**Задание 1.**

**Вениамин Кандидатов хочет стать космонавтом. Он узнал, что для этого ему необходимо пройти по ряду параметров, по каждому из которых даётся оценка**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Оценка**  **Тест** | **Хорошо** | **Удовлетворительно** | **Неудовлетворительно** |
| 1 | Вес | [75-90] кг | [70-75) кг, (90-100] кг | > 100 кг, < 70 кг |
| 2 | Рост | [170-185] см | [160-170) см, (185-190] см | > 190 см, < 160 см |
| 3 | Возраст | [25-35] лет | [23-25) лет, (35-37] лет | > 37 лет, < 23 лет |
| 4 | Зрение | 1 | - | < 1 |
| 5 | Курение | отсутствует | - | присутствует |
| 6 | Терапевт | Не больше 2 болезней | 3 болезни | Больше 3 болезней |
| 7 | Психолог | 0 болезней | 1 болезнь | Более 1 болезни |
| 8 | Тест «Вес и вредные привычки» | Все случаи, кроме того, что описано в «удовлетворительно» и «неудовлетворительно» | Не удовлетворяет критерию из «неудовлетворительно», и у кандидата есть простуда и/или вирусы, и его вес больше 110 кг | Кандидат курит, у кандидата простуда и/или вирусы, и его вес больше 120 кг или меньше 60 кг |
| 9 | Тест «Странный» | Имя кандидата начинается с буквы «П» | Не удовлетворяет критерию из «хорошо» и возраст кандидата больше 68 лет | Если не удовлетворяет критериям из «хорошо» и «удовлетворительно» |
| 10 | Тест «Математический» | Не удовлетворяет критерию из «неудовлетворительно», и рост кандидата делится нацело на 2 | Все случаи, кроме того, что описано в «хорошо» и «неудовлетворительно» | Рост делится нацело на 3, и у кандидата насморк |

\* квадратная скобка – значение включено в диапазон, круглая – не включено

Терапевт проверяет наличие следующих болезней:

* Насморк (cold)
* Бронхит (bronchitis)
* Вирусы (virus)
* Аллергия (allergy)
* Ангина (quinsy)
* Бессонница (insomnia)

Психиатр проверяет наличие следующих болезней:

* Алкоголизм (alcoholism)
* Бессонница (insomnia)
* Наркомания (narcomania)
* Травмы (injury)

**На основании тестирования принимается решение. Кандидат не проходит тест, если:**

- имеется хотя бы 1 неудовлетворительный результат

- имеется 3 или более удовлетворительных результатов

**Программа должна:**

* принять на вход информацию:
  + Имя (не пустая строка)
  + вес в кг (целое число, больше 0)
  + рост в см (целое число, больше 0)
  + Возраст (целое число, больше 0)
  + Зрение (дробное число, от 0 до 1)
  + Разделённый пробелами список вредных привычек и болезней (может быть пустым)
* Если входные данные неверны, нужно указать, какой из параметров неверный и почему
* Выдать результат:
  + Если кандидат прошел тестирование, должна показаться строка «Кандидат <имя> подходит»
  + Если кандидат не прошел тестирование, должно показаться сообщение со списком проблем кандидата в формате:  
      
    «Кандидат <имя> не прошел тестирование. Проблемы:

\* <проблема> (<удовлетворительно/неудовлетворительно>)

\* <проблема> (<удовлетворительно/неудовлетворительно>)»

Например,

«Кандидат <имя> не прошел тестирование. Проблемы:

\* Рост кандидата меньше 170 см (удовлетворительно)

\* Кандидат курит (неудовлетворительно)

\* Возраст кандидата больше 37 лет (неудовлетворительно)»

* Спросить, хотим ли мы протестировать других кандидатов. При несогласии программа должна завершать выполнение, при согласии – переходить к чтению входных данных

**Замечания:**

* Каждый тест должен являться самостоятельным (не должен зависеть от результатов предыдущих тестов и не должен влиять на результаты последующих тестов)
* Приложение может иметь любой интерфейс (консоль, WinForms, WPF, …)
* Необходимо обеспечить лёгкость изменения, удаления тестов кандидатов, а также добавления новых тестов кандидатов без изменения (или с минимальными изменениями) существующего кода
* Задача рассчитана на применение некоторых принципов ООП

