

Федеральное государственное образовательное бюджетное учреждение
высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ
РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)**

**Кафедра информационных технологий
Факультет информационных технологий и анализа больших данных**

Документ подписан усиленной неквалифицированной электронной подписью
Организация: Финансовый университет при Правительстве РФ
Утверждено: Проректор по учебной и методической работе Е.А. Каменева
Сертификат: FIO1+yUlbi2bL66+yj6xw6EkLq5OaVQi
Дата: 15.10.2025 г.

Е.Ю. Клочков

Веб-разработка

Рабочая программа дисциплины

для студентов, обучающихся по направлению подготовки:

09.03.04 - Программная инженерия,

Образовательная программа «Инженер-разработчик программного обеспечения»

Профиль:

«Инженер-разработчик программного обеспечения»

Рекомендовано

*Факультет информационных технологий и анализа больших данных
(протокол № 03 от 16.12.2025 г.)*

Одобрено

*Кафедра информационных технологий
(протокол № 12 от 03.12.2025 г.)*

© Москва 2025

СОДЕРЖАНИЕ

1.	Наименование дисциплины	3
2.	Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине	3
3.	Место дисциплины в структуре образовательной программы	6
4.	Объем дисциплины (модуля) в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся	6
5.	Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий	7
5.1.	Содержание дисциплины	7
5.2.	Учебно-тематический план	12
5.3.	Содержание семинаров	13
6.	Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине	15
6.1.	Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы	15
6.2.	Перечень вопросов, заданий, тем для подготовки к текущему контролю	17
7.	Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине	24
8.	Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины	40
9.	Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины	40
10.	Методические указания для обучающихся по освоению дисциплины	42
11.	Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем	56
12.	Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине	57

1. Наименование дисциплины

«Веб-разработка».

2. Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине

Код компетенции	Наименование компетенции	Индикаторы достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции
ПКП-5	Способность проектировать и реализовывать интеллектуальные информационные системы	Демонстрирует знания основных методов машинного обучения и интеллектуального анализа данных, применяет готовые инструменты для создания интеллектуальных алгоритмов	знать: основные методы ML (классификация, регрессия, кластеризация) и готовые инструменты (TensorFlow.js, ML5.js) для их реализации в веб-приложениях уметь: применять готовые ML-библиотеки на JavaScript для решения типовых задач (распознавание изображений, анализ текста) в веб-интерфейсах
		Понимает особенности интеллектуальных информационных систем в части операций разработки, развертывания и сопровождения	знать: особенности жизненного цикла ML-приложений (обучение моделей, инференс, A/B тестирование, мониторинг дрейфа данных) в контексте веб-разработки уметь: организовывать процесс разработки и деплоя веб-приложений с ML-компонентами, включая версионирование моделей и метрик
		Адаптирует практики создания программных продуктов, в том числе командные, для интеллектуальных информационных систем	знать: особенности agile-практик (спринты, стендапы, ретроспективы) при разработке ML-проектов и специфику командных ролей (data scientist, ML engineer, frontend developer) уметь: адаптировать процессы код-ревью, тестирования и CI/CD для проектов, содержащих как код, так и ML-модели
		Организовывает сбор и подготовку данных для систем машинного обучения,	знать: методы сбора, разметки, аугментации и нормализации данных для обучения моделей в

