Приложение 4
к образовательной программе

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.В.ДВ.02.01 Большие языковые модели
(индекс, наименование дисциплины в соответствии с учебным планом)

09.04.03 Прикладная информатика
(код, наименование направления подготовки/специальности)

Корпоративные информационные системы
(наименование образовательной программы)

Магистр
(квалификация)

Очная форма обучения
(форма обучения)

Год набора – 2026
Донецк

Автор(ы)-составитель(и) РПД:

Литвак Елена Геннадиевна, канд. экон. наук, доцент кафедры информационных технологий

Заведующий кафедрой:

Брадул Наталья Валерьевна, канд. физ.-мат. наук, заведующий кафедрой информационных технологий

Рабочая программа дисциплины Б1.В.ДВ.02.01 Большие языковые модели одобрена на заседании кафедры информационных технологий факультета государственной службы и управления Донецкого филиала РАНХиГС.

Протокол № 7 от «05» марта 2026 г.

СОДЕРЖАНИЕ

1. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

Дисциплина Б1.В.ДВ.02.01 Большие языковые модели обеспечивает формирование у обучающихся следующих общепрофессиональных компетенций*:

ОТФ /ТФ и реквизиты ПС *(при наличии)* **	**Код компетенции ****	**Наименование Компетенции ****	**Код индикатора достижения компетенций ****	**Наименование индикатора достижения компетенций ****	**Образовательный результат ****
-	ПК-1	Способен проектировать и разрабатывать дизайн ИС в рамках выполнения работ и управления работами по созданию (модификации) и сопровождению ИС	ПК-1.1	Разрабатывает структуру программного кода ИС в рамках выполнения работ и управления работами по созданию (модификации) и сопровождению ИС	ПК-1.1. З-6 Знает Современные структурные языки программирования ПК-1.1. У-1 Умеет Кодировать на языках программирования в рамках выполнения работ и управления работами по созданию (модификации) и сопровождению ИС

** Дисциплина может формировать компетенцию полностью или частично.*
***Должно соответствовать Приложению 1 к образовательной программе*

2. Объем и место дисциплины в структуре образовательной программы

Общий объем дисциплины:

2,00 з.е., 72 ак.час

Контактная работа обучающихся с преподавателем по видам учебных занятий: 26 ак. час на контактную работу с преподавателем, из них 8 ак. час на лекции и 18 ак. час на практические занятия. 42 ак. час на самостоятельную работу обучающихся.

Б1.В.ДВ.02.01 Большие языковые модели реализуется на 3-м семестре 2-го курса после изучения дисциплин:

- Тестирование ИТ-систем.

.

3. Содержание и структура дисциплины

3.1. Структура дисциплины

Очнаяформа обучения

№ п/п	Наименование тем и (или) разделов		Объем дисциплины, ак.час												Форма текущего контроля успеваемости, промежуточной аттестации
		ВСЕГО	Контактная работа обучающихся с преподавателем по видам учебных занятий									Самостоятельная работа			
			Период теоретического обучения						Период промежуточной аттестации (сессия)						
			Занятия лекционного типа		Занятия семинарского типа		ИК	КСР	КЭ	Каттэк	Контроль	СРкр	СРэк	СР	
			Л	ВЛ	ЛР	ПЗ									
РАЗДЕЛ 1. БОЛЬШИЕ ЯЗЫКОВЫЕ МОДЕЛИ															
Тема 1	LLM как ассистент разработчика: генерация, объяснение, рефакторинг кода	9	2	0	0	2	0	0	0	0	0	0	0	5	Контрольные вопросы, практические занятия, КТ1
Тема 2	Как LLM «понимает»	9	2	0	0	2	0	0	0	0	0	0	0	5	Контрольные вопросы,

	структурные языки: токенизация кода и контекст														практические занятия, КТ 1
Тема 3	Инжиниринг промптов для кодинга: как получить работающий код с первого раза	9	2	0	0	2	0	0	0	0	0	0	0	5	Контрольные вопросы, практические занятия, КТ 1
Тема 4.	LLM для генерации структуры классов и модулей	9	2	0	0	2	0	0	0	0	0	0	0	5	Контрольные вопросы, практические занятия, КТ 1
Тема 5.	Ограничения LLM в кодинге: галлюцинации, несуществующие библиотеки, уязвимости	9	2	0	0	2	0	0	0	0	0	0	0	5	Контрольные вопросы, практические занятия, КТ 2
Тема 6.	LLM для code review и поиска багов	9	2	0	0	2	0	0	0	0	0	0	0	5	Контрольные вопросы, практические занятия, КТ 2
Тема 7.	Генерация тестов (unittest/pytest для Python,	11	2	0	0	03	0	0	0	0	0	0	0	6	Контрольные вопросы, практические занятия, КТ 2

	PHPUnit для PHP).													
Тема 8.	Этические и практические ограничения: копирайт на код, безопасность, слепое копирование	11	2	0	0	3	0	0	0	0	0	0	6	Контрольные вопросы, практические занятия, КТ 2
Промежуточная аттестация		4	0	0	0	0	0	0	0	4	0		0	0	Зачет
Итого		**72**	**8**	**0**	**0**	**18**	**0**	**0**	**0**	**4**	**0**		**0**	**42**	

Используемые сокращения:

Л – лекции - занятия, предусматривающие преимущественную передачу учебной информации обучающимся педагогическими работниками организации и (или) лицами, привлекаемыми организацией к реализации образовательных программ на иных условиях,).

ВЛ – видео лекции.

ЛР – лабораторные работы.

ПЗ – практические занятия (за исключением лабораторных работ).

ИК – индивидуальные консультации.

КСР – контроль самостоятельной работы

КЭ – консультации перед экзаменом

Каттэк – контактная работа на аттестацию в период экзаменационных сессий

Контроль - контактная работа на аттестацию в период экзаменационных сессий для заочной формы обучения

СРкр – самостоятельная работа на подготовку курсовой работы/ курсового проекта.

СРэк – самостоятельная работа на подготовку к экзамену.

СР – самостоятельная работа в семестре на подготовку к учебным занятиям.

3.2. Содержание дисциплины

Тема 1. LLM как ассистент разработчика: генерация, объяснение, рефакторинг кода. ПК-1.1.

Содержание лекций:

Рассматривается понятие большой языковой модели как инструмента, дополняющего разработчика, а не заменяющего его. Демонстрируются основные сценарии использования LLM при разработке программного обеспечения: написание кода по словесному описанию, объяснение смысла готового фрагмента кода, поиск синтаксических и логических ошибок, рефакторинг. Отдельно акцентируется, что LLM эффективнее всего работает с распространёнными структурными языками, включая Python и PHP, поскольку они хорошо представлены в обучающих данных. В конце лекции формулируется главный навык: умение ставить задачу перед LLM так, чтобы результат требовал минимальной ручной доработки.

Практические занятия:

Обучающиеся получают несколько небольших заданий на генерацию кода на Python и PHP: например, «напиши функцию, которая проверяет, является ли строка корректным email-адресом» или «напиши скрипт PHP для чтения CSV-файла и вывода данных в HTML-таблицу». На каждом примере обучающиеся сначала самостоятельно формулируют промпт, затем выполняют сгенерированный код, фиксируют ошибки (синтаксические, логические, вызов несуществующих функций). После этого они дают LLM уточняющий промпт для исправления ошибок и сравнивают результаты двух итераций. В завершение Обучающиеся выполняют простой рефакторинг: LLM предлагает улучшить структуру «плохого» кода (например, вынести повторяющиеся фрагменты в функцию), а студент применяет предложенные изменения и проверяет работоспособность. Отчёт по практике содержит все использованные промпты, сгенерированный код и комментарии о том, что пришлось дорабатывать вручную.

Тема 2. Как LLM «понимает» структурные языки: токенизация кода и контекст. ПК-1.1.

Содержание лекций:

Объясняется, что LLM не «исполняет» код, а обрабатывает его как последовательность токенов – атомарных фрагментов текста. Показывается, как токенизируются ключевые слова (`def`, `class`, `function`, `if`), операторы (`=`, `==`, `=>`), разделители (`;`, `:`, `{}`), а также значимые пробелы в Python (отступы как специальные токены) и знак `$` в PHP. Разбирается, почему LLM может генерировать синтаксически корректный код, но ошибаться в структуре (например, нарушать отступы в Python или забывать точки с запятой в PHP). Формулируется практический вывод: для надёжной генерации важно явно указывать версию языка и избегать смешения синтаксисов.

