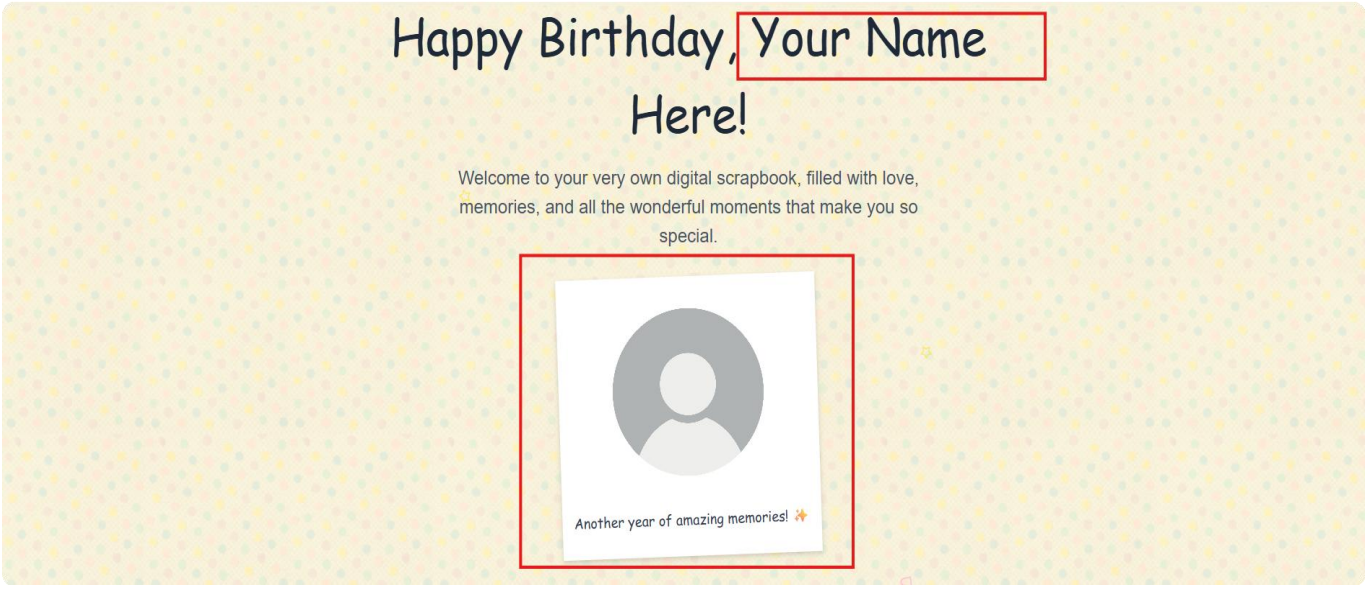
Hero Component

These images illustrate the `Hero.jsx` component, showing its code structure and two variations of its visual appearance, demonstrating its customizable nature.

Hero.jsx

src > components > Hero.jsx > ...

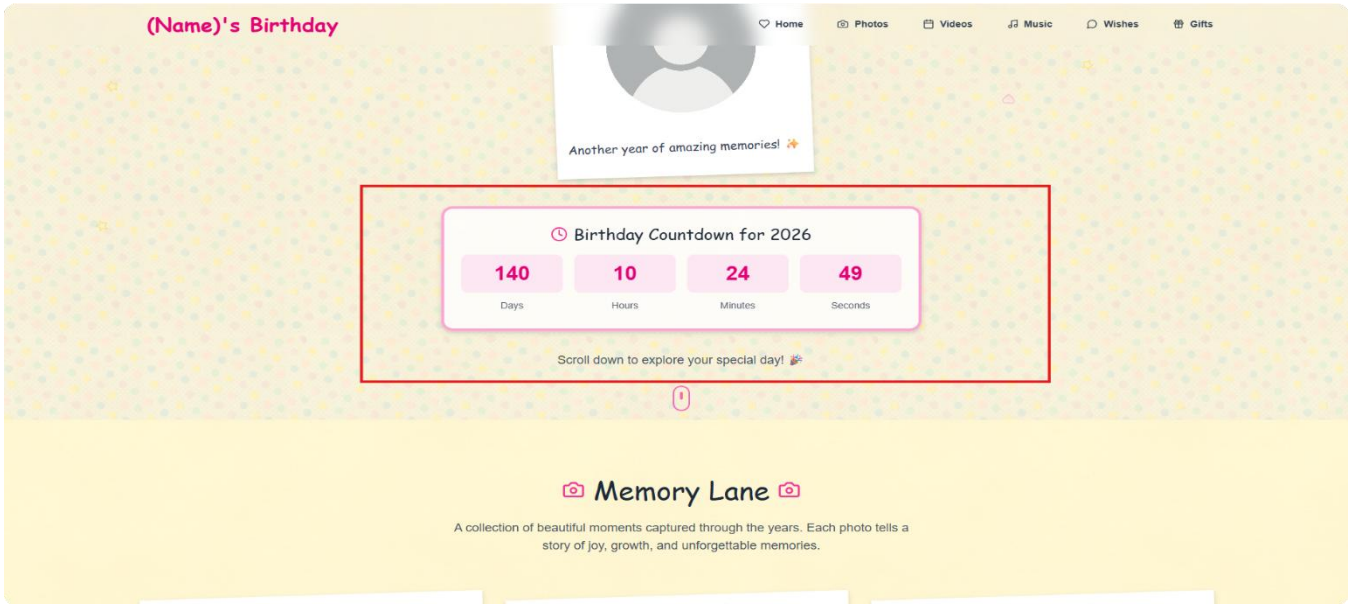
```
10  const Hero = () => {
159         </div>
160
161         <div className="container mx-auto px-4 text-center relative z-10">
162           <motion.div
163             variants={containerVariants}
164             initial="hidden"
165             animate="visible"
166             className="max-w-4xl mx-auto"
167           >
168             <motion.div variants={itemVariants} className="mb-8 mt-14">
169               <h1 className="handwritten text-responsive-xl text-gray-800 mb-4">
170                 Happy Birthday, {celebrantName}!
171               </h1>
172               <p className="text-responsive-md text-gray-600 max-w-2xl mx-auto">
173                 Welcome to your very own digital scrapbook, filled with love, memories,
174                 and all the wonderful moments that make you so special.
175               </p>
176             </motion.div>
177
178             <motion.div variants={itemVariants} className="mb-12 flex justify-center">
179               <div className="polaroid-frame scrapbook-shadow max-w-sm">
180                 <div className="w-full h-64 rounded-lg overflow-hidden mb-4">
181                   <img
182                     src={default_pic}
```



