A
PROJECT REPORT
ON
"STRINGS – A Digital Art Gallery"

Submitted to
Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur

In the Partial Fulfillment of
B. Com. Computer Application Final Year

Submitted By
Mr. Doshant Mahesh Giradkar

Under the Guidance of
Prof. Shripad Dixit

G. S. College of Commerce, Wardha
2023-24

G. S. College of Commerce, Wardha

Certificate

This is certify that **Mr. Doshant M. Giradkar** has completed their project on the topic **"String: A Digital Art Gallery"** prescribed by the **Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur** for **B.Com. Computer Application Final Year** course in **G. S. College of Commerce, Wardha**.

Mr. Doshant M. Giradkar
Prof. Shripad Dixit

Dr. Revati Bangre
Co-ordinator

Dr. Arundhati Ninawe
Principal

External Examiner

Internal Examiner

Declaration

I, **Mr. Doshant M. Giradkar** hereby honestly declare that the work entitled **“String: A Digital Art Galary”** submitted by me at **G. S. College of Commerce, Wardha** in partial fulfillment of the requirement for the award of B. Com. Computer Application Degree by **Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur** has not been submitted elsewhere for the award of any degree, during the academic session **2023-24.**

Mr. Doshant Mahesh Giradkar

Date:

Place:

ACKNOWLEDGEMENT

I, take this opportunity to express our gratitude and wholehearted thanks to project guide **Prof. Shripad Dixit**, Co-ordinator **Dr. Revati Bangre** for her guidance throughout this work. I am very much thankful to her for their constant encouragement, support, and kindness. I am also grateful to our teachers **Prof. Shailesh Janbandhu, Prof. Harsha Gangavane** and **Prof. Amol Raut** for their encouragement, help and support from time to time.

I wish to express our sincere thanks to Principal, **Dr. Arundhati Ninawe** for providing us wide range of opportunities, facilities, and inspiration to gather professional knowledge and material without which this project could not have been completed.

Mr. Doshant M. Giradkar

Date:

Place:

INDEX

Introduction

In today's digital age, the world of art has transcended physical boundaries, and the internet has become a vast canvas for artists to showcase their talents. "Strings: A Digital Art Gallery" is an innovative web application that aims to provide a platform for artists to connect, collaborate, and share their artwork with a global audience.

The project's primary objective is to create an engaging and immersive digital space where artists can curate their own virtual galleries, upload their masterpieces, and present their creative process. By fostering a community of like-minded individuals, Strings encourages artistic exploration, constructive feedback, and the celebration of diverse artistic styles.

Objectives

The key objectives of the "Strings: A Digital Art Gallery" project are as follows:

1. **Artist Profiles and Artwork Sharing**:

Provide a user-friendly platform where artists can create personalized profiles and seamlessly upload their artwork, showcasing their unique styles and techniques.

2. **Artist Connectivity**:

Foster a vibrant community where artists can connect, follow each other's work, and engage in meaningful discussions about art, techniques, and inspiration.

3. **Art Discovery and Appreciation**:

Implement features that enable users to discover and appreciate artwork from artists around the world, fostering a sense of global artistic exploration.

4. **Community Engagement**:

Build an engaging platform that encourages interactions, likes & dislikes, and support among artists, nurturing a thriving creative ecosystem.

Preliminary System Analysis

Identification of Need:

In the contemporary art world, traditional physical galleries and exhibitions may pose limitations in terms of accessibility, reach, and cost. "Strings: A Digital Art Gallery" addresses this need by providing a virtual platform that transcends geographical boundaries, allowing artists to showcase their work to a global audience without the constraints of physical space.

Preliminary Investigation:

Extensive research was conducted to understand the existing digital art platforms, their strengths, and limitations. User feedback and industry trends were analyzed to identify the key features and functionalities required to create an engaging and user-friendly digital art gallery.

Feasibility Study:

A comprehensive feasibility study was undertaken to evaluate the project's viability, considering the following aspects:

- **Technical Feasibility**:

  The project leverages widely adopted web technologies, including HTML, CSS, JavaScript, Node.js, and PostgreSQL, ensuring technical feasibility and scalability.

- **Economic Feasibility**:

  The project's cost-effective development and hosting solutions make it economically viable, with potential revenue streams through premium features or advertisements.

- **Operational Feasibility**:

  The user-friendly interface and intuitive design of the platform ensure operational feasibility and ease of use for both artists and viewers.

Project Category

"Strings: A Digital Art Gallery" is a Full Stack Web Application, encompassing both front-end and back-end components, as well as a database for efficient data management.

Software and Hardware Requirements

Software Requirements

Front-end Development: HTML, CSS, JavaScript

Back-end Development: Node.js

Database Management: PostgreSQL

Integrated Development Environment (IDE): Visual Studio Code

Web Browsers: Latest versions of web browsers (e.g., Google Chrome, Mozilla Firefox)

