СУ “Св. Климент Охридски”,
Факултет по математика и информатика,
Катедра Информационни Технологии

Разпределена система за управление на курсове по програмиране с автоматично оценяване на решения

Здравко Иванов Гюров, РСМТ, ФН: 26379
Научен ръководител: Стоян Велев

София, 2022

Съдържание

I. Списък на съкращенията

API - Application Programming Interface
WYSIWYG - What You See Is What You Get

II. Речник на термините

Application Programming Interface - проложно-програмен интерфейс
Online Judge - онлайн система за автоматично оценяване на решения на задачи по програмиране
Java Servlet - програми, които работят върху уеб сървър, те са междинното ниво между заявките, идващи от уеб браузъра и базите от данни или други приложения на уеб-сървъра
plug-in - софтуерент компонент, който представлява “приставка”, която се инсталира в допълнение към съществуващо софтуерно приложение, за да предостави допълнителна функционалност
What You See Is What You Get - системи, при които съдържанието по време на редактиране изглежда като крайния резултат

III. Списък на фигурите

Asd

IV. Списък на таблиците

Asd

1. Увод

Обратната връзка за функционалната коректност, ефективност и придържане към конвенции за стил и добри практики е ключова за изграждането на знания и умения в курс по програмиране. Регулярната обратна връзка и особено възможността студентите сами да проверяват и подобряват итеративно решенията си, са възможни единствено чрез автоматизиране на процеса по тестване и оценяване. С увеличаването на броя практически курсове по програмиране и броя студенти през изминалите години, тези практики стават все по-наложителни за подпомагане на преподавателския състав да бъде по-полезен и да предоставя персонална помощ за разрешаване на по-сложни казуси със спечеленото време.

Целта на дипломната работа е да се проектира и реализира разпределена система за управление на курсове по програмиране с автоматично оценяване на решения.

Структура на дипломната работа

В глава **2. Обзор на подобни системи** ще разгледаме няколко системи, които покриват изискванията ни до някаква степен. След това ще ги сравним и ще съпостави техните плюсове и минуси и най-накрая ще обобщим защо те няма да ни свършват работа.

В глава **3. Проектиране на системата** ще разгледаме функционалните и нефункционалните изискванията на системата и как ще бъдат изпълнени и предоставени.

В глава **4. Реализиране на системата** ще разгледаме конкретните технологии, с които ще бъде имплементирана системата и защо точно те са избрани. Ще видим също и основните API-та и как може един потребител да работи със системата

В глава **5. Експерименти и анализ на резултатите** ще разгледаме различни тестове, които са били приложени на системата за да се симулира реална работна среда и ще направим разбор на резултатите.

В последната глава **6. Заключение** ще направим обобщение, ще видим как системата ще влезе в употреба и ще разгледаме възможни подобрения и бъдещо развитие на системата.

Дефиниции

X наричаме У

2. Обзор на подобни системи

Подобни системи

В тази глава ще разгледаме едни от най-популярните приложения и уеб услуги, които покриват възможно най-много от изискванията ни. Първо, нека определим кои са тези най-важни за нас функционални и нефункционални изисквания, които ще действат като критерии за сравнение.

1. Системата трябва да ни предоставя възможността да управляваме много на брой курсове (стотици, хиляди, а може и повече, за момента не се интересуваме от конкретен брой).
2. Всеки курс трябва да може да съдържа голямо количество задания (десетки, това отново е приблизително число).

3. Системата трябва да предоставя ясен и лесен начин за работа с предадените решения.
4. Системата трябва да дава възможност на преподавателския състав да проверява и оценява решенията по интуитивен начин.
5. Системата трябва да може да проверява решенията на студентите автоматично.
6. Системата трябва да може да улавя признаци на плагиатство у решенията на студентите.
7. Системата трябва да е гъвкава в отношение на технологиите, които се преподават в курса. Целта е да намерим една система, която може да се използва за всички ситуации, която ще е удобна за всички преподаватели и ще стане позната на студентите.
8. Системата трябва да дава свобода на лекторите сами да управляват курсовете си без намеса на администратор.
9. Системата трябва да е възможно най-достъпна финансово както за преподавателите, така и за студентите.
10. Системата трябва да е достъпна 24/7.
11. Системата трябва да е устойчива.
12. Системата трябва да е лесна за използване.

