Contents

1 Phase-2 Report: A Detailed Technical Progress Overview

Authors: *Anton Hibl, Ethan Wilcox*

1.1 Introduction

Phase-2 of the project has built upon the solid foundation laid in Phase-1, with a focus on expanding functionalities, refining user experience, and enhancing security measures. This stage was marked by iterative development, rigorous testing, and effective collaboration among the team members.

1.2 Frontend Development

1.2.1 Features and Functionalities

- Gallery Component Development: The Gallery component was designed and developed to display individual ArtworkCards, providing an immersive browsing experience for the users.

- ArtworkCard and ArtworkDetail Components Development: The representation of individual artworks and their detailed view was implemented, allowing users to explore and appreciate the artworks in depth.

- UserAccount Component Development: The UserAccount component was developed for managing user profile details and uploaded artworks, offering users a personalized and manageable space.
- Transition to Material-UI: To ensure a more consistent and robust UI, the components were refactored to use Material-UI instead of Fluent UI.

1.2.2 Design Components and Wireframes

The wireframes were updated to reflect the new components and changes in the user interface. The components were further refined to ensure a cohesive and intuitive user experience.

1.3 Backend Development

1.3.1 Features and Functionalities

- Enhanced User Authentication: The user authentication process was improved with the addition of a GET endpoint for the '/register' route, providing a graphical way for users to sign up.
- Artwork Model: An Artwork model was implemented to manage the artworks in the database, facilitating efficient data handling.
- VerifyToken Middleware: A middleware function was developed to verify tokens, enhancing the security of the application by ensuring only authenticated users can access certain routes.

1.3.2 Secure Design Considerations

- Token Verification: The VerifyToken middleware was used to verify the 'auth-token' in the headers of requests, ensuring secure access to certain endpoints.
- Secure Routing: Routes were implemented for all the client components, ensuring a secure and seamless navigation experience for the users.

1.4 Integration of Client and Server

The backend and frontend were integrated by connecting the API with the frontend components. The user flow was streamlined, providing a smooth and intuitive experience from registration to browsing the gallery.

1.5 Test-Driven Development (TDD)

The testing suite was expanded to cover the new functionalities:

- Unit Tests: Additional unit tests were written for the new functions and methods.
- Integration Tests: New integration tests were developed to test the interactions between the new components and the backend.
- End-to-End Tests: The end-to-end tests were updated to cover the new user flow and functionalities.

1.6 Changes and Adjustments

As the project evolved, several changes and adjustments were made:

- Transition to Material-UI: The decision to switch from Fluent UI to Material-UI was made to ensure a more consistent and robust user interface.
- Routing Adjustments: The routing was adjusted to reflect the new user flow and to ensure secure access to certain routes.

1.7 Challenges and Overcoming Them

Phase-2 presented its own set of challenges:

- UI Library Transition: The transition from Fluent UI to Material-UI required a thorough understanding of the new library and careful refactoring of the components.
- Secure Routing: Implementing secure routing required a nuanced approach and careful testing to ensure only authenticated users can access certain routes.

1.8 Lessons Learned

Key takeaways from Phase-2 include:

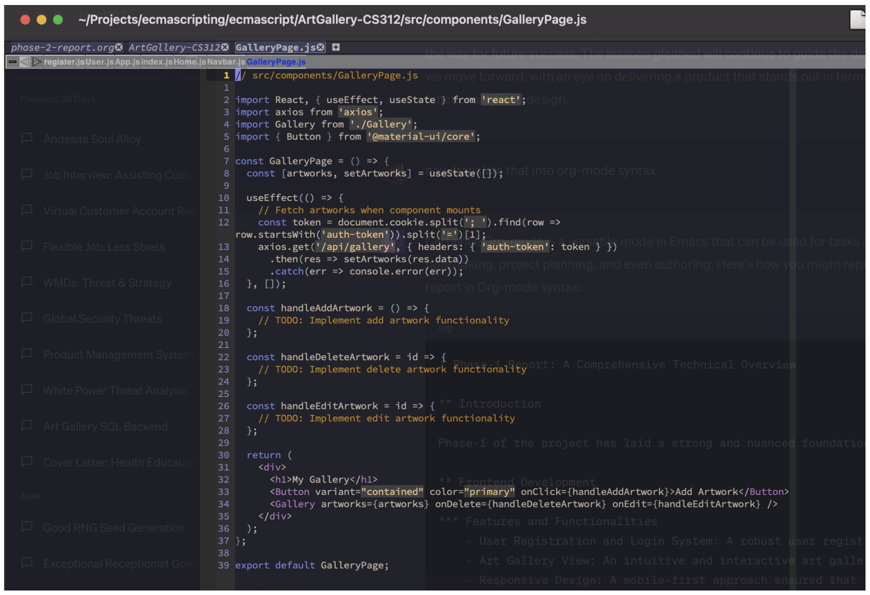
- The Importance of Flexibility: The ability to adapt to changes and new requirements played a crucial role in the successful completion of this phase.