***Node.js Package Manager (NPM)*:** For managing external libraries and dependencies

Hardware Requirements

Processor: Dual-core processor or higher

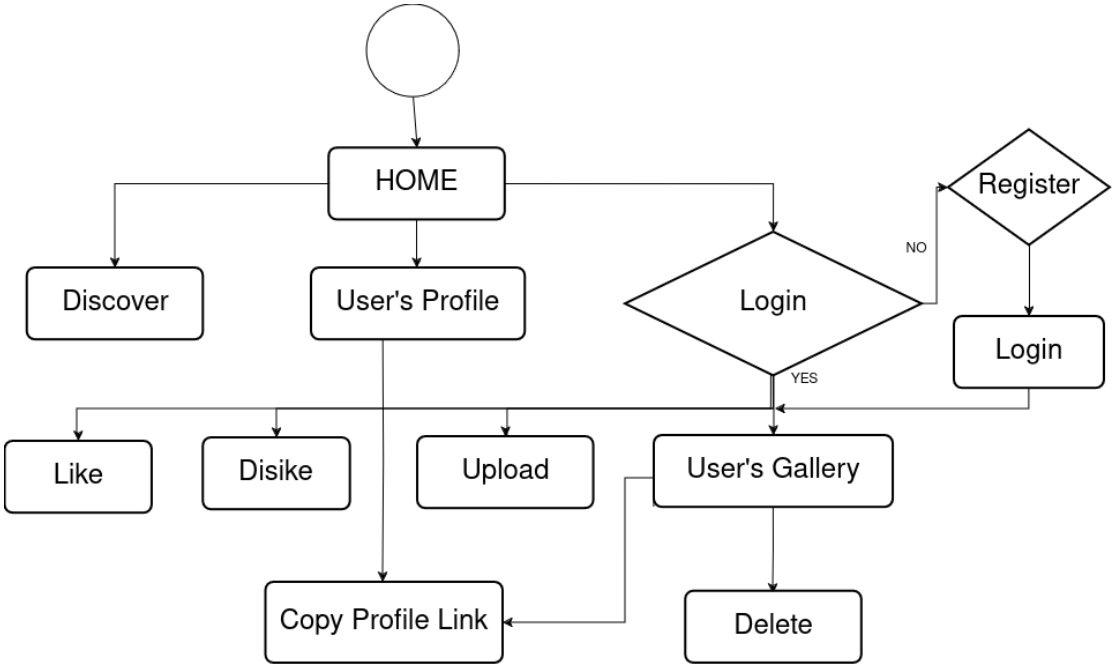
RAM: 2 GB or more

Hard Disk Space: 8 GB or more of free space

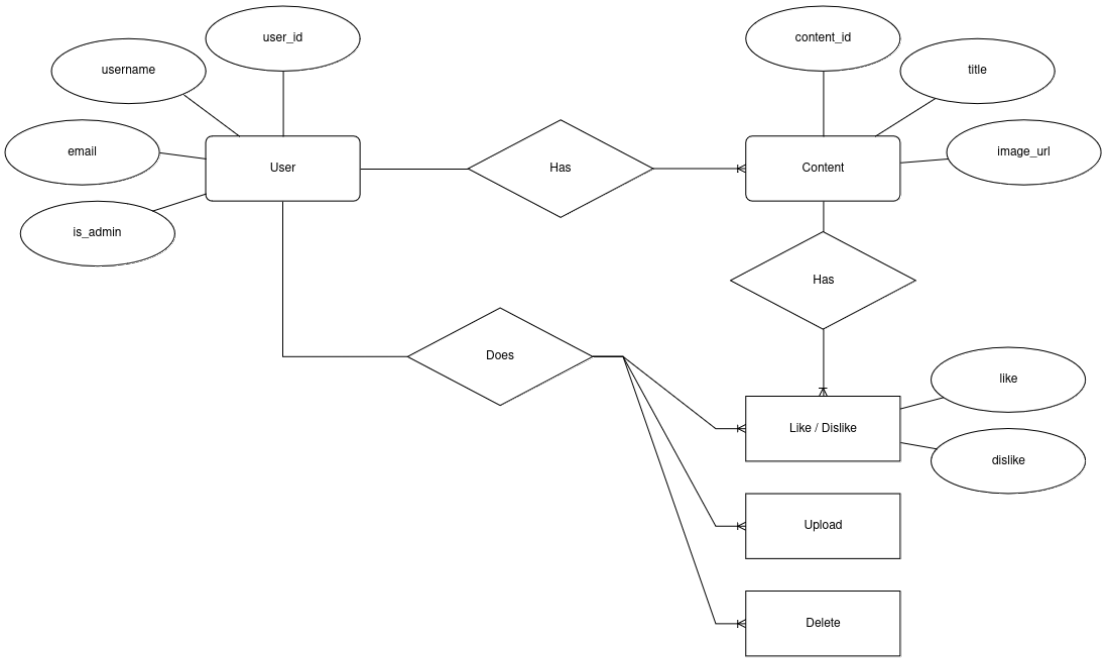
Internet Connectivity: Stable internet connection for development and deployment

Detailed System Analysis

Data-Flow Diagram:



Entity-Relationship Diagram:



Data Structures and Tables

The project utilizes a relational database management system (PostgreSQL) to store and manage data efficiently. The following tables are used:

1. **Users Table**
    - Stores user information, including username, email, password hash, and admin status.

Feild	Type	Description
user_id	INTEGER	NOT NULL AUTO INCREMENT PRIMARY KEY
username	VARCHAR(50)	NOT NULL UNIQUE *(50 characters)*
email	VARCHAR(255)	NOT NULL UNIQUE *(255 characters)*
is_adm	BOOLEAN	NOT NULL
pass_hash	VARCHAR(255)	NOT NULL *(255 characters)*

2. **Content Table**
    - Stores information about the uploaded artwork, including the content ID, user ID, title, and image path.

Feild	Type	Description
content_id	BIGINT	NOT NULL AUTO INCREMENT PRIMARY KEY UNIQUE
user_id	INTEGER	NOT NULL FOREIGN KEY
title	VARCHAR(255)	NOT NULL *(Size 255 characters)*
img	VARCHAR(255)	NOT NULL UNIQUE *(Size 255 characters)*

3. **Actions Table**
    - Stores user interactions with the artwork, such as likes and dislikes, along with the associated content ID and user ID.

Feild	Type	Description
content_id	BIGINT	NOT NULL FOREIGN KEY
user_id	INTEGER	NOT NULL FOREIGN KEY
_like	BOOLEAN	
_dislike	BOOLEAN	

System Design and Source Code

Controllers

1. server.js:

```
require("dotenv").config() //module to read enviornment variables
const cors = require("cors") //dependency for ExpressJS
const express = require("express") //ExpressJS - node framework to build web backend
const ejsLayouts = require("express-ejs-layouts") //template engine EJS
const bodyParser = require("body-parser") //middle ware to parse body
const session = require("express-session") //middleware to create sessions
const flash = require("express-flash") //middleware to store data
const passport = require("passport") //middleware to setup cookie

// Creating express app
const app = express()

// Initializing cookie
const initCookie = require("./cookieConfig")
initCookie(passport)

// Importing Route Controllers
const indexRouter = require("./controller/index")
const userRouter = require("./controller/user")
const adminRouter = require("./controller/admin")
const actionRouter = require("./controller/actions")

// Middleware
app.use(session({
    secret: "secretproject", //keyword to encrypt session
    resave: false,
    saveUninitialized: false
})) // Setting up sessons options
app.use(flash())
app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: true }))
app.use(cors())
app.use(passport.initialize())
app.use(passport.session())
app.set('view engine', 'ejs')
app.set("views", __dirname + "/views")
app.set("layout", 'layouts/layout')
app.use(ejsLayouts)
```

```
// Setting Up Routers
app.use(express.static('public'))
app.use('/', indexRouter)
app.use("/user", userRouter)
app.use("/admin", adminRouter)
app.use('/actn', actionRouter)

// Listening on port defined in enviornment variable // Default PORT:4000
app.listen(process.env.PORT || 4000, () => {
   console.log("Server Started At Port " + process.env.PORT || 4000)
})
```