След като сме въвели тази основа, може да преминем към приложенията.
За този анализ са подбрани както системи, които са били в експлоатация дълго време и са се доказали като едни от най-добрите за времето си, така и новонавлезли системи, използващи модерни и иновативни подходи и технологии, а именно:

- Web-CAT
- codePost
- GitHub Classroom

Web-CAT

Web-CAT е гъвкава и приспособима система за оценяване, която е предназначена да обработва задания по програмиране.
Web-CAT е програма, която върви на сървър и предоставя възможностите си чрез уеб интерфейс. Всички дейности свързани с решенията на заданията, обратната връзка, разглеждане на резултатите и оценяване се извършват в уеб браузър. Всички административни дейности на инструкторите свързани с курсовете, заданията и оценяването също се извършват в уеб браузър, дори системните административни дейности се извършват по този начин.
Преди създаването на Web-CAT са съществували и други системи за автоматично оценяване, но те обикновено са се фокусирали над определяне дали изходният код на студента произвежда желания резултат. Това са така наречените Online Judges. Web-CAT оригинално е била проектирана като автоматична система за оценяване с общо предназначение, но преди да се завърши първоначалната версия, нейните автори са решили, че искат по-скоро да поддържат дейности по тестване на софтуера на студентите, отколкото да оценяват работата на студенти чрез просто сравнение на резултатите.

Основни силни страни

Сигурност (Security)
На първо място, системата е проектирана да поддържа сигурна работа. Нейният потребителки модел включва подход за удостоверяване, базиран на приставки с отворен API, така че администраторът може да избере една от няколко вградени стратегии за удостоверяване или дори да предостави персонализирана(custom). Услугите на Web-CAT са предпазени едновременно от специфични права на ниво потребител и от система за контрол на достъпа на ниво роля. Разпознаването на злонамерени студентски програми и предпазването от тях идва по подразбиране. Цялостността на данните се поддържа от политики за сигурност в системата и от услугите предоставени от релационна база данни.

Преносимост (Portability)

Web-CAT е приложение, написано на Java, което дава на кода висока степен на преносимост. То може да бъде пакетирано и разпространявано като Java Servlet приложение, което ще работи под всеки съвместим сървлет контейнер като Apache Tomcat. Когато е пакетиран като сървлет, Web-CAT може да се разпространява като единичен .war файл (сервлет уеб архив), който включва всички необходими зависимости.

Разтегливост (Extensibility)
Може би най-голямата сила на Web-CAT е вродената гъвкавост и разширяемост, вградени в неговата ахитектура. Приложението е проектирано с напълно plug-in базирана архитектура. Основни функционалности могат да бъдат добавени без промяна на код в съществуващата система, а всички съществуващи възможности на Web-CAT се реализират в няколко плъгина. Приложението е напълно неутрално по отношение на езика. Web-CAT се използва за оценяване на решения написани на Java, C++, Prolog и други. В допълнение, системата е напълно неутрална по отношение на това как се оценяват заданията и каква обратна връзка се връща на студентите.

Ръчно оценяване
Web-CAT се справя с всички автоматизирани задачи, които инструкторът иска да се извършват за да се обработват студентските решения, но приложението също има вградена поддръжка за задачите за ръчно оценяване. Преподавателският състав може да пише коментари и предложения на студентите за техните решения, може да добавя или премахва точки директно в HTML изгледа на изходния код на студента по WYSIWYG начин. Студентите се уведомяват автоматично по имейл, когато ръчното оценяване на тяхното решение е завършено, за да могат да го разгледат.

~(Edwards, Stephen. “What is Web-CAT?”)

Опит със системата

След разглеждане на документацията на Web-CAT виждаме защо тя е една от най-използваните системи от рода си. Системата е доста близко до това, което търсим, но за да направим цялостен и изчерпателен анализ, трябва също да се използва, за да се потвърди, че действително се предоставят всички тези функционалности. Също трябва да я съпоставим с нашите изисквания.

