PROJECT SYNOPSIS & REPORT

Paint Application

Submitted By

Divyansh Mishra
6210040013, BCA

Submitted To

Mr. Akshit Kumar
PSR Gov. Degree College Baijnath

Certificate

This is to certify that the project report titled **"Paint Application"** submitted by **Divyansh Mishra** is a bonafide record of the work carried out by the student in partial fulfillment of the requirements for the award of the Degree of **Bachelor of Computer Applications (BCA)**.

The project was completed under my guidance and supervision at **Pandit Sant Ram Government Degree College Baijnath.**

Signature of the Supervisor

Name of the Supervisor

Acknowledgment

I would like to express my sincere gratitude to Mr. Akshit, my project supervisor, for their invaluable guidance and support throughout this endeavor. Their mentorship has been instrumental in shaping the outcome of this project.

I am thankful to the faculty of the **BCA** at **Pandit Sant Ram Govt. Degree College Baijnath** for their dedication to imparting knowledge and fostering an environment conducive to learning.

I appreciate my classmates and colleagues for the collaboration, feedback, and camaraderie, which have enriched this project.

Finally, I acknowledge my family and friends' unwavering support and encouragement, which has been a driving force throughout this journey.
The collective efforts and contributions of these individuals and organizations have been instrumental in the completion of this project.

Student Name: Divyansh Mishra

Project Name: Paint Application

UID: 6210040013

Class Roll No: 21BCA041

Index

Abstract

This project report presents the design and development of a comprehensive paint application built using **HTML**, **CSS**, and **JavaScript**. The primary objective of this project was to create a user-friendly and feature-rich web-based application that allows users to create and manipulate digital artwork with ease.

The paint application offers a range of essential drawing tools, including **brushes**, **lines**, **rectangles**, and **circles**, empowering users to express their creativity through various stroke sizes and colors. Additionally, the application supports multiple layers, enabling users to work on complex compositions while maintaining flexibility and control over individual elements.

The project's implementation leveraged the power of the **Canvas API**, a built-in HTML5 feature that facilitates dynamic rendering of graphics and animations. Through careful planning and efficient coding practices, the application delivers a seamless and responsive user experience, ensuring smooth drawing and real-time visual feedback.

Furthermore, the report delves into the project's planning and development phases, including requirements gathering, system design, and implementation details. It also outlines the testing methodologies employed to ensure the application's functionality, usability, and performance.

The paint application not only serves as a practical tool for digital artists and hobbyists but also demonstrates the capabilities of web technologies in creating interactive and visually appealing applications. The project showcases the integration of various programming concepts and techniques, making it a valuable educational resource for aspiring web developers and designers.

Objective of the Project and Modules

Objectives

Develop a versatile paint application

The primary objective of the project is to create a feature-rich paint application that caters to the needs of both casual users and professional artists. The application aims to provide a wide range of drawing tools, editing features, and customization options to enhance user creativity and productivity.

Intuitive user interface

One of the key objectives is to design an intuitive and user-friendly interface that allows users to navigate seamlessly through the application. Emphasis is placed on accessibility and ease of use, ensuring that users of all skill levels can comfortably utilize the software.

Modular design for scalability

The project aims to adopt a modular design approach, dividing the application into distinct modules or components. This facilitates scalability and maintainability, allowing for easier implementation of new features and enhancements in the future.

Modules

Drawing Canvas Module

This module forms the core of the paint application, providing users with a blank canvas to draw and create artwork. It includes functionalities such as drawing tools, color selection, brush customization, and layer management.

Toolbox Module

The toolbox module encompasses various drawing tools and editing features, including brushes, pencils, erasers, shapes, text, and image manipulation tools. Each tool is designed to serve a specific purpose and enhance user creativity.

User Interface Module

This module focuses on the design and implementation of the graphical user interface (GUI) of the application. It includes components such as menus, toolbars, dialogs, and panels, all designed to enhance user interaction and experience.