- Value of Security: A consistent focus on security ensured a product that not only delivers in terms of functionality and design but also in terms of user privacy and data integrity.

1.9 Conclusion

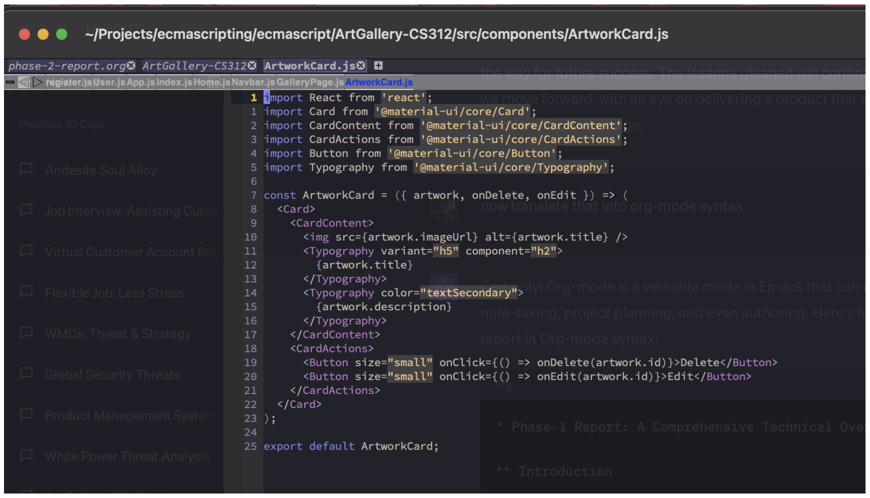
Phase-2 of the project has seen the successful expansion of functionalities, refinement of user experience, and enhancement of security measures. The team's commitment to best practices, continuous learning, and collaboration has led to the successful completion of this phase and set the stage for the final phase of the project. The lessons learned will/would continue to guide the development as we move forward, with a focus on delivering a product that excels in terms of quality, functionality, and design.

1.10 Screenshots


~/Projects/ecmascripting/ecmascript/ArtGallery-CS312/src/components/GalleryPage.js
phase-2-report.org
ArtGallery-CS312
GalleryPage.js
register.js User.js App.js index.js Home.js Navbar.js GalleryPage.js

```
// src/components/GalleryPage.js

import React, { useEffect, useState } from 'react';
import axios from 'axios';
import Gallery from './Gallery';
import { Button } from '@material-ui/core';

const GalleryPage = () => {
  const [artworks, setArtworks] = useState([]);

  useEffect(() => {
    // Fetch artworks when component mounts
    const token = document.cookie.split('; ').find(row =>
row.startsWith('auth-token')).split('=')[1];
    axios.get('/api/gallery', { headers: { 'auth-token': token } })
      .then(res => setArtworks(res.data))
      .catch(err => console.error(err));
  }, []);

  const handleAddArtwork = () => {
    // TODO: Implement add artwork functionality
  };

  const handleDeleteArtwork = id => {
    // TODO: Implement delete artwork functionality
  };

  const handleEditArtwork = id => {
    // TODO: Implement edit artwork functionality
  };

  return (
    <div>
      <h1>My Gallery</h1>
      <Button variant="contained" color="primary" onClick={handleAddArtwork}>Add Artwork</Button>
      <Gallery artworks={artworks} onDelete={handleDeleteArtwork} onEdit={handleEditArtwork} />
    </div>
  );
};

export default GalleryPage;
```