След използване на системата 8 месеца (4 като студент и 4 като асистент) като кратко обобщение може да се каже, че системата изпълнява добро количество от желаните функционалности, но определено може да се желае доста повече.
Първо, започвайки с това как изобщо един лектор може да се сдобие с това приложение, за да го използва за своя курс по програмиране. Тъй като това е приложение, а не уеб услуга, преподавателят ще трябва сам да се погрижи за сервирането на това приложение до външния свят. Алтернативно би могло да има една инстанция на това приложение за целия факултет или университет, така може да има системен администратор, който да е посветен на тази дейност. Хостването на приложението няма да бъде безплатно, а и освен административните дейности в самата система, ще има и усилия за поддържане на самото приложение работещо 24/7 дори при голямо натоварване.
Приемайки, че вече има сервирана система за използване, следваща стъпка би била регистрация на потребители, както лектори и асистенти, така и студенти. Това става ръчно, като за това отново се грижи системният администратор. Този процес е доста досаден, времеемък и склонен към грешки.
При създаване на курсове и задания не може да се пишат дълги описания или условия, така че тази дейност трябва да се извършва в друга система. След създаване на задание, потребителите виждат само името му, което може да се кликне. Това води към друга страница, в която студентите могат да си качат решението като .zip архив. Работата със zip архиви за контрол на версиите на решенията на студентите е много неудобна. Докато решават задачата, студентите постоянно правят промени по решенията си, дооправят бъгове и качват нова версия на решението си, което е напълно нормално. За една задача те могат да имат голям брой версии, което става трудно за следене. Те

трябва да знаят в коя версия в кой архив им се намира. Също какви точно промени са направени в определена версия и още много други проблеми свързани с контрол на версията на изходен код. След като си качат архива с финалната версия в системата не излиза ясно и видимо кода, който се съдържа там и за да си сигурен, че си качил правилната версия трябва да си изтеглиш решението, да го разархивираш и да го разгледаш цялостно, което е голямо неудобство. В допълнение, от гледна точка на преподавателския състав, когато някой студент иска помощ, неговото решение отново трябва да бъде свалено и разглеждано локално.
При проверка на решения се вижда така наречения WYSIWYG интерфейс, който звучи много удобен и интуитивен, но на практика бъгове в потребителският интерфейс го правят доста неудобен и труден за ползване. Доста неприятно и произволно, но често явление е да си на един клик от страница за срив.
Срещат се и сериозни технически ограничения, примерно при задания свързани с упражняване и тестване на знанията на студентите за многонишково програмиране. Изпитват се затруднения при стартиране на множество нишки.
Накратко, системата може да свърши работа, но си личи, че не използва модерни подходи и технологии и определено звучи по-добре отколкото изглежда.

codePost

CodePost е система за автоматично оценяване на решения на софтуерни задачи, която също дава възможността за коментиране на кода на студентите и предоставя различни инструменти за даване на обратна връзка. Тя е уеб услуга и е напълно безплатна за лектори в университети. CodePost се използва в момента от Бостънският университет, Принстънският университет, Университетът на Айова, Университетът Корнел и други.

Това обаче не е просто още един инструмент за оценяване. Процесът за даване на обратна връзка е изграден от нулата с една конкретна цел в предвид

и тя е да направи преподавателите отлични в това което правят, а именно да обучават следващото поколение програмисти.
Системата е бърза и лесна за използване. Тя включва комплексни и напреднали функции, които подобряват обучението и спестяват време.

Мисията на codePost е да помогне на преподавателите по компютърни науки да предоставят изключителна обратна връзка на студентите относно техните положени усилия по програмиране. Оригиналната система е разработена в Принстънския университет от екип от студенти. Тогава всички изпитвания са били провеждани и преглеждани напълно ръчно с лист и химикал. Целта е била този процес да се дигитализира.

Основни силни страни

Коментиране на код
Една от основните цели на на codePost е да въведе анотиране на кода с минимални усилия. Могат да се добавят лесни за четене коментари, които могат да са както персонализирани, така и стандартизирани рубрики. Поддържа се също и коментиране на Jupyter тетрадки и .pdf документи.

Автоматично оценяване
Друга важна мисия за създателите на codePost е да предоставят функционалност за лесно писане на тестове и изпълняването им срещу студентските решения. Идеята е ефективно да могат да се откриват грешките дори в курсове с много голям брой студенти. Преподавателският състав може да се възползват от прости тестове, които не изискват никакво писане на код. Могат също да се пишат и по-гъвкави тестове използвайки кратки скриптове. Всички тестове се изпълняват на codePost сървъри като всички основни езици за програмиране се поддържат.
CodePost е доста гъвкава и мощна система, която предоставя много различни варианти за осъществяването на самото автоматично оценяване. Тестовете могат да се изпълняват директно на сървъри, предоставени от системата, но те също могат да се изпълняват и на сървъри предоставени от потребителя. Резултатите от тестовете пък могат да се показват или като обекти или като

