Общие требования

В этом разделе описаны общие требования к проекту.

Исходный код

- В проекте должен использоваться GIT как система контроля версий и управления исходным кодом;
- В данном проекте должен использоваться репозиторий https://bitbucket.org/ или https://github.com/ (возможно использование альтернативы, по договоренности с ментором(исходя из контекста под ментором подразумевается тут и далее Product Owner, уточню сегодня у Шатовкой)). У каждого члена команды должна быть своя учетная запись. Список логинов должен быть предоставлен ментору с тем, чтобы он мог распознать каждого члена команды. Команда должна предоставить доступ к репозиторию BitBucket GIT ментору, в случае невозможности (в случае ограничения на количество аккаунтов, связанное с необходимостью дополнительной оплаты) создания лишнего аккаунта, один из членов команды должен предоставить собственные данные для входа. Использование личных хранилищ запрещено;
- Команда должна использовать систему мониторинга задач предоставляемую GitHub или BitBucket для менеджмента тасков и багов. Можно использовать любую другую бесплатную Agile доску;
- В проекте, для контроля качества написания исходного кода необходимо использовать инструменты статического анализа. При этом, проект должен быть разработан с отсутствием «warnings» инструмента статического анализа.
    - FindBugs и checkstyle нужно использовать для проектов на Java;
    - ReSharper и Style Cop для проектов по .Net.
- Опционально: Как средство непрерывной интеграции может быть использовано TeamCity или Jenkins;
- Опционально: В качестве средства статического анализа для любых типов проектов может быть использован Sonar, но по предварительной договоренности с ментором.

Рабочий процесс

Каждая task или bug-fix должна следовать четкому рабочему алгоритму. Этот алгоритм должен быть согласован с ментором. ИНАЧЕ:

Для task’ов.

- Таска создана в трэкере (github\bitbucket\etc.) (New);
- Таска взята девелопером на разработку (Assigned);
- Таска готова к осмотру (на этом этапе – «проверка» исходного кода) (For review);
- Таска взята девелопером на осмотр (In review) – отсюда таска может продвинутся вперед или назад – на доработку девелоперу, который за нее «в ответе»;
- Таска готова к функциональному тестированию (Ready to test);
- Таска тестируется (In test) – возможно ее продвижение назад (Assigned) или вперед (Done);
- Таска закрыта (Done).

Все приведенные выше этапы жизненного цикла таски должны быть выполнены разными людьми.

Для багов:

- New or Reopened or Cloned;
- Assigned;
- In review;
- Ready;
- In test;
- Done.

Качество

- Каждый итоговый продукт должен быть тестирован;
- Каждая разработанная feature должна быть тестирована;
- Определение статуса DONE: feature разработана, код проверен, функциональная составляющая оттестирована и удовлетворяет требованиям;
- Для каждой «фичи» и каждого найденного бага должны быть созданы Test Cases (короткая история, как это должно работать, маленькая User Story). Test Cases – часть финальной демонстрации. Test Cases нужно хранить в отдельной папке (в папке проекта).
- Осмотр кода каждой разработанной фичи должен осуществляться членом команды, который не принимал участия в ее разработке;
- На демо, в разработанных фичах не должно находиться багов «тяжестью» средний или выше;

- Каждый участник команды должен принимать участие в тестировании;
- Не менее 60% когда необходимо покрывать юнит-тестированию.

Отчетность и демонстрация

- Детальные User Stories для проекта должны быть обсуждены в команде и согласованы с ментором;
- Product Backlog должен находится со всеми остальными файлами проекта в репозитории;
- Время demonstration meetings должно быть согласовано с ментором. Рекомендуется проводить такие небольшие демо еженедельно;
- Еженедельные отчеты должны быть отправлены ментору, содержа в себе следующую информацию:
  - Статус (green/yellow/red) с небольшим описанием;
  - Список features в разработке;
  - Список уже разработанных features;
  - Число всех test cases готовых ко времени отправки отчета;
  - Число всех юнит-тестов, сделанных ко времени отправки отчета;

Задание

Framework для работы с графами.

Обзор

Цель этого проекта – предоставить гибкий framework для работы с графами, включая решение всех типичных задач.

Требования

Framework должен поддерживать следующий функционал:

- Загружать граф из текстового файла в виде списка смежности или матрицы смежности по выбору пользователя;
- Подсчет минимальных разрезов графа;
- Поиск в ширину;
- Поиск в глубину;
- Нахождение кратчайших путей;
- Вычисление сильных компонент;
- Топологическая сортировка;
- Минимальное остовное дерево.