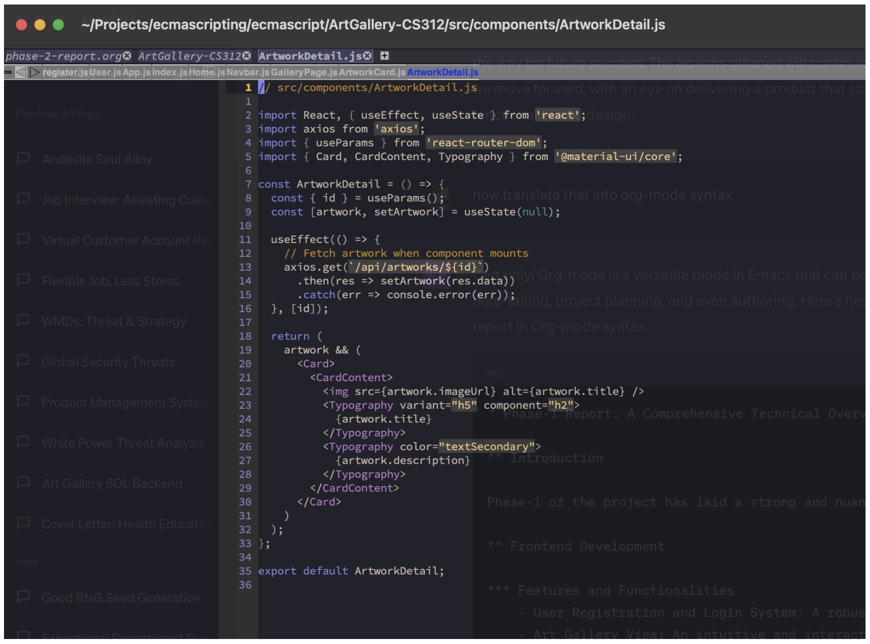
Figure 1: GalleryPage component



~/Projects/ecmascripting/ecmascript/ArtGallery-CS312/src/components/ArtworkCard.js

```javascript
import React from 'react';
import Card from '@material-ui/core/Card';
import CardContent from '@material-ui/core/CardContent';
import CardActions from '@material-ui/core/CardActions';
import Button from '@material-ui/core/Button';
import Typography from '@material-ui/core/Typography';

const ArtworkCard = ({ artwork, onDelete, onEdit }) => (
  <Card>
    <CardContent>
      <img src={artwork.imageUrl} alt={artwork.title} />
      <Typography variant="h5" component="h2">
        {artwork.title}
      </Typography>
      <Typography color="textSecondary">
        {artwork.description}
      </Typography>
    </CardContent>
    <CardActions>
      <Button size="small" onClick={() => onDelete(artwork.id)}>Delete</Button>
      <Button size="small" onClick={() => onEdit(artwork.id)}>Edit</Button>
    </CardActions>
  </Card>
);

export default ArtworkCard;
```

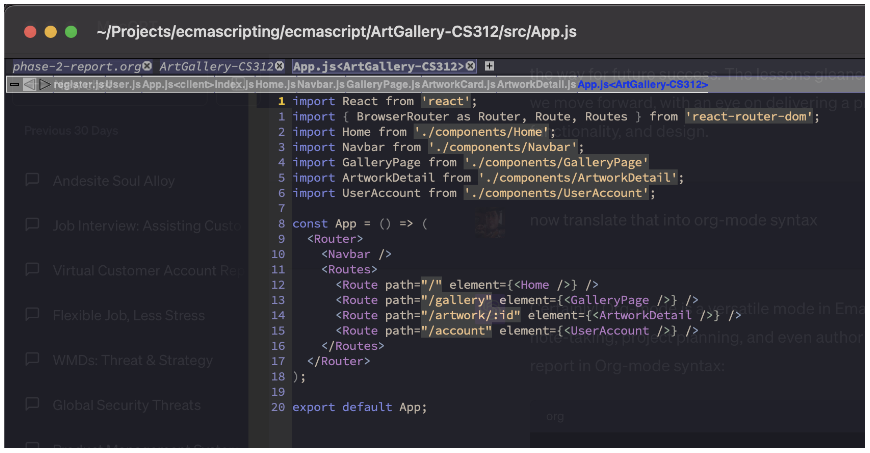
Figure 2: ArtworkCard component



~/Projects/ecmascripting/ecmascript/ArtGallery-CS312/src/components/ArtworkDetail.js

```javascript
// src/components/ArtworkDetail.js

import React, { useEffect, useState } from 'react';
import axios from 'axios';
import { useParams } from 'react-router-dom';
import { Card, CardContent, Typography } from '@material-ui/core';

const ArtworkDetail = () => {
  const { id } = useParams();
  const [artwork, setArtwork] = useState(null);

  useEffect(() => {
    // Fetch artwork when component mounts
    axios.get(`/api/artworks/${id}`)
      .then(res => setArtwork(res.data))
      .catch(err => console.error(err));
  }, [id]);

  return (
    artwork && (
      <Card>
        <CardContent>
          <img src={artwork.imageUrl} alt={artwork.title} />
          <Typography variant="h5" component="h2">
            {artwork.title}
          </Typography>
          <Typography color="textSecondary">
            {artwork.description}
          </Typography>
        </CardContent>
      </Card>
    )
  );
};

export default ArtworkDetail;
```