Hero.jsx 7, M

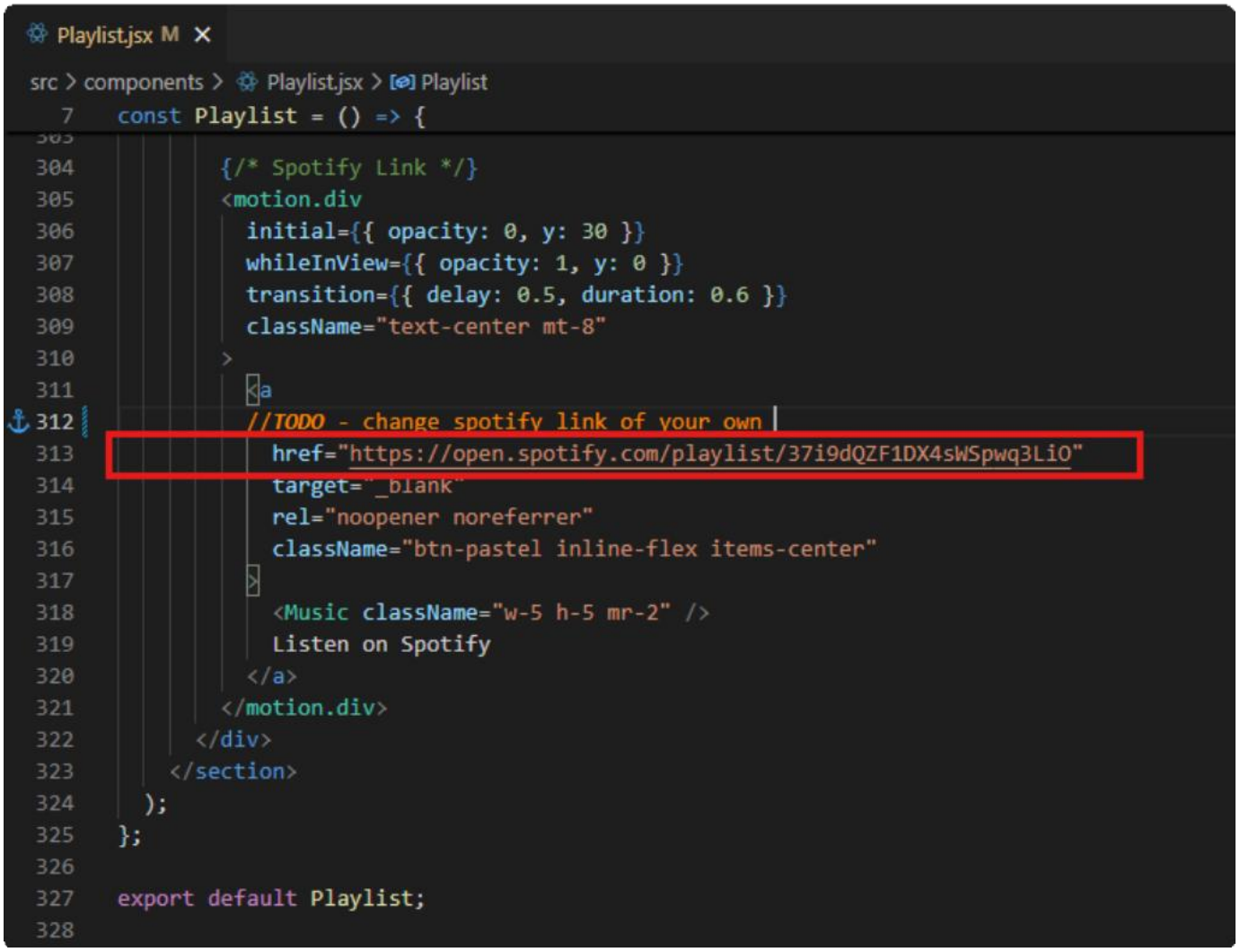
src > components > Hero.jsx > Hero

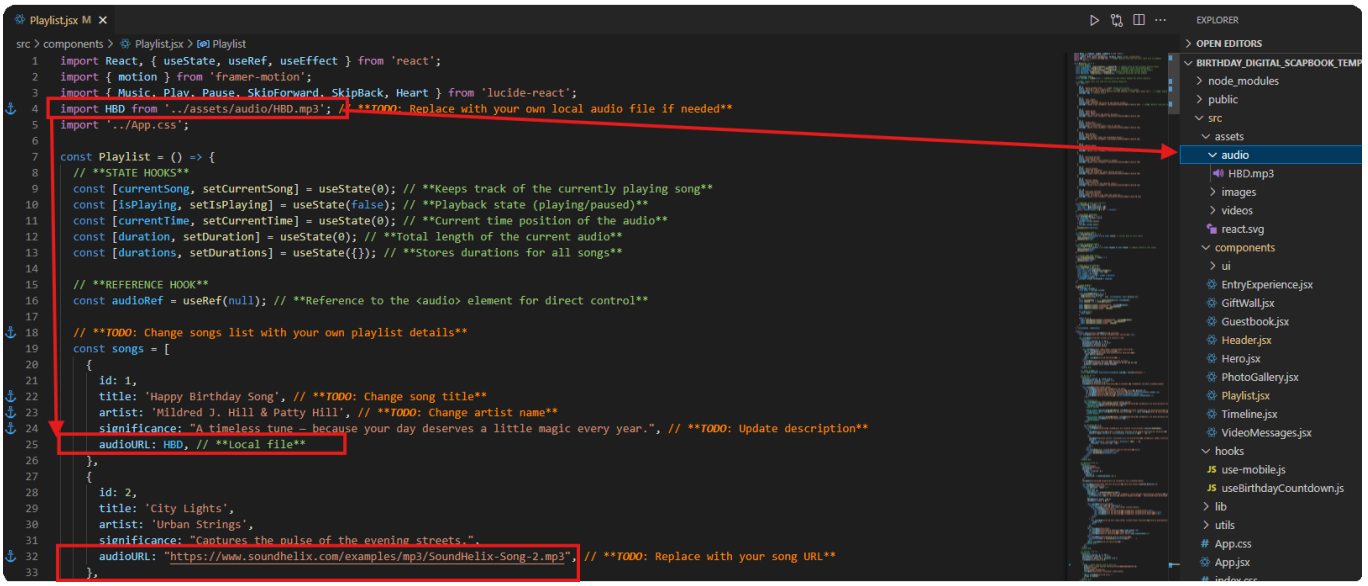
```jsx
import React, { useState, useEffect } from 'react';
import { motion } from 'framer-motion';
import Confetti from 'react-confetti';
import { useWindowSize } from '@react-hook/window-size';
import { Clock, Heart, Star } from 'lucide-react';
// **TODO: Change to your desired image**
import default_pic from '../assets/images/Hide/user_icon.jpg';
import '../App.css';

const Hero = () => {
  // **TODO: Change this name**
  const celebrantName = 'Your Name Here';

  // **TODO: Set birthday month & day (MM-DD)**
  const birthdayDate = '01-01';

  const [timeLeft, setTimeLeft] = useState({
    days: 0,
    hours: 0,
    minutes: 0,
    seconds: 0,
  });

  const [isBirthday, setIsBirthday] = useState(false);
  const [countdownFinished, setCountdownFinished] = useState(false);
  const [hasStartedCelebration, setHasStartedCelebration] = useState(false);
  const [currentYear, setCurrentYear] = useState(new Date().getFullYear());
  const [width, height] = useWindowSize();
```

