Список проектов

1. Инструмент для составления словарей на английском языке

Требуется реализовать инструменты, которые бы позволяли:

- составлять документы с правильной транскрипцией и переводом выбранных преподавателем слов с помощью плагина для Google Chrome
- составлять набор слов в lingvaleo из docx с помощью консольного приложения
- обновлять существующие наборы (имя набора, состав набора) с помощью консольного приложения

Технологии: Python3, Flask, Mongodb, Docker, js

deliverables: Плагин для Google Chrome, который позволяет по клику добавлять слово в документ, ищет при этом транскрипцию слова и его перевод (или позволяет внести эти данные самостоятельно)

Консольное приложение для Linux, которое позволяет из документа docx определенного формата составить набор слов в Lingua Leo, а также обновить этот набор.

2. Онлайн-курс на Stepik по работе в Google Docs и Google Tables

Требуется создать онлайн-курс на платформе Stepik.org для работы с Google Docs и Google Tables. В качестве теории можно загружать текстовую/видео информацию. В качестве задач необходимо создать автоматически проверяемые задания, при выполнении которых пользователь должен работать в Google Docs и Google Tables, а на Stepik система проверки должна проверять правильность его работы.

Технологии: Python3, Flask, Mongodb, Docker, js

deliverables: Набор автоматически проверяемых заданий в онлайн-курсе по работе в Google Docs и Google Tables, набор теоретических степов.

3. Telegram bot для для студентов 1го курса

Требуется создать telegram bot'a для студентов 1го курса. Вопросы, которые должны быть освещены ботом (информация касается предметов Информатика и Программирование):

- “Я не защитил л.р./не сдал курсовую в срок - что мне делать?”
- “Попал на допсессию - что делать?”
- “Хочу улучшить оценку после сессии - что делать?” (вопрос касается дня качества)
- Расчет баллов по рейтинговой системе для студента/отображение информации о том, что нужно сделать на конкретную оценку
- Сроки защит
- и др.

Технологии: Python3, Flask, Mongodb, Docker, js

deliverables: Набор скриптов для сборки docker образа и запуска docker контейнера с Telegram bot'ом.

4. Автоматическая проверка содержимого пуллреквеста у студента

На данный момент есть инструмент, который позволяет после создания пуллреквеста проверить пуллреквест на соответствие определенным правилам. Требуется дополнить этот инструмент проверками содержимого пуллреквестов:

- Проверять содержимое пуллреквеста на степике: автоматически подставлять код студента в соответствующую задачу на степике и проверять, была ли эта задача успешно решена
- Закрывать пуллреквест студента с соответствующей формулировкой, если пуллреквест не прошел.

Также есть инструмент, который позволяет посмотреть статистику студента 1го курса на Stepik. Требуется дополнить этот инструмент отображением статистики:

- Показывать дату создания каждого пуллреквеста по ЛР, дату мержа пуллреквеста преподавателем
- Показывать историю пуллреквеста после нажатия соответствующей кнопки (Например, “Показать больше”)

Технологии: Python3, Flask, Mongodb, Docker, js

deliverables: Дополнения уже готовых веб-приложений, позволяющие следить за статистикой в Гитхаб и проверять содержимое пуллреквестов.

5. Инструмент выдачи заданий

Для защиты ЛР студент должен получить задание от преподавателя по соответствующей ЛР. Необходимо создать веб-приложение, которое позволяет:

- Авторизоваться через Github аккаунт как студент, или как преподаватель
- Отобразить для студента его прогресс: количество Л.Р., которые были защищены студентом, количество попыток защиты данной Л.Р.
- Получить задание к соответствующей л.р. (задание должно быть не тем же, что студент получал ранее)
- Прикрепить фотографию решенного задания/ссылку на исходный код в репозитории/исходный код
- Послать статус задания: Решено (можно проверять), Не решено
- Преподаватель должен видеть таблицу студентов: ФИО, статус каждого студента: Получил Задачу (отображать номер л.р.), Не получил, Решил, Не Решил. При этом должна быть возможность получить доп информацию о студенте при нажатии кнопки “Подробнее”: информация о всех попытках.
- Должна быть возможность отслеживать изменение фокуса приложения.
- Преподаватель может изменить статус задачи на Зачтено и Не Зачтено

- В начале пары приложение должно автоматически становиться доступным для просмотра заданий студентом, в конце пары - недоступным. Преподаватель должен иметь возможность управлять этим.

Технологии: Python3, Flask, Mongodb, Docker, js

deliverables: Веб-приложение, позволяющее студенту получить задание и узнать свой статус и позволяющее преподавателю следить за прогрессом студентов.