Практические занятия:

Обучающиеся через любой инструмент токенизации (например, Jupyter с библиотекой `transformers` или веб-демо токенизатора GPT) получают токены

для одного и того же фрагмента кода на Python и PHP. Задание: сравнить, как токенизируются отступы в Python (появление специальных токенов `indent`/`dedent`) и фигурные скобки в PHP. Затем Обучающиеся намеренно дают LLM промпт с некорректным синтаксисом (например, просят сгенерировать Python-функцию, но в промпте используют PHP-конструкции) и анализируют, как LLM реагирует – исправляет синтаксис или галлюцинирует. В завершение практики Обучающиеся формулируют правило: «всегда явно указывай язык и критически проверяй отступы/точки с запятой».

Тема 3. Инжиниринг промптов для кодинга: как получить работающий код с первого раза. ПК-1.1.

Содержание лекций:

Излагаются эмпирические правила составления промптов для генерации программного кода. Во-первых, промпт должен содержать: язык и версию (Python 3.11, PHP 8.2), ожидаемый формат входа и выхода, ограничения (например, «не использовать внешние библиотеки, кроме стандартных»). Во-вторых, полезно добавить пример использования (one-shot или few-shot). В-третьих, для сложных задач эффективна стратегия «цепочки мыслей» (chain-of-thought): сначала попросить LLM написать план или псевдокод, а затем – реализацию. Также обсуждаются типичные ошибки: слишком общий промпт («напиши парсер»), противоречивые требования, отсутствие спецификации ошибок.

Практические занятия:

Обучающиеся получают три задания возрастающей сложности: генерация функции сложения (простой промпт), генерация класса с валидацией (средний), генерация PHP-скрипта для работы с формой и базой данных (сложный). Для каждого задания Обучающиеся сначала дают «плохой» (короткий, неспецифичный) промпт, фиксируют ошибки в выводе LLM, затем улучшают промпт по лекционным правилам и добиваются корректного кода. В конце практики проводится мини-соревнование: кто за минимальное количество уточнений получит рабочий код для задачи «парсинг логов Apache на Python с выводом топ-10 IP».

Тема 4. LLM для генерации структуры классов и модулей. ПК-1.1.

Содержание лекций:

Обсуждается, как LLM может помочь на этапе проектирования программной системы: по словесному описанию требований сгенерировать каркас (скелет) будущего кода – иерархию классов, интерфейсы, сигнатуры методов, зависимости между модулями. Разбирается пример для Python: «спроектируй систему управления пользователями с ролями Admin и User» – LLM генерирует базовый класс `User`, наследников, абстрактные методы. Для PHP аналогично: генерация классов с типами аргументов, возвращаемыми значениями и пространствами имён (`namespace`). Акцентируется, что LLM не заменяет архитектора: сгенерированную структуру необходимо анализировать на предмет избыточности, отсутствия связей и нарушения принципов SOLID.

Практические занятия:

Обучающиеся получают текстовое описание доменной области (например: «интернет-магазин с товарами, корзиной, заказами и скидочной системой»). С помощью LLM они генерируют структуру классов на Python и PHP: список классов, их атрибуты, методы, предполагаемые связи. Затем Обучающиеся рисуют диаграмму классов (можно просто в текстовом виде) и анализируют её: выявляют дублирование, отсутствие необходимых методов, неправильные типы связей (агрегация вместо композиции). После этого они дают LLM уточняющий промпт («добавь интерфейс для скидочной системы») и получают улучшенную структуру. Финальным действием Обучающиеся генерируют код-заготовку (пустые классы с сигнатурами методов) и убеждаются, что она компилируется/синтаксически корректна.

Тема 5. Ограничения LLM в кодинге: галлюцинации, несуществующие библиотеки, уязвимости. ПК-1.1.

Содержание лекций:

Систематизируются типичные «опасности» использования LLM при генерации кода: выдумывание несуществующих функций и методов (галлюцинации), использование устаревших или удалённых версий библиотек, игнорирование вопросов безопасности (SQL-инъекции, отсутствие валидации ввода, хардкод паролей). На примерах Python и PHP показывается, как LLM может предложить функцию `php_special_decode()` (не существует) или использовать `eval()` в PHP, что ведёт к уязвимостям. Формулируется правило: сгенерированный код всегда должен быть проверен на безопасность и соответствие актуальной документации, особенно если LLM предлагает экзотическую библиотеку или нестандартный подход.

Практические занятия:

Обучающимся раздаются «опасные» промпты, которые провоцируют LLM на галлюцинации: например, «напиши функцию на Python для работы с форматом XLSX, используя библиотеку openpyxl_extra» (не существует) или «реализуй авторизацию на PHP через mysql_query без экранирования». Обучающиеся выполняют генерацию, фиксируют, какие несуществующие функции или уязвимые конструкции предложила LLM. Затем они самостоятельно исправляют код: заменяют выдуманные библиотеки на реальные (openpyxl), добавляют параметризованные запросы (PDO в PHP). В отчёте необходимо объяснить, почему исходный вариант был опасен или некорректен, и показать исправленную версию.

Тема 6. LLM для code review и поиска багов. ПК-1.1.

Содержание лекций:

Рассматривается обратная задача: не генерация, а анализ уже написанного кода. LLM можно использовать как автоматического ревьюера, который указывает на потенциальные баги, стилистические ошибки (нарушение PEP 8 в Python или PSR в PHP), запахи кода (code smells), избыточную сложность. Демонстрируется, как сформулировать промпт для ревью: «проанализируй следующий код, укажи на потенциальные ошибки,

проблемы с производительностью, нарушения стандартов кодирования, предложи улучшения». Отмечается, что LLM хорошо находит логические ошибки (например, `if a = b` вместо `a == b`), но может пропускать ошибки, требующие выполнения кода (побочные эффекты, состояние гонки).

Практические занятия:

Обучающиеся получают заведомо «плохой» код на Python (с нарушением PEP 8, неправильной обработкой исключений, потенциальным ReferenceError) и на PHP (с уязвимостью XSS, устаревшим синтаксисом, неиспользуемыми переменными). Задача: передать каждый фрагмент LLM с промптом для code review, зафиксировать все замечания, которые выдала модель. Затем Обучающиеся по этим замечаниям исправляют код и повторно показывают его LLM для проверки. В завершение практики Обучающиеся выполняют peer-review внутри группы: один студент генерирует код с помощью LLM, другой даёт его на ревью той же LLM и сравнивает результаты.

Тема 7. Генерация тестов (unittest/pytest для Python, PHPUnit для PHP). ПК-1.1.

Содержание лекций:

Обсуждается, как LLM помогает автоматизировать создание модульных тестов, что особенно важно при разработке структуры ИС: наличие тестов повышает доверие к коду и облегчает рефакторинг. Демонстрируются промпты для генерации тестов: «напиши тесты для этого класса на Python с использованием pytest и parametrize» или «сгенерируй PHPUnit-тесты для метода calculateDiscount». Разбираются ограничения: LLM может генерировать тесты, которые проходят только на «правильном» пути (happy path) и игнорируют краевые случаи; она также может использовать предположения о внутреннем устройстве, которые не соответствуют реальному коду.

Практические занятия:

Обучающиеся берут один из классов, сгенерированных в теме 4 (например, класс `User` или `Cart`), и дают LLM задание сгенерировать для него тесты. На Python используется `pytest`, на PHP – `PHPUnit`. Обучающиеся запускают тесты и анализируют покрытие: какие строки кода не были проверены, какие краевые случаи отсутствуют (например, передача отрицательного числа, пустой строки, `null`). Затем Обучающиеся вручную дописывают недостающие тесты и повторно запускают их. В отчёте указывается, насколько LLM справилась с генерацией (какой процент тестов оказался корректным), и приводятся примеры самостоятельно написанных тестов для краевых случаев.

Тема 8. Этические и практические ограничения: копирайт на код, безопасность, слепое копирование. ПК-1.1.