2. index.js

```
const express = require("express");
const router = express.Router();
const db = require("../model/db");
const multr = require("multer");

// Content Pages
router.get("/", (req, res) => {
  db.query(
          `SELECT content.content_id, content.title, content.img, content.user_id,
users.username,
        COALESCE(SUM(CASE WHEN actions._like THEN 1 ELSE 0 END), 0) AS
total_likes,
         COALESCE(SUM(CASE WHEN actions._dislike THEN 1 ELSE 0 END), 0)
AS total_dislikes
       FROM content
       JOIN users ON content.user_id = users.user_id
       LEFT JOIN actions ON content.content_id = actions.content_id
            GROUP BY content.content_id, content.title, content.img, content.user_id,
users.username
       ORDER BY content.content_id DESC;`,
    (err, results) => {
      if (err) throw err;

      let homePageData = results.rows;
      if (typeof req.user != "undefined") {
        res.render("index", {
          user: req.user.username,
          userid: req.user.user_id,
          resp: results.rows,
          activeDir: req.path,
          homePageData,
        });
      } else
        res.render("index", {
          activeDir: req.path,
          homePageData,
        });
    }
  );
});

router.get("/discover", (req, res) => {
  db.query(
```

```
        `SELECT content.content_id, content.title, content.img, content.user_id, users.username,
        COALESCE(SUM(CASE WHEN actions._like THEN 1 ELSE 0 END), 0) AS total_likes,
        COALESCE(SUM(CASE WHEN actions._dislike THEN 1 ELSE 0 END), 0) AS total_dislikes
       FROM content
       JOIN users ON content.user_id = users.user_id
       LEFT JOIN actions ON content.content_id = actions.content_id
           GROUP BY content.content_id, content.title, content.img, content.user_id, users.username
       ORDER BY total_likes DESC, total_dislikes ASC;`,
    (err, results) => {
     if (err) throw err;

     let discoverPageData = results.rows;
     if (typeof req.user != "undefined")
      res.render("discover", {
       user: req.user.username,
       userid: req.user.user_id,
       activeDir: req.path,
       resp: results.rows,
       discoverPageData,
      });
     else
      res.render("discover", {
       activeDir: req.path,
       discoverPageData,
      });
    }
  );
});

router.get("/my-gallery", (req, res) => {
  if (typeof req.user != "undefined")
    db.query(
          `SELECT content.content_id, content.title, content.img, content.user_id, users.username,
           COALESCE(SUM(CASE WHEN actions._like THEN 1 ELSE 0 END), 0) AS total_likes,
          COALESCE(SUM(CASE WHEN actions._dislike THEN 1 ELSE 0 END), 0) AS total_dislikes
          FROM content
          JOIN users ON content.user_id = users.user_id
```

```
          LEFT JOIN actions ON content.content_id = actions.content_id
          WHERE users.username = '${req.user.username}'
             GROUP BY content.content_id, content.title, content.img, content.user_id,
users.username
          ORDER BY total_likes DESC, total_dislikes ASC;`,
      (err, results) => {
       if (err) throw err;
       else {
        let galleryPage = results.rows;
        res.render("gallery", {
         user: req.user.username,
         userid: req.user.user_id,
         activeDir: req.path,
         resp: results.rows,
         galleryPage,
        });
       }
      }
    );
  else res.redirect("/user/login");
});

router.get("/view/:username", (req, res) => {
  db.query(
          `SELECT content.content_id, content.title, content.img, content.user_id,
users.username,
            COALESCE(SUM(CASE WHEN actions._like THEN 1 ELSE 0 END), 0)
AS total_likes,
          COALESCE(SUM(CASE WHEN actions._dislike THEN 1 ELSE 0 END), 0)
AS total_dislikes
          FROM content
          JOIN users ON content.user_id = users.user_id
          LEFT JOIN actions ON content.content_id = actions.content_id
          WHERE users.username = '${req.params.username}'
             GROUP BY content.content_id, content.title, content.img, content.user_id,
users.username
          ORDER BY total_likes DESC, total_dislikes ASC;`,
    (err, results) => {
     if (err) {
      req.flash("err_msg", err)
      res.render("user", {
       user: req.params.username,
       userid: req.user.user_id,
       activeDir: req.path,
```

```
          resp: results.rows,
        });
      } else if (results.rowCount < 1) {
        req.flash("err_msg", "User or User's page doesn't exits")
        res.render("user", {
          user: req.params.username,
          userid: req.user.user_id,
          activeDir: req.path,
          resp: results.rows,
        });
      } else {
        let userPage = results.rows;
        res.render("user", {
          user: req.params.username,
          userid: req.user.user_id,
          resp: results.rows,
          activeDir: req.path,
          userPage,
        })}})})

router.post("/view", (req, res) => {
  console.log(req.body);
  res.redirect(`/view/${req.body.username}`)
})

const storeImg = multr.diskStorage({
  destination: (req, file, cb) => {
    cb(null, "./public/images");
  },
  filename: (req, file, cb) => {
    cb(null, Date.now() + "-" + file.originalname);
  },
  onFileUploadStart: (file) => {
    if (file.mimetype == 'image/jpg' || file.mimetype == 'image/jpeg' || file.mimetype ==
'image/png' || file.mimetype == 'image/webp') {
      return true;
    } else {
      return false;
    }
  }
});

const upload = multr({ storage: storeImg });
router.get("/upload", (req, res) => {
```

```
  if (typeof req.user != "undefined")
    res.render("upload/upload", {
      user: req.user.username,
      userid: req.user.user_id,
      resp: results.rows,
      activeDir: req.path,
    });
  else {
    res.redirect("/user/login");
  }
});

router.post("/upload", upload.single("file"), (req, res) => {
  ext = (req.file.filename).split('.').pop()
  if (typeof req.user != "undefined") {
    if (ext == 'jpg' || ext == 'jpeg' || ext == 'png' || ext == 'webp' || ext == 'avif') {
      let { title } = req.body;
      db.query(
        `INSERT INTO content (user_id, title, img)
        VALUES ($1, $2, '${req.file.filename}')`,
        [req.user.user_id, title],
        (err, results) => {
          if (err) throw err
          else {
            req.flash("success_msg", "Image Uploaded!!")
            res.redirect("/my-gallery")
          }
        }
      )
    } else {
      req.flash("err_msg", "Invalid File Type, please check file extension!!")
      res.redirect("/my-gallery")
    }
  } else {
    res.redirect("/user/login");
  }
});
module.exports = router;
```

3. user.js :

```
const { SHA256 } = require("crypto-js");
const express = require("express");
const router = express.Router();
const db = require("../model/db");
const passport = require("passport");

router.get("/login", (req, res) => {
  res.render("userForm/login", {
    activeDir: req.path,
  });
});

router.post(
  "/login",
  passport.authenticate("local", {
    successRedirect: "/",
    failureRedirect: "/user/login",
    failureFlash: true,
  })
);

router.post("/logout", (req, res, next) => {
  req.logOut((err) => {
    if (err) return next(err);
    req.flash("success_msg", "You have Logged Out!!");
    res.redirect("/user/login");
  });
});

router.get("/register", (req, res) => {
  res.render("userForm/register", {
    activeDir: req.path,
  });
});

router.post("/register", (req, res) => {
  let errors = [];
  let { username, pswd, confirm_pswd, email } = req.body;
  if (pswd !== confirm_pswd) errors.push({ message: "Password Didn't Match" });
  if (pswd.length < 8)
    errors.push({ message: "Password Should Atleast Be 8 Characters Long" });
  if (errors.length > 0) {
    res.render("userForm/register", { errors });
```

```
  } else {
    let pswd_hash = SHA256(pswd).toString();
    db.query(
        `SELECT * FROM users WHERE username='${username}' OR email_id='$
{email}'`,
      (err, results) => {
        if (results.rows.length > 0) {
          errors.push({ message: "USERNAME or EMAIL ID Already Exists!" });
          res.render("userForm/register", { errors });
        } else {
          db.query(
            `INSERT INTO users (username, email_id, pass_hash, is_adm)
                    VALUES ($1, $2, $3, $4)`,
            [username, email, pswd_hash, false],
            (err, results) => {
              if (err) throw err;
              else {
                req.flash(
                  "success_msg",
                  "You are now registered, Please Login"
                );
                res.redirect("/user/login");
              }
            }
          );
        }
      }
    );
  }
});

module.exports = router;
```

