

ИТМО

Разработка модуля рекомендации подсказок для системы автоматизированной проверки задач по программированию

Автор: Кривоносов Егор Дмитриевич
Научный руководитель: Перл Иван Андреевич
к.т.н., доцент ф-та ПИиКТ Университета ИТМО

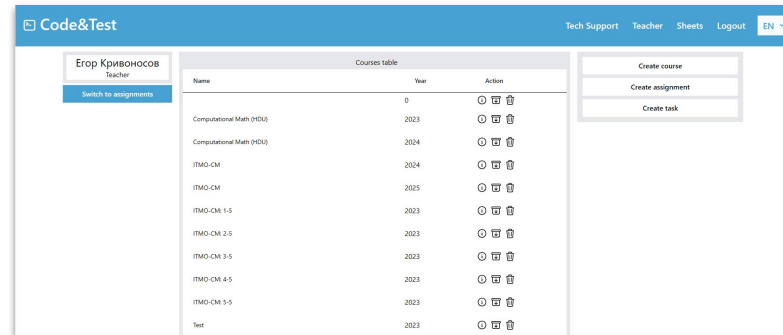
Санкт-Петербург, 2025

Что такое Code&Test?

Платформа **Code&Test** – разрабатывается в Университете ИТМО (лаборатория COSM) как открытое решение для сферы образования. **Платформа автоматизированной проверки тестовых заданий по программированию** предоставляет возможность преподавателям создавать собственные задания, а студентам – выполнять их онлайн во время учебного процесса.

Основные возможности платформы:

- создание курсов и заданий для обучения студентов программированию;
- тестирование кода студентов;
- сбор данных по выполненным заданиям для оценивания;
- проверка на антиплагиат.



Student	Course	1			2			3			4			5		
		Q1	Q2	P1	Q1	Q2	P1	Q1	Q2	P1	Q1	Q2	P1	Q1	Q2	P1
P32222	1	66			1	90		2	96		0	87.75		0	70.45	
P32312	0	99			0	60		0	96		0	87.75		0	99.63	
P32301	1															
P32301	0	75.62			0	71.36		0	96		0	87.75		0	74.92	
P32311	1	66			1											
P32312	1	66			1	61.36		2	72		1	100		2	96.4	70.45

Проблемы:



1. Студенты **1–2 курса** часто повторяют **одни и те же ошибки** из-за недостатка опыта.
2. Без своевременного обозначения ошибок **формируются неправильные привычки**, которые трудно исправить в будущем.
3. Начинающим студентам бывает сложно **самостоятельно разобраться с IT-инструментами для анализа кода** из-за сложной установки и настройки.
4. **Преподаватель не всегда может предоставить** каждому студенту **оперативную и одинаково качественную обратную связь**.

Анализ аналогов:

- Платформы, такие как **HackerRank, Codeforces, Codewars, LeetCode, CodeRun**, **не предоставляют подсказок**, ограничиваясь стандартным онлайн-judge поведением.



Решение:

- Разработка модуля для платформы Code&Test, который анализирует код студентов, выявляет ошибки и предлагает рекомендации по их исправлению с использованием больших языковых моделей.

Реальное применение похожих решений:

- Внедрение систем анализа кода, таких как CBR-Insight, позволяет получать мгновенную обратную связь без ожидания проверки третьими лицами, что актуально с точки зрения современных образовательных технологий ([Ludwig et al., 2022](#)).

- **Цель:**

- повысить качество и скорость оценки кода студентов за счёт автоматического выявления ошибок, генерации рекомендаций по их устранению и автоматизированного составления отчётов.

- **Задачи:**

- провести анализ существующих методов анализа кода (статического и динамического);
- спроектировать архитектуру модуля и определить его функциональные возможности;
- исследовать генеративные ИИ для автоматического рефакторинга и генерации отчетов;
- разработать и протестировать прототип модуля;
- интегрировать модуль в платформу Code&Test и протестировать его на реальных данных во время обучения;
- оценить эффективность модуля на основе собранных данных и обратной связи от студентов и преподавателя.





