Для повышения оценки до "хорошо" необходимо выполнить задание 1, 2, 3:

1) Необходимо провести анализ исходного кода приложения с целью выявления недостатков проектирования и кода. Проект для анализа нужно выложить в Git. Минимальное число проанализированных недостатков: 6 (типы недостатков должны быть разные).

Каждый найденный недостаток необходимо оформить в виде отдельного Issue с описанием следующих характеристик:
Тип недостатка (если определен в литературе).
К чему данная проблема может привести.
Из-за чего могла возникнуть данная проблема.
Если проблема относится к участку кода, необходимо добавить ссылку на этот участок/ процитировать его.

Пример:

Примером может быть public class Team. Чем он только не занимается - и с UI-элементами работает, и за экономику игры отвечает, и рейтинг команды меняет и общую логику поведения содержит …

```
public class Team : MonoBehaviour
{
    //много кода
}
```

Почему это плохо:

1. Нарушается принцип единственной ответственности. Он зависит от многих компонентов, логика поведения которых может измениться - придется менять и весь класс, отлавливать и искать в нем ошибки (примерно, как снежный ком)
2. В нем сложно ориентироваться - найти нужную функцию и т.д.
3. Скорее всего в некоторых функциях я повторяю то, что было уже реализовано (хотя бы частично в другом методе)

2) Необходимо покрыть Unit-тестами недостатки кода, обнаруженные при выполнении лабораторной работы №1: требуется полное покрытие ДВУХ недостатков модульными тестами (реализация всех приемов из книги Совершенный код - глава 22.3 Приемы тестирования + комментарии, где какой прием использован). (минимум 8 тестов на недостаток). К реквесту с тестами необходимо прикрепить видео с успешным прохождением тестов.

3) Необходимо провести рефакторинг недостатков, которые были обнаружены при выполнении лабораторной работы №1.
Рефакторинг каждого недостатка должен быть оформлен в виде отдельного (или группы коммитов) коммита в git.