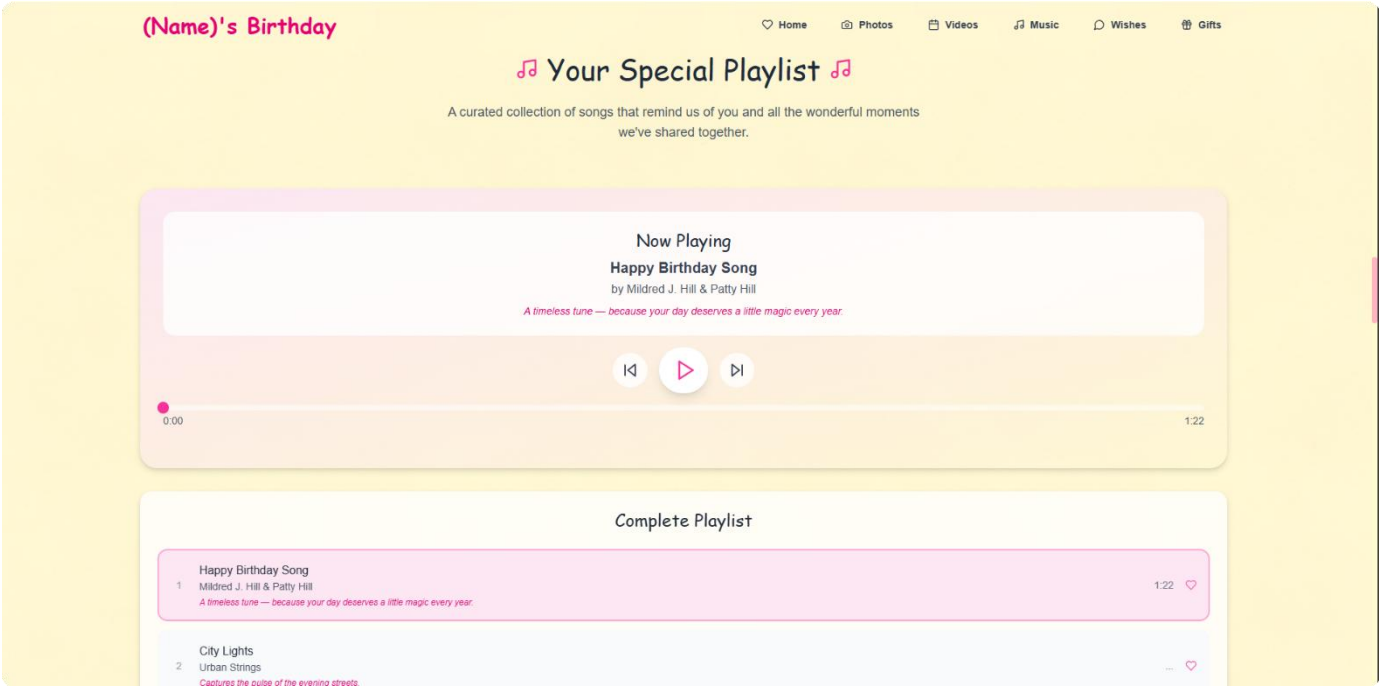


Playlist Component

These screenshots demonstrate the `Playlist.jsx` component, including its code structure, how local audio files are referenced, and the visual appearance of the music player and Spotify integration.




Playlist.jsx M
src > components > Playlist.jsx > Playlist
import React, { useState, useRef, useEffect } from 'react';
import { motion } from 'framer-motion';
import { Music, Play, Pause, SkipForward, SkipBack, Heart } from 'lucide-react';
import HBD from '../assets/audio/HBD.mp3'; // **TODO: Replace with your own local audio file if needed**
import '../App.css';
const Playlist = () => {
// **STATE HOOKS**
const [currentSong, setCurrentSong] = useState(0); // **Keeps track of the currently playing song**
const [isPlaying, setIsPlaying] = useState(false); // **Playback state (playing/paused)**
const [currentTime, setCurrentTime] = useState(0); // **Current time position of the audio**
const [duration, setDuration] = useState(0); // **Total length of the current audio**
const [durations, setDurations] = useState({}); // **Stores durations for all songs**
// **REFERENCE HOOK**
const audioRef = useRef(null); // **Reference to the <audio> element for direct control**
// **TODO: Change songs list with your own playlist details**
const songs = [
{
id: 1,
title: 'Happy Birthday Song', // **TODO: Change song title**
artist: 'Mildred J. Hill & Patty Hill', // **TODO: Change artist name**
significance: "A timeless tune — because your day deserves a little magic every year.", // **TODO: Update description**
audioURL: HBD, // **Local file**
},
{
id: 2,
title: 'City Lights',
artist: 'Urban Strings',
significance: "Captures the pulse of the evening streets.",
audioURL: "https://www.soundhelix.com/examples/mp3/SoundHelix-Song-2.mp3", // **TODO: Replace with your song URL**
},
EXPLORER
OPEN EDITORS
BIRTHDAY_DIGITAL_SCAPBOOK_TEMP
node_modules
public
src
assets
audio
HBD.mp3
images
videos
react.svg
components
ui
EntryExperience.jsx
GiftWall.jsx
Guestbook.jsx
Header.jsx
Hero.jsx
PhotoGallery.jsx
Playlist.jsx
Timeline.jsx
VideoMessages.jsx
hooks
use-mobile.js
useBirthdayCountdown.js
lib
utils
App.css
App.jsx


(Name)'s Birthday
Home
Photos
Videos
Music
Wishes
Gifts
Your Special Playlist
A curated collection of songs that remind us of you and all the wonderful moments we've shared together.
Now Playing
Happy Birthday Song
by Mildred J. Hill & Patty Hill
A timeless tune — because your day deserves a little magic every year.
0:00
1:22
Complete Playlist
1
Happy Birthday Song
Mildred J. Hill & Patty Hill
A timeless tune — because your day deserves a little magic every year.
1:22
2
City Lights
Urban Strings
Captures the pulse of the evening streets.

