BHAVYA ANAND

Project Progress Report

Project Title: File Management System
Reporting Period: *11th December,2023 to 15th December,2023.*

Key Objectives:

Project Setup: Establish the project structure, dependencies, and initial configurations.
Backend Development: Implement basic backend functionality, including user authentication, database connection, and API endpoints.
Security Measures: Integrate security features such as password hashing using bcrypt and JWT token generation for authorization.
Utility Components: Create utility classes and components for handling asynchronous operations, API responses, and errors.
Middleware Integration: Add necessary middleware, such as CORS, for proper API endpoint access.

Accomplishments:

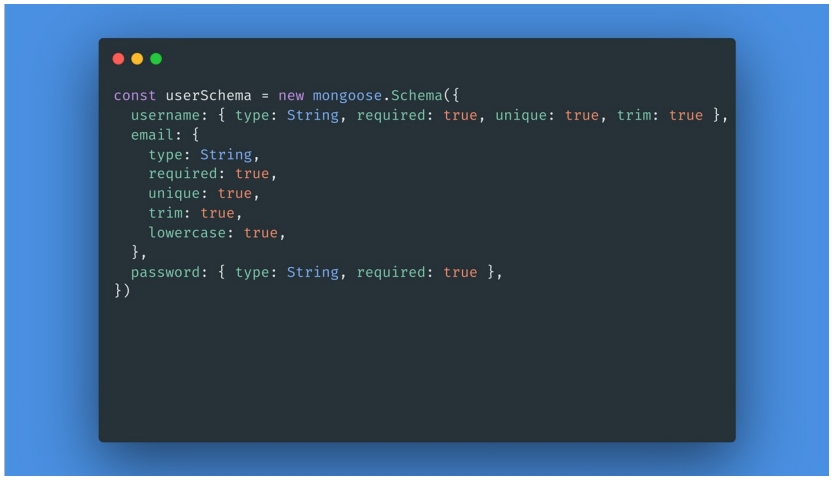
[Bhavya Anand]

Project Setup:

Directory Structure: Created a well-organized directory structure for the project, separating frontend and backend components.
Configuration Setup: Established initial configurations for React JS and Tailwind CSS, ensuring a smooth development environment.

Backend Development:

Express and Node.js Setup: Implemented the backend using Express and Node.js, enabling efficient server-side operations.
Database Connection: Successfully connected the backend to the MongoDB database, laying the foundation for data storage.
User Model: Developed a robust user model, including necessary attributes and relationships with other entities (files, folders).



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

BHAVYA ANAND

API Endpoints: Established initial API endpoints for user authentication and basic CRUD operations on files and folders.

Error Handling: Implemented a consistent error handling mechanism using utility components for better code maintainability.

```
class ApiError extends Error {
  constructor(
    statusCode,
    message = 'Something went wrong',
    error = [],
    stack = ''
  ) {
    super(message)
    this.statusCode = statusCode
    this.data = null
    this.message = message
    this.success = false
    this.errors = errors
    if (stack) {
      this.stack = stack
    } else {
      Error.captureStackTrace(this, this.constructor)
    }
  }
}
export default ApiError
```

Security Measures:

Password Hashing: Integrated bcrypt to hash user passwords, enhancing security measures within the user model.

```
userSchema.pre('save', async function (next) {
  const user = this
  if (user.isModified('password') || user.isNew) {
    try {
      const salt = await bcrypt.genSalt(10)
      const hashedPassword = await bcrypt.hash(user.password, salt)
      user.password = hashedPassword
      next()
    } catch (error) {
      return next('Error while encrypting the password', error)
    }
  } else {
    return next()
  }
})
```

JWT Token Generation: Implemented JWT token generation for user authorization, enhancing the overall security of the system.

```
userSchema.methods.generateAccessToken = function () {
  return jwt.sign(
    { _id: this._id, username: this.username },
    process.env.ACCESS_TOKEN_SECRET,
    { expiresIn: process.env.ACCESS_TOKEN_EXPIRY }
  )
}
userSchema.methods.generateRefreshToken = function () {
  return jwt.sign({ _id: this._id }, process.env.REFRESH_TOKEN_SECRET, {
    expiresIn: process.env.REFRESH_TOKEN_EXPIRY,
  })
}
```

BHAVYA ANAND

Utility Components:

Async Handler: Created a utility class to handle asynchronous operations, ensuring a more streamlined and readable codebase.
API Response Component: Developed a reusable component for consistent API response formatting, promoting code consistency.
API Errors Component: Implemented a utility component to handle API errors, ensuring a standardized error response structure.

Middleware Integration:

CORS Middleware: Added CORS middleware to the backend to manage cross-origin resource sharing, facilitating proper API endpoint access.
Index.js Integration: Integrated all components and configurations in the index.js file for a centralized and organized project structure.

Miscellaneous:

1. Conducted thorough testing of backend functionalities to identify and address potential bugs.
2. Documented the setup and configuration processes for future reference.

Overall Progress:

1. The project has made significant strides in its initial setup and backend development.
2. Security measures, including password hashing and JWT token generation, have been successfully implemented.
3. Utility components have been created to enhance code readability and maintainability.
4. Middleware integration, particularly CORS, ensures proper API endpoint access.

Upcoming Goals:

File/Folder Management: Implement file and folder management functionalities, including creation, deletion, and retrieval.
Enhanced Authentication: Strengthen user authentication mechanisms and explore multi-factor authentication options.
Creation of all the routes: All the routes such as /upload, /update, /download will be created.
Frontend Development: Begin developing frontend components for user interactions and seamless integration with backend services.

Roadblocks/Challenges:

While the setup and initial development proceeded smoothly, potential challenges may arise during the integration of frontend components and advanced functionalities.

BHAVYA ANAND

Changes to Project Plan:
No significant changes to the original project plan; however, continuous adjustments will be made based on evolving requirements and feedback.

Risks:

Technical Risks:
Address potential technical challenges that may arise during the integration of frontend components.

Security Risks:
Regularly assessing and updating security measures to mitigate potential vulnerabilities.

Action Items:

File/Folder Management:

1. Implement CRUD operations for files and folders.
2. Establish secure file upload and download mechanisms.

Enhanced Authentication:

1. Explore and implement additional authentication measures.
2. Conduct security audits to identify potential vulnerabilities.

Frontend Development:

1. Initiate the development of frontend components for user interactions.
2. Ensure seamless integration with backend APIs.

Performance Monitoring:

1. Implement metrics tracking for key performance indicators.
2. Set up monitoring tools for continuous performance assessment.

Attachments:

Click here to access the GitHub repository for the project.

(All the codes have been already pushed to github)