Статический анализ

- синтаксические ошибки
- нарушения стиля
- потенциальные уязвимости
- нарушения архитектурных принципов



Динамический анализ

- ошибки выполнения
- проблемы с памятью
- замедления и неэффективность
- ошибки в логике программы

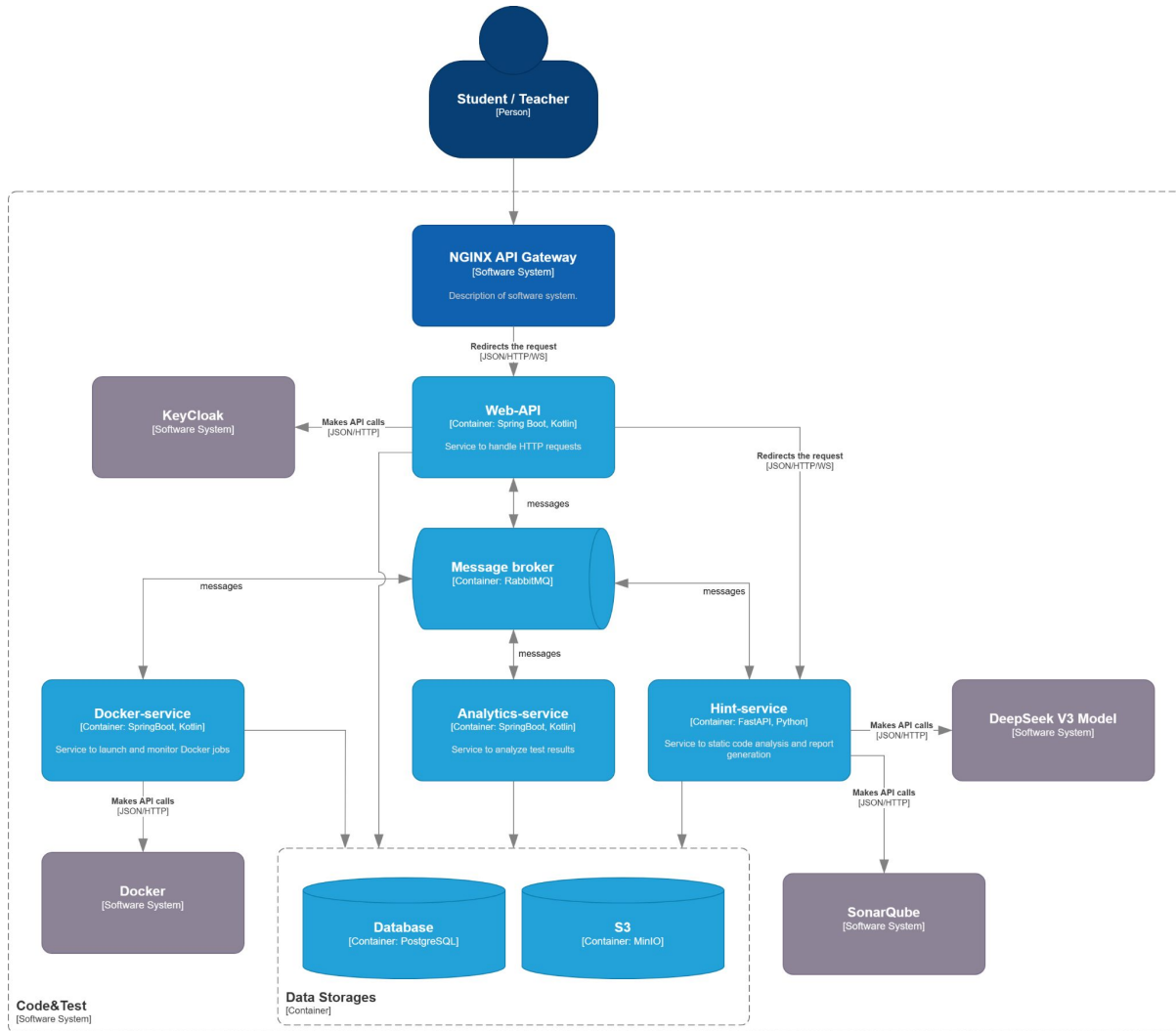


Совместное применение

- позволяет более полно выявлять **широкий спектр** ошибок
- **статика – правила, динамика – поведение**
- комбинированный подход даёт комплексную картину качества кода

[1] Ernst M. D. Static and dynamic analysis: Synergy and duality //WODA 2003: ICSE Workshop on Dynamic Analysis. – 2003. – С. 24-27.