прости текстови файлове. При извеждането на резултатите може автоматично да се добавят или изваждат точки за всеки тест, може да се добавят обяснения за всеки тест и също ако тестът е пропаднал, може да се показват подсказки. Преподавателите, не винаги искат да показват резултатите от тестовете на студентите, примерно ако искат да ги принудят те да се постарят да си тестват самостоятелно решението максимално. Това също е възможно, както и в допълнение може да се изпълняват тестове при качване на решение, за да се предотвратят ситуации, в които решението на студента дори не се компилира. Системата позволява да се посочват лимити на количеството тестове, които могат да се изпълняват. Могат да се задават и изисквания за имената на файловете, които трябва да се качат. Друга позитивна черта е и факта, че тестовете се изпълняват изолирано, така че един неуспешен тест няма да повлияе на другите.

Отворена и опретивно съвместима
Уеб услугата предоставя доста функционалности, но все пак всички те са доста базови. Потребителите обаче могат да пишат скриптове, в които да комуникират с codePost API-то, което позволява интегриране с множество системи като Moss, GitHub, Repl.it, CodePen и други.

~(codePost)

Опит със системата

Разглеждайки документацията добиваме усещане за една доста сериозна и професионална система, която вече се е утвърдила в много от най-добрите университети в света. Модерният и информативен сайт с документацията покачва очакванията ни и за самата уеб услуга.
След използване на системата 4 месеца като асистент основните придобити впечатления са, че това е една доста съвременна система, която използва най-новите технологии на пазара за да предостави максимална сигурност, бързодействие и удобство. В сравнение с Web-CAT характеристиките, които изпъкват най-много определено са липсата на грешки и интуитивността при работа с уеб интерфейсът. Тези неща обаче далеч не значат, че codePost е

перфектен продукт. Места, в които Web-CAT се проваля, действително са подобрени тук, но пък и обратното важи за повече от една функционалност.

Първото нещо, което трябва се направи от преподавател, който иска да използва системата е да се регистрира, което става просто с имейл и парола. След това той трябва да се свърже с администратор, който поддържа системата, и да докаже, че наистина е лектор в университет, за да може да му се дадат учителски права. Чак след това ще може да създава курсове и задания. Това не е автоматичен процес и никога не се знае колко време би отнел, но все пак е разбираем за сигурността на системата.

След сдобиването с учителски профил, лектора може да създаде курс и ръчно трябва да добави всички студенти в него и да даде асистентски права на асистентите. Интерфейсът е много прост, минималистичен и красив. Това прави работата доста приятна. Има малко на брой изгледи, което намаля шанса за объркване или “изгубване”. В страничната лента на сайта има връзка към документацията и кратки видеа, които показват как се работи със сайта. Лентата присъства във всеки изглед, което е изключително удобно. С по един клик може да се разгледат всички курсове, всички задания в даден курс и предадените решения.
Студентите могат да качат своето решение като архив. Това е едно от основните неудобства. Има алтернатива да се използва GitHub за съхраняване на кода, но това е функционалност, която не идва по подразбиране, а трябва преподавателя да напише скрипт за интеграция с GitHub, който ще осъществи тегленето на изходния код на студентите. Там обаче ще се появят допълнителни проблеми свързани с видимостта на хранилищата на студентите. Те или ще трябва да са публични и лесно видими за всички или трябва да са лични, но тогава пък ще има допълнителни затруднения с изтеглянето на кода. С решения в .zip архиви идват същите проблеми, които се срещат и при Web-CAT.
Преминавайки към оценяването на решения и писането на обратна връзка за студентите, виждаме може би най-добрата функционалност на codePost, а именно уеб редактора на код, чрез който се пишат директно коментари върху кода на студента. Друга функционалност, която спестява много време и усилия

за синхронизиране на преподавателския състав са рубриките от съобщения за обратна връзка. Тя дава възможност да се създадат рубрики представляващи различни видове грешки, примерно грешки свързани с бързодействие или грешки свързани с чист код и други. След това могат да се попълват тези рубрики с конкретни съобщения и точките, които ще бъдат отнети при това нарушение. Тези съобщения ще могат да се използват от всички оценяващи заданието. Един минус на този изглед е, че не се дава възможност на студентите да теглят решенията си, а само да ги разглеждат.
Основните негативни характеристики освен работата със .zip архиви са и фактът, че codePost е уеб услуга и при срив на системата преподавателите нямат какво да направят, освен да чакат. Това е изключително неудобно ако се случи в критичен момент като край на срок за предаване на някакъв проект или домашно. Друго затруднение, с което ще се сблъскат лекторите е извличането на резултатите от тестовете, които са се изпълнили срещу решенията на студентите. Ако се използват тестови пакети, които са по-сложни от прости входно-изходни тестове, то за самото извличане и показване на резултатите ще трябва да се специфичен скрипт за тази цел. Всички тези недостатъци са помислени и избегнати в Web-CAT.