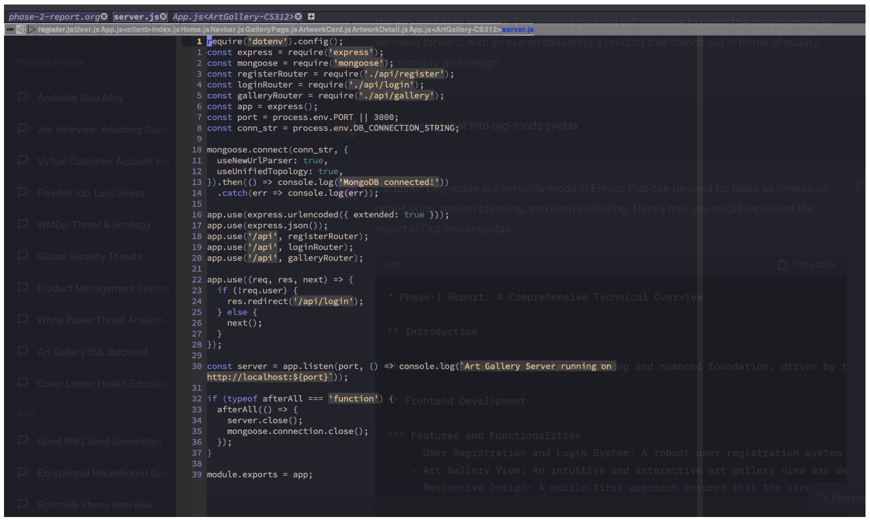
Figure 3: ArtworkDetail component



```
~/Projects/ecmascripting/ecmascript/ArtGallery-CS312/src/App.js
```

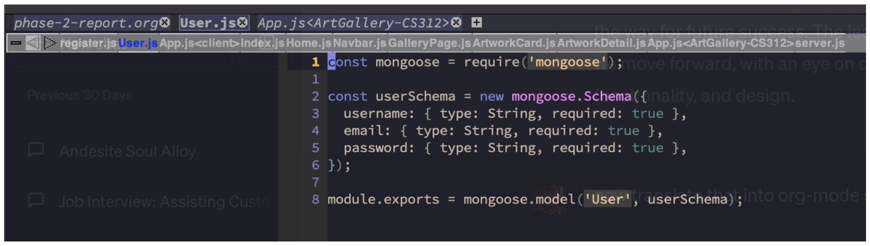
```javascript
import React from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import Home from './components/Home';
import Navbar from './components/Navbar';
import GalleryPage from './components/GalleryPage'
import ArtworkDetail from './components/ArtworkDetail';
import UserAccount from './components/UserAccount';

const App = () => (
  <Router>
    <Navbar />
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/gallery" element={<GalleryPage />} />
      <Route path="/artwork/:id" element={<ArtworkDetail />} />
      <Route path="/account" element={<UserAccount />} />
    </Routes>
  </Router>
);

export default App;
```

