**Document Name:** Definition of Done/Style Guide

**Product Name:** MuscleMinder

**Definition of Done (DoD)**

For our FitTech Innovators team working on MuscleMinder, the Definition of Done includes the following criteria:

**1. Code Quality and Standards**

- Code has been reviewed by at least one other team member.

- Code is documented and follows the team's coding standards.

- Code is free from known bugs and passes all unit tests.

**2. Testing**

- All unit tests have been written and passed.

- Integration tests have been conducted and passed.

- User acceptance tests have been completed and passed.

- No critical or high-severity bugs remain unresolved.

**3. Functionality**

- The feature meets the acceptance criteria defined in the user story.

- The feature has been demonstrated to the Product Owner and has been accepted.

- All related documentation (user guides, release notes) has been updated.

**4. Performance**

- Performance criteria are met, including response times and load handling.

- The feature does not degrade the performance of the existing system.

**5. Security**

- Security standards and best practices have been adhered to.

- No security vulnerabilities are introduced by the new feature.

**6. Usability**

- The features have been tested for usability and accessibility.

- Feedback from usability testing has been addressed.

**7. Deployment**

- The feature has been deployed to a staging environment.

- Deployment scripts and configurations are updated.

- The feature is ready for deployment to production.

**8. Documentation**

- Relevant documentation is updated, including user manuals, README, and API documentation.

- Change logs are updated to reflect the new feature.

**Style Guide**

The Style Guide provides guidelines for maintaining consistency and quality across the project. For MuscleMinder, the Style Guide includes the following:

**1. Code Formatting**

- Follow standardized guidelines and protocols for code structure.

- Use consistent indentation (4 spaces per indentation level).

- Use meaningful variable and function names that clearly describe their purpose.

**2. Comments and Documentation**

- Write comments to explain the purpose of code sections.

- Document important design decisions.

**3. Version Control**

- Used Github for version control.

- Commit messages are to be clear and concise

**4. UI/UX Design**

- Follow the design specifications provided by team agreements.

- Ensure consistency in UI elements (buttons, forms, icons).

- Prioritize accessibility, ensuring all features are usable by individuals with disabilities.

**5. API Development**

- Use RESTful principles for API design.

- Ensure all endpoints are working properly.

- Validate all inputs and handle errors gracefully.

**6. Testing**

- Write unit tests for new features and bug fixes.

- Use descriptive names for test cases and organize tests logically.

**7. Performance Optimization**

- Optimize code for performance where necessary.

- Avoid premature optimization, focus on readability and maintainability first.

**8. Security**

- Follow security guidelines.

- Regularly review and update dependencies to address security vulnerabilities.

- Use secure coding practices, including input validation and output encoding.

**9. Collaboration and Communication**

- Maintain open and transparent communication within the team.

- Use project management tools to track progress and tasks.

- Regularly participate in team meetings, stand-ups, and retrospectives.