4. actions.js

```
const router = require("express").Router();
const db = require("../model/db");
require("dotenv").config();

router.post("/like/:id", async (req, res) => {
  if (typeof req.user != 'undefined') {
    db.query(
      `
      INSERT INTO actions values(${req.user.user_id}, ${req.params.id}, true, false);
    `,
      (err) => {
        if (err) {
          db.query(
            `
          UPDATE actions SET
          _like = ${true},
          _dislike = ${false}
          WHERE content_id = ${req.params.id} AND user_id = ${
              req.user.user_id
            };
        `,
            (err) => {
              if (err) {
                res
                  .status(422)
                  .send(`Something went wrong!!
                        <br><hr><br>
                        go to home page? <a href='/'>[Y]<a>
                        `);
              } else {
                res.status(200);
              }
            }
          );
        }
      }
    );
  } else {
    res
      .status(422)
      .send(
        "please login first<br><hr><br>go to home page? <a href='/'>[Y]<a>"
      );
```

```
  }
});

router.post("/dislike/:id", (req, res) => {
  if (typeof req.user != 'undefined') {
    db.query(
      `
      INSERT INTO actions values(${req.user.user_id}, ${
        req.params.id
      }, ${false}, ${true});
    `,
      (err) => {
        if (err) {
          db.query(
            `
          UPDATE actions SET
          _like = ${false},
          _dislike = ${true}
          WHERE content_id = ${req.params.id} AND user_id = ${
              req.user.user_id
            };
        `,
            (err) => {
              if (err) {
                res
                  .status(422)
                  .send(`Something went wrong!!`);
              } else {
                res.status(200);
              }
            }
          );
        }
      }
    );
  } else {
    res
      .status(422)
      .send(
        "<script> alert('Please login first')</script>"
      );
  }
});
```

```
router.post("/delete/:id", (req, res) => {
  if (typeof req.user != 'undefined') {
    db.query(
      `
    DELETE FROM content
    WHERE user_id = ${req.user.user_id} AND content_id = ${req.params.id};
    `,
      (err) => {
        if (err) {
          res
            .status(422)
            .send(`Something went wrong!!`);
        } else {
          res.status(200);
        }
      }
    );
  } else {
    res
      .status(422)
      .send(
        "please login first<br><hr><br>go to home page? <a href='/'>[Y]<a>"
      );
  }
});

module.exports = router;
```

5. cookieConfig.js

```
const LocalStrategy = require("passport-local").Strategy;
const db = require("./model/db");
const { SHA256 } = require("crypto-js");

function init(passport) {
  const authenticateUser = (username, password, done) => {
    db.query(
      `SELECT * FROM users WHERE username = '${username}'`,
      (err, results) => {
        if (err) throw err;

        if (results.rows.length > 0) {
          const user = results.rows[0];
          if (SHA256(password).toString() === user.pass_hash) {
            return done(null, user);
          } else {
            return done(null, false, { message: "Wrong Username or Password" });
          }
        } else {
          return done(null, false, { message: "Incorrect Username" });
        }});
  };
  passport.use(
    new LocalStrategy(
      {
        usernameField: "username",
        passwordField: "pswd",
      },
      authenticateUser
    ));
  passport.serializeUser((user, done) => {
    done(null, user.user_id);
  });
  passport.deserializeUser((user_id, done) => {
    db.query(
      `SELECT * FROM users WHERE user_id = ${user_id}`,
      (err, results) => {
        if (err) throw err;

        return done(null, results.rows[0]);
      });
  });
} module.exports = init;
```

Static:

1. index.css

```
@import url('https://fonts.googleapis.com/css2?family=Dancing+Script:wght@400..700&display=swap');

img {
   pointer-events: none;
}

.post {
        margin: 0.5rem;
        width: 32vw;
        display: flex;
        height: fit-content;
        flex-direction: column;
}

.post-image {
        display: block;
        height: 34vh;
        min-width: 40%;
}

.post .btn {
        font-size: 1.5rem;
}

.card-body {
        display: flex;
        flex: 1;
        flex-direction: column;
}

.text-body-secondary {
        flex: 1;
        overflow: hidden;
}

.post-ratings-container {
        display: flex;
        justify-content: end;
        padding: 12px 0;
```

```
}

.post-rating {
	display: flex;
	align-items: center;
	cursor: default;
}

.post-rating:not(:last-child) {
	margin-right: 12px;
}

.post-rating-selected>.post-rating-button,
.post-rating-selected>.post-rating-count {
	color: #009578;
}

.post-rating-button {
	margin-right: 6px;
	cursor: pointer;
	color: #555555;
}

.post-rating:not(.post-rating-selected)>.post-rating-button:hover {
	color: #000000;
}

.btn {
	color: white;
}

.navbar-brand {
	font-size: 2.25rem;
	font-family: "Dancing Script", cursive;
	font-weight: 600;
	user-select: none;
}

.about-site {
	margin: 1rem;
	vertical-align: middle;
}

.about-site h2 {
```

```
        display: inline;
        color: rgb(255, 255, 255);
}

.about-site h3 {
        display: inline;
        color: rgb(202, 202, 202);
}

.about-site p {
        font-size: 1.25rem;
}

.about-site a {
        text-decoration: none;
        color: gray;
}

.about-site .card {
        width: 50vw;
}

.card-title {
        text-overflow: ellipsis;
        white-space: nowrap;
        overflow: hidden;
}

.view-page-heading {
        margin: .5rem;
        padding: .75rem;
        width: fit-content;
        background-color: gray;
        color: rgb(23, 26, 39);
        border-radius: .5rem;
        user-select: none;
}

.link-floating-button {
        font-size: 2rem;
        position: fixed;
        bottom: 2rem;
        right: 2rem;
}
```

Views:

1. layout.ejs

```
<!doctype html>
<html lang="en" data-bs-theme="dark">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>STRINGS - Digital Art SNS</title>
  <link href="/bootstrap.min.css" rel="stylesheet">
  <link rel="stylesheet" href="/index.css">
  <script src="https://unpkg.com/htmx.org@1.9.11"></script>
  <script>
    function like(content_id) {
      post = document.querySelector("#".concat(content_id))
      rating = post.querySelector(".post-rating-button-like")
      dislike_rating = post.querySelector(".post-rating-button-dislike")
      const count = post.querySelector(".post-rating-count")

      if (rating.classList.contains("post-rating-selected")) {
        rating.classList.remove('btn-primary')
        rating.classList.add('btn-secondary')
        rating.classList.remove("post-rating-selected")
        // dislike_rating.classList.add('btn-primary')
        // dislike_rating.classList.remove('btn-secondary')

      } else {
        rating.classList.add('btn-primary')
        rating.classList.remove('btn-secondary')
        rating.classList.add("post-rating-selected")
        dislike_rating.classList.add('btn-secondary')
        dislike_rating.classList.remove('btn-primary')

      }
        const response = fetch(`/actn/like/${content_id.replace(/^item/, '')}`, { method:
"post" })
    }

    function dislike(content_id) {
      post = document.querySelector("#".concat(content_id))
      rating = post.querySelector(".post-rating-button-dislike")
      like_rating = post.querySelector(".post-rating-button-like")
```

```
      const count = post.querySelector(".post-rating-count")

      if (rating.classList.contains("post-rating-selected")) {
        rating.classList.remove('btn-primary')
        rating.classList.add('btn-secondary')
        rating.classList.remove("post-rating-selected")
        // like_rating.classList.add('btn-primary')
        // like_rating.classList.remove('btn-secondary')

      } else {
        rating.classList.add('btn-primary')
        rating.classList.remove('btn-secondary')
        rating.classList.add("post-rating-selected")
        like_rating.classList.add('btn-secondary')
        like_rating.classList.remove('btn-primary')

      }
    }

    function deleteContent(content_id) {
      const post = document.querySelector("#" + String(content_id));
      fetch(`/actn/delete/${content_id.replace(/^item/, '')}`, { method: "post" });
    }

  </script>
</head>

<body>
  <div class="container-fluid" style="margin-top: 6rem;">
    <%- body %>
  </div>
  <%- include('../partials/navbar.ejs') %>
    <script src="/bootstrap.bundle.min.js"></script>
</body>

</html>
```