Settings and Preferences Module

This module enables users to customize their painting environment according to their preferences. It includes options for adjusting brush settings, canvas properties, keyboard shortcuts, and other application settings.

Project Category and Tools

Project Category

The paint application falls under the category of graphics software, specifically within the domain of digital art and design tools. It is designed to provide users with a platform for creating, editing, and manipulating digital artwork, ranging from illustrations and sketches to complex compositions.

Tools and Technologies Used

Programming Languages

JavaScript (ES6+): The entire functionality of the paint application is implemented using vanilla JavaScript (ES6+), without the use of any external frameworks or libraries. This approach emphasizes native browser capabilities and ensures lightweight, efficient code execution.

HTML5 and CSS3

HTML5 and CSS3 are utilized for structuring the application's user interface and styling elements, respectively. **HTML5** provides the foundation for creating interactive web content, while **CSS3** enhances the visual presentation and layout of the paint application.

Version Control

Git: Git is used for version control, enabling collaborative development, code management, and tracking of project changes. Platforms like GitHub or GitLab may be utilized for hosting the project repository.

Integrated Development Environment (IDE)

Visual Studio Code: Visual Studio Code (VS Code) is chosen as the code editor for its lightweight yet powerful features, extensive plugin ecosystem, and support for JavaScript and web development. It provides a comfortable environment for writing, debugging, and testing JavaScript code.

By leveraging these tools and technologies, the paint application is developed using vanilla JavaScript, emphasizing simplicity, performance, and compatibility across different web browsers. This approach allows for greater control over the application's behavior and ensures a seamless digital painting experience for users without relying on external frameworks or libraries.

Software Requirement Specification (SRS)

Introduction to SRS

The Software Requirement Specification (SRS) outlines the functional and non-functional requirements of the paint application. It serves as a blueprint for the development team, guiding the design, implementation, and testing phases of the project.

Functional Requirements

Drawing Tools

The application shall provide a variety of drawing tools, including brushes, pencils, erasers, shapes (e.g., lines, rectangles, circles), and text tools.
Users shall be able to select different brush sizes, shapes, and colors for drawing and editing artwork.

Canvas Management

The application shall offer canvas management features, allowing users to create new canvases, save artwork, and export images in various formats (e.g., PNG, JPEG, SVG).

Layer Support

The application shall support multiple layers, enabling users to organize artwork into separate layers for better organization and editing flexibility.
Users shall be able to create, delete, rearrange, and merge layers, as well as adjust layer opacity and blending modes.

User Interface

The user interface shall be intuitive and user-friendly, featuring menus, toolbars, dialogs, and panels for easy navigation and access to drawing tools and features.
Tooltips, hover effects, and contextual menus shall be utilized to provide guidance and enhance the user experience.

Non-Functional Requirements

Performance

The application shall exhibit responsive performance, with smooth rendering and minimal latency during drawing operations.
Load times for opening and saving files shall be optimized to ensure efficient user interaction.

Compatibility

The application shall be compatible with modern web browsers, including Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. The application shall adhere to web accessibility standards (WCAG) to ensure accessibility for users with disabilities.

Constraints

Software Requirement

Browser Limitations

The application's functionality may be limited by the capabilities and constraints of web browsers, particularly in terms of performance, memory usage, and compatibility with certain features (e.g., file system access).

Hardware Requirements

Users may require hardware components such as a mouse, touchpad, or stylus for optimal interaction with the application, especially during drawing and editing tasks.

Process Model

For this project, the process model suitable is the Iterative Waterfall Model. The iterative waterfall model was chosen for the paint application's development due to its structured yet adaptable framework. By breaking the process into sequential phases, it provided clear direction while allowing for iterative refinement. This flexibility accommodated evolving requirements and design iterations, ensuring the project remained focused and responsive to change. Ultimately, the model's balance between structure and adaptability contributed to the successful development of the paint application.