Архитектура решения





Ключевые критерии:

- качество генерации отчетов на естественном языке (MMLU-Pro);
- результаты тестов в кодировании (MBPP + HumanEval-X);
- производительность (кол-во запросов в минуту, если API и скорость ответа);
- доступность API (в РФ) или возможность развернуть локальную модель;
- максимальное количество токенов (на ввод и вывод);
- возможность дообучения (fine-tuning) (может понадобиться в будущем).

Модели:

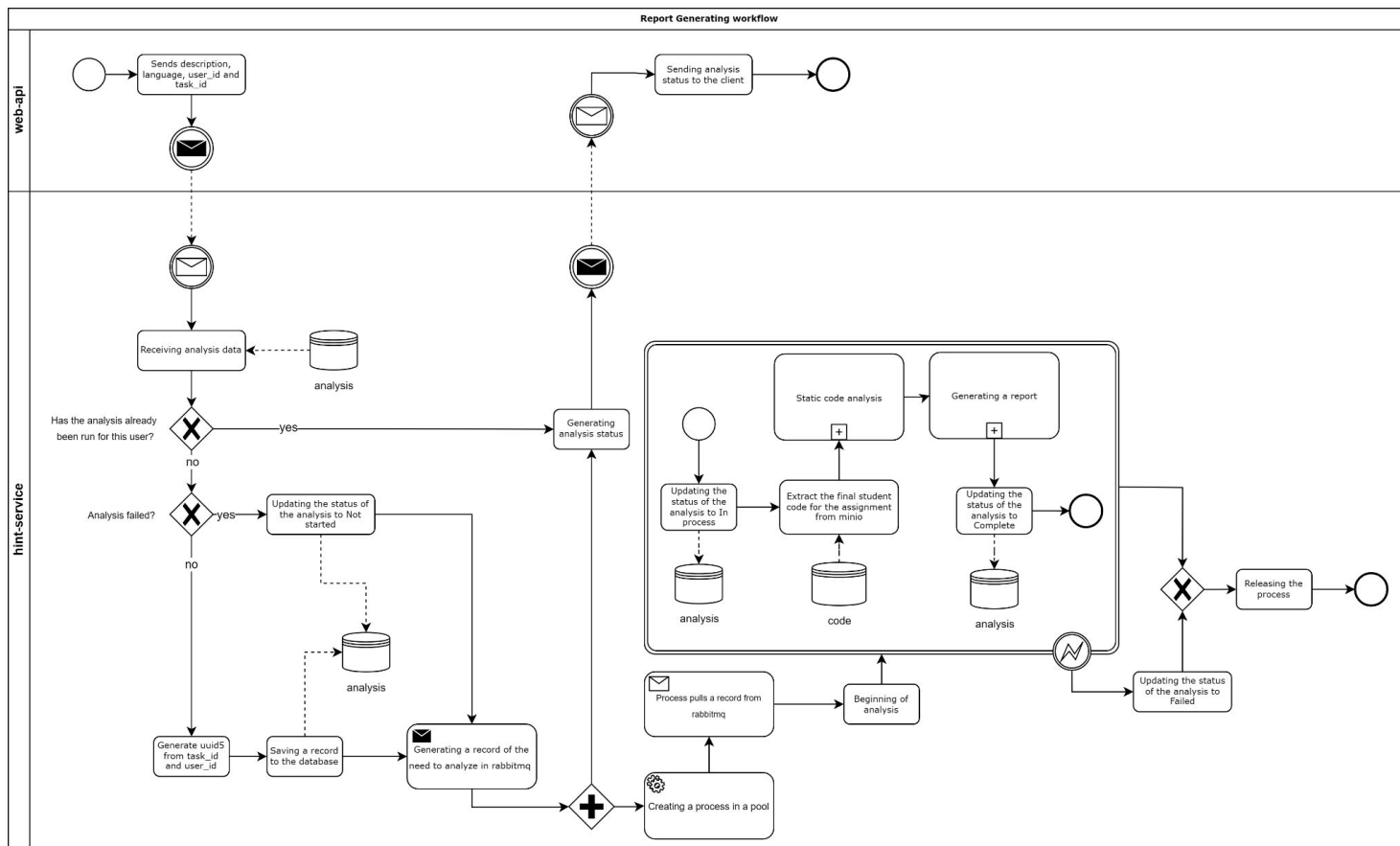
- Gemini 1.5, Gemini 2.0 (Google DeepMind);
- YandexGPT 5 (Яндекс);
- GPT-4o и младшие модели (OpenAI);
- Claude 3.5 Sonnet (Anthropic);
- Llama 3 (Meta);
- DeepSeek v3 (DeepSeek AI).

