Contents

1 Phase-1 Report: A Comprehensive Technical Overview

1.1 Introduction

Phase-1 of the project has laid a strong and nuanced foundation, driven by the vision of delivering a high-quality, secure, and efficient product. This stage was characterized by meticulous planning, thoughtful design, extensive testing, and close connection between different facets of the development team.

1.2 Frontend Development

1.2.1 Features and Functionalities

- User Registration and Login System: A robust user registration system was developed with input validation, secure password handling using cryptographic hashing, and a seamless user experience.
- Art Gallery View: An intuitive and interactive art gallery view was designed, with capabilities for users to browse, filter, and view artworks in various arrangements.
- Responsive Design: A mobile-first approach ensured that the user interface would adapt across different devices and screen sizes.

1.2.2 Design Components and Wireframes

The wireframes created during this phase provided a visual blueprint for the interface design. Components were mapped out to ensure cohesion across different sections of the application, with an emphasis on usability and aesthetics.

1.3 Backend Development

1.3.1 Features and Functionalities

- User Authentication and Authorization: Utilizing JWT tokens, a secure authentication system was implemented, ensuring data integrity and user privacy by protecting and hashing passwords.
- UI: Navigation and homepage components were completed and will ensure a complete and unified experience for visitors to the gallery.
- Database Design: A simple yet effective ER schema was designed and translated into JavaScript, which has established a solid foundation for data relations and integrity.

1.3.2 Secure Design Considerations

- Data Validation: Extensive validation rules were applied to prevent SQL injections.
- Encryption: Encryption and specifically hashing techniques were used to secure sensitive data like passwords.

1.4 Integration of Gulp

Gulp was integrated into the workflow, providing automation for repetitive tasks such as minification, compilation, unit testing, and linting. This not only streamlined the development process but also ensured consistent code quality; now we are able to ensure the build system stays the same across the local and remote environments and ensure all changes work immediately after they are pushed. Overall this helps debugging and reduces time to solve issues in our experience.

1.5 Test-Driven Development (TDD)

Through TDD, a detailed suite of tests was developed using Jest and Supertest:

- Unit Tests: For individual functions and methods.
- Integration Tests: To test interactions between different parts of the application.
- End-to-End Tests: To simulate real user scenarios and test the application as a whole.

The continuous testing cycle facilitated early detection of issues and robust code quality. We specifically use two suites to test either the front-end(jest) or the back-end(supertest).

1.6 Changes and Adjustments

The initial phase of development was accompanied by a few unforeseen challenges and insights that required changes:

- Design Alterations: The initial ER schema underwent several iterationschanges to match the requirements while maintaining it's simplicity.
- Technology Adjustments: Decisions regarding the tools and libraries were refined based on compatibility and performance considerations; We did not originally know we would choose jest, supertest or gulp.

1.7 Challenges and Overcoming Them

Phase-1 was not without its hurdles:

- Integration Challenges with Gulp: The integration required a deep understanding of the tool and its plugins. This was overcome through extensive documentation review and experimentation.
- Security Concerns: Ensuring a secure design required an attentive approach, this was achieved through following best practices like password hashing and then using the appropriate libraries to deal with these nuances.