		в том числе потоковых, онлайн обучения	браузерной среде уметь: создавать инструменты сбора пользовательских данных (аннотации, фидбэк) и подготавливать их для обучения или дообучения моделей
ПКП-6	Способность вести разработку программных систем в команде, вести эффективную коммуникацию	Придерживается рекомендованного в команде стиля написания кода, проводит рефакторинг написанного кода, в том числе, другими членами команды, проводит код-ревью	знать: стандарты и соглашения по написанию кода на JavaScript/TypeScript (ESLint, Prettier), принципы чистого кода и паттерны рефакторинга уметь: проводить код-ревью по чек-листу, предлагать улучшения кода, проводить рефакторинг существующего кода, с сохранением функциональности
		Документирует код, архитектуру, дизайн, требования к программной системе с использованием принятой в команде системы технических стандартов	знать: виды технической документации (JSDoc, архитектурные решения ADR, user stories, диаграммы) и инструменты для их ведения (Swagger, Storybook, Miro) уметь: создавать документацию разных уровней: от инлайн-комментариев до архитектурных решений и пользовательских инструкций
		Использует инструментальные средства командной разработки программного обеспечения, контроля версий, отслеживания ошибок, планирования процессов разработки в зависимости от принятой в команде методологии разработки	знать: инструменты и практики Git (ветвление, мерж, конфликты), таск-трекеры (Jira, Trello), CI/CD (GitHub Actions), и методологии (Scrum, Kanban) уметь: эффективно использовать Git в команде, вести таски в трекере, настраивать базовые CI-пайплайны и участвовать в agile-ритуалах
		Выстраивает эффективную двустороннюю коммуникацию с нетехническими специалистами по предмету разработки предметной области в целом	знать: принципы нетехнической коммуникации: избегание жаргона, визуализация, storytelling, активное слушание, техники вопросов уметь: объяснять технические решения бизнесу, формулировать требования, проводить презентации продукта, задавать уточняющие вопросы и резюмировать договоренности
		Коммуницирует задачи	знать: методы оценки задач (story

		разработки между членами команды, оценивает трудоемкость и сложность выполнения задач разработки, ставит задачи и контролирует их выполнение, в том числе в письменной формализованной форме	points, часы), техники декомпозиции (user story → tasks), форматы постановки задач (SMART, INVEST), инструменты контроля (daily sync, отчеты) уметь: декомпонировать фичи на задачи, оценивать их сложность, ставить задачи коллегам, отслеживать прогресс и проводить демо результатов
--	--	---	--

3. Место дисциплины в структуре образовательной программы

Дисциплина «Веб-разработка» относится к «Профилю "Инженер-разработчик программного обеспечения"».

4. Объем дисциплины (модуля) в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся

Очно-заочная форма обучения

Вид учебной работы по дисциплине	Всего (в з/е и часах)	Семестр 5 (в часах)	Семестр 6 (в часах)
Общая трудоёмкость дисциплины	8/288	144	144
Контактная работа- Аудиторные занятия	52	26	26
Лекции	16	8	8
Семинары, практические занятия	36	18	18
Самостоятельная работа	236	118	118
Вид текущего контроля	Проектная работа, Проектная работа	Проектная работа	Проектная работа
Вид промежуточной аттестации	Зачет, Экзамен	Зачет	Экзамен

5. Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий

5.1. Содержание дисциплины

Тема 1. Введение в веб разработку - основы HTML и CSS, знакомство с JavaScript. Вёрстка. Семантика и лейаут.

Введение в разработку.

Что такое HTML и CSS.

Базовые CSS-свойства.

Больше CSS.

Знакомство с JavaScript.

Вёрстка — продолжение дизайна.

Инструменты разработчика.

Файлы в проекте - структура файлов, пути к файлам.

Семантика и глобальные атрибуты.

Шрифты и типографика.

Флексбокс-вёрстка.

Позиционирование элементов.

Grid Layout. Кодстайл.

Основы Bash и Git.

Основы Git: создание репозитория, коммиты, ветки.

Code style: базовые правила форматирования кода.

Комментарии в коде: однострочные и многострочные.

Тема 2. Вёрстка. Доступность интерфейсов и подходы к организации стилей.

Настройка страницы и метаданных.

Внешний встраиваемый контент: видео, iframe, API.

Дополнение блочной модели.

Псевдоклассы и псевдоэлементы.

Доступность интерфейсов.

Разметка форм.

Селекторы.

Стилизация форм.

Продвинутый Git и Bash.

Документирование CSS: комментарии к сложным стилям.

Accessibility (a11y) документация: ARIA-атрибуты, screen readers.

Git: работа с issues, базовые workflow.

Тема 3. Вёрстка. Адаптивность и графика.

Резиновая и адаптивная вёрстки.

Растровая графика.

CSS-переменные.

Единицы измерения и функции.

Grid Layout, часть 2.

Разработка интерфейса для разных устройств.

Кроссбраузерность, вендорные префиксы.