GitHub Classroom

Github Classroom е доста подобна система на вече разгледаните системи - Web-CAT и codePost. При нея целта също е да автоматизира курсовете и да позволи на преподавателите да се концентрират върху обучението. Тя позволява лесно управление и организиране на курсове. Също така предоставя възможност за следене и управление на задания в уеб интерфейсът, автоматично оценяване на решения и помагане на студентите, когато срещнат някакъв проблем и не могат сами да стигнат до добро решение. Интересното и очевидно качество на GitHub Classroom обаче е, че тя се базира на GitHub - популярен и стандартен инструмент, който се използва от всички софтуерни разработчици ежедневно. Git и GitHub автоматично решават всички проблеми, свързани с предаването на решения в .zip формат, тъй като те са създадени точно с тази цел, а именно контрол на версиите на изходен код. Ученето и

работата с тази система, ще даде и безценен опит на студентите и ще ги научи на добри практики за работния процес, заради нейната същност.

Основни силни страни

Безболезнено оценяване

GitHub спестява време на преподавателския състав като използва автоматично тестване за оценяване на задания. Тестовете се пускат при всяко качване на код и студентите могат веднага да видят как са се справили, позволявайки им бързо да направят нужните промени в кода си за да стигнат до решението. Преподавателят има специален изглед, в който лесно вижда как се справя всеки студент в курса в даден момент, колко от тестовете минават успешно за всеки един от тях и дали изобщо са качвали някакъв код изобщо.

Даване на ценна обратна връзка

Преподавателите имат правата да разглеждат изцяло GitHub хранилищата на студентите, и не само това, те дори са админи в тях. Това означава, че могат да виждат изходния код, история на качването на решения от студента и различни графики, които показват цялостната активност в това хранилище. Учителите могат да изискват конкретни промени по кода, да оставят общи коментари и дори да дават обратна връзка ред по ред.

Огромна видимост върху студентската работа

GitHub Classroom показва ясно на преподавателския състав, когато учениците забият, за да могат да се отзоват на помощ. Системата също предоставя функционалността да се създават както индивидуални задания, така и групови, което е разлика в сравнение с Web-CAT и codePost, където има само индивидуални. Историята към всяко хранилище прави проноса на всеки един от групата пределно ясен. Студенти, които показват извънредно висока активност определено ще бъдат оценени, както и тези, които не са допринесли много за решението на заданието.

Допълнителни характеристики

Системата, бидейки една от най-популярните инструменти за разработване на софтуер, която има над 73 милиона потребителя (към март 2022г.), няма никакви проблеми с мащабирането и поддържането дори на стотици студенти в един курс. Преподавателите могат спокойно да оставят автоматичното разпределяне на задания и автоматичното тестване на решенията да свършат тежката работа за тях.
GitHub Classroom също прави създаването на задания със стартов код и разпределянето им до студентите лекота.
С фокус над честността и почтеността, системата позволява на лекторите да направят хранилищата лични или да ги оставят публични, по тяхна преценка.

~ (GitHub)

Използване на системата

Watch videos and retell
GitHub Classroom е уеб услуга, с която се работи доста лесно. При наличието на опит в GitHub, учителят вече е наясно с над 50% от функционалностите, менютата и бутончетата, с които той ще трябва да работи. Системата се базира на GitHub организации, хранилища, задаване на права в тях и GitHub работни процеси, които са сравнително нови за GitHub, но са нещо доста познато като цяло в софтуерното инженерство.
За да започне да използва системата, преподавателят трябва да посети уеб страницата на GitHub Classroom, където ще му бъде поискано да се удостовери като за тази цел може да използва вече съществуващ GitHub акаунт или да си направи нов. След влизане в системата, излиза изглед, в който има бутон за създаване на нова класна стая. Преди създаването на нова стая, първо трябва да се направи нова организация, това представлява технически акаунт, чиято цел е да групира хранилища с една и съща цел и предназначение. Тази организация ще групира студентските хранилища като съответно тяхната обща черта ще е, това че принадлежат на един и същи курс или използвайки термините - класна стая. Преди да премине към създаването на класната стая, учителят има възможност да покани хора в организацията и евентуално да ги направи администратори, така те ще могат да му помагат с управляването на

курсовете. Най-накрая учителят може да продължи със създаването на класната стая. Тази операция започва с избирането на асистенти, което не е финално и винаги може да се добавят и премахват в по-късен етап. За да се завърши процеса с присъединяването към организацията или класната стая, администраторите и асистентите трябва да посетят специален линк, който трябва да им бъде разпространен от лектора.