**Дополнительно (не обязательно):**

* Написать unit тесты. Покрытие кода тестами должно быть не менее 70%

**ВАЖНО!!! Вы можете не тратить время на рутинный набор всех тестов кандидатов и покрытие их кода unit-тестами в случае, если ваше архитектура вашего решения действительно обеспечивает расширяемость и тестируемость, и если все указанные тесты кандидатов реализуемы в рамках архитектуры. Если данные условия выполнены, вы можете реализовать только тесты кандидата под номерами 1, 6, 9 и несколько unit-тестов для них в качестве примера.**

**Навыки:** объектно-ориентированное проектирование, SOLID, unit тесты

**Задание 2.**

Клементий Тараканов решил основать стартап «Что мы едим» - первую в мире социальную сеть для любителей рассказать о том, как они вкусно кушают. Для разработки сайта он обратился к вам. Сайт должен состоять из двух страниц:

1. На Странице 1 можно:
   1. Ввести имя (обязательное поле)
   2. Ввести email (обязательное поле)
   3. Выбрать блюдо из выпадающего списка блюд (обязательное поле)
   4. Нажать кнопку «Добавить новое блюдо»
   5. Нажать кнопку «Рассказать о том, что я ем»
2. При нажатии на кнопку «Добавить новое блюдо» появляется диалог, в котором:
   1. Можно ввести название блюда
   2. Можно нажать «Добавить»
   3. Если блюдо с таким именем уже существует, диалог не закрывается, и показывается сообщение «Это блюдо уже кто-то когда-то ел» (красным цветом)
   4. Если блюда с таким именем нет, диалог закрывается, а добавленное блюдо появляется в выпадающем списке блюд
3. При нажатии на кнопку «Рассказать о том, что я ем»:
   1. Если не все обязательные поля заполнены, рядом с незаполненным полем показывается соответствующее сообщение
   2. Если всё заполнено верно, система сохраняет данные о пользователе и том, что он ел (если пользователь с таким сочетанием **имени** и **email** не существует, его необходимо создать; нужно запоминать, когда данный пользователь съел данное блюдо), и показывается Страница 2
4. На Странице 2 показывается сообщение:

Если пользователь не был зарегистрирован в системе ранее (совпадать должны одновременно имя и email), то показывается сообщение:

«<**имя**>, мы рады приветствовать вас в нашем сообществе! Вы только что съели <**имя блюда**>! За сегодня <**имя блюда**> было съедено <**число раз, сколько тот же продукт съели за сегодня**> раз!»

Если пользователь уже существовал, то показывается сообщение:

«<**имя**>, с возвращением! Вы только что съели <**имя блюда**>! Всего вы съели <**имя блюда**> <**число раз, сколько данное блюдо было съедено данным пользователем**> раз! За сегодня <**имя блюда**> было съедено <**число раз, сколько тот же продукт съели за сегодня**> раз!»»

Под сообщением должна показываться «лента», в которой будет размещаться информация о том, кто, что и когда ел. «Лента» должна быть отсортирована по времени по убыванию. Каждое сообщение в ленте должно быть в формате «HH:mm dd/MM/yyyy <имя> (<email>) съел <имя блюда>»

Пример «ленты»:

*Также люди едят:*

*12:25 17/05/2018 Василий (*[*test@test.ru*](mailto:test@test.ru)*) съел Спаржа*

*11:55 17/05/2018 Василий (*[*test@test.ru*](mailto:test@test.ru)*) съел Сосиска*

*17:07 15/05/2018 Теодор (teodor.dh@gmail.com) съел Картошка*

**Замечания:**

* Данные должны сохраняться в БД (выберите любую удобную БД)
* Необходимо использовать Entity Framework для работы с данными БД
* Необходимо использовать платформу ASP.NET MVC 5 (или ASP.NET MVC 4)
* Можно использовать frontend фреймворки (bootstrap, jQuery, любые другие)
* Желательно ограничить число записей в «ленте» (15-20 самых «свежих» записей)

**Навыки:** ASP.NET MVC, MVC, Entity Framework, frontend разработка