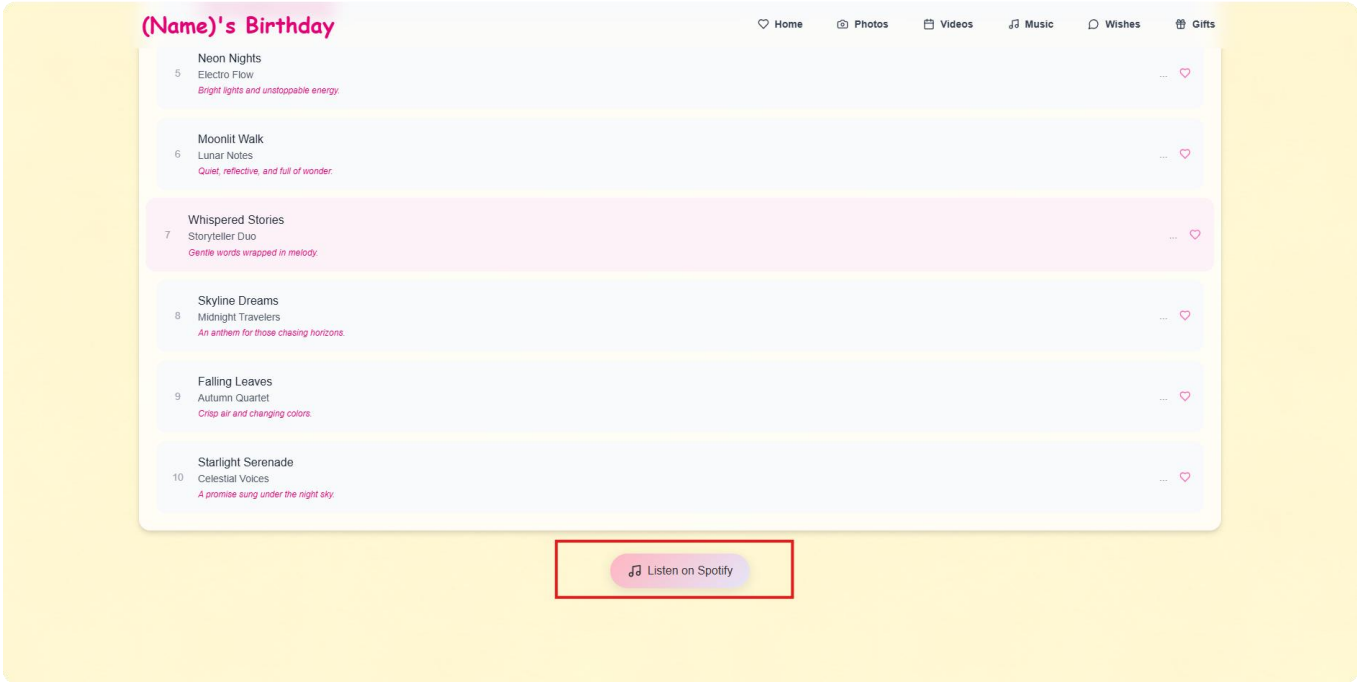
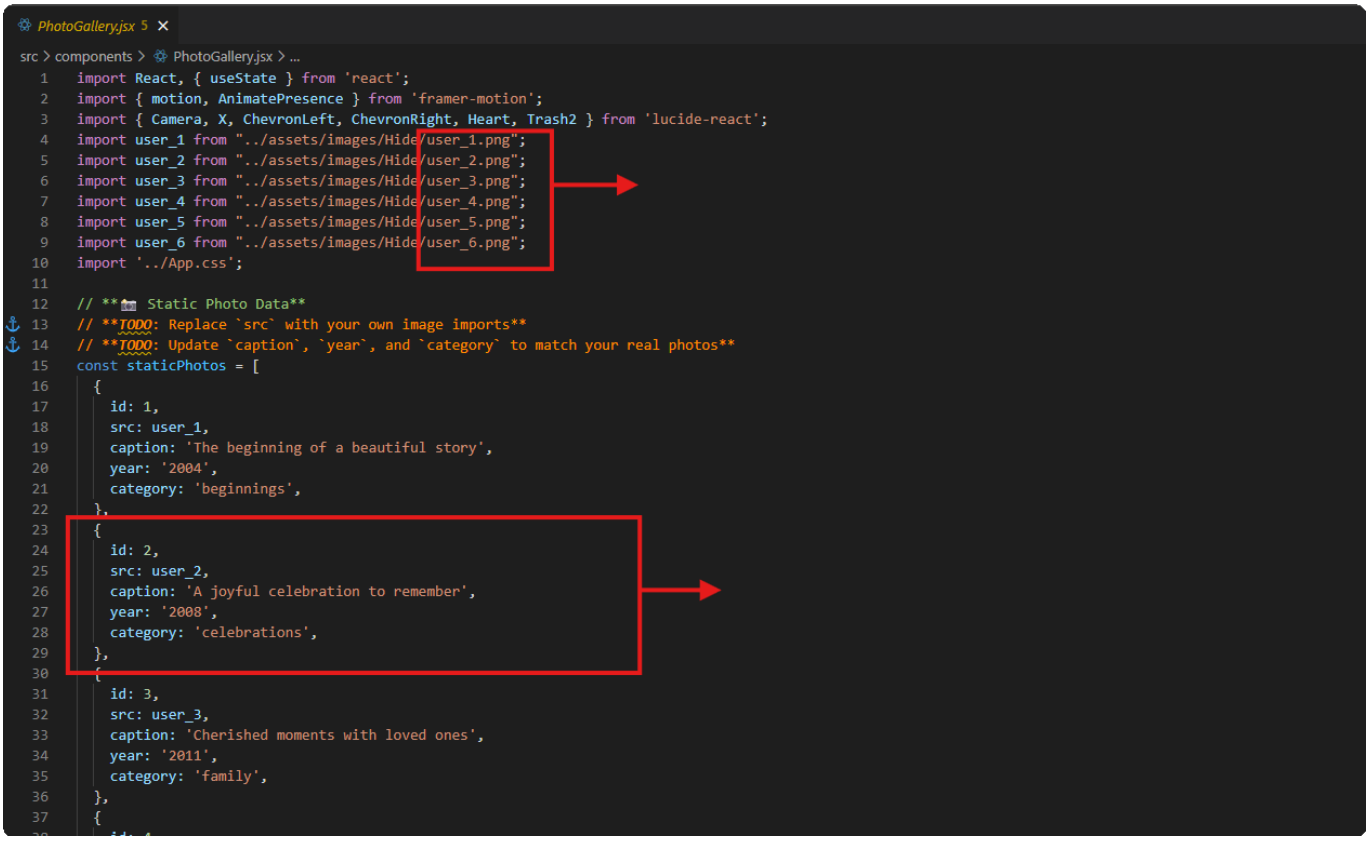
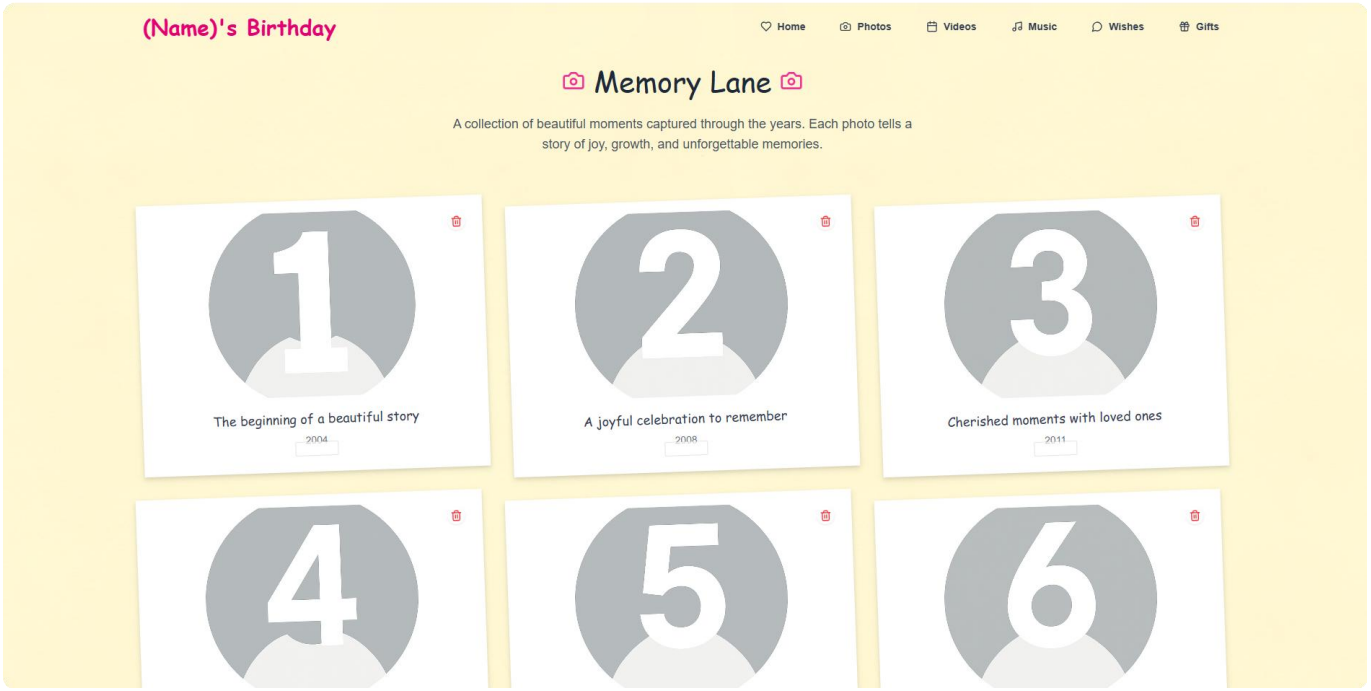