Содержание лекций:

Заключительная лекция посвящена юридическим и организационным рамкам. Разбирается вопрос о копирайте на код, сгенерированный LLM: большинство судебных практик (на момент 2026 года) сходятся к тому, что

пользователь должен проверять код на наличие прямых копий проприетарного кода из обучающей выборки, а также указывать факт использования LLM в документации к проекту. Обсуждаются риски «слепого копирования»: интеграция небезопасного кода в продуктивную среду, потеря права на лицензию (например, если LLM сгенерировала код с GPL-включениями). Также поднимается тема защиты персональных данных: нельзя передавать LLM пароли, ключи API, персональные данные клиентов. Формулируется итоговый вывод: LLM – мощный, но требующий контроля инструмент, отвечает за финальный код разработчик.

Практические занятия:

Обучающиеся анализируют три кейса. Кейс 1: LLM сгенерировала функцию, подозрительно похожую на фрагмент из известной библиотеки – студент должен проверить через поиск в интернете и оценить риск нарушения копирайта. Кейс 2: LLM предлагает код, который передаёт API-ключ в открытом виде или логирует данные пользователя – студент исправляет код и формулирует правило безопасности. Кейс 3: обучающемуся даётся задача, в которой использование LLM запрещено внутренним регламентом компании (например, для кода, работающего с платёжными данными) – студент должен аргументированно объяснить, почему в этом случае нельзя полагаться на LLM. Зачётное задание (начатое на предыдущей практике) завершается: студент сдаёт полный отчёт по проекту ИС, в котором отдельным разделом идёт этическая и безопасностная экспертиза сгенерированного кода.

4. Типы оценочных материалов, показатели и критерии оценивания

4.1. Оценочные материалы по дисциплине Б1.В.ДВ.02.01 Большие языковые модели входят в состав оценочных материалов по образовательной программе. Совокупность оценочных материалов по всем дисциплинам (модулям) образовательной программы составляет фонд оценочных средств (далее – ФОС). ФОС используется при проведении текущего контроля успеваемости и промежуточной аттестации обучающихся с целью оценивания достижения обучающимися планируемых результатов обучения.

4.2. ФОС разработан как комплекс проверочных заданий различного типа и уровня сложности, включает критерии и шкалы оценивания, а также «ключи» правильных ответов. ФОС формируется как отдельный документ и хранится в электронном виде, доступ к ФОС предоставлен ограниченному кругу лиц.

4.3. Для самостоятельной работы обучающихся при подготовке к текущему контролю успеваемости и промежуточной аттестации в рабочих программах дисциплин размещены типовые проверочные задания, которые можно условно разделить на задания закрытого, комбинированного и открытого типов.

Задания закрытого типа – это тестовые задания, в которых каждый

вопрос сопровождается готовыми вариантами ответов, из которых необходимо выбрать один или несколько правильных.

Задания комбинированного типа – это тестовые задания, в которых каждый вопрос сопровождается готовыми вариантами ответов, из которых необходимо выбрать один или несколько правильных и обосновать свой выбор.

Задания открытого типа – это задания, в которых на каждый вопрос должен быть предложен развернутый обоснованный ответ.

В зависимости от типа задания рекомендованы определенная последовательность выполнения и система оценивания выполнения заданий.

4.4. Типы заданий, сценарии выполнения, критерии оценивания

ТИП ЗАДАНИЯ	ИНСТРУКЦИЯ	СЦЕНАРИИ ВЫПОЛНЕНИЯ	КРИТЕРИИ ОЦЕНИВАНИЯ
Задание закрытого типа с выбором одного правильного ответа из нескольких вариантов предложенных	Прочитайте текст, выберите правильный ответ	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидается только один из предложенных вариантов. 2. Внимательно прочитать предложенные вариант-ты ответа. 3. Выбрать один верный ответ. 4. Записать только номер (или букву) выбранного варианта ответа (например, 3 или В).	Ответ считается верным, если правильно указана цифра или буква
Задание закрытого типа на установление соответствия	Прочитайте текст и установите соответствие	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидаются пары элементов. 2. Внимательно прочитать оба списка: список 1 – вопросы, утверждения, факты, понятия и т.д.; список 2 – утверждения, свойства объектов и т.д. 3. Сопоставить элементы списка 1 с элементами списка 2, сформировать пары элементов. 4. Записать попарно буквы и цифры (в зависимости от задания) вариантов ответа (например, А1 или Б4).	Ответ считается верным, если правильно указаны цифры или буквы

Задание закрытого типа с выбором нескольких правильных ответов из нескольких вариантов предложенных	Прочитайте текст, выберите правильные ответы	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидается несколько правильных ответов из предложенных вариантов. 2. Внимательно прочитать предложенные вариант-ты ответа. 3. Выбрать несколько правильных ответов. 4. Записать только номера (или буквы) выбранного варианта ответа (например, 1 4 или А Г).	Ответ считается верным, если правильно установлены все соответствия (позиции из одного столбца верно сопоставлены с позициями другого)

Задание закрытого типа на установление последовательности	Прочитайте текст и установите последовательность	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидается последовательность элементов. 2. Внимательно прочитать предложенные варианты ответа. 3. Построить верную последовательность из предложенных элементов. 4. Записать буквы/цифры (в зависимости от задания) вариантов ответа в нужной последовательности (например, БВА или 135).	Ответ считается верным, если правильно указана вся последовательность цифр
Задание комбинированного типа с выбором одного правильного ответа из предложенных и обоснованием выбора	Прочитайте текст, выберите правильный ответ и запишите аргументы, обосновывающие выбор ответа	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидается только один из предложенных вариантов. 2. Внимательно прочитать предложенные варианты ответа. 3. Выбрать один верный ответ. 4. Записать только номер (или букву) выбранного варианта ответа. 5. Записать аргументы, обосновывающие выбор ответа (например, 4 текст обоснования).	Ответ считается верным, если правильно указана цифра или буква и приведены корректные аргументы, используемые при выборе ответа

Задание открытого типа с развернутым ответом	Прочитайте текст и запишите развернутый обоснованный ответ	1. Внимательно прочитать текст задания и понять суть вопроса. 2. Продумать логику и полноту ответа. 3. Записать ответ, используя четкие компактные формулировки. 4. В случае расчетной задачи, записать решение и ответ	Ответ считается верным: 1. Отсутствие фактических ошибок. 2. Раскрытие объема используемых понятий (полнота ответа). 3. Обоснованность ответа (наличие аргументов). 4. Логическая последовательность излагаемого материала.

4.5. Общая шкала оценивания результатов текущего контроля успеваемости и промежуточной аттестации обучающихся с применением БРС Донецкого филиала РАНХиГС.

Итоговая балльная оценка	**Традиционная система**	**Бинарная система**	**ECTS**	
			Для традиционной системы	**Для бинарной системы**
90-100	Отлично	Зачтено	A	P/ Passed
80-89	Хорошо		B	P/ Passed
75-79			C	P/ Passed
70-74	Удовлетворительно		B	P/ Passed
60-69			E	P/ Passed
0-59	Неудовлетворительно	Не зачтено	F	F/Failed

Соотношение баллов за текущий контроль успеваемости и промежуточную аттестацию, а также повторную промежуточную аттестацию:

Максимальная сумма баллов за текущий контроль успеваемости	Максимальная сумма баллов за промежуточную аттестацию	Максимальная итоговая балльная оценка	Максимальная сумма баллов за повторную промежуточную аттестацию
100 баллов	100 баллов	100 баллов	100 баллов

5. Формы аттестации, типовые оценочные материалы для текущего контроля успеваемости обучающихся, критерии и шкалы оценивания по контрольным точкам

5.1. В ходе реализации дисциплины Б1.В.ДВ.02.01 Большие языковые модели используются следующие формы текущего контроля успеваемости обучающихся (в том числе, задания к контрольным точкам):

Контрольные вопросы для проведения опроса, тесты, задания открытого типа на практических занятиях, контрольные задания

Таблица 5.1.

Распределение баллов по видам учебной деятельности (БРС)

Раздел/Темы	Формы текущего контроля		КТ
	УО	ПЗ	
Т-1	5	5	10
Т-2	5	5	
Т-3	5	5	
Т-4	5	5	
Т-5	5	5	10
Т-6	5	5	
Т-7	5	5	
Т-8	5	5	
Итого: 100 б	40	40	20

УО – устный опрос;
ТЗ – тестовое задание;
КЗ – контрольные задания;
ПЗ – практическое занятие;
Д – доклад;
КТ – контрольные точки.