Сравнение больших языковых моделей (LLM)

<div> <div>Модель</div> <div>Параметры</div> </div>	Gemini 1.5-Flash	Gemini 2.0-Flash	YandexGPT 5	GPT-4o	Claude 3.5-Sonnet	Llama 3	DeepSeek v3
Качество генерации отчетов на естественном языке (MMLU-Pro)	64.1%	77.6%	68.0%	77.9%	78.0%	73.3%	81.3%
Результаты тестов в кодировании (MBPP + HumanEval-X)	75.6% 84.7%	82.3% 90.2%	80.2% 69.1%	87.6% 87.2%	89.4% 81.7%	82.3% 72.0%	87.6% 86.6%
Производительность (кол-во запросов в минуту, скорость ответа)	~15 RPM (бесплатно) ~1000 RPM (платно) Отклик ~13-80 сек.	~15 RPM (бесплатно) ~4000 RPM (платно) Отклик ~7-47.2 сек.	До 1000 RPM (бизнес-тарифы) Отклик ~8-20 сек.	~50 RPM (бесплатно) ~500-5000 RPM (платно, зависит от подписки) Отклик ~8-30 сек.	~20-500 RPM Отклик ~10-40 сек.	Отклик ~15 сек. (на мощном GPU)	Отклик ~15 сек.
Доступ API (в РФ) / бесплатная локальная модель	Ограниченный доступ из РФ, только через прокси; нет локальной версии	Ограниченный доступ из РФ, только через прокси; нет локальной версии	Полный доступ в РФ; бесплатный API для физических лиц	Ограниченный доступ из РФ, только через прокси; нет локальной версии	Ограниченный доступ из РФ, только через прокси; нет локальной версии	Полный доступ в РФ; есть локальная версия	Полный доступ в РФ; есть локальная версия
Максимальное количество токенов	in: ~2 097 152 out: до ~8 192	in: ~2 097 152 out: до ~32 768	in: ~32 768 out: до ~32 768	in: ~128 000 out: до ~4 096	in: ~200 000 out: до ~4 096	in: ~8 192 out: до ~8 192	in: ~128 000 out: до ~16 000
Поддержка Fine-Tuning	-	-	-	-	-	+	+



Процесс генерации отчета



Интерфейс: как это видит студент

Code&Test

CoursesTasksStudentsTop IssuesAssignments

preprod-test-username Preprod-test-user

TaskSubmitted solutionsCustom RunReport

Report on Code Analysis

Static Analysis Findings

- The code lacks a proper file header, which is a blocker issue.
- The class is not placed in a named package, which is a minor issue.
- The class contains a public constructor, which is a major issue for utility classes.
- The method `solveByPowerMethod` has a high cognitive complexity (24) and cyclomatic complexity (13), exceeding the recommended thresholds.
- The method contains multiple redundant variables and assignments, such as `prevLambda` and `unusedVar`.
- The method uses magic numbers (e.g., `388.8`, `0.25`) without clear explanation.
- The method contains unnecessary type casting and redundant calculations.
- The method lacks proper indentation and uses inconsistent brace placement.
- The method includes an empty method `foo` with no implementation, which is a critical issue.