6. Инструмент для сбора метеорологических данных

Необходимо разработать консольное приложение, которое позволит собирать данные о погоде (температуру, сила и направление ветра, влажность, облачность) в указанном городе.

Приложение должно:

1. собирать информацию за определенный промежуток времени;
2. получать информацию с заданным шагом (почасовая, дневная, за месяц, и т.д.);
3. сохранять данные в виде таблицы;
4. при отсутствии каких-либо данных проводить интерполяцию данных;
5. создавать статистику данных (среднее за выбранный интервал, минимум и максимум, моду);
6. рисовать график погоды за выбранный период времени.

Технологии: Python3

deliverables: Консольное приложение, на вход которому поступает город, в котором необходимо собирать данные, начальная и конечная дата, и частота сбора данных. Далее приложение обращается к погодному сервису (на выбор выполняющего), который может предоставить необходимые данные, и формирует таблицу с данными и файл со статистикой по данным.

7. Визуальный редактор карт для симулятора Duckietown

Необходимо разработать веб- / кроссплатформенное GUI-приложение для создания и редактирования карт для симулятора среды Duckietown в формате yaml. Требуемая функциональность:

1. открытие, отображение и сохранение карт,
2. палитра возможных блоков,
3. отмена изменений,
4. масштабирование карты,
5. копирование фрагментов карты
6. экспорт карты в png,
7. вычисление характеристик карты (протяженность дорог, количество перекрестков, наличие тупиков …),
8. *вычисление количества необходимых материалов для того, чтобы карту воспроизвести в виде полигона (сколько нужно какой изоленты, блоков, знаков и т.д.).

Технологии: YAML, Python, Docker(в небольших количествах), Duckietown.

deliverables: приложение, позволяющее создавать / редактировать существующие карты в комфортном режиме + эмулятор нормально функционирует с такими картами.

8. Генератор карт для симулятора Duckietown

Необходимо создать сценарий командной строки, позволяющий генерировать случайные карты для симулятора Duckietown исходя из набора параметров:

1. размер поля,
2. количество перекрестков,
3. ограничение на тип перекрестков,
4. количество петель,
5. количество пешеходов на прямой участок дороги,
6. количество знаков,
7. максимальная длина дороги между перекрестками,
8. *разные схемы компоновки дорог

При этом, генерируемые карты должны быть не просто набором случайных блоков, а максимально корректными с точки зрения дорожного движения.

Также необходимо иметь возможность статической валидации (проверки корректности соответствия формату) для сгенерированных карт.

Технологии: YAML, Python, Docker(в небольших количествах), Duckietown.

deliverables: приложение командной строки, позволяющее сгенерировать корректную карту (== от которой не ломается симулятор Duckietown) по набору параметров.

9. Визуализация экспериментов для симулятора Duckietown

Необходимо разработать приложение (или расширить возможности симулятора Duckietown), которое позволит фиксировать видеозаписи проводимых экспериментов. Требования:

1. формат видео mp4,
2. эксперимент фиксируется в конфигурации вид сверху,
3. все боты нумеруются,
4. для каждого бота отображается траектория,
5. справа от поля (карты) на однотонном фоне отображаются
    1. счетчик времени модели,
    2. по каждмоу боту:
        1. текущие координаты (x y + углы ориентации) ,
        2. пройденное расстояние,
        3. количество времени за пределами разметки (== выезд за пределы проезжей части),
        4. количество времени простоя (== скорость бота == 0),
6. *экспорт всех данных симуляции из формате rosbag + прикладывать само видео в rosbag.

Технологии: Python, Docker, Duckietown, ROS.

deliverables: патч, либо отдельное приложение, позволяющее фиксировать результаты

моделирования в симуляторе Duckietown на видео.

10. Составитель оглавлений / указателей терминов по курсу в Stepik

Задача: создать приложение, которое будет автоматически составлять оглавление / алфавитный указатель для курса по Stepik. Функциональность:

- авторизация с помощью OAuth;
- получение данных онлайн-курса из Stepik;
- парсинг, стемминг и фильтрация шумовых слов для текстовых степов;
- выделение заголовков для текстовых степов;
- составление оглавления (модули, уроки, заголовки степов);
- составление алфавитного указателя терминов (список терминов - каждый термин являетяя ссылкой на степ, где он был упомянут);
- выделение звуковой дорожки видео;
- распознавание текста звуковой дорожки;
- расширение алфавитного указателя распознанным текстом;
- формирование html-страницы с оглавлением и указателем.

Технологии: Python, Stepik API.

deliverables: приложение командной строки, формирующее html оглавления и указателя.