Figure 4: App.js



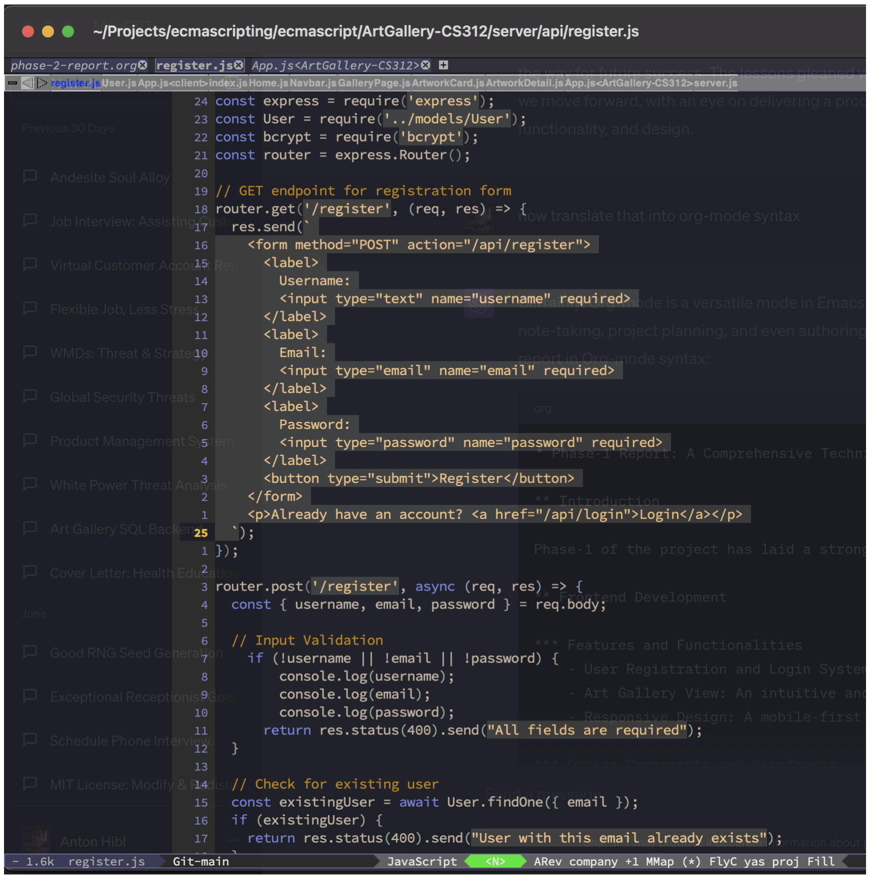
```javascript
require('dotenv').config();
const express = require('express');
const mongoose = require('mongoose');
const registerRouter = require('./api/register');
const loginRouter = require('./api/login');
const galleryRouter = require('./api/gallery');
const app = express();
const port = process.env.PORT || 3000;
const conn_str = process.env.DB_CONNECTION_STRING;

mongoose.connect(conn_str, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => console.log('MongoDB connected!'))
  .catch(err => console.log(err));

app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use('/api', registerRouter);
app.use('/api', loginRouter);
app.use('/api', galleryRouter);

app.use((req, res, next) => {
  if (!req.user) {
    res.redirect('/api/login');
  } else {
    next();
  }
});

const server = app.listen(port, () => console.log(`Art Gallery Server running on http://localhost:${port}`));

if (typeof afterAll === 'function') {
  afterAll(() => {
    server.close();
    mongoose.connection.close();
  });
}

module.exports = app;
```

Figure 5: server.js



```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  username: { type: String, required: true },
  email: { type: String, required: true },
  password: { type: String, required: true },
});

module.exports = mongoose.model('User', userSchema);
```

Figure 6: index.js

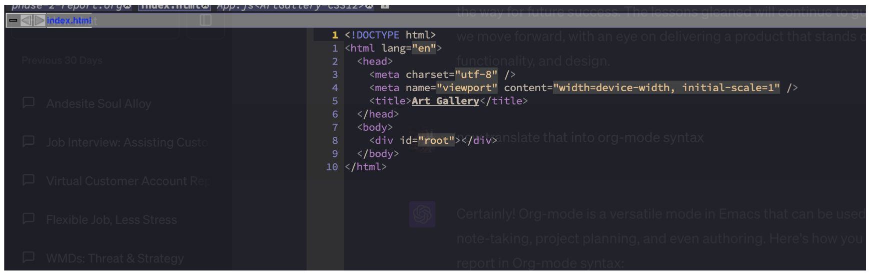


~/Projects/ecmascripting/ecmascript/ArtGallery-CS312/server/api/register.js

```
const express = require('express');
const User = require('../models/User');
const bcrypt = require('bcrypt');
const router = express.Router();

// GET endpoint for registration form
router.get('/register', (req, res) => {
  res.send(`
    <form method="POST" action="/api/register">
      <label>
        Username:
        <input type="text" name="username" required>
      </label>
      <label>
        Email:
        <input type="email" name="email" required>
      </label>
      <label>
        Password:
        <input type="password" name="password" required>
      </label>
      <button type="submit">Register</button>
    </form>
    <p>Already have an account? <a href="/api/login">Login</a></p>
  `);
});

router.post('/register', async (req, res) => {
  const { username, email, password } = req.body;

  // Input Validation
    if (!username || !email || !password) {
        console.log(username);
        console.log(email);
        console.log(password);
      return res.status(400).send("All fields are required");
  }

  // Check for existing user
  const existingUser = await User.findOne({ email });
  if (existingUser) {
    return res.status(400).send("User with this email already exists");
```

```
- 1.6k  register.js   Git-main                JavaScript  <N>  ARev company +1 MMap (*) FlyC yas proj Fill
```

Figure 7: register.js



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Art Gallery</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

Figure 8: index.html