Dynamic Analysis Findings

- Execution Time: `0.08 sec`
- Memory Usage: `38,528 KB`
- Inefficient Nested Loops: The code performs matrix-vector multiplication in a nested loop, which has $O(n^3)$ complexity and can be slow for large matrices.
- Redundant Calculations: Some calculations (e.g., repeated normalization, recomputation of `ab` vector) increase the computational load.
- Memory: The use of multiple temporary arrays (`b New`, `ab`) and lack of in-place operations lead to increased memory allocation.

Refactored Code

```
class Result {
    public static double solveByPowerMethod(int n, double[][] matrix) {
        double[] b = new double[n];
        for (int i = 0; i < n; i++) {
            b[i] = 1.0;
        }
        double lambda = 0.0;
        double prevLambda = 0.0;
        double tol = 1e-7;
        // ...
    }
}
```

Code

Java

Libs

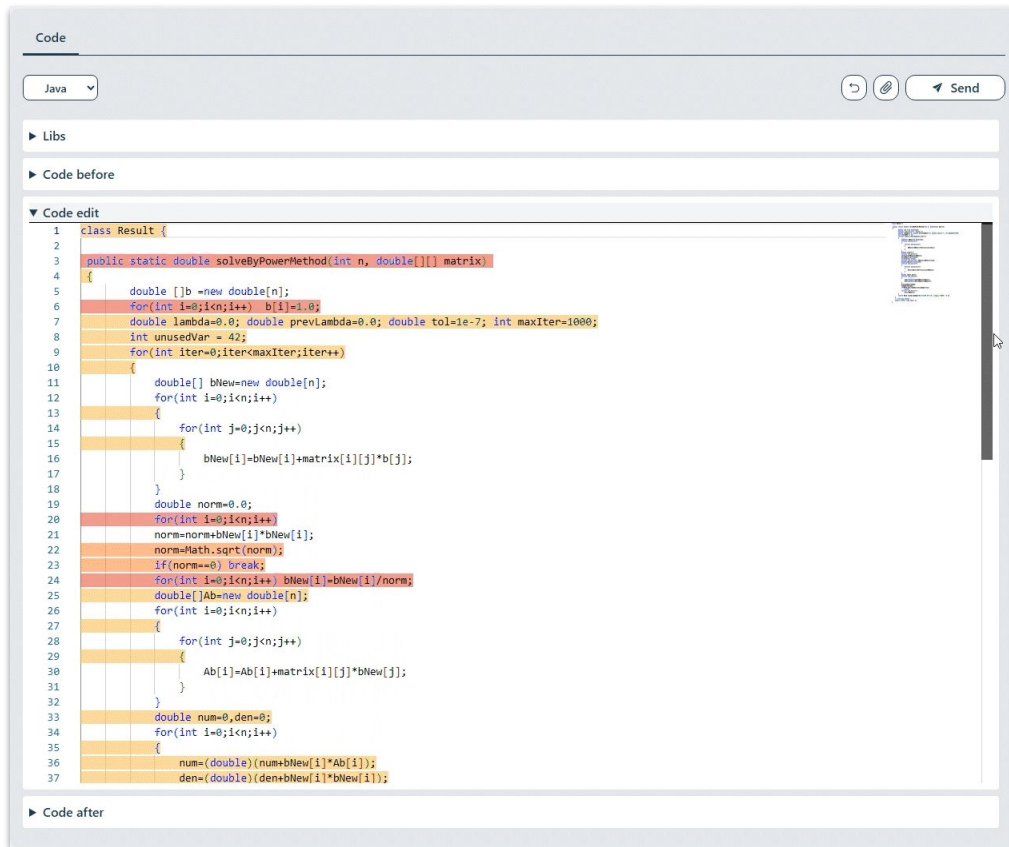
Code before

Code edit

```
1 class Result {
2
3     public static double solveByPowerMethod(int n, double[][] matrix) {
4     {
5         double[] b = new double[n];
6         for (int i = 0; i < n; i++) {
7             b[i] = 1.0;
8         }
9         double lambda = 0.0; double prevLambda = 0.0; double tol = 1e-7; int maxIter = 1000;
10        int unusedVar = 42;
11        for (int iter = 0; iter < maxIter; iter++)
12        {
13            double[] bNew = new double[n];
14            for (int i = 0; i < n; i++)
15            {
16                for (int j = 0; j < n; j++)
17                {
18                    bNew[i] = bNew[i] + matrix[i][j] * b[j];
19                }
20            }
21            double norm = 0.0;
22            for (int i = 0; i < n; i++)
23            {
24                norm += Math.pow(bNew[i], 2);
25            }
26            norm = Math.sqrt(norm);
27            if (norm == 0) break;
28            for (int i = 0; i < n; i++) bNew[i] = bNew[i] / norm;
29            double[] abNew = new double[n];
30            for (int i = 0; i < n; i++)
31            {
32                for (int j = 0; j < n; j++)
33                {
34                    abNew[i] = abNew[i] + matrix[i][j] * bNew[j];
35                }
36            }
37            double num = 0, den = 0;
38            for (int i = 0; i < n; i++)
39            {
40                num = (double) (num + bNew[i] * abNew[i]);
41                den = (double) (den + bNew[i] * bNew[i]);
42            }
43            lambda = num / den;
44            prevLambda = lambda;
45            b = bNew;
46            // ...
47        }
48    }
49 }
```

Code after

Интерфейс: как это видит студент (блок кода)