11. Задачи на iverilog

Опираясь на книгу "Цифровая схемотехника и архитектура компьютера" необходимо разработать ряд задач на программирование электронных устройств в IVerilog.

- создание самих задач (код эталонных решений, код неверных решений, код test_bench - проверочная программа на IVerilog);
- скрипты для сборки студенческих решений,
- скрипт для запуска студенческих решений и проверки их через test_bench,
- скрипт для прогона интеграционных тестов (== протестить работу каждой задачи на ее эталонных и неверных решениях),
- (hard) рандомизация условий задач и проверок;

Задачи для реализации:

- N-входовые вентили (И, ИЛИ, XOR, NOR, NAND),
- светофор,
- двойной светофор на X-образном перекрестке,
- двойной светофор на T-образном перекрестке,
- N-входовой мультиплексор,
- N-входовой демультиплексор,
- ....

Описание самого Verilog.

Технологии: IVerilog, Docker, Python, Bash.

deliverables: набор задач (формулировка, решения, проверочный код (test bench), а также скрипты для проверки решений и прогона эталонных решений.

12. Расширение Chrome для улучшения UI github

Реализуйте расширение (расширения) для браузера Chrome, позволяющее улучшить удобство для Github.

Задачи:

- создание репозиториев для групп МОЭВМ (необходимо указать год и количество групп для каждого направления) с защищенной веткой master (т.е. веткой, в которую нельзя коммитить),
- отображение кнопки “Добавить задачу ” на странице проекта (agile-доски),
- кнопка для копирования задачи (на странице просмотра задачи),
- отображение комментариев к задачам при их просмотре на странице проекта,
- отображение дат последних коммитов всех пользователей при просмотре главной страницы репо,
- отображение наличия конфликта на иконке пулл-реквеста при просмотре в списке PR,
- пометка карточек без связанных PR на странице проекта,
- пометка карточек с конфликтом в связанном PR на странице проекта,

Технологии: github api, javascript, html, css.

deliverables: готовое к публикации расширение.

13. Инструмент для проверки человека на присутствие в БД

Необходимо разработать консольное приложение обработки видеопотока, определения лиц в кадре, и проверки лица на присутствие в БД изображений. Задачи:

- реализовать обработку потокового видео
- реализовать алгоритмы обнаружения лиц на изображении
- реализовать алгоритм сопоставления обнаруженных лиц и лиц из БД
- генерация отчета по работе

Технологии: Python3, OpenCV, Keras, TensorFlow

deliverables: Консольное приложение, которое при запуске должно получать снятое видео или потоковое видео с камеры. Во время работы определяет наличие людей из БД в кадре. В конце формирование отчета, который должен включать список лиц, которые были обнаружены и время их первого и последнего появления в кадре, количество неопознанных лиц.

14. Обфускатор кода написанного на языке C

Необходимо разработать консольное приложение для обфускации написанного кода. Информация по обфускации: http://citforum.ru/security/articles/obfus/ Приложение должно получать уже написанный код на языке C и проводить обфускацию. Должна быть настройка

степени обфускации и выбор того, какие типы обфускации используются. Технологии: Python3

deliverables: Консольное приложение? получающее файл с расширением .c и параметры обфускации. Параметрами обфускации могут быть: переименование переменных, изменение представление чисел, добавление лишних ничего не делающих строчек кода, замена константных выражений вызовом функций, а также степень обфускации (например, сколько лишних строчек добавляется, размер новых имен переменных, и.т.д.). По завершении программы программа создает новый файл с запутанным кодом.

15. Сервис по проверке документов на соответствие шаблону

Необходимо разработать web-сервис, позволяющий проверить соответствие документа и документа шаблона. разработать разметку для документа шаблона (примером такого шаблона могут быть шаблоны оформления отчетов с сайта etu.ru)

- разработать алгоритмы анализа текста и метаданных файла
- сравнение документа с шаблоном (размер шрифтов, цвет текста, отступы, нумерация страниц, и.т.д.)
- реализовать формирование отчета по итогам сравнения

Технологии: Python3, Flask, js

deliverables: Сервис должен получать 2 документа (форматов docx, doc, odt, rtf. будет плюсом если будет реализовано и для pdf) один из которых является документом-шаблоном. Далее сервис проводит сравнение первого документа на соответствие шаблону представленному во втором документе. При несоответствии должна выводиться информация о том, что именно и где не соответствует.

From:
http://se.moevm.info/ - **se.moevm.info**

Permanent link:
http://se.moevm.info/doku.php/courses:mse:project_list

Last update: **2022/12/10 09:08**