Подходы к написанию вёрстки. Продвинутый Git.

Документирование адаптивных breakpoints.

Git: разрешение конфликтов, pull requests.

Командная работа над вёрсткой: code review чек-лист.

Тема 4. Вёрстка - декорирование, подходы и инструменты.

Введение в векторную графику SVG.

Переходы и 2D-трансформации. Анимации.

Вариативные шрифты.

Декорирование.

3D трансформации.

Современные интерактивные элементы.

Документирование анимаций и переходов.

Инструменты для совместной работы: Figma, Miro.

Презентация решений по анимациям нетехническим специалистам.

Тема 5. Введение в JavaScript, базовые типы, работа с DOM.

Введение в JavaScript.

Как находить решения проблем.

Примитивы. Начало.

Знакомство с DOM.

Создание, добавление и удаление элементов в DOM.

Методы работы с данными, условия, циклы.

JSDoc: документирование функций.

Рефакторинг: базовые принципы улучшения кода.

Оценка сложности задач: time estimation для скриптов.

Тема 6. Основы JavaScript. Сборка кода и модульность. Валидация форм.

Как находить решения проблем.

Примитивы, продолжение.

Продолжение знакомства с DOM.

Создание, добавление и удаление элементов в DOM.

Методы работы с данными, условия, циклы.

Хранилище.

localStorage и sessionStorage.

Дебаггинг JavaScript.

Модули в JS.

Сборка кода на Vite.

Валидация форм.

Регулярные выражения.

Валидация форм и сборка кода.

Введение в машинное обучение в браузере: TensorFlow.js демо.

Сбор данных через формы: подготовка датасетов.

Документирование валидационных правил.

Git: ветвление для экспериментов с ML.

Тема 7. Работа с API в JS.

Асинхронность. Работа с API.

ML API: работа с готовыми моделями (OpenAI, Google Vision).

Документирование API-интеграций: Swagger/OpenAPI basics.

Потоковая обработка данных для ML: WebSocket + данные.

Коммуникация с backend-разработчиками по API контрактам.

Тема 8. Typescript.

Введение в TypeScript.

Основы TypeScript.

Использование ИИ в разработке.

TypeScript для ML: типизация данных и моделей.

Документирование типов и интерфейсов.

Особенности разработки ML-систем: версионирование моделей.

Планирование задач для ML-компонентов: декомпозиция.

Эксплуатация ML-решений: мониторинг качества предсказаний.

5.2. Учебно-тематический план

№ п/п	Наименование тем(разделов) дисциплины	Трудоемкость в часах				
		Всего	Контактная работа - Аудиторная работа			Самостоя тельная работа
			Общая, в т.ч.:	Лекции	Семинары, практическ ие занятия	
1	Введение в веб разработку - основы HTML и CSS, знакомство с JavaScript. Вёрстка. Семантика и лейаут.	38	8	2	6	30
2	Вёрстка. Доступность интерфейсов и подходы к организации стилей.	36	6	2	4	30
3	Вёрстка. Адаптивность и графика.	36	6	2	4	30
4	Вёрстка - декорирование, подходы и инструменты.	34	6	2	4	28
5	Введение в JavaScript, базовые типы, работа с DOM.	38	8	2	6	30
6	Основы JavaScript. Сборка кода и модульность. Валидация форм.	36	6	2	4	30
7	Работа с API в JS.	36	6	2	4	30
8	Typescript.	34	6	2	4	28
	Итого	288	52	16	36	236