The screenshot shows a code editor window titled "Code" with a "Java" dropdown menu. Below the menu are icons for undo, redo, and a "Send" button. The editor displays a Java class named "Result" with a static method "solveByPowerMethod". The code is annotated with various markers: yellow boxes for minor style issues, orange boxes for quality or performance problems, and red boxes for serious errors. The code includes nested loops for matrix multiplication and normalization, and a final calculation of a determinant-like value.

```
1 class Result {
2
3     public static double solveByPowerMethod(int n, double[][] matrix)
4     {
5         double []b =new double[n];
6         for(int i=0;i<n;i++) b[i]=1.0;
7         double lambda=0.0; double prevLambda=0.0; double tol=1e-7; int maxIter=1000;
8         int unusedVar = 42;
9         for(int iter=0;iter<maxIter;iter++)
10        {
11            double[] bNew=new double[n];
12            for(int i=0;i<n;i++)
13            {
14                for(int j=0;j<n;j++)
15                {
16                    bNew[i]=bNew[i]+matrix[i][j]*b[j];
17                }
18            }
19            double norm=0.0;
20            for(int i=0;i<n;i++)
21            {
22                norm=norm+bNew[i]*bNew[i];
23                if(norm==0) break;
24            }
25            for(int i=0;i<n;i++) bNew[i]=bNew[i]/norm;
26            double[] Ab=new double[n];
27            for(int i=0;i<n;i++)
28            {
29                for(int j=0;j<n;j++)
30                {
31                    Ab[i]=Ab[i]+matrix[i][j]*bNew[j];
32                }
33            }
34            double num=0,den=0;
35            for(int i=0;i<n;i++)
36            {
37                num+=(double)(num+bNew[i]*Ab[i]);
38                den+=(double)(den+bNew[i]*bNew[i]);
39            }
40        }
41    }
42 }
```



– незначительные замечания, которые влияют только на читаемость или оформление кода (например, форматирование, имена переменных).



– ошибки, которые могут привести к ухудшению качества кода, его поддерживаемости или производительности (например, плохие практики, нарушение стиля, неэффективный код).



– серьезные проблемы, которые могут привести к ошибкам или сбоям, но не останавливают работу программы прямо сейчас (например, потенциальные уязвимости).



– критические ошибки, влияющие на работоспособность приложения (например, возможные баги, которые вызывают сбои или утечки данных).

Code&Test | Courses Tasks Students **Top Issues** Assignments preprod-test-username Preprod-test-user

Common Student Issues

Java

Clear Table

Rule Name	Rule Code	Count ↑
An open curly brace should be located at the end of a line	java:S1105	7
Magic numbers should not be used	java:S109	5
Control structures should use curly braces	java:S121	5

Introduction

Control structures are code statements that impact the program's control flow (e.g., if statements, for loops, etc.)