2. partials/navbar.ejs

```
<nav class="navbar navbar-expand-lg bg-body-tertiary" style="position: fixed; top: 0px; width: 100%;">
  <div class="container-fluid">
    <a class="navbar-brand">Strings</a>
     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent"
                aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <% if (typeof activeDir!='undefined' ) { %>
          <li class="nav-item">
             <a class="nav-link <%= activeDir =='/'? 'active': '' %>" aria-current="page" href="/">Home</a>
          </li>
          <li class="nav-item">
                    <a class="nav-link <%=activeDir=='/discover' ? 'active' : '' %>" href="/discover">Discover</a>
          </li>
          <li class="nav-item">
            <a class="nav-link <%=activeDir=='/my-gallery' ? 'active' : '' %>" href="/my-gallery">My Gallery</a>
          </li>
          <% } %>
      </ul>
      <form class="d-flex" action="/view" style="margin-right: 1rem;" method="post">
                        <input type="text" name="username" class="form-control" placeholder="username"
                       aria-label="enter username" aria-describedby="addon-wrapping" style="margin-right: .25rem;" required>
        <button class="btn btn-outline-success" type="submit" aria-label="View User">View </button>
      </form>
        <h1 style="margin-top: .5rem; color: rgb(66, 66, 66); font-weight: 500; user-select: none;"> | </h1>
      <a class="btn btn-outline-success" href="/upload" style="margin: .25rem; margin-left: 1rem;">Upload</a>
      <% if (typeof user=='undefined' ) { %><%- include('./_login_btn.ejs') %>
      <% } %> <% if (typeof user !='undefined' ) { %> <%- include('./_logout_btn.ejs') %> <% } %> </div>   </div> </nav>
```

3. partials/_login_btn.ejs

```
<a class="btn btn-outline-success" href="/user/login">Log In</a>
```

4. partials/_logout_btn.ejs

```
<form action="/user/logout" method="post">
  <button class="btn btn-outline-success" type="submit">Log Out</button>
</form>
```

5. upload.ejs

```
<form action="/upload" method="post" enctype="multipart/form-data">
  <h3 style="margin: .5rem;">
    Upload,
    <small class="text-body-secondary">Your Art</small>
  </h3>
  <div class="input-group mb-3">

    <span class="input-group-text" id="inputGroup-sizing-default">Title</span>
        <input type="text" name="title" class="form-control" aria-label="Sizing example input"
      aria-describedby="inputGroup-sizing-default" required>
  </div>
  <div class="mb-3">
              <input class="form-control" type="file" name="file" id="file" accept=".png,.jpg,.jpeg,.webp,.avif" required>
  </div>

  <div class="col-12">
    <button class="btn btn-primary" type="submit">Upload</button>
  </div>
</form>
```

6. login.ejs

```
<div class="user-form">
   <form action="/user/login" method="post">
      <div>
         <h3 style="margin: .5rem;">
            Log In,
            <small class="text-body-secondary">to existing account</small>
         </h3>
         <ul>
            <% if (messages.success_msg) { %>
               <li style="color: green;">
                  <%= messages.success_msg %>
               </li>
               <% } %>
                  <% if (messages.error) { %>
                     <li style="color: red;">
                        <%= messages.error %>
                     </li>
                     <% } %>


         </ul>
             <div class="input-group flex-nowrap" style="margin: 1rem .5rem; width: 100%;">
            <span class="input-group-text" id="addon-wrapping">👤</span>
                           <input type="text" name="username" class="form-control" placeholder="Username*"
                     aria-label="enter username" aria-describedby="addon-wrapping" required>
         </div>
             <div class="input-group flex-nowrap" style="margin: 1rem .5rem; width: 100%;">
            <span class="input-group-text" id="addon-wrapping">🔑</span>
                            <input type="Password" name="pswd" class="form-control" placeholder="Password*"
                     aria-label="enter password" aria-describedby="addon-wrapping" required>
         </div>
           <button type="submit" class="btn btn-primary" style="margin: .5rem;">Log In</button> <br />
                           <span style="margin: .5rem;">don't have an account? <a href="/user/register">Create one</a></span>
      </div>
   </form></div>
```

7. register.ejs

```
<div class="user-form">
   <form action="/user/register" method="post">
      <div>
         <h3 style="margin: .5rem;">
            Register,
            <small class="text-body-secondary">a new account</small>
         </h3>
         <% if (typeof errors!='undefined' ) { %>
            <ul>
               <% errors.forEach((error)=> { %>
                  <li style="color: red;">
                     <%= error.message %>
                  </li>
                  <% }) %>
            </ul>
            <% } %>
               <div class="input-group flex-nowrap" style="margin: 1rem .5rem; width:
100%;">
                  <span class="input-group-text" id="addon-wrapping">👤</span>
                        <input type="text" name="username" class="form-control"
placeholder="Username*"
                     aria-label="enter username" aria-describedby="addon-wrapping"
required>
               </div>
               <div class="input-group flex-nowrap" style="margin: 1rem .5rem; width:
100%;">
                  <span class="input-group-text" id="addon-wrapping">🔑</span>
                        <input type="Password" name="pswd" class="form-control"
placeholder="New Password*"
                     aria-label="enter password" aria-describedby="addon-wrapping"
required>
               </div>
               <div class="input-group flex-nowrap" style="margin: 1rem .5rem; width:
100%;">
                  <span class="input-group-text" id="addon-wrapping">🔑</span>
                   <input type="Password" name="confirm_pswd" class="form-control"
placeholder="Confirm Password*"
                      aria-label="confirm pasword" aria-describedby="addon-wrapping"
required>
               </div>
               <div class="input-group flex-nowrap" style="margin: 1rem .5rem; width:
100%;">
                  <span class="input-group-text" id="addon-wrapping">📧</span>
```

```
                    <input type="email" name="email" class="form-control" placeholder="Email ID*"
                     aria-label="enter email id" aria-describedby="addon-wrapping" required>
                </div>
<button type="submit" class="btn btn-primary" style="margin: .5rem;">Register</button> <br />
                    <span style="margin: .5rem;">Log in to existing account? <a href="/user/login">Click here</a></span>
      </div>
   </form>
</div>
```

8. index.ejs

```
<div class="about-site">
  <center>
    <h2>Welcome To Strings, </h2>
     <h3>A Gallery For Digital Art</h3> <a data-bs-toggle="collapse" href="#about" role="button"
      aria-expanded="false" aria-controls="collapseExample">...</a>
    <div class="collapse" id="about">
      <div class="card card-body">
        <p>
           In an era where every artist’s dream is to stand out, Strings emerges as the bridge between creators and
            their global audience. Whether you wield a stylus or a mouse, this digital gallery invites you to showcase
           your masterpieces, share your process, and find kindred spirits who resonate with your vision.
        </p>
        <p>
          Curate your digital gallery. Showcase your best work, from that mesmerizing digital landscape to the
         quirky pixel portrait. Each piece is a thread in your artistic journey on Strings.
        </p>
      </div>
    </div>
  </center>
</div>
<div style="display: flex; flex-wrap: wrap">
  <% homePageData.forEach(card=> { %>
    <div class="post card" id="item<%= card.content_id %>">
           <img src="images/<%= card.img %>" class="card-img-top post-image" alt="Loading..." />
```

```
      <div class="card-body">
        <h5 class="card-title" style="flex: 1">❝ <%= card.title %> </h5>
        <p class="text-body-secondary">
          ~ <%= card.username %>
        </p>
        <div class="d-inline-flex gap-1">
          <div class="post-ratings-container">
            <div class="post-rating" style="display: inline;">
              <button type="button" class="btn btn-secondary post-rating-button-like post-rating-button" onclick="like(`item<%= card.content_id %>`)">👍</button>
              <span class="post-rating-count">
                <%= card.total_likes %> : <%= card.total_dislikes %>
              </span>
            </div>
            <div class="post-rating" style="display: inline;">
                <button type="button" class="btn btn-secondary post-rating-button-dislike post-rating-button" onclick="dislike(`item<%= card.content_id %>`)">👎</button>
            </div>
          </div>
        </div>
      </div>
    </div>
    <% }); %>
</div>
```