1.8 Lessons Learned

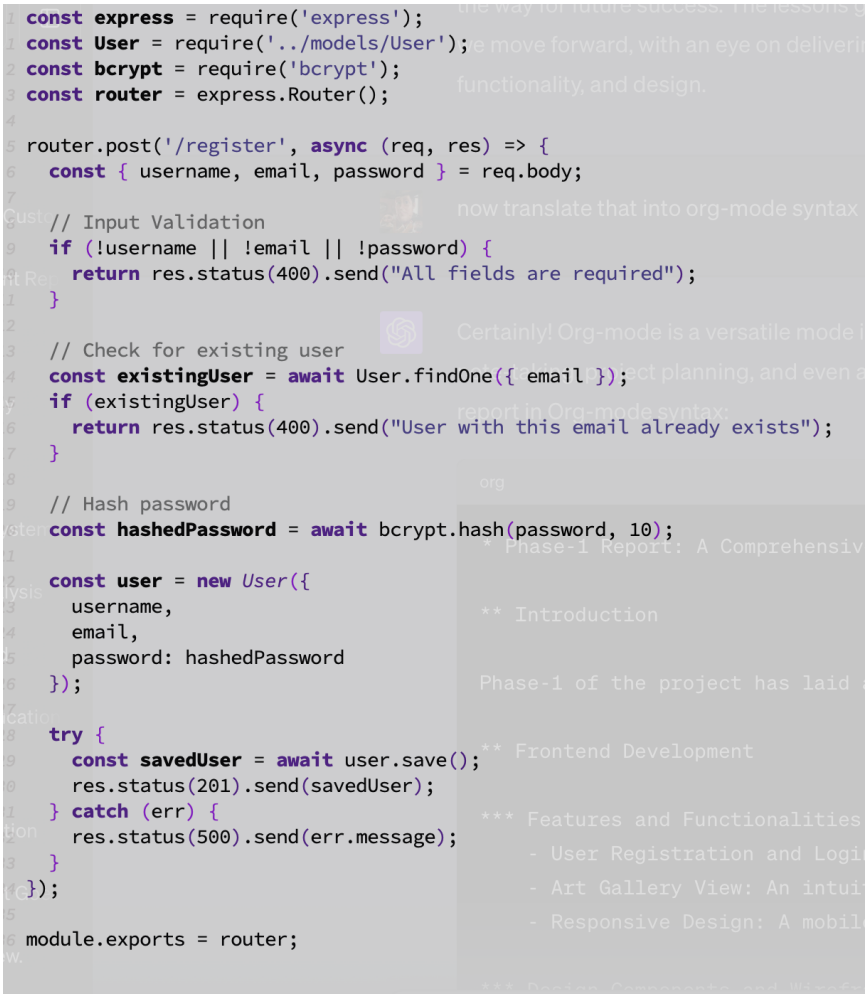
Key takeaways from Phase-1 include:

- The Importance of Planning: Comprehensive planning played a pivotal role in keeping the project on track.
- History tracking: code change tracking tools like GitHub fostered a cohesive development environment to review and track the project and it's changes over time.
- Emphasis on Quality: A consistent focus on code quality, security, and usability ensured a productset of deliverables aligned with high standards.

1.9 Screenshots

1.10 Conclusion

Phase-1 of the project has set the stage for a well-architected, secure, and user-centric application. The team's commitment to best practices, innovation, and continuous improvement has not only led to the successful completion of this phase but also paved the way for future success. The lessons gleaned will continue to guide the development as we move forward, with an eye on delivering a product that stands out in terms of quality, functionality, and design.



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

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

  // Input Validation
  if (!username || !email || !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");
  }

  // Hash password
  const hashedPassword = await bcrypt.hash(password, 10);

  const user = new User({
    username,
    email,
    password: hashedPassword
  });

  try {
    const savedUser = await user.save();
    res.status(201).send(savedUser);
  } catch (err) {
    res.status(500).send(err.message);
  }
});

module.exports = router;
```

Figure 1: User Registration Logic

```
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 2: User Schema(Code)