Project Scheduling

Project Scheduling Overview

Project scheduling involves the planning and organization of tasks, resources, and timelines to ensure the successful completion of the paint application development. This section presents the scheduling methodology, including the Program Evaluation and Review Technique (PERT) analysis and the GANTT chart representation.

GANTT Chart

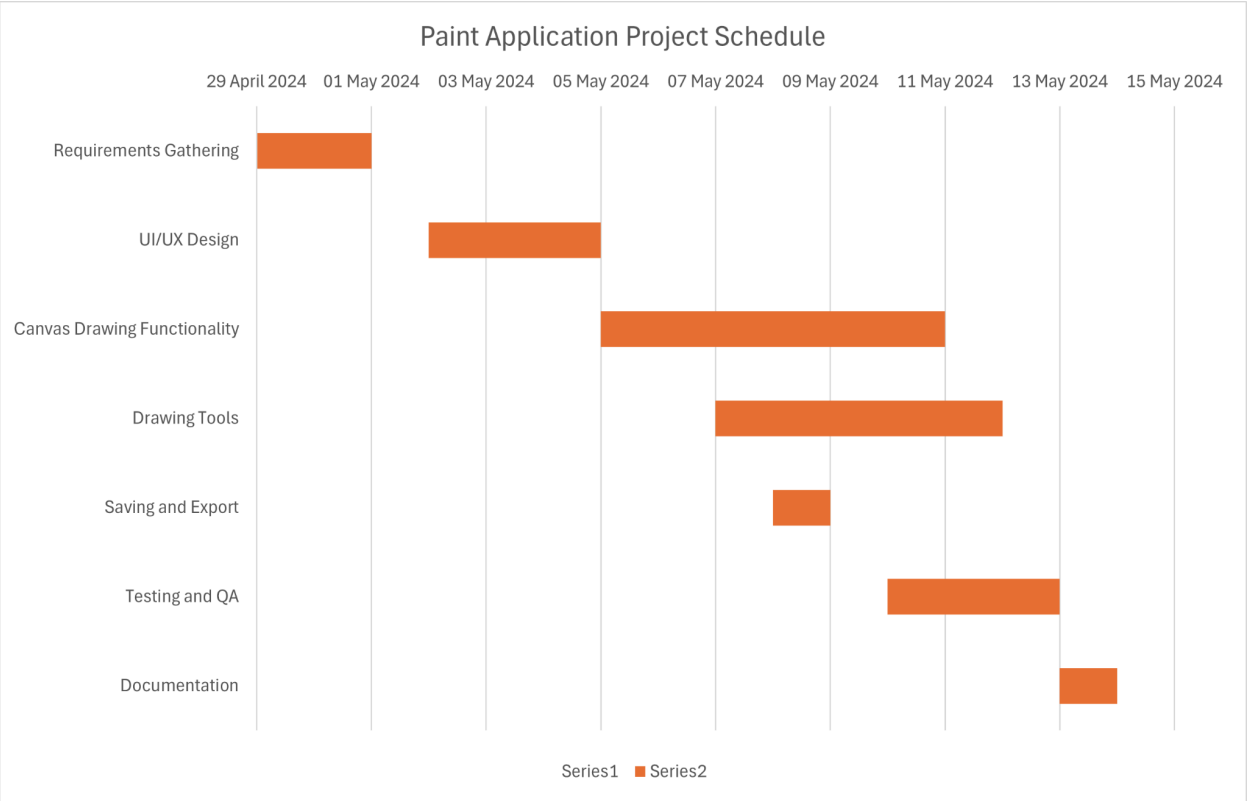
A GANTT chart is a visual representation of project tasks, timelines, and dependencies, displayed as horizontal bars on a time scale. The GANTT chart for the paint application development outlines the following key elements.

1. **Task List**: A list of project tasks, including their names, durations, start dates, and end dates.