9. discover.ejs

```
<div style="display: flex; flex-wrap: wrap">
  <% discoverPageData.forEach(card=> { %>
    <div class="card post" id="item<%= card.content_id %>">
            <img src="images/<%= card.img %>" class="card-img-top post-image" alt="Loading..." />
      <div class="card-body">
        <h5 class="card-title" style="flex: 1">❝ <%= card.title %>
        </h5>
        <p class="text-body-secondary">
          ~ <%= card.username %>
        </p>
        <div class="d-inline-flex gap-1">
          <div class="post-ratings-container">
            <div class="post-rating like" style="display: inline;">
                <button type="button" class="btn btn-secondary post-rating-button post-rating-button-like"
              onclick="like(`item<%= card.content_id %>`)">👍</button>
            <span class="post-rating-count">

              <%= card.total_likes %> : <%= card.total_dislikes %>

            </span>
          </div>
          <div class="post-rating dislike" style="display: inline;">
                <button type="button" class="btn btn-secondary post-rating-button post-rating-button-dislike"
              onclick="dislike(`item<%= card.content_id %>`)">👎</button>
          </div>
        </div>
      </div>
    </div>
    <% }); %>
</div>
```

10. gallery.ejs

```
<ul>
  <% if (messages.success_msg) { %>
    <li style="color: green;">
      <%= messages.success_msg %>
    </li>
    <% } %>
      <% if (messages.err_msg) { %>
        <li style="color: red;">
          <%= messages.err_msg %>
        </li>
        <% } %>
</ul>


<div style="display: flex; flex-wrap: wrap">
  <% galleryPage.forEach(card=> { %>
    <div class="post card" id="item<%= card.content_id %>">
            <img src="images/<%= card.img %>" class="card-img-top post-image"
alt="Loading..." />
      <div class="card-body">
        <h5 class="card-title" style="flex: 1">❝ <%= card.title %>
        </h5>
        <p class="text-body-secondary">
          ‣ <%= card.username %> ‣
        </p>
        <div class="d-inline-flex gap-1">
          <div class="post-ratings-container">
            <div class="post-rating" style="display: inline;">
                  <button type="button" class="btn btn-secondary post-ratting-button-like
post-rating-button"
              onclick="like(`item<%= card.content_id %>`)">👍</button>
            <span class="post-rating-count">

              <%= card.total_likes %> : <%= card.total_dislikes %>

            </span>
          </div>
          <div class="post-rating" style="display: inline;">
              <button type="button" class="btn btn-secondary post-ratting-button-dislike
post-rating-button"
              onclick="dislike(`item<%= card.content_id %>`)">👎</button>
          </div>
          <div style="display: inline;">
```

```
                        <button type="button" class="btn btn-secondary delete-btn" onclick="deleteContent(`item<%= card.content_id %>`)">🚮</button>
          </div>
        </div>
       </div>
     </div>
   </div>
   <% }); %>
</div>

<script>
  function cpy_to_clipboard() {
    link = window.location.host + `/view/<%= user %>`
    navigator.clipboard.writeText(link)
  }
</script>
<button class="btn btn-success link-floating-button" onclick="cpy_to_clipboard()">📋</button>
```

11. user.ejs

```
<h2 class="view-page-heading">
  <%= user %>'s Gallery %>
</h2>

<ul>
  <% if (messages.err_msg) { %>
    <li style="color: red;">
      <%= messages.err_msg %>
    </li>
    go to home page? <a href="/" style="text-decoration: none;">[Y]</a>
    <% } %>
</ul>

<div style="display: flex; flex-wrap: wrap">
  <% if (typeof userPage !='undefined' ) {%>
    <% userPage.forEach(card=> { %>
      <div class="post card" id="item<%= card.content_id %>">
            <img src="/images/<%= card.img %>" class="card-img-top post-image"
alt="Loading..." />
        <div class="card-body">
          <h5 class="card-title" style="flex: 1">❝ <%= card.title %> </h5>
          <p class="text-body-secondary">
            ~ <%= card.username %>
          </p>
          <div class="d-inline-flex gap-1">
            <div class="post-ratings-container">
              <div class="post-rating" style="display: inline;">
                  <button type="button" class="btn btn-secondary post-ratting-button-like
post-rating-button" onclick="like(`item<%= card.content_id %>`)">👍</button>
              <span class="post-rating-count">

              <%= card.total_likes %> : <%= card.total_dislikes %>

              </span>
            </div>
            <div class="post-rating" style="display: inline;">
              <button type="button" class="btn btn-secondary post-ratting-button-dislike
post-rating-button" onclick="dislike(`item<%= card.content_id %>`)">👎</button>
            </div>
          </div>
        </div>
      </div>
    </div>
```

```
    <% }) %>
      <% } %>
</div>

<script>
  function cpy_to_clipboard() {
    link = window.location.host + `/view/<%= user %>`
    navigator.clipboard.writeText(link)
  }
</script>
<button class="btn btn-success link-floating-button" onclick="cpy_to_clipboard()">📋</button>
```

Model:

1. db_connection.js

```
const { Pool } = require("pg")
require("dotenv").config()

// Setting Up Connection With Database
const db = new Pool({
    user: process.env.DB_USER,
    password: process.env.DB_PASS,
    port: process.env.DB_PORT,
    host: process.env.DB_HOST,
    database: "strings"
})

module.exports = db // Exporting Connection with database
```

2. database.sql

```
CREATE DATABASE strings;

CREATE TABLE users(
   user_id SERIAL PRIMARY KEY,
   username varchar(32) NOT NULL UNIQUE,
   email_id varchar(64) NOT NULL,
   pass_hash VARCHAR(255) NOT NULL,
   is_adm BOOLEAN NOT NULL,
   UNIQUE (email_id, username)
);

CREATE TABLE content (
   user_id INT NOT NULL,
   content_id BIGSERIAL PRIMARY KEY NOT NULL,
   title varchar(125) NOT NULL,
   img varchar(255) NOT NULL,
   FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);

CREATE TABLE actions (
   user_id INT NOT NULL,
   content_id INT NOT NULL,
   _like BOOLEAN,
   _dislike BOOLEAN,
        FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE
CASCADE,
    FOREIGN KEY (content_id) REFERENCES content(content_id) ON DELETE
CASCADE
);
```

Modules

The "Strings: A Digital Art Gallery" project is divided into the following modules:

1. ***Login and Registration Module***

- Provides forms for user registration and authentication
- Implements secure password hashing with sha256 hashing algorithm and user session management
- Validates if username and email already exists, before registration.
-

2. ***Upload Module***

- Allows authenticated users to upload their artwork
- Implements file validation and storage mechanisms

3. ***Like and Dislike Module***

- Enables users to express their appreciation or dislike for artwork
- Updates the Actions table with user interactions