Photo Gallery Component

These images show the `PhotoGallery.jsx` component, detailing its code for managing static photo data and the visual layout of the photo gallery.

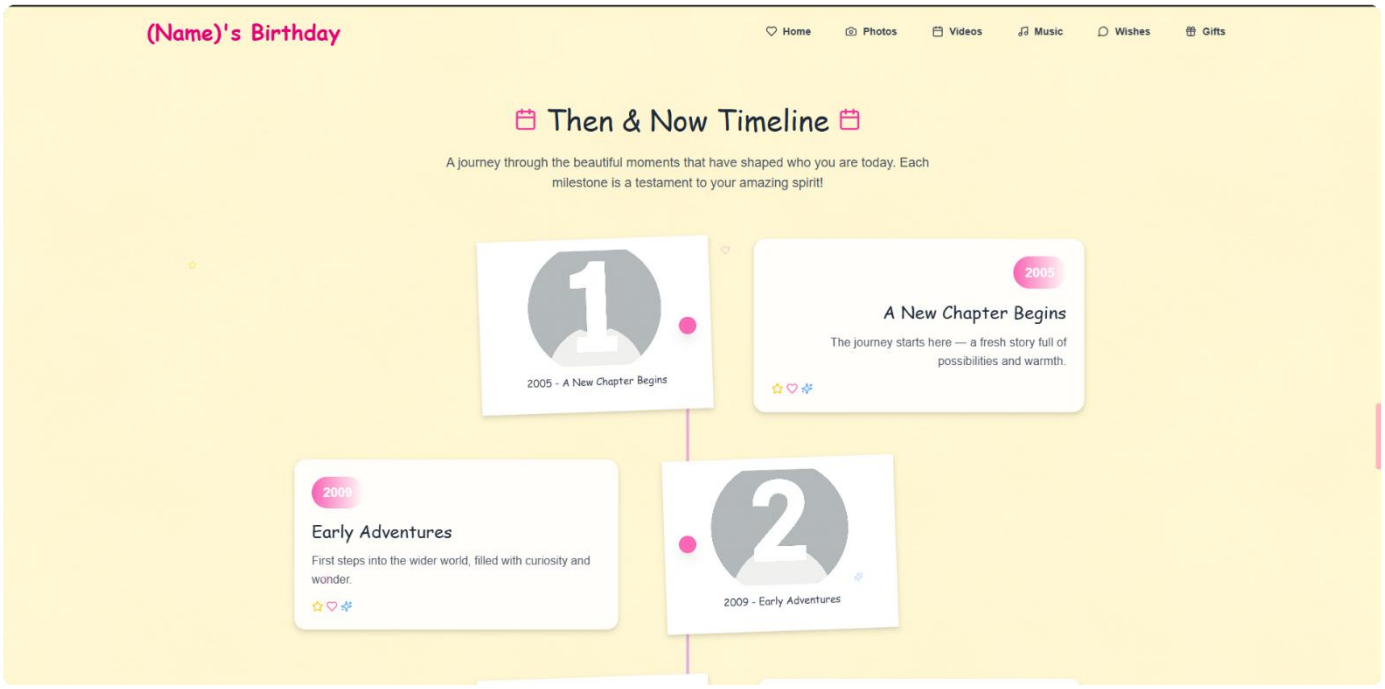


```
PhotoGallery.jsx 5
src > components > PhotoGallery.jsx > ...
1  import React, { useState } from 'react';
2  import { motion, AnimatePresence } from 'framer-motion';
3  import { Camera, X, ChevronLeft, ChevronRight, Heart, Trash2 } from 'lucide-react';
4  import user_1 from "../assets/images/Hide/user_1.png";
5  import user_2 from "../assets/images/Hide/user_2.png";
6  import user_3 from "../assets/images/Hide/user_3.png";
7  import user_4 from "../assets/images/Hide/user_4.png";
8  import user_5 from "../assets/images/Hide/user_5.png";
9  import user_6 from "../assets/images/Hide/user_6.png";
10 import '../App.css';
11
12 // **📸 Static Photo Data**
13 // **TODO: Replace `src` with your own image imports**
14 // **TODO: Update `caption`, `year`, and `category` to match your real photos**
15 const staticPhotos = [
16   {
17     id: 1,
18     src: user_1,
19     caption: 'The beginning of a beautiful story',
20     year: '2004',
21     category: 'beginnings',
22   },
23   {
24     id: 2,
25     src: user_2,
26     caption: 'A joyful celebration to remember',
27     year: '2008',
28     category: 'celebrations',
29   },
30   {
31     id: 3,
32     src: user_3,
33     caption: 'Cherished moments with loved ones',
34     year: '2011',
35     category: 'family',
36   },
37   {
```