5.3. Содержание семинаров

Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
Введение в веб разработку - основы HTML и CSS, знакомство с JavaScript. Вёрстка. Семантика и лейаут.	Установка и настройка Visual Studio Code IDE. Базовые элементы. HTML и CSS. Теги HTML - Заголовки, Абзац, Ссылка, Родители и дети. Структура HTML-документа. Связь CSS и HTML. Тег <style>. Связь CSS и HTML. CSS-файл. Подключение JavaScript к HTML. Как семантическая разметка влияет на SEO и доступность?	Решение и разбор задач.
Вёрстка. Доступность интерфейсов и подходы к организации стилей.	Описание сайта для браузеров и поисковых машин. API RuTube. Виджеты-информеры. Теги <video> и <audio>. Разные типы селекторов. Стилизация полей ввода, кнопок и ярлыков. Псевдоклассы валидации. Стилизация выпадающих списков.	Решение и разбор задач.
Вёрстка. Адаптивность и графика.	Форматы растровой графики, оптимизация изображений, плотность пикселей. Переопределение переменных. Переменная внутри другой функции. Адаптив без медиазапросов. Синтаксис медиазапросов. Характеристики устройств. Выражения от контейнера. Синтаксис. Кросс-браузерность. Ветки в Git.	Решение и разбор задач.
Вёрстка - декорирование, подходы и инструменты.	Введение в векторную графику. Экспорт и оптимизация SVG. Стили внутри SVG. Переходы и 2D-трансформации.	Решение и разбор задач.
Введение в JavaScript, базовые типы, работа с DOM.	Вывод на страницу и в консоль, подключение JS к странице. Числа, операции с числами. Строки, конкатенация, приведение числа к строке. Переменные, типы переменных. Условия, больше-меньше, булевы операции. Знакомство с DOM. Введение. Управление содержимым: свойства .innerHTML и .textContent.	Решение и разбор задач.
Основы JavaScript. Сборка кода и модульность. Валидация форм.	Объединение и преобразование в строку. Добавление и удаление первого и последнего элементов. Управление элементами на любых позициях. Перебор массива. Методы forEach и map. Функции обратного вызова и их аргументы. Дебаггинг JavaScript. Типы ошибок. Поиск документации. Отладка через debugger.	Решение и разбор задач.
Работа с API в JS.	Асинхронные операции. Синхронные и асинхронные колбэки. Протокол HTTP. Запросы	Решение и разбор задач.

	из JavaScript. Формат JSON. HTTP-запрос и HTTP-ответ.	
Typescript.	Динамическая типизация. Самодокументированный код через JSDoc. Статическая типизация и TypeScript. Проверка типов во время исполнения. Полнота программного интерфейса. Приведение типов в TypeScript. Дженирики. Типизация DOM-элементов и их событий. Выбор инструментов для работы с нейросетями. Основы промт-инжиниринга. Поиск информации и объяснение теории. Отладка кода. Документирование. Генерация тестовых данных и автотестов.	Решение и разбор задач.

6. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине

6.1. Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы

Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
Введение в веб разработку - основы HTML и CSS, знакомство с JavaScript. Вёрстка. Семантика и лейаут.	Блоки, Отступы, Подпись к обложке. Массивы, Случайные числа, Функции с аргументами, Возвращаемое значение. Выбор и изменение элементов страницы. Реакция на действия пользователя. Булевы величины. Объекты. Подключение внешних библиотек. Циклы. Релиз.	Работа с основной и дополнительной учебной литературой, а также с интернет-источниками по теме, исследование, анализ, решение задач.
Вёрстка. Доступность интерфейсов и подходы к организации стилей.	Мета: настройки страницы. Фавикон. Разметка OpenGraph. Псевдоклассы и псевдоэлементы. Поля ввода с тегом <input>. Минимальные и максимальные значения. Поля загрузки, сброса и отправки данных. Поля ввода с другим синтаксисом. Ярлыки. Передаваемые значения. Поля множественного и единичного выбор.	Работа с основной и дополнительной учебной литературой, а также с интернет-источниками по теме, исследование, анализ, решение задач.
Вёрстка. Адаптивность и графика.	Растровая графика. Картинки на выбор (браузера). image-set() - загрузка фоновых изображений, в зависимости от условий. loading="lazy" — ленивая загрузка. Значение «по умолчанию» - «запасное значение» внутри переменной. Тёмная тема на CSS-переменных. Grid Layout - продолжение. Размеры неявных строк и колонок. Свойство grid-auto-flow. Функции minmax() и fit-content(). Параметры auto-fill и auto-fit.	Работа с основной и дополнительной учебной литературой, а также с интернет-источниками по теме, исследование, анализ, решение задач.
Вёрстка - декорирование, подходы и инструменты.	Анимации. Вариативные шрифты. Декорирование. 3D трансформации. Современные интерактивные элементы.	Работа с основной и дополнительной учебной литературой, а также с интернет-источниками по теме, исследование, анализ, решение задач.
Введение в JavaScript, базовые типы, работа с DOM.	Массивы, создание, доступ по индексу, длина массива. Циклы for, while, do. . . while. Функции: определение и вызов. Примитивные типы данных. Строки, числа и специальные числовые значения. Управление содержимым: свойства . innerHTML и . textContent. Создание, добавление и удаление	Работа с основной и дополнительной учебной литературой, а также с интернет-источниками по теме, исследование, анализ,