4. ***Delete Module***

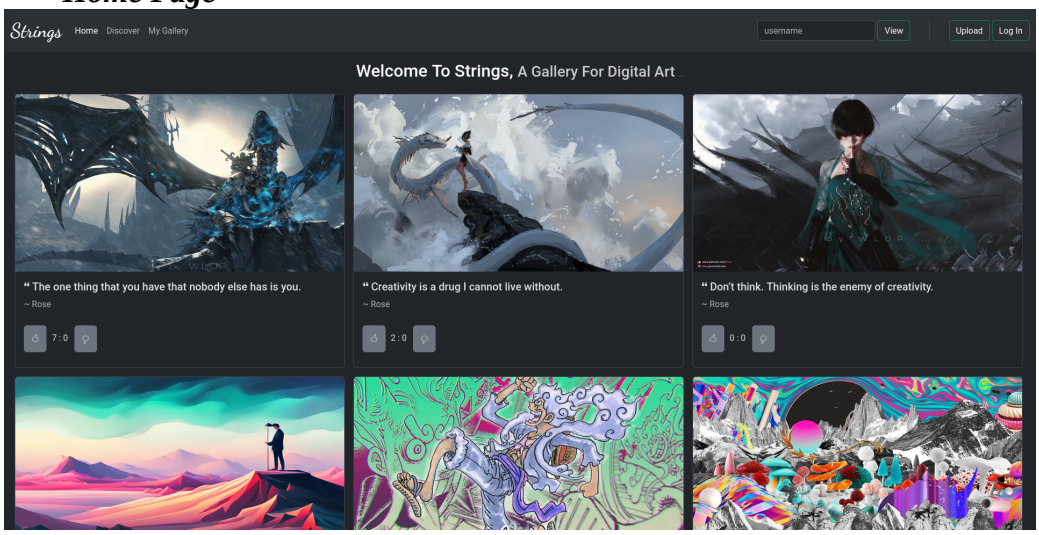
- Allows users to delete their own uploaded artwork
- Provides administrative privileges to delete inappropriate content

5. ***Content Module***

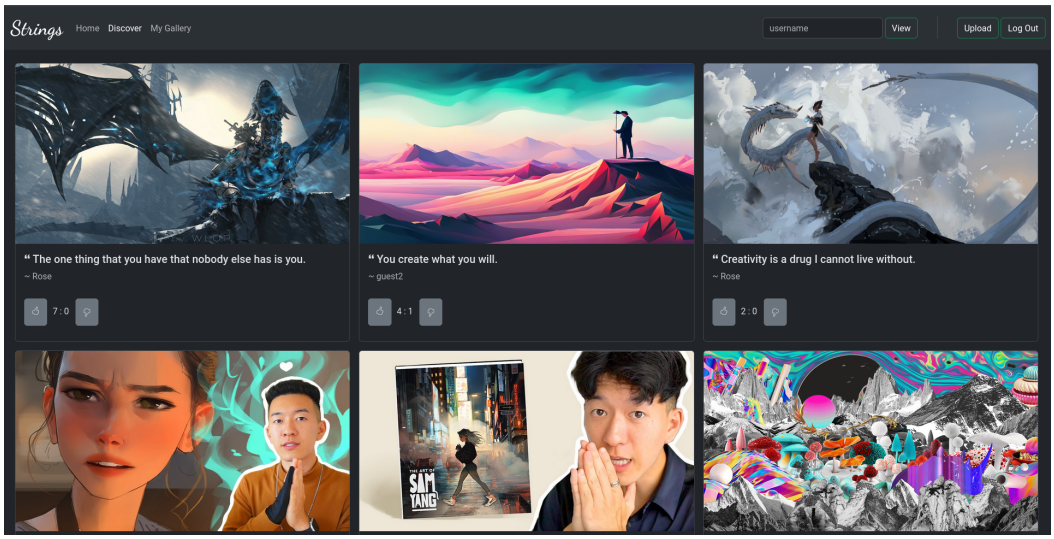
- Retrieves and displays artwork from the database
- Implements pagination and filtering mechanisms for efficient content navigation