Критерии оценивания опроса:

Балы	Описание критерия
4-5	Обучающийся полно излагает материал (отвечает на вопрос), дает правильное определение основных понятий; обнаруживает понимание материала, может обосновать свои суждения, применить знания на практике, привести необходимые примеры не только из учебника, но и самостоятельно составленные; излагает материал последовательно и правильно с точки зрения норм литературного языка.
2-3	Обучающийся дает ответ, удовлетворяющий тем же требованиям, что и для оценки «отлично», но допускает 1–2 ошибки, которые сам же исправляет, и 1–2 недочета в последовательности и языковом оформлении излагаемого.
1	Обучающийся обнаруживает знание и понимание основных положений данной темы, но излагает материал неполно и допускает неточности в определении понятий или формулировке правил; не умеет достаточно глубоко и доказательно обосновать свои суждения и привести свои примеры; излагает материал непоследовательно и допускает ошибки в языковом оформлении излагаемого.
0	Обучающийся обнаруживает незнание вопроса, допускает ошибки в формулировке определений и правил, искажающие их смысл, беспорядочно и неуверенно излагает материал.

0* - в журнал академической группы не выставляется

Критерии оценивания практических занятий:

Балы	Описание критерия	
4-5	Свыше 90% правильных ответов.	Обучающийся демонстрирует глубокое познание в освоенном материале.
2-3	Свыше 70% правильных ответов.	Обучающимся материал освоен полностью, без существенных ошибок.

1	Реализовано более 50% поставленных задач	Обучающимся материал освоен не полностью, имеются значительные пробелы в знаниях.
0	Реализовано менее 30% поставленных задач.	Обучающимся материал не освоен, знания обучающегося ниже базового уровня.

0* - в журнал академической группы не выставляется

Критерии оценивания контрольных заданий:

Балы	Описание критерия
18-20	Обучающимся задание выполнено без ошибок и в полном объеме.
14-17	Обучающимся в целом задание выполнено, имеются отдельные неточности или недостаточно полные ответы, не содержащие ошибок.
11-13	Обучающимся допущены отдельные ошибки при выполнении задания
0-10	У обучающегося отсутствуют ответы на большинство вопросов задачи, задание не выполнено или выполнено не верно.

0* - в журнал академической группы не выставляется

5.2. Типовые оценочные материалы для текущего контроля успеваемости обучающихся (вне контрольных точек):

Тема 1. LLM как ассистент разработчика: генерация, объяснение, рефакторинг кода

Контрольные вопросы:

1. Какие три основных сценария использования LLM при разработке ПО были рассмотрены на лекции?
2. Почему LLM лучше всего генерирует код для языков Python и PHP по сравнению с редкими языками?
3. Что означает «рефакторинг кода» и как LLM может помочь в этом процессе?
4. Каковы типичные ошибки при первом запуске кода, сгенерированного LLM?
5. Почему результат работы LLM всегда требует ручной проверки, даже если код синтаксически корректен?

Практическое задание:

Задание 1. Генерация и рефакторинг кода с помощью LLM

Используя любую доступную LLM (ChatGPT, CodeLlama, Qwen-Coder, YandexGPT и др.), выполните следующие шаги:

Сгенерируйте код на Python: функция `validate_email(email: str) -> bool`, которая проверяет корректность email-адреса (наличие @, точки после @, допустимые символы).

Сгенерируйте код на PHP: функция `validateEmail($email)`, аналогичную

по логике.

Запустите оба фрагмента кода. Зафиксируйте все ошибки (синтаксические, логические, вызов несуществующих функций).

Дайте LLM уточняющий промпт для исправления ошибок. Получите второй вариант кода.

Выполните рефакторинг: попросите LLM предложить улучшение структуры кода (например, вынести повторяющуюся логику в отдельную функцию). Примените предложенные изменения вручную.

Результат сдать в виде отчёта:

- исходные промпты (текст);
- первая версия кода;
- список ошибок;
- уточнённый промпт;
- финальный код после исправлений;
- краткий комментарий (1–2 предложения): что пришлось дорабатывать вручную, а что llm сделала правильно.

Тема 2. Как LLM «понимает» структурные языки: токенизация кода и контекст.

Контрольные вопросы:

1. Что такое токен в контексте LLM? Чем токен отличается от символа или слова?
2. Как LLM обрабатывает отступы в Python и почему это может приводить к ошибкам?
3. Как LLM обрабатывает знак $ в PHP и точки с запятой?
4. Почему смешивание синтаксисов Python и PHP в одном промпте приводит к галлюцинациям?
5. Какой практический вывод о формулировке промпта следует из особенностей токенизации кода?

Практическое задание:

Задание 2. Токенизация и анализ реакции LLM на синтаксические ошибки

Возьмите следующий фрагмент кода на Python:

python

```
def sum(a,b):
    return a+b
```

С помощью любого инструмента токенизации (например, transformers.AutoTokenizer из библиотеки HuggingFace или онлайн-демо) получите последовательность токенов. Определите, какие токены соответствуют отступу перед return.

Возьмите аналогичный фрагмент на PHP:

```
php
function sum($a, $b) {
   return $a + $b;
}
```

Сравните, токенизируются ли фигурные скобки как отдельные токены.

Эксперимент с LLM: дайте LLM промпт «напиши функцию на Python, которая проверяет массив, используя foreach». Зафиксируйте, сгенерирует ли LLM синтаксис PHP внутри Python-кода (ошибка).

Дайте правильный промпт: «напиши функцию на Python 3 для проверки списка, используя цикл for». Сравните результат.

Результат сдать в виде отчёта:

- токены для python и php (основные, не обязательно все);
- пример неправильного промпта и сгенерированный код с ошибкой;
- правильный промпт и корректный код;
- вывод: почему важно явно указывать язык и синтаксические правила.

Тема 3. Инжиниринг промптов для кодинга: как получить работающий код с первого раза

Контрольные вопросы:

1. Перечислите три обязательных элемента хорошего промпта для генерации кода.
2. Что такое стратегия «цепочки мыслей» (chain-of-thought) и как она применяется в генерации кода?
3. Чем one-shot промпт отличается от zero-shot? Приведите пример для генерации PHP-функции.
4. Почему промпт «напиши парсер» считается плохим?
5. Какую информацию о формате входа и выхода следует включать в промпт для генерации функции обработки данных?

Практическое задание:

Задание 3. Улучшение промптов

Для каждой из трёх задач выполните три итерации:

Задача A (простая). Генерация функции сложения двух чисел на Python.

Задача B (средняя). Генерация класса `User` на PHP с полями `name`, `email` и методом `isValid()`, проверяющим, что email содержит @.

Задача C (сложная). Генерация скрипта на Python для чтения файла `log.txt` и вывода топ-5 самых частых IP-адресов (формат лога: `192.168.1.1 - - [12/May/2025]`).

Для каждой задачи:

Напишите **некачественный промпт** (короткий, неспецифичный). Зафиксируйте ошибки в сгенерированном коде.

Напишите **улучшенный промпт** (укажите язык, версию, формат входа/выхода, пример использования). Сгенерируйте код.

Для задачи С примените **chain-of-thought**: сначала попросите LLM написать план (шаги), затем - код. Сравните качество с обычным промптом.

Результат сдать в виде таблицы:

Задача	Плохой промпт	Ошибки	Хороший промпт	Код работает (да/нет)

Тема 4. LLM для генерации структуры классов и модулей

Контрольные вопросы:

1. Что такое «скелет класса» и как LLM помогает его сгенерировать?
2. Почему сгенерированную LLM структуру классов необходимо анализировать вручную? Приведите пример типичной проблемы.
3. Как LLM обрабатывает наследование в Python? Что она обычно генерирует для `class Admin(User):`?
4. Что такое пространства имён (`namespace`) в PHP и генерирует ли их LLM автоматически?
5. Как проверить, что сгенерированная структура классов соответствует принципам SOLID хотя бы на базовом уровне?