	элементов в DOM.	решение задач.
Основы JavaScript. Сборка кода и модульность. Валидация форм.	Отбор элементов массива: метод filter. Методы some, every, find. Сведение массива. Метод reduce. Сортировка массива. Способы создания функции. Стрелочные функции. Создание объектов и запись свойств. Операторы: delete, inПеребор свойствОбработка событий. Инкапсуляция и модулиЧто такое модуль и как его использоватьЛокальный серверБиблиотека пакетов NPM. Подключение сборщика Vite к проекту. Как подключать ресурсы в проекте с Vite и конфигурировать сборку. Связываем JS-методы валидации с DOM. Взаимодействие с другими элементами DOM.	Работа с основной и дополнительной учебной литературой, а также с интернет-источниками по теме, исследование, анализ, решение задач.
Работа с API в JS.	Таймеры. Event Loop. Promise. Работа с API. Введение. Инструменты: вкладка Network. Стандартная отправка формы.	Работа с основной и дополнительной учебной литературой, а также с интернет-источниками по теме, исследование, анализ, решение задач.
Typescript.	Добавляем TypeScript в проект. Настройка TypeScript. Инструментарий TS. Типизация стандартных объектов JS. Файлы деклараций d. ts. Использование и создание библиотек. Генерация фрагментов кода.	Работа с основной и дополнительной учебной литературой, а также с интернет-источниками по теме, исследование, анализ, решение задач.

6.2. Перечень вопросов, заданий, тем для подготовки к текущему контролю

Примерная тематика проектной работы 5 семестр:

"Адаптивный лендинг для реального бизнеса/события/продукта"

1. Лендинг для местного кафе/ресторана
2. Сайт-портфолио для фотографа/дизайнера
3. Промо-страница для технологического стартапа
4. Сайт для благотворительного мероприятия
5. Интернет-визитка для специалиста (резюме-сайт)
6. Лендинг для образовательного курса
7. Сайт для спортивного мероприятия
8. Промо нового гаджета/продукта

Пример задания проектной работы:

Проект: "Лендинг для экологичного стартапа EcoTech"

Контекст: Компания EcoTech запускает новый продукт — умную систему полива растений. Нужен продающий лендинг для привлечения первых клиентов.

Требования:

1. Адаптивность (обязательно):
 - Mobile-first подход
 - 3 breakpoints: mobile (320px), tablet (768px), desktop (1200px)
 - Удобная навигация на всех устройствах
2. Семантическая вёрстка:
 - Использовать минимум 10 различных семантических тегов
 - Корректная структура заголовков (h1-h6)

- Валидный HTML (проверить через validator.w3.org)

3. Стилизация:

- CSS-переменные для цветовой схемы (зелёная тематика)
- Flexbox для основных блоков
- Grid Layout для галереи или сетки преимуществ
- Анимации: плавные hover-эффекты, появление элементов при скролле

4. Доступность:

- ARIA-атрибуты для интерактивных элементов
- Контрастность текста не менее 4.5:1
- Полная клавиатурная навигация
- Оптимизация:
- Оптимизированные изображения (WebP с fallback)
- Минимизированные CSS файлы
- Ленивая загрузка изображений ниже fold

6. Интерактивность (базовый JavaScript):

- Работающая форма подписки на email
- Кнопка "Back to top"
- Переключение темы (dark/light mode)
- Галерея с табами/слайдером

Обязательные секции:

1. Hero-секция с главным заголовком и СТА-кнопкой
2. О продукте (3-4 преимущества с иконками)
3. Как это работает (таймлайн или шаги)
4. Галерея/демо