Form Design

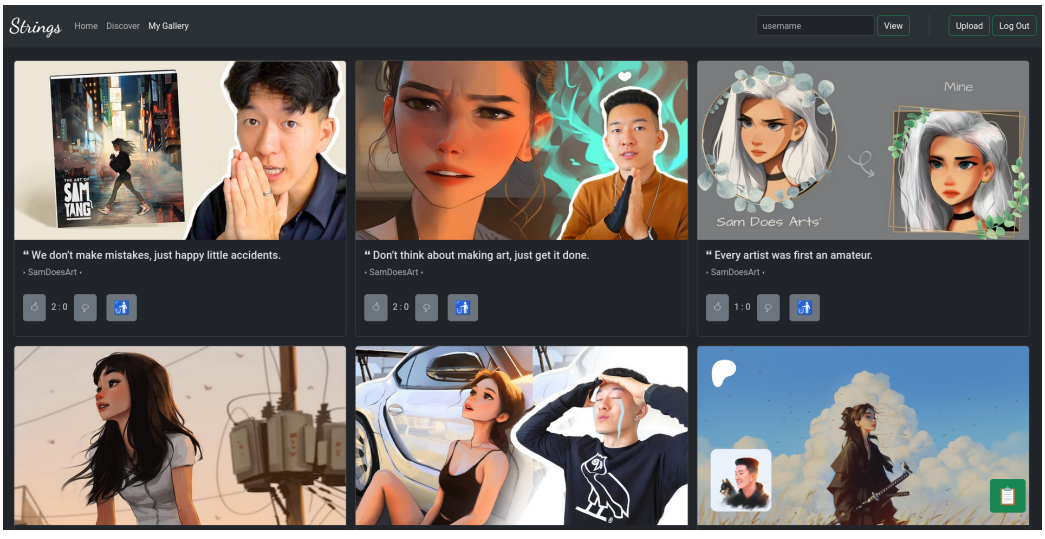
- ***Home Page***



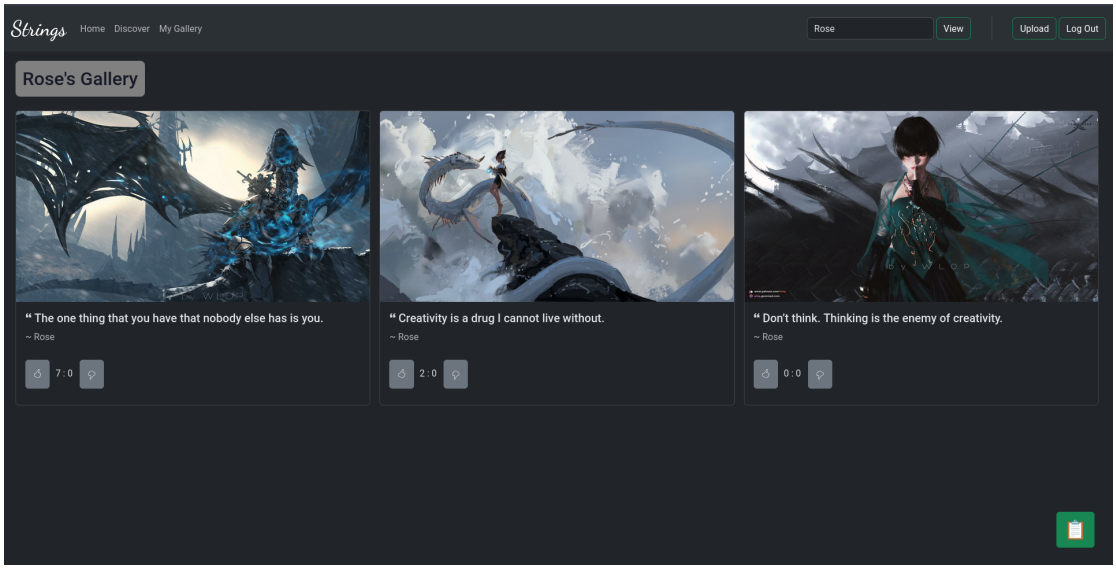
- ***Discover Page***



- ***User’s Gallery***

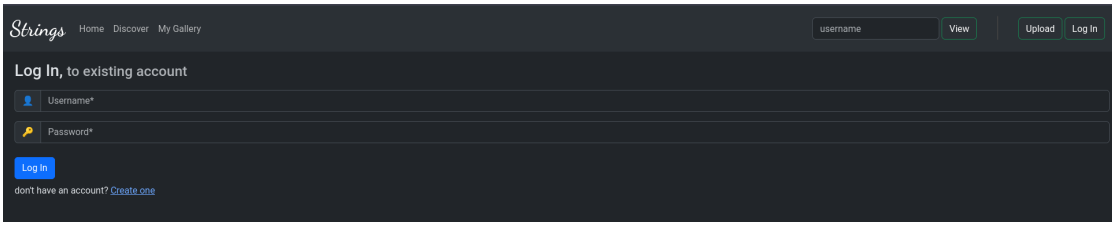


- ***Other Users's Gallery***

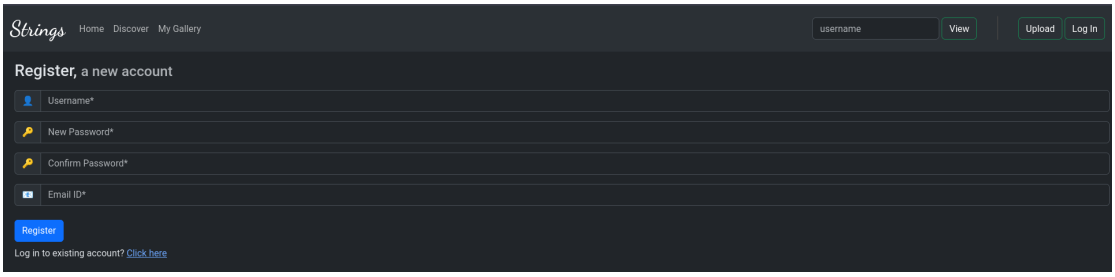


Input and Output Screens

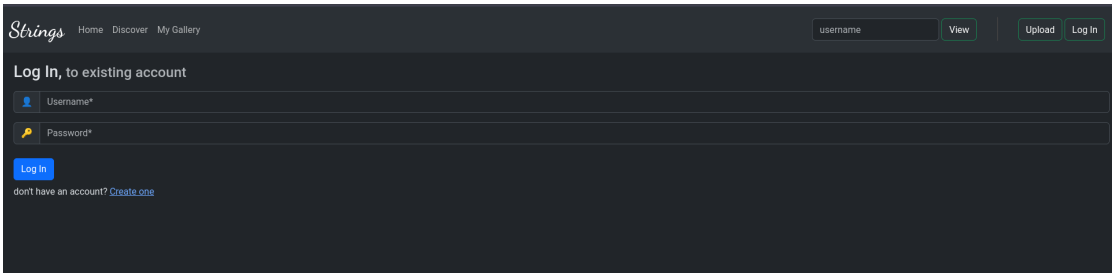
- ***User Login Page***



- ***User Registration Page***



- ***Upload Page***



Testing and Validation Checks

Throughout the development process, rigorous testing and validation measures were implemented to ensure the system's functionality, reliability, and security. The following testing strategies were employed:

- **Unit Testing:**

  Individual components and functions were tested in isolation to identify and resolve errors or bugs.

- **Integration Testing:**

  Modules were integrated and tested to ensure seamless functionality and compatibility.

- **User Interface Testing:**

  The user interface was thoroughly tested for usability, responsiveness, and accessibility.

- **Security Testing:**

  Vulnerability assessments and penetration testing were conducted to identify and mitigate potential security risks.

- **Performance Testing:**

  The application was tested under various load conditions to evaluate its performance and scalability.

System Security Measures

Ensuring the security and privacy of user data was a top priority in the development of "Strings: A Digital Art Gallery." The following security measures were implemented:

- ***Secure Authentication:***

  Strong password hashing algorithms and secure session management techniques were used to protect user credentials and prevent unauthorized access.

- ***Data Encryption:***

  Sensitive user data, such as passwords, was encrypted using industry-standard encryption algorithms.

- ***Input Validation:***

  All user inputs were validated and sanitized to prevent injection attacks and other security vulnerabilities.

- ***Access Control:***

  Role-based access control mechanisms were implemented to restrict access to sensitive data and functionalities based on user roles.

- ***Secure File Upload:***

  Uploaded artwork was validated and stored securely, with measures in place to prevent the upload of malicious files.

Implementation, Evaluation, and Maintenance

The "Strings: A Digital Art Gallery" project was implemented using an Agile development methodology, allowing for iterative development, continuous integration, and frequent feedback loops. The implementation process involved several stages, including requirements gathering, system design, development, testing, and deployment.

Throughout the implementation phase, regular meetings and progress reviews were conducted to ensure alignment with project objectives and to address any challenges or concerns that arose.

Upon completion, the project was thoroughly evaluated to assess its success in achieving the defined objectives. User feedback and performance metrics were analyzed to identify areas for improvement and potential future enhancements.

To ensure the long-term sustainability and success of the platform, a comprehensive maintenance plan was developed. This plan includes regular security updates, bug fixes, performance optimizations, and the integration of new features and functionalities based on user feedback and industry trends.

Future Scope of the Project

The "Strings: A Digital Art Gallery" project has significant potential for future growth and expansion. Some potential enhancements and additional features that could be considered include:

1. **Virtual Exhibitions and Events:**

- Organize virtual exhibitions and events where artists can showcase their work and interact with attendees in real-time.

2. **Commissioned Art:**

- Introduce a feature that allows users to commission artwork from their favorite artists, fostering collaboration and supporting artists financially.

3. **Augmented Reality Integration:**

- Explore the integration of augmented reality (AR) technology to provide an immersive experience for viewers, allowing them to visualize artwork in their physical spaces.

4. **Mobile Application Development:**

- Develop a mobile application to extend the platform's reach and provide a seamless experience for artists and viewers on-the-go.

Conclusion

The "Strings: A Digital Art Gallery" project has successfully achieved its objectives of creating an engaging and immersive digital platform for artists to showcase their talents, connect with like-minded individuals, and foster a vibrant community of art enthusiasts. Through its user-friendly interface, powerful features, and robust security measures, the application has demonstrated its potential to revolutionize the way art is experienced and shared in the digital realm.

By providing a virtual space for artists to curate their own galleries, upload their masterpieces, and present their creative processes, Strings has bridged the gap between artists and their global audience, transcending physical boundaries and limitations. The application's emphasis on artist connectivity and community engagement has paved the way for constructive feedback, artistic exploration, and the celebration of diverse artistic styles.

The project's success can be attributed to its comprehensive system analysis, meticulous design, and rigorous testing and validation processes. The implementation of industry-standard technologies and best practices has ensured the platform's scalability, reliability, and adaptability to emerging trends and user requirements.

As the project moves forward, the future scope outlined in this report presents exciting opportunities for growth and innovation. Integrating social media, virtual exhibitions, commissioned art, artist portfolios, augmented reality, and mobile applications will not only enhance the user experience but also solidify Strings' position as a leading digital art platform.

The "Strings: A Digital Art Gallery" project is a testament to the power of technology in democratizing art and fostering a global creative community. By providing a virtual canvas for artists to showcase their talents and connect with like-minded individuals, the project has taken a significant step towards shaping the future of art in the digital age.