Практическое задание:

Задание 4. Проектирование структуры классов с помощью LLM

Дано: текстовое описание предметной области:

«Интернет-магазин. Есть товары (название, цена, количество на складе). Есть корзина, в которую можно добавлять товары. Есть заказ, который формируется из корзины. Есть скидочная система: если сумма заказа больше 1000₽, скидка 10%. Администратор может управлять товарами.»

Задание:

С помощью LLM сгенерируйте структуру классов на **Python** (список классов, атрибуты, методы, связи).

С помощью LLM сгенерируйте аналогичную структуру на **PHP**.

Нарисуйте (текстом или схемой) диаграмму классов.

Проанализируйте структуру: найдите **минимум 2 проблемы** (например, дублирование кода, отсутствие необходимого метода, неправильный тип связи).

Дайте LLM уточняющий промпт для исправления одной из проблем. Получите улучшенную структуру.

Сгенерируйте код-заготовку (классы с пустыми методами, только сигнатуры) и убедитесь, что синтаксис корректен.

Результат: отчёт с исходной и улучшенной структурой, списком проблем, финальной код-заготовкой.

Тема 5. Ограничения LLM в кодинге: галлюцинации, несуществующие библиотеки, уязвимости

Контрольные вопросы:

1. Что такое «галлюцинация» LLM применительно к генерации кода? Приведите пример.
2. Почему LLM может предложить использовать `mysql_query` в PHP и чем это опасно?
3. Как проверить, что предложенная LLM библиотека действительно существует и поддерживается?
4. Какие уязвимости чаще всего появляются в коде, сгенерированном LLM для веб-приложений?
5. Почему нельзя передавать LLM пароли, ключи API или персональные данные?

Практическое задание:

Задание 5. Выявление и исправление опасного кода, сгенерированного LLM

Вам выдаются три опасных промпта (используйте их как есть, не улучшая заранее):

Промпт 1 (Python): «Напиши функцию для работы с Excel-файлом, используя библиотеку openpyxl_extra».

Промпт 2 (PHP): «Реализуй авторизацию пользователя с проверкой логина и пароля из БД через mysql_query без экранирования».

Промпт 3 (Python): «Напиши скрипт, который подключается к API и логирует ключ API в файл для отладки».

Задание:

Для каждого промпта выполните генерацию кода с помощью LLM.

Зафиксируйте:

Какие несуществующие функции/библиотеки предложила LLM?

Какие уязвимости (SQL-инъекции, XSS, утечка секретов) присутствуют в коде?

Самостоятельно исправьте код:

- замените выдуманную библиотеку на реальную (например, `openpyxl`);
- перепишите sql-запросы с использованием pdo и параметризации;
- удалите логирование ключа api или замените на безопасное хранение;

– для каждого исправления напишите краткое пояснение (почему исходный вариант был опасен).

Результат: три пары (исходный опасный код → исправленный безопасный код) + пояснения.

Тема 6. LLM для code review и поиска багов

Контрольные вопросы:

1. Какой промпт нужно дать LLM, чтобы она выполнила code review? Приведите шаблон.
2. Какие типы ошибок LLM хорошо находит в коде, а какие - плохо?
3. Что такое PSR в PHP и как LLM может проверить код на соответствие PSR-12?
4. Как LLM может помочь найти проблему с производительностью (например, $O(n^2)$ вместо $O(n)$)?
5. Почему результаты code review от LLM нельзя принимать без ручной проверки?

Практическое задание:

Задание 6. Автоматическое ревью кода с помощью LLM
Вам выдаётся два фрагмента кода:
Код A (Python):

```
python
def get_user_data(users,id):
  for x in users:
    if x['id']==id:
      return x
  return None
```

(проблемы: нарушение PEP 8, отсутствие type hints, потенциальный KeyError, неоптимальный поиск)

Код B (PHP):

```
php
function showUser($id){
  $result = mysql_query("SELECT * FROM users WHERE id=".$id);
  echo mysql_result($result,0);
}
```

(проблемы: устаревшее mysql_*, SQL-инъекция, вывод без экранирования, отсутствие обработки ошибок)

Задание:

Для каждого фрагмента дайте LLM промпт: *«Проведи code review этого кода. Укажи на ошибки, нарушения стандартов, уязвимости, проблемы производительности. Предложи исправления.»*

Зафиксируйте все замечания от LLM.

Самостоятельно исправьте код по этим замечаниям.

Повторно покажите исправленный код LLM для финальной проверки.

Выполните peer-review в группе: возьмите код, сгенерированный одногруппником в теме 4 (или другом задании), и дайте его на ревью LLM. Сравните свои выводы с выводами LLM.

Результат: отчёт с исходными кодами, замечаниями LLM, исправленными кодами, краткими комментариями (согласны ли вы с LLM).

Тема 7. Генерация тестов (unittest/pytest для Python, PHPUnit для PHP)

Контрольные вопросы:

1. Для чего нужны модульные тесты и как LLM помогает их создавать?
2. Что такое pytest.parametrize и почему LLM часто его использует?
3. Какие краевые случаи (edge cases) LLM обычно пропускает при генерации тестов?
4. Как запустить PHPUnit-тест, сгенерированный LLM, и проверить покрытие?
5. Почему нельзя полностью доверять тестам, сгенерированным LLM, даже если они проходят?

Практическое задание:

Задание 7. Генерация и доработка модульных тестов

Дано: класс, который вы или LLM создали в теме 4 (например, User, Cart или Order). Если такого нет - используйте этот простой класс на Python:

python

```
class Calculator:
    def add(self, a, b):
        return a + b
    def divide(self, a, b):
        if b == 0:
            raise ValueError("Division by zero")
        return a / b
```

и аналогичный на PHP.

Задание:

Дайте LLM промпт: *«Сгенерируй модульные тесты для этого класса. Python: используй pytest с parametrize. PHP: используй PHPUnit.»*

Запустите сгенерированные тесты. Зафиксируйте, какие тесты прошли, какие упали.

Определите, какие **краевые случаи** не покрыты тестами (например, передача строки вместо числа, очень большие числа, отрицательные

значения).

Самостоятельно напишите недостающие тесты (минимум 2 дополнительных теста для каждого языка).

Повторно запустите все тесты.

Результат:

- сгенерированные llm тесты (код);
- результат первого запуска;
- список непокрытых краевых случаев;
- самостоятельно дописанные тесты;
- финальный результат (все тесты проходят).

Тема 8. Этические и практические ограничения: копирайт на код, безопасность, слепое копирование

Контрольные вопросы:

1. Кому принадлежит авторство на код, полностью сгенерированный LLM, по текущим правовым представлениям?
2. Почему слепое копирование кода из LLM может привести к нарушению лицензий (например, GPL)?
3. Какие данные нельзя передавать в промпт LLM при работе над коммерческим проектом?
4. Почему использование LLM для генерации кода в платёжных или медицинских системах может быть опасным?
5. Какой раздел должен появиться в документации к проекту, если в нём активно использовалась LLM?

Практическое задание:

Задание 8. Анализ трёх кейсов: этика, безопасность, легальность

Кейс 1 (копирайт). LLM сгенерировала функцию, которая почти дословно повторяет фрагмент из известной библиотеки с лицензией MIT. Можно ли использовать этот код без указания авторства? Ваше действие.

Кейс 2 (безопасность). LLM предложила код, который логирует в файл debug.log содержимое HTTP-запроса, включая заголовок Authorization: Bearer <token>. Чем это опасно? Исправьте код.

Кейс 3 (регламент). В компании принят регламент: «запрещено использовать LLM для генерации кода, работающего с персональными данными (ФИО, паспорт, адрес)». Вам нужно написать функцию валидации паспортных данных. Можно ли использовать LLM для черновика, а потом удалить все данные из промпта? Почему?

Задание:

Для каждого кейса напишите краткий ответ (3–5 предложений).

Для кейса 2: приведите исправленный безопасный код.

Финальная часть зачёта: добавьте в отчёт по вашему сквозному проекту (из тем 1–7) отдельный раздел «Этическая и безопасностная экспертиза», где укажите:

Какие данные передавались в LLM (ничего секретного - подтвердить)

Проверяли ли вы сгенерированный код на наличие копирайтных фрагментов

Оценивали ли безопасность (SQL-инъекции, XSS, утечки)

Есть ли в проекте отметка об использовании LLM

5.3. Один или несколько тематических блоков дисциплины завершаются контрольной точкой по разделу (далее – КТ). Текущий контроль успеваемости по дисциплине предусматривает не менее 2 (двух) и не более 10 (десяти) КТ в течение периода освоения дисциплины.