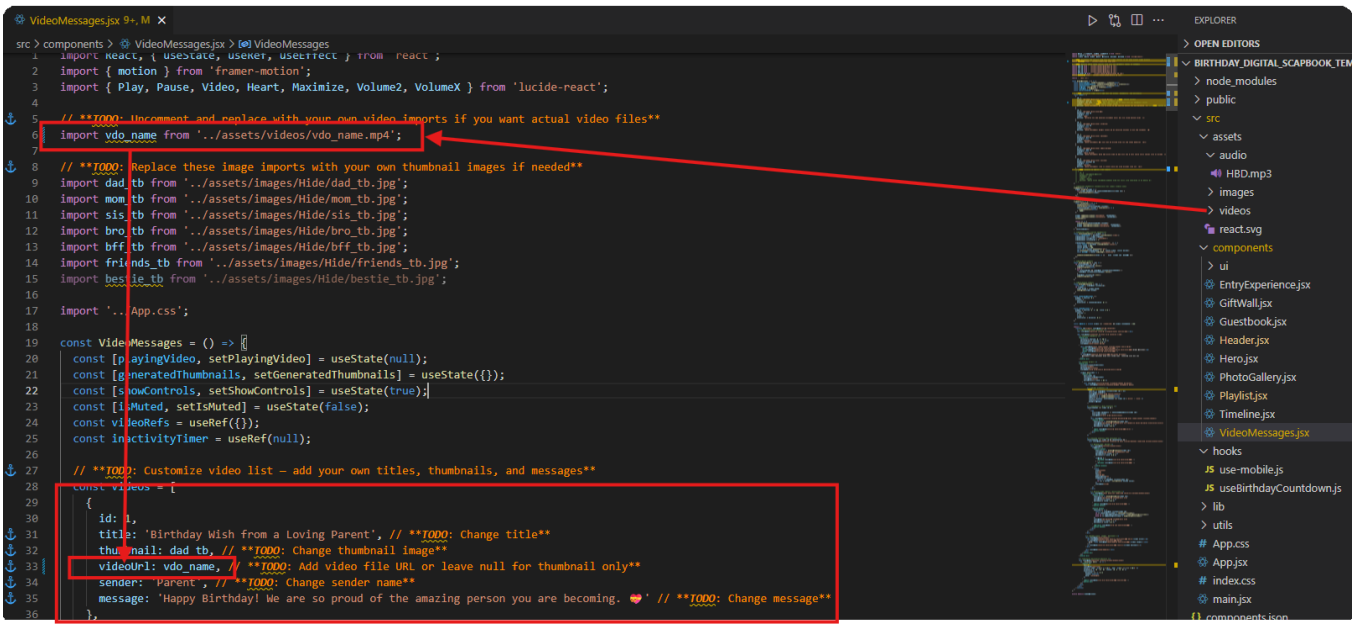
Timeline Component

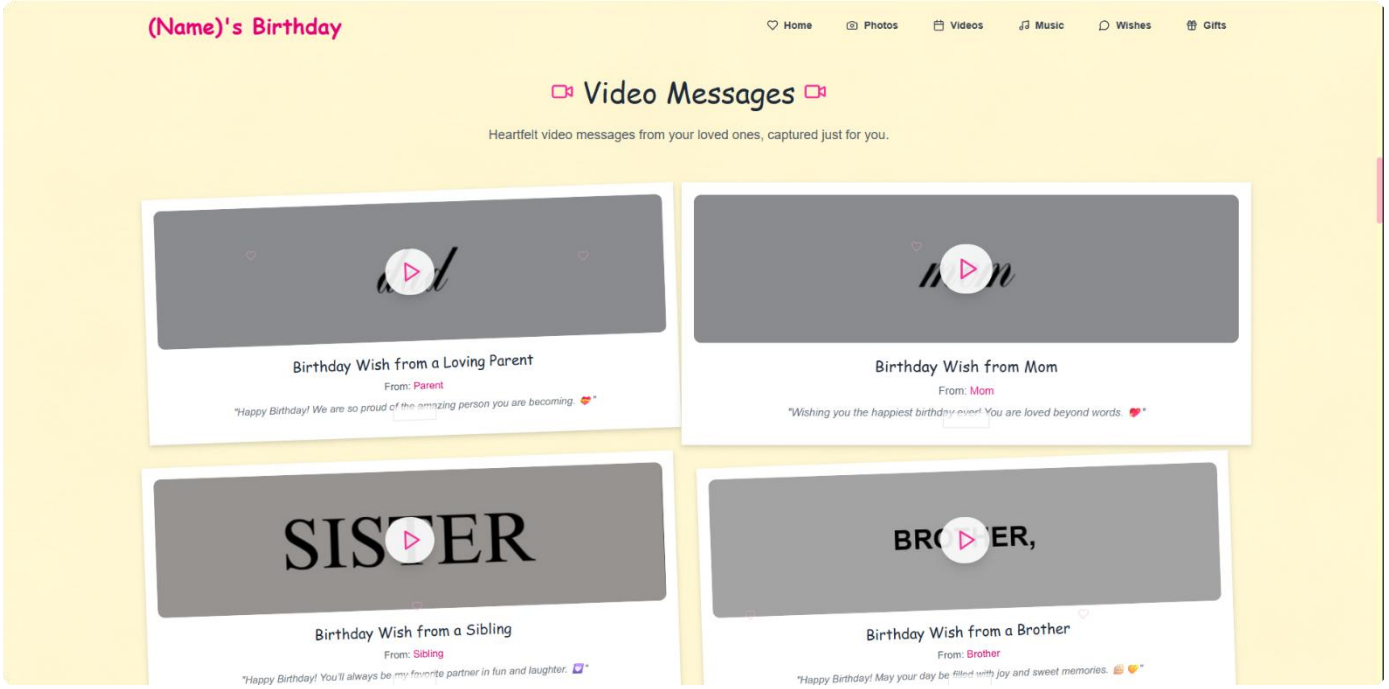
This screenshot provides a visual of the "Then & Now Timeline" component, showcasing how milestones are presented.