2. **Timeline**: A horizontal time scale representing the project duration, typically divided into days, weeks, or months.

3. **Task** Bars: Horizontal bars representing the duration of each task, positioned along the timeline according to their start and end dates.

The Chart



Conclusion

Project scheduling using the GANTT chart provides a systematic approach to planning, organizing, and tracking the paint application development. By identifying critical tasks, dependencies, and timelines, project managers can allocate resources effectively, mitigate risks, and ensure timely project delivery.

Entity-Relationship Diagram

Introduction

The Entity-Relationship (ER) diagram illustrates the entities, attributes, and relationships within the paint application's database schema. It provides a visual representation of the data model, helping identify the entities and their attributes, and the relationships between them.
The entity-relationship data model is based on a perception of a real-world that consists of a collection of basic objects called entities and of relationships among these objects. An entity is an “object” in the real world that is distinguishable from other objects. E.g. each customer is an entity and rooms can be considered to be entities. Entities are described by a set of attributes. E.g. the attributes Room no. and Room type describe a particular Room in a hotel. The set of all entities of the same type and the set of all relationships of the same type are termed as an entity set and relationship set respectively

ER Diagram Components

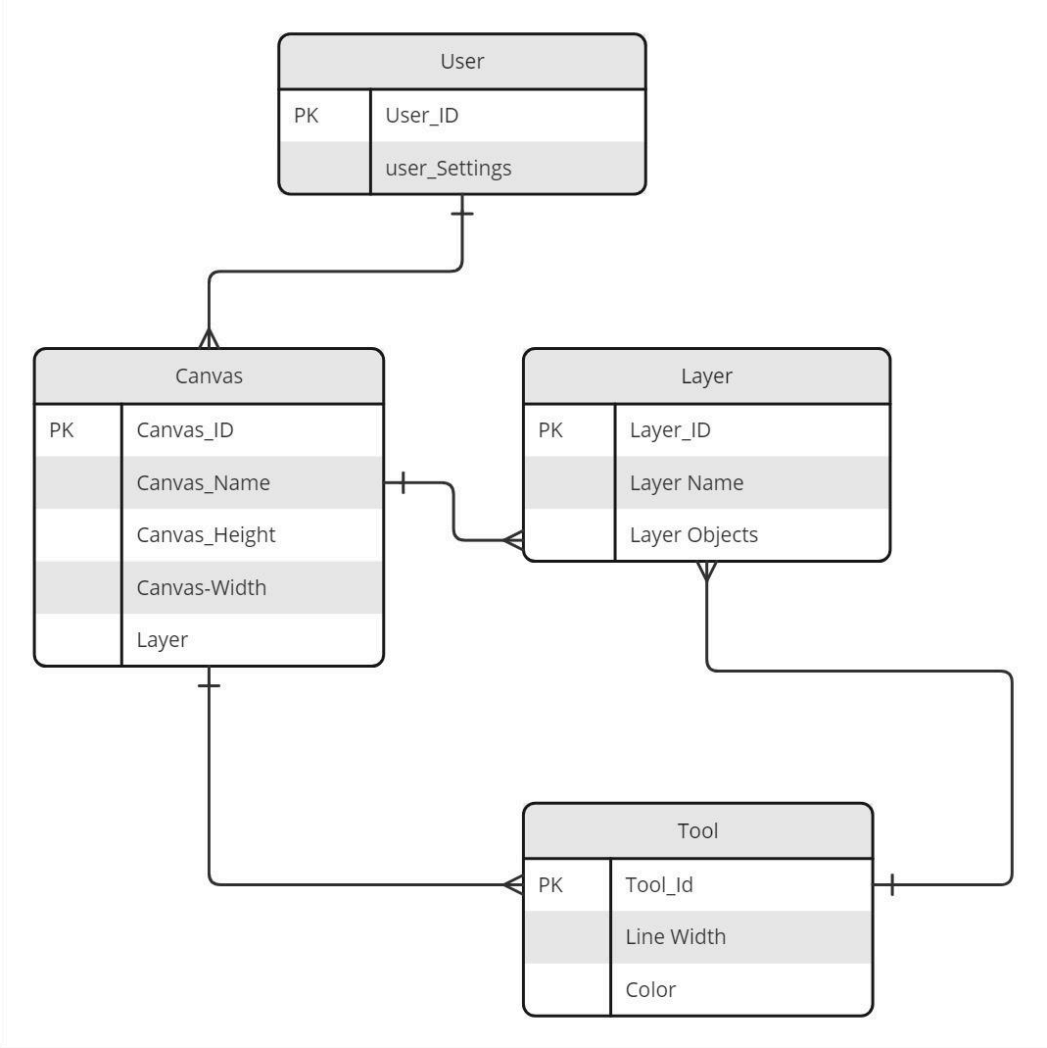
1. **Entity** (User, Canvas, Layer, Drawing Tool, Color Palette)
    a. Entities represent the real-world objects or concepts that the database stores information about. They are typically nouns and serve as the main building blocks of the ER diagram. Examples of entities in a paint application may include User, Canvas, Layer, Drawing Tool, and Color Palette.