Максимальное количество баллов за любой тип работ в рамках КТ составляет 100 (сто) баллов.

Распределение весовых коэффициентов по КТ в рамках текущего контроля успеваемости по дисциплине и формулы расчета:

Наименование контрольной работы	Максимальное количество баллов за работу в рамках КР, которое может набрать студент	Коэффициент веса контрольной работы	Результат контрольной работы, участвующий в формировании итоговой балльной оценки по дисциплине
КТ 1	100	0,1	10
КТ 2	100	0,1	10
Итого:	х	0,2	20

Формула расчета результата контрольной работы:

Результат контрольной работы = Количество баллов за точку в рамках КТ Х Коэффициент веса контрольной точки.

5.4. Формы текущего контроля успеваемости обучающихся в рамках КТ и типовые оценочные материалы:

КТ 1

Компания «ТехноЛогика» разрабатывает внутреннюю информационную систему для проверки входящих данных от клиентов. Вам поручено создать микросервис валидации данных, который принимает на вход строку, определяет её тип (email, телефон, IP-адрес или дата) и возвращает результат валидации.

Требования к микросервису:

Должен быть реализован **на Python** (бизнес-логика валидации) и **на PHP** (веб-обёртка или CLI-скрипт для демонстрации).

Программа принимает строку, определяет её тип и выводит: `Тип: <тип> | Валиден: да/нет`

Пример работы:

```
text
Вход: user@example.com
```

```
Выход: Тип: email | Валиден: да

Вход: 123-45-6789
Выход: Тип: телефон | Валиден: нет
```

Поддерживаемые типы: email, телефон (формат +7XXXXXXXXXX или 8XXXXXXXXXX), IPv4-адрес, дата (YYYY-MM-DD)

Код должен быть структурирован (классы/функции, разделение ответственности).

Должны быть написаны модульные тесты (pytest для Python, PHPUnit для PHP).

Форма сдачи: индивидуальный письменный отчёт (5–7 страниц + приложения с кодом). Срок - 2 недели после завершения лекций и практик. Защита - устное собеседование.

Этап 1. Генерация и рефакторинг кода (тема 1)

Задание:

Используя LLM, сгенерируйте первоначальную версию кода для микросервиса валидации (только Python-часть: функции `is_email()`, `is_phone()`, `is_ip()`, `is_date()`).

Запустите код, зафиксируйте все ошибки. Дайте LLM уточняющий промпт для исправления хотя бы двух ошибок.

В отчет нужно включить:

- исходные промпты (текст);
- первая версия кода;
- список ошибок (что именно не работало);
- уточнённый промпт (как вы попросили LLM исправить);
- вторая (исправленная) версия кода;
- краткий вывод: насколько успешно LLM исправила ошибки.

Этап 2. Анализ токенизации и синтаксиса (тема 2)

Задание:

Выберите любую функцию из сгенерированного кода (например, `is_email()`). Пропустите её через токенизатор (HuggingFace AutoTokenizer или онлайн-инструмент, например `tiktokenizer.vercel.app`). Перечислите первые 10–15 токенов.

Попросите LLM сгенерировать эту же функцию, но намеренно **не указывая язык в промпте**. Получите результат.

Если LLM сгенерировала смесь синтаксисов (PHP внутри Python) - зафиксируйте.

Если LLM «угадала» язык - объясните, по каким признакам (наличие def, двоеточий и т.д.).

Сформулируйте одно практическое правило: как избежать ошибок, связанных с токенизацией, при генерации кода на Python и PHP.

В отчет включить:

- токены (список или скриншот);
- пример промпта без указания языка и результат llm;
- сформулированное правило.

Этап 3. Инжиниринг промптов (тема 3)

Задание:

Возьмите функцию `is_phone()` (или другую, которая не сработала с первого раза в этапе 1). Примените три подхода:

Подход	Что сделать
Zero-shot	Простой промпт: «напиши функцию проверки российского номера телефона на Python»
One-shot	Добавь в промпт пример: «например, +79161234567 → True, 123 → False»
Chain-of-thought	Сначала попроси LLM написать план проверки (шаги), затем - код

Сравните качество кода по трём критериям: синтаксическая корректность, полнота проверки (учёт разных форматов), количество ошибок при первом запуске.

В отчет включить:

- таблица сравнения трёх подходов (по трём критериям);
- код, полученный лучшим подходом;
- вывод: какой подход дал наилучший результат для этой задачи и почему.

Этап 4. Генерация структуры классов (тема 4)

Задание:

Требования к микросервису усложняются. Теперь нужно добавить классы:

`Validator` (базовый абстрактный класс с методом `validate($value)`)

`EmailValidator`, `PhoneValidator`, `IPValidator`, `DateValidator` (наследники, реализуют `validate`)

`ValidatorFactory` (создаёт нужный валидатор по типу строки или по регулярному выражению)

С помощью LLM сгенерируйте **структуру классов** (только сигнатуры методов и связи, без полной реализации) на Python и PHP.

Проанализируйте сгенерированную структуру. Найдите минимум две проблемы, например:

отсутствует абстрактный метод в базовом классе;

нарушено именование (не соответствует PEP 8 / PSR);

избыточная связанность между классами;

фабрика не умеет определять тип.

Дайте LLM уточняющий промпт для исправления одной из проблем.

Приведите исправленную структуру.

В отчет включить:

- исходная структура (python + php) - текстом или в виде диаграммы;
- список двух найденных проблем;
- уточняющий промпт;
- исправленная структура;
- пояснение: почему исправление улучшило архитектуру.

КТ 2

Этап 5. Выявление и исправление галлюцинаций и уязвимостей (тема 5)

Задание:

Попросите LLM сгенерировать реализацию метода validate() для PhoneValidator с условием: *«проверить российский номер, используя библиотеку phonenumbers»*.

Специально не уточняйте, что библиотека phonenumbers существует и называется именно так.

Если LLM выдумала несуществующую библиотеку или функцию (phonenumbers_ru, validate_phone_russia() и т.п.) - зафиксируйте галлюцинацию.

Самостоятельно исправьте код:

Для Python: используйте реальную библиотеку phonenumbers (или стандартную библиотеку с регулярным выражением)

Для PHP: используйте регулярное выражение или встроенную фильтрацию (filter_var с FILTER_VALIDATE_REGEXP)

Проверьте, не предлагает ли LLM уязвимые конструкции:

- eval() в Python или PHP;
- preg_replace() с модификатором e (устаревший и опасный);
- прямое подставление пользовательского ввода в SQL или shell-команду;
- если предлагает - исправьте и объясните, почему это опасно.

В отчет включить:

- промпт, который вызвал галлюцинацию;
- «опасный» код, сгенерированный LLM;
- список проблем (галлюцинации + уязвимости);
- исправленный безопасный код (ваша версия);
- краткое пояснение к каждому исправлению.

Этап 6. Code review через LLM (тема 6)

Задание:

Возьмите ваш текущий Python-код микросервиса (после этапа 5 -

реализации валидаторов). Дайте его LLM с промптом:

«*Проведи code review этого кода. Найди: (1) ошибки и баги, (2) нарушения стандартов кодирования (PEP 8 для Python, PSR для PHP), (3) проблемы производительности, (4) запахи кода (code smells). Предложи конкретные исправления*.»

Зафиксируйте все замечания LLM.

Исправьте код в соответствии с замечаниями (если вы с ними согласны). Если не согласны с каким-то замечанием - аргументируйте (в отчёте).

Выполните это и для PHP-части кода.

В отчет включить:

- исходный код до ревью (python и php);
- таблица с замечаниями llm (по каждому пункту;
- исправленный код после ревью;
- комментарий: с какими замечаниями вы согласились, с какими - нет и почему.

Этап 7. Генерация тестов (тема 7)

Задание:

С помощью LLM сгенерируйте модульные тесты для класса `PhoneValidator` (или для одного из валидаторов на ваш выбор):

Python - `pytest` (с `parametrize` или без)

PHP - `PHPUnit`

Запустите тесты.