Video Messages Component

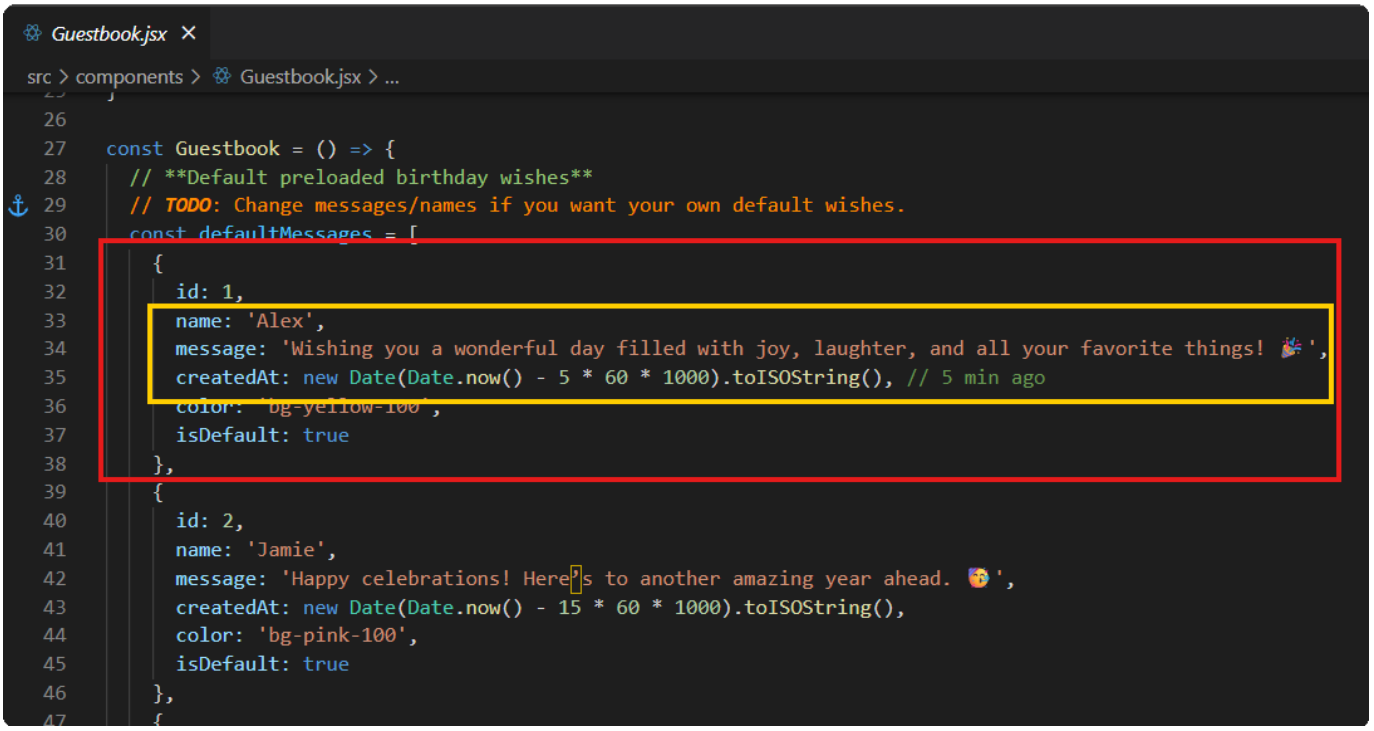
These images display the `VideoMessages.jsx` component, including its code for video management and the visual presentation of video messages.





Guest Book Component

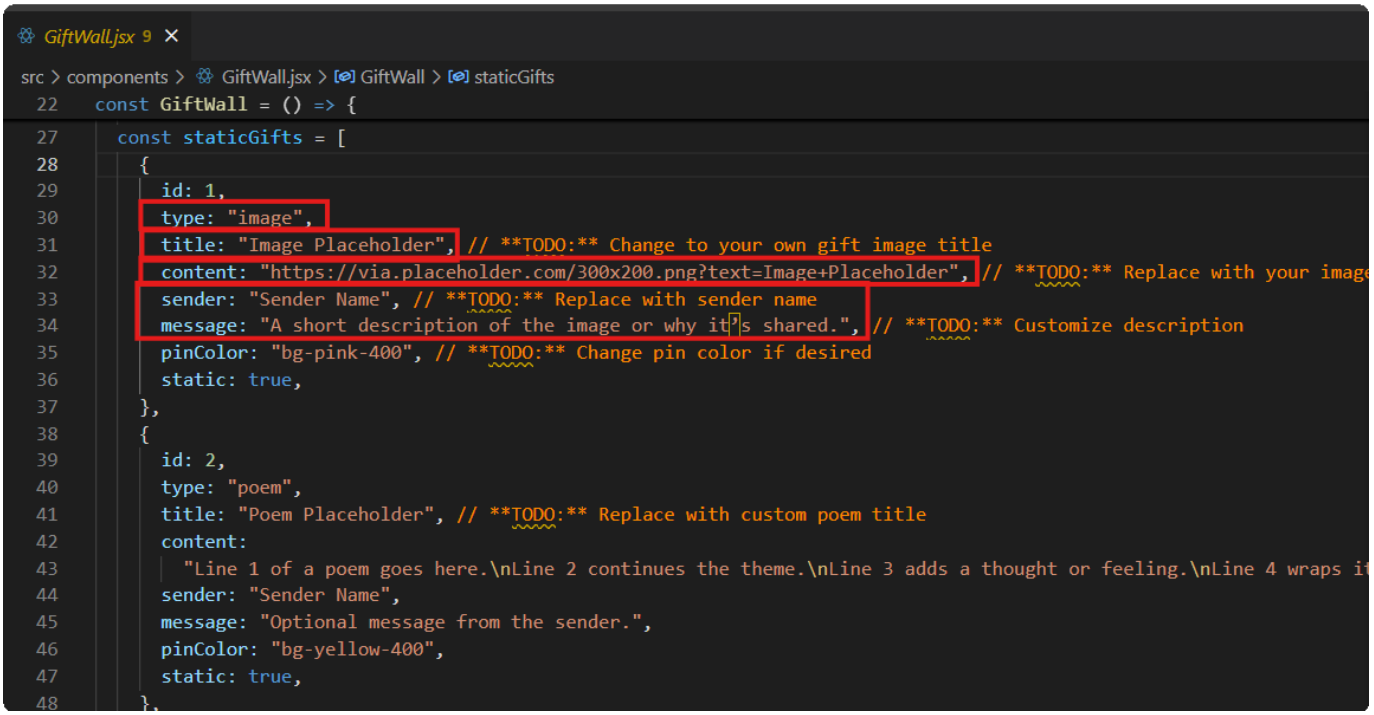
This screenshot shows the `GuestBook.jsx` component, specifically how default messages are structured within the code.