2. **Attributes**
    a. Attributes are the properties or characteristics of entities. They describe the details or qualities of an entity. Each entity has one or more attributes that define it. Attributes are typically represented as ovals connected to their respective entity by a line. Examples of attributes for the User entity might include UserID, Username, Email, and Password.

3. **Relationships**
    a. Relationships represent the connections or associations between entities. They define how entities are related to each other and how they interact. Relationships are typically represented as lines connecting two entities, with labels indicating the nature of the relationship (e.g., one-to-one, one-to-many, many-to-many). For example, a User may have multiple Canvases, indicating a one-to-many relationship between User and Canvas entities

4. **Keys:**
    a. Keys are attributes or combinations of attributes that uniquely identify each instance of an entity. They ensure data integrity and help establish relationships between entities. In an ER diagram, keys are often designated as primary keys (PK) or foreign keys (FK). The primary key uniquely identifies each record in an entity, while foreign keys establish relationships between entities by referencing the primary key of another entity.

ER Diagram For Paint Application



The above Entity Relationship (ER) diagram represents the data model for a paint application. Here are some key observations and conclusions about this ER diagram:

1. **User Entity:** The User entity represents the application's users. It has attributes like User_ID and User_Settings, which can store user-specific preferences or settings.

2. **Canvas Entity:** The Canvas entity is central to the application, as it represents the drawing area or canvas. It has attributes like Canvas ID, Canvas Name, Canvas Height, and Canvas Width to define the canvas dimensions. Notably, it has a one-to-many relationship with the Layer entity, indicating that a canvas can have multiple layers.
3. **Layer Entity:** The Layer entity represents individual layers within the canvas. Each layer has a unique Layer ID, Layer Name, and Layer Objects attribute. The Layer Objects attribute likely stores the drawing objects (e.g., shapes, lines, brush strokes) present on that particular layer.
4. **Tool Entity:** The Tool entity represents the various drawing tools available in the application, such as brushes, pencils, shapes, etc. It has attributes like Tool_Id, Line Width, and Color, which define the properties of the selected tool.
5. **Relationships:**
    a. The User entity has a one-to-many relationship with the Canvas entity, indicating that a user can have multiple canvases.
    b. The Canvas entity has a one-to-many relationship with the Layer entity, meaning a canvas can have multiple layers.
    c. The Layer entity has a one-to-many relationship with the Tool entity, suggesting that a layer can have multiple drawing objects created with different tools.
6. **Normalization:** The ER diagram is normalized, with entities having their own unique identifiers (primary keys) and relationships established between them to avoid data redundancy.

The ER diagram provides a logical and structured representation of the data model for a paint application. It captures the essential entities, their attributes, and the relationships between them.