Implementation Report - Sprint 1

1. Summary of Implemented Features

In Sprint 1, the core functionality of the LockBox password manager was developed. The primary features implemented include:

- User Registration and Login: The backend supports registration and login functionality. Users can register by providing a username, email, and password. The password is securely hashed using bcrypt before storing it in the database. Login functionality verifies user credentials and issues a successful login response upon validation.

- Database Integration: MongoDB is used as the database for storing user information. Mongoose was utilized for schema modeling and interacting with the MongoDB database.

- API Endpoints:
  - POST /register: Registers a new user with email and password validation.
  - POST /login: Allows users to log in using email and password validation.

- Validation: Input data is validated using express-validator to ensure proper user inputs for registration and login.

- Error Handling: Error responses for validation issues and user-related errors (e.g., user already exists, invalid credentials) are appropriately sent to the client.

2. Coding Standards

To ensure code consistency and readability, the following coding standards were implemented:

- ESLint: The ESLint configuration enforces a consistent coding style across all JavaScript files, such as indentation, variable declarations, and usage of semicolons.
- Prettier: Code formatting is standardized using Prettier, which ensures that the code remains clean and adheres to the defined style guide.
- Error Handling: Proper error messages and status codes are returned for invalid inputs or issues during user registration and login.
- Modularity: The code is structured into smaller, reusable modules (e.g., user models, validation) to maintain separation of concerns.

3. Review Process

A code review was conducted to ensure code quality and adherence to best practices. The key points addressed during the review were:

- Code Structure: The project structure was reviewed to ensure that files are logically organized. The models folder contains all database-related logic, while the main server logic is located in index.js.
- Error Handling: The error handling mechanisms were reviewed to ensure that the application correctly responds to user errors and system failures.
- Testing: Basic testing via Postman was conducted to validate that the registration and login endpoints are functioning correctly. It was agreed that more extensive unit tests will be implemented in future sprints.

After the review, several minor code improvements were made based on reviewer feedback. These changes were committed and pushed to the repository.

4. Testing Approach

The following testing approach was followed for Sprint 1:

- Manual Testing: All features were tested manually using Postman to ensure that the registration and login functionality works as expected.
  - Registration: Tested by submitting valid and invalid data to ensure proper validation and user creation.
  - Login: Tested by submitting correct and incorrect credentials to verify the response and error handling.

- Automated Testing (Future): Automated unit tests using Jest will be added in future sprints to increase code coverage and ensure that features continue to function correctly as the project evolves.

5. Challenges Encountered

The following challenges were encountered and successfully addressed during the sprint:

- MongoDB Connection: Initially, there were issues with connecting to MongoDB due to incorrect connection strings. This was resolved by configuring the connection string correctly and ensuring MongoDB was running locally.

- Password Hashing: Ensuring that passwords were hashed properly using bcrypt took some trial and error. Once hashed, the login functionality required careful handling of password comparisons to avoid security issues.

- API Validation: Input validation for both registration and login required careful attention to ensure that the inputs met the expected format. Express-validator was integrated to handle these validation requirements.

- Code Formatting and Linting: Setting up ESLint and Prettier together took some time to configure correctly. However, once set up, these tools ensured a clean and consistent codebase.