Root Cause

While not technically incorrect, the omission of curly braces can be misleading and may lead to the introduction of errors during maintenance.

In the following example, the two calls seem to be attached to the `if` statement, but only the first one is, and `checkSomething` will always be executed:

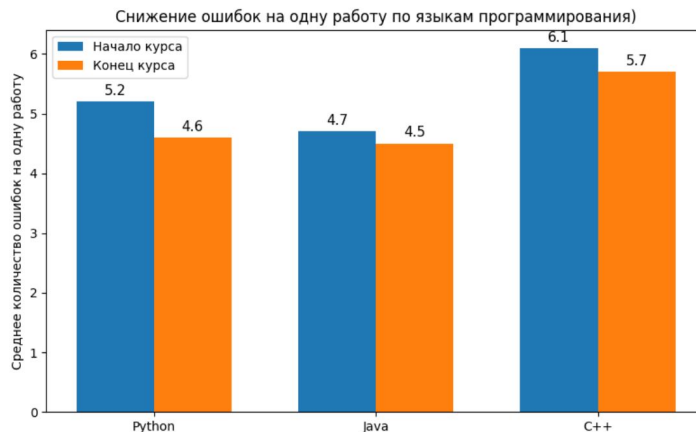
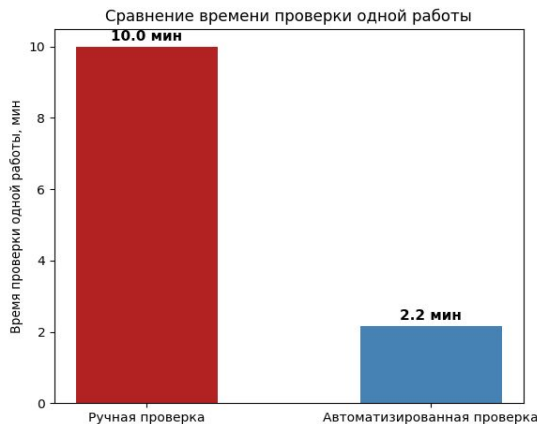
```
if (condition) // Noncompliant
    executeSomething();
    checkSomething();
```

Adding curly braces improves the code readability and its robustness:

```
if (condition) {
    executeSomething();
    checkSomething();
}
```

Результаты тестирования и обратная связь

- Время проверки одной работы **сократилось с 10 минут (ручная) до 2 минут 10 секунд (автоматизированная)**.
- После интеграции модуля **снизилось среднее количество ошибок** в последующих работах студентов (в среднем на 7-11%).
- Модуль **поддерживает 11 языков программирования** и **формирует отчёты на трёх естественных языках:** русском, английском и китайском.
- Преподаватели отметили **снижение нагрузки и повышение объективности оценки решений**, а также высказали пожелание добавить возможность включать / выключать статический анализ до отправки на проверку в рамках отдельных заданий.
- Студенты отмечают улучшение интерфейса и полезность подробных рекомендаций по их коду.





- Проведен анализ существующих методов статического и динамического анализа кода, а также аналогов образовательных платформ.
- Проведено исследование и возможности интеграции генеративной языковой модели.
- Спроектирована архитектура и реализован модуль.
- Модуль интегрирован в платформу Code&Test, протестирован на реальных данных.

Цель была достигнута и все задачи выполнены, что подтвердилось при тестировании и полученной обратной связи от студентов и преподавателей.

Возможное развитие модуля:

- внедрение дообучения языковой модели (fine-tuning) на реальных студенческих данных для повышения точности и индивидуализации рекомендаций;
- формирование индивидуальной траекторий исправления ошибок с учётом истории попыток конкретного пользователя;
- для преподавателей — автоматическое формированием тем для разбора на занятиях.

Кривоносов Е.Д. (науч. рук. Перл И.А.) Исследование возможности применения LLM для генерации персонализированных подсказок на платформе Code&Test // Сборник тезисов докладов конгресса молодых ученых. Электронное издание. – СПб: Университет ИТМО, [2025].
URL: <https://kmu.itmo.ru/digests/article/15675>