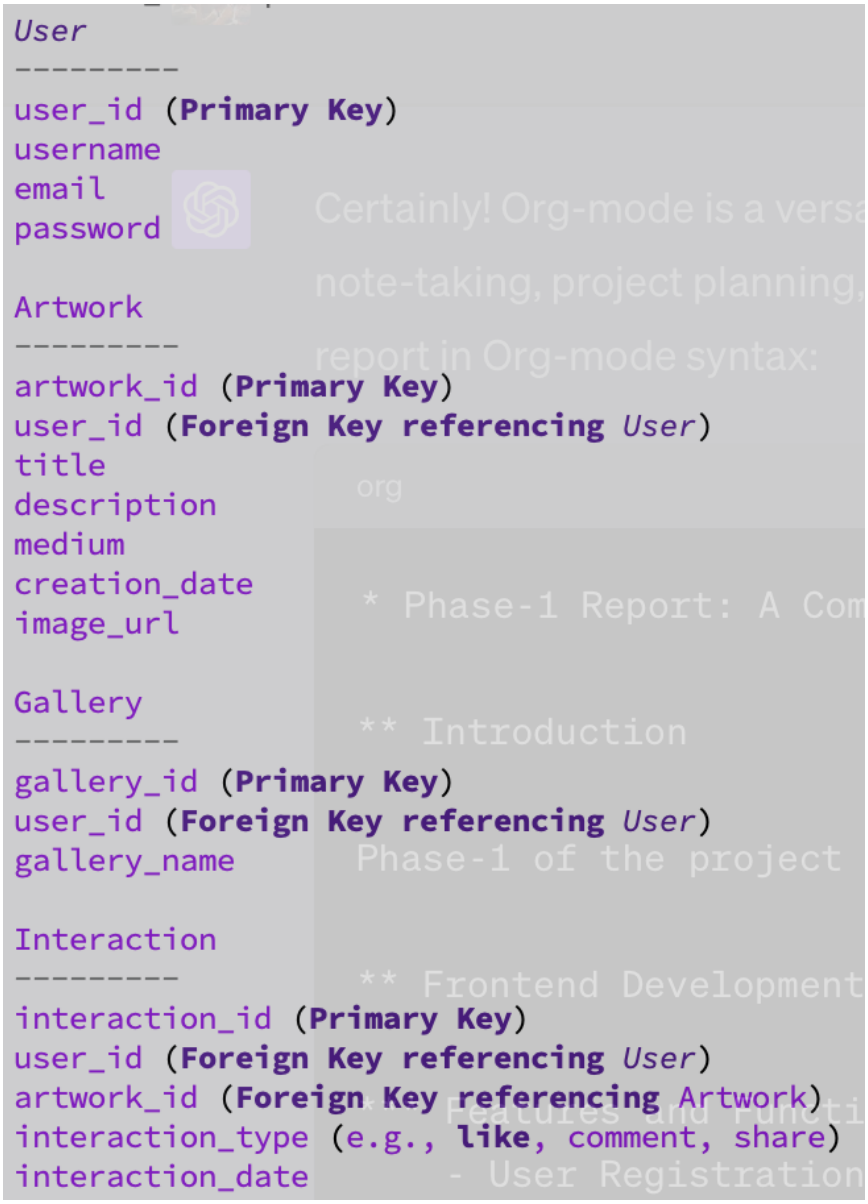


Figure 3: User Schema(ER Template)

```jsx
import React from 'react';
import { Nav } from '@fluentui/react';

const links = [
  {
    name: 'Home',
    url: '/',
    key: 'key1',
  },
  {
    name: 'Login',
    url: '/login',
    key: 'key2',
  },
  {
    name: 'Register',
    url: '/register',
    key: 'key3',
  },
];

const Navbar = () => (
  <Nav groups={[{ links }]} />
);

export default Navbar;
```

Figure 4: Navigation Logic

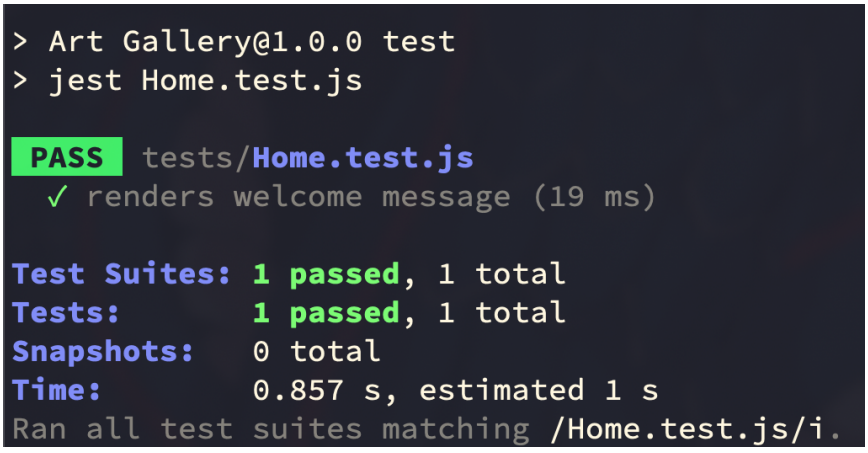


Figure 5: Tests Example