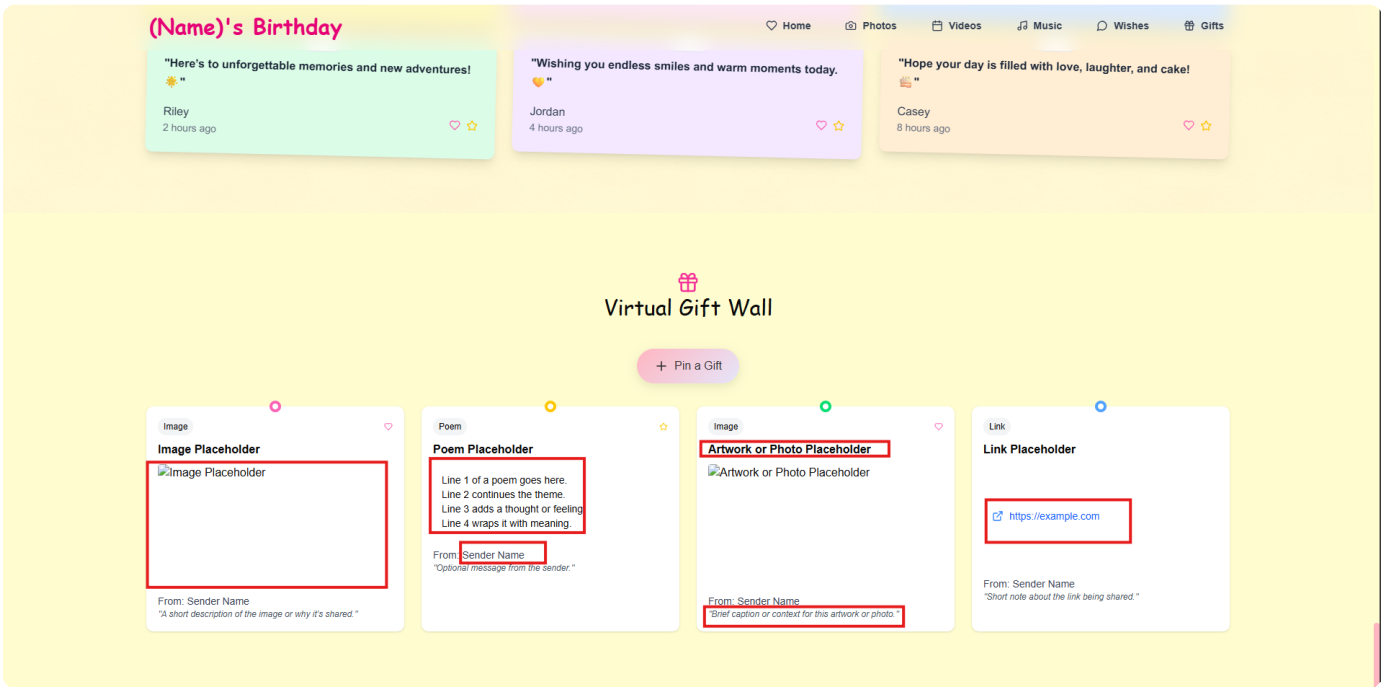
```jsx
// Guestbook.jsx
// src > components > Guestbook.jsx > ...

const Guestbook = () => {
  // **Default preloaded birthday wishes**
  // TODO: Change messages/names if you want your own default wishes.
  const defaultMessages = [
    {
      id: 1,
      name: 'Alex',
      message: 'Wishing you a wonderful day filled with joy, laughter, and all your favorite things! 🎉',
      createdAt: new Date(Date.now() - 5 * 60 * 1000).toISOString(), // 5 min ago
      color: 'bg-yellow-100',
      isDefault: true
    },
    {
      id: 2,
      name: 'Jamie',
      message: 'Happy celebrations! Here's to another amazing year ahead. 🥳',
      createdAt: new Date(Date.now() - 15 * 60 * 1000).toISOString(),
      color: 'bg-pink-100',
      isDefault: true
    },
```

Gift Wall Component

These images illustrate the `GiftWall.jsx` component, showing its code for defining different gift types and the visual layout of the virtual gift wall.



```jsx
// GiftWall.jsx 9
// src > components > GiftWall.jsx > GiftWall > staticGifts

const GiftWall = () => {
  const staticGifts = [
    {
      id: 1,
      type: "image",
      title: "Image Placeholder", // **TODO:** Change to your own gift image title
      content: "https://via.placeholder.com/300x200.png?text=Image+Placeholder", // **TODO:** Replace with your image
      sender: "Sender Name", // **TODO:** Replace with sender name
      message: "A short description of the image or why it's shared.", // **TODO:** Customize description
      pinColor: "bg-pink-400", // **TODO:** Change pin color if desired
      static: true,
    },
    {
      id: 2,
      type: "poem",
      title: "Poem Placeholder", // **TODO:** Replace with custom poem title
      content:
        "Line 1 of a poem goes here.\nLine 2 continues the theme.\nLine 3 adds a thought or feeling.\nLine 4 wraps it
      sender: "Sender Name",
      message: "Optional message from the sender.",
      pinColor: "bg-yellow-400",
      static: true,
    },
```



Wishlist Component

This screenshot shows the visual appearance of the Birthday Wishes section, which includes the guest book and potentially a wishlist.