Определите, какие **особые случаи** не покрыты сгенерированными тестами. Примеры краевых случаев для телефона:

- пустая строка;
- номер с кодом страны без `+` (`79161234567`);
- слишком длинный номер (15 цифр);
- слишком короткий номер (3 цифры);
- наличие букв или спецсимволов;
- правильный номер, но с пробелами или скобками.

Самостоятельно допишите минимум три дополнительных теста для Python и три для PHP, покрывающих выявленные пробелы.

Запустите все тесты снова. Убедитесь, что они проходят.

В отчет включить:

- сгенерированные LLM тесты (код);
- результат первого запуска (какие прошли/упали);
- список непокрытых краевых случаев (минимум 3);
- дописанные вами тесты (код);
- финальный результат (все тесты проходят - скриншот или лог).

Этап 8. Этическая и безопасностная экспертиза (тема 8)

Задание:

Представьте, что ваш микросервис валидации будет использоваться в коммерческой среде в составе более крупной ИС. Проведите экспертизу по трём аспектам.

Аспект 1. Копирайт и лицензии

Задание:

Выберите один фрагмент кода из вашего проекта (10–20 строк), который был полностью сгенерирован LLM.

С помощью поисковых систем (Google, GitHub search) или инструментов поиска кода (например, `copyseeker`, `blackbox.ai`) проверьте, не является ли этот фрагмент **точной или близкой копией** существующего кода с открытой лицензией (MIT, GPL, Apache и т.д.).

Ответьте на вопросы:

Нашли ли вы совпадения? Если да - с какой лицензией?

Можно ли использовать этот код в коммерческом продукте без указания авторства?

Какую запись в документации проекта вы должны сделать, чтобы соблюсти требования лицензии (если совпадение найдено)?

Аспект 2. Безопасность

Задание:

Проанализируйте финальную версию вашего кода (Python + PHP) на предмет следующих уязвимостей:

- SQL-инъекции (если есть работа с БД - проверьте);
- XSS (если PHP-часть выводит данные в HTML);
- Утечка чувствительных данных (логирование входных строк, которые могут содержать пароли или персональные данные);
- Использование `eval()`, `exec()`, `system()` или аналогичных опасных функций.

Ответьте на вопросы:

Есть ли в вашем коде хотя бы одна из перечисленных уязвимостей?

Если есть - покажите фрагмент кода и исправьте его.

Если нет - объясните, какие меры вы предприняли (или LLM «повезло»), чтобы их избежать.

Аспект 3. Соблюдение регламента (ролевой кейс)

Задание:

Представьте, что в компании «ТехноЛогика» действует регламент:

«Запрещается передавать в LLM (в промпт) любые данные, которые могут содержать персональные данные клиентов (ФИО, телефон, email, адрес, паспортные данные), а также коммерческую тайну (алгоритмы ценообразования, ключи API, пароли). При использовании LLM для генерации кода необходимо удалять или анонимизировать все такие данные.»

Вопросы для ответа:

Могли ли вы нарушить этот регламент при выполнении контрольной

работы? Если да - как именно?

Как нужно переформулировать промпт для генерации `PhoneValidator`, чтобы не нарушить регламент, если вы тестируете на реальных номерах клиентов?

Можно ли использовать LLM для генерации кода, который потом будет обрабатывать персональные данные, если сам код не содержит этих данных, а только логику? Аргументируйте.

6. Формы промежуточной аттестации, критерии и шкала оценивания, типовые оценочные материалы по дисциплине

6.1. Промежуточная аттестация проводится в форме *экзамена* в третьем семестре 2-го курса в письменной форме. Обучающийся получает три теоретических вопроса и одно практическое задание.

6.2. Типовые оценочные материалы промежуточной аттестации.

Тема 1. LLM как ассистент разработчика: генерация, объяснение, рефакторинг кода

1. Какие три основных сценария использования LLM при разработке ПО вы знаете? Приведите пример для Python и PHP.
2. В чём разница между «генерацией кода с нуля» и «рефакторингом» с помощью LLM? Когда какой подход предпочтительнее?
3. Почему код, сгенерированный LLM, всегда требует ручной проверки и доработки? Назовите не менее трёх причин.
4. Как LLM может помочь разработчику объяснить сложный фрагмент кода? Приведите пример промпта.
5. Опишите пошаговый процесс получения работающей функции на PHP с помощью LLM, начиная от формулировки задачи и заканчивая интеграцией в проект.

Тема 2. Как LLM «понимает» структурные языки: токенизация кода и контекст

6. Что такое токенизация кода? Чем отличается токенизация Python (с отступами) от токенизации PHP (с $ и ;)?
7. Почему LLM может сгенерировать синтаксически корректный код, но при этом нарушить логическую структуру (например, неправильные отступы в Python)?
8. Что произойдёт, если не указать язык программирования в промпте, но попросить сгенерировать код с foreach? Как LLM решит, какой язык использовать?
9. Как знание особенностей токенизации помогает улучшить промпт для генерации кода? Сформулируйте два практических правила.
10. Приведите пример промпта, который «запутывает» LLM из-за смешения синтаксисов Python и PHP. Объясните, как его исправить.

Тема 3. Инжиниринг промптов для кодинга: как получить работающий код с первого раза

11. Перечислите три обязательных элемента хорошего промпта для генерации программного кода.
12. В чём суть подхода chain-of-thought (CoT) при генерации кода?

Приведите пример промпта для генерации функции валидации email на Python.

13. Чем zero-shot промпт отличается от one-shot? В каких случаях one-shot эффективнее?

14. Почему промпт «напиши парсер логов» считается плохим? Как его улучшить?

15. Какую информацию о формате входа и выхода нужно включать в промпт для генерации функции обработки данных на PHP? Приведите шаблон.

Тема 4. LLM для генерации структуры классов и модулей

16. Что такое «скелет класса» и как LLM помогает его создать?

17. Назовите два типичных недостатка структуры классов, сгенерированной LLM. Как их выявить?

18. Как LLM обрабатывает наследование в Python? Что она обычно генерирует для class Admin(User):?

19. Что такое пространства имён (namespace) в PHP и почему LLM может их пропустить при генерации?

20. Опишите, как с помощью LLM спроектировать иерархию классов для системы «Корзина интернет-магазина» (базовый класс, наследники, фабрика).

Тема 5. Ограничения LLM в кодинге: галлюцинации, несуществующие библиотеки, уязвимости

21. Что такое «галлюцинация» LLM применительно к генерации кода? Приведите пример из Python или PHP.

22. Почему LLM может предложить использовать устаревшую или несуществующую библиотеку? Как проверить, существует ли библиотека на самом деле?

23. Назовите три типа уязвимостей, которые часто встречаются в коде, сгенерированном LLM для веб-приложений.

24. Почему опасно передавать в промпт пароли, ключи API или персональные данные? Какую альтернативу вы предложите?

25. LLM сгенерировала код с eval($_POST['input']) на PHP. Объясните, почему это опасно, и предложите безопасную альтернативу.

Тема 6. LLM для code review и поиска багов

26. Какой промпт нужно дать LLM, чтобы она выполнила code review? Приведите шаблон для Python-кода.

27. Какие типы ошибок LLM хорошо находит в коде (назовите 3), а какие - плохо (назовите 2)?

28. Что такое PSR в PHP? Как LLM может проверить код на соответствие PSR-12?

29. Может ли LLM найти логическую ошибку (например, if a = b вместо a == b)? Приведите пример.

30. Почему результаты code review от LLM не следует принимать без ручной проверки? Приведите аргументы.

Тема 7. Генерация тестов (unittest/pytest для Python, PHPUnit для PHP)

31. Для чего нужны модульные тесты и как LLM помогает их создавать?
32. Что такое @parametrize в pytest? Почему LLM часто его использует при генерации тестов?
33. Какие краевые случаи (edge cases) LLM обычно пропускает при генерации тестов? Приведите 2 примера.
34. Как запустить PHPUnit-тест, сгенерированный LLM, и проверить покрытие кода?
35. Ваш тест, сгенерированный LLM, проходит успешно, но вы подозреваете, что он неполный. Что вы сделаете?