Следваща стъпка би била да се добавят студенти към класната стая. GitHub предоставя множество интеграции с външни системи за управление на обучението, така че това може да стане по много различни начини. Някои от тях са Google Classroom, Canvas, Moodle, Sakai както и други подобни. По-прост начин за добавяне на студентите е, чрез обикновен текстов файл, като примерно може да се напишат факултетните номера на студентите (или друг идентификатор като 3 имена или имейл) по 1 на ред. Завършвайки този процес, студентите трябва да влязат в системата и после в класната стая, като тогава трябва да изберат към кой факултетен номер да се свържат. Тук се появява и проблем, защото всеки студент може да се свърже, с който и да е идентификатор и този проблем трябва да се решава ръчно.

Последното функционалност, която завършва целия процес на провеждане на курс по програмиране с автоматично оценяване е създаването на заданията. Това е процес от 3 стъпки, който включва задаване на име, крайна дата, определяне дали заданието ще е индивидуално или групово и също задаване на видимост на хранилищата. Втората стъпка е да се зададе шаблонно хранилище, което съдържа изходен код, който ще се репликира автоматично в хранилището на всеки студент. Тази стъпка не е задължителна, тогава хранилищетата ще са празни по начало. Последната трета стъпка е да се добавят тестовете, които ще се изпълняват срещу решенията на студентите. Те могат да са написани на какъвто и да е език. Тук срещаме друг огромен проблем, за да се изпълняват тези тестове като работен процес на GitHub, те трябва да са репликирани във всяко хранилище, тоест студентите ще могат както да ги гледат и да нагласят решенията си по тях без да мислят изчерпателно за проблема, така и директно да ги редактират, което напълно обезсмисля автоматичното тестване, защото след оценяването и крайния срок

на заданието, ще трябва преподавателският състав да прегледа хранилището на всеки студент за злонамерени активности.

Накратко, GitHub Classroom точно както предните 2 системи, покрива голяма част от изискваните функционалности като се справя с доста от проблемите по по-иновативен и впечатляващ начин, но и тази система има 1-2 големи недостатъка, които бихме желали да избегнем.

Сравнителен анализ

Сравнителен анализ (+/-)

Обобщение

Обобщение - защо тези няма да ни свършат работа

Да опиша всяка система, като покривам всяка една от 12-те точки отгоре и дали системата предоставя тази функционалност или не и накрая след 5те описания да направя табличка с 12те точки ясно да се вижда +/-

3. Проектиране на системата

Да има диаграми, архитектурна, дб ,флоу

4. Реализиране на системата

Технологии, защо съм ги избрал
Да опиша основните апита
Снимки на ui-a

5. Експерименти и анализ на резултатите

Asd

6. Заключение

Постигнати резултати

Проучени са системи за …

Проектирана е система за …

Реализирана е система …

Приноси(научни/научно-приложни/приложни)

Научни - свравнителен анализ на съществуващи системи

Научно-приложни - проектиране

Приложни - имплементацията

Апробация - системата ще се ползва за …

Насоки за бъдеща работа, перспективи

Използвана литература

codePost. “codePost.io.” *codePost: Autograder and code review for computer science courses*, https://codepost.io/. Accessed 24 March 2022.

Edwards, Stephen. “What is Web-CAT?” *Web-CAT*, 1 October 2020, https://web-cat.org/projects/Web-CAT/WhatIsWebCat.html. Accessed 22 March 2022.

GitHub. “Basics of setting up GitHub Classroom.” *GitHub Docs*, https://docs.github.com/en/education/manage-coursework-with-github-classroom/get-started-with-github-classroom/basics-of-setting-up-github-classroom. Accessed 24 March 2022.

GitHub. “GitHub Classroom.” *GitHub Classroom*, https://classroom.github.com/. Accessed 24 March 2022.

Google corp/org name. “Google title.” *Google website title*, 14 March 2022, https://google.com. Accessed 14 March 2022.