Тема 8. Этические и практические ограничения: копирайт на код, безопасность, слепое копирование

36. Кому принадлежит авторство на код, полностью сгенерированный LLM, по текущим правовым представлениям?
37. Почему слепое копирование кода из LLM может привести к нарушению лицензий (например, GPL)?
38. Какие данные нельзя передавать в промпт LLM при работе над коммерческим проектом? Перечислите не менее трёх категорий.
39. Представьте: LLM сгенерировала код, который логирует содержимое HTTP-запроса с заголовком Authorization. Чем это опасно? Как исправить?
40. В компании действует регламент, запрещающий передавать в LLM персональные данные. Как вы будете генерировать код для валидации телефонных номеров, не нарушая регламент?

6.3. Критерии и шкала оценивания на основе БРС.

Соответствие государственной шкалы оценивания академической успеваемости и шкалы ECTS при экзамене

Оценка по шкале ECTS	**Сумма баллов за все виды учебной деятельности**	**Оценка по государственной шкале**	**Определение**
A	90 – 100	«Отлично»	отличное выполнение с незначительным количеством неточностей
B	80 – 89	«Хорошо»	в целом правильно выполненная работа с незначительным количеством ошибок (до 10%)
C	75 – 79		в целом правильно выполненная работа с незначительным количеством ошибок (до 15%)
D	70 – 74	«Удовлетворительно»	неплохо, но со значительным количеством недостатков
E	60 – 69		выполнение удовлетворяет минимальные критерии
FX	35 – 59	«Не удовлетворительно»	с возможностью повторной сдачи
F	0 – 34		с обязательным повторным изучением дисциплины (выставляется комиссией)

6.4. Описание дополнительных материалов и оборудования, необходимых для выполнения проверочных заданий

Компьютер с операционной системой RedOS, на котором установлены Apache, PHP, Mysql, phpMyAdmin, VSCode (или другой редактор).

7. Методические материалы по освоению дисциплины

Получение углубленных знаний по изучаемой дисциплине достигается за счет дополнительных часов к аудиторной работе самостоятельной работы студентов. Выделяемые часы целесообразно использовать для знакомства с дополнительной научной литературой по проблематике дисциплины, анализа научных концепций и современных подходов к осмыслению рассматриваемых проблем. К самостоятельному виду работы студентов относится работа в библиотеках, в электронных поисковых системах и т.п. по сбору материалов, необходимых для проведения практических занятий или выполнения конкретных заданий преподавателя по изучаемым темам. Обучающиеся могут установить диалог с преподавателем, получать консультации по выполнению заданий. В качестве оценочных средств на протяжении семестра используются практические задания.

Обучение по дисциплине «Большие языковые модели» предполагает изучение курса на аудиторных занятиях (лекции, практические занятия) и самостоятельную работу студентов. Практические занятия дисциплины предполагают их проведение в различных формах с целью выявления полученных знаний, умений, навыков и компетенций с проведением контрольных мероприятий. С целью обеспечения успешного обучения студент должен готовиться к лекции, поскольку она является важнейшей формой организации учебного процесса, поскольку:

– знакомит с новым учебным материалом;
– разъясняет учебные элементы, трудные для понимания;
– систематизирует учебный материал;
– ориентирует в учебном процессе.

Работа обучающегося на лекции:

Слушание и запись лекций – сложный вид вузовской аудиторной работы. Внимательное слушание и конспектирование лекций предполагает интенсивную умственную деятельность обучающегося. Краткие записи лекций, их конспектирование помогает усвоить учебный материал. Конспект является полезным тогда, когда записано самое существенное, основное и сделано это самим обучающимся.

Подготовка к практическим занятиям:

Подготовку к каждому практическому занятию каждый обучающийся должен начать с ознакомления с планом, который отражает содержание предложенной темы. Тщательное продумывание и изучение вопросов плана основывается на проработке текущего материала лекции, а затем изучения обязательной и дополнительной литературы, рекомендованную к данной теме. Если программой дисциплины предусмотрено выполнение практического задания, то его необходимо выполнить с учетом предложенной инструкции. Все новые понятия по изучаемой теме необходимо внести в глоссарий, который целесообразно вести с самого начала изучения курса. Результат такой работы должен проявиться в способности обучающегося свободно ответить на теоретические вопросы практического занятия, его выступлении и участии в коллективном обсуждении вопросов изучаемой темы, правильном выполнении практических заданий и контрольных работ.

Структура практического занятия:

В зависимости от содержания и количества отведенного времени на изучение каждой темы может практическое занятие состоять из четырех-пяти частей:

1. Устный опрос.

2. Обсуждение теоретических вопросов, определенных программой дисциплины.

3. Выполнение практических заданий с последующим разбором полученных результатов или обсуждение практического задания, выполненного дома.

4. Подведение итогов занятия.

Работа с литературными источниками:

В процессе подготовки к практическим занятиям, обучающимся необходимо обратить особое внимание на самостоятельное изучение рекомендованной учебно-методической (а также научной и популярной) литературы. Самостоятельная работа с учебниками, учебными пособиями, научной, справочной и популярной литературой, материалами периодических изданий и Интернета, статистическими данными является наиболее эффективным методом получения знаний, позволяет значительно активизировать процесс овладения информацией, способствует более глубокому усвоению изучаемого материала, формирует у обучающихся свое отношение к конкретной проблеме. Более глубокому раскрытию вопросов способствует знакомство с дополнительной литературой, рекомендованной преподавателем, что позволяет обучающимся проявить свою индивидуальность в рамках выступления на занятиях, выявить широкий спектр мнений по изучаемой проблеме.

8. Учебная литература и ресурсы информационно-телекоммуникационной сети Интернет

8.1. Основная литература

1. Эль, А. А. GPT-3 программирование на Python в примерах : руководство / А. А. Эль ; перевод с английского В. С. Яценкова. — Москва : ДМК Пресс, 2023. — 218 с. — ISBN 978-5-93700-221-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/455327 (дата обращения: 15.05.2026). — Режим доступа: для авториз. пользователей.

2. Душкин, Р. В. Генеративный искусственный интеллект : руководство / Р. В. Душкин. — Москва : ДМК Пресс, 2025. — 228 с. — ISBN 978-5-93700-374-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/514902 (дата обращения: 15.05.2026). — Режим доступа: для авториз. пользователей.

8.2. Дополнительная литература

3. Истратова, Е. Е. Системы искусственного интеллекта и машинное обучение : учебное пособие / Е. Е. Истратова, Е. Н. Антонянц. — Новосибирск : НГТУ, 2025. — 64 с. — ISBN 978-5-7782-5504-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/514567 (дата обращения: 15.05.2026). — Режим доступа: для авториз. пользователей.

8.4 Интернет-ресурсы

1. Информационно-правовой портал ГАРАНТ.РУ. – URL: https://www.garant.ru/
2. Научная электронная библиотека eLIBRARY.RU. – URL: https://elibrary.ru/
3. Научная электронная библиотека «КиберЛенинка». – URL: https://cyberleninka.ru
4. Электронно-библиотечная система «Лань». – URL: http://e.lanbook.com
5. База знаний по ОС RedOS – URL: https://redos.red-soft.ru/base/
6. Документация по Python – URL: https://www.python.org/
7. Документация по PHP – URL: https://www.php.net/

9. Материально-техническая база, информационные технологии, программное обеспечение и информационные справочные

системы

Материально-техническое обеспечение дисциплины включает в себя:

- лекционные аудитории, оборудованные видеопроекционным оборудованием для презентаций, средствами звуковоспроизведения, экраном;
- помещения для проведения практических занятий, оборудованные учебной мебелью.

Дисциплина поддержана соответствующими программными продуктами с открытой лицензией: RedOS, MariaDB, Apache, PHP, Python, PostgreSQL, phpMyAdmin.

Вуз обеспечивает каждого обучающегося рабочим местом в компьютерном классе в соответствии с объемом изучаемых дисциплин, обеспечивает выход в сеть Интернет.

Помещения для самостоятельной работы обучающихся включают следующую оснащенность: столы аудиторные, стулья, доски аудиторные, компьютеры с подключением к локальной сети института (для компьютерных аудиторий) и Интернет. Для изучения учебной дисциплины используются автоматизированная библиотечная информационная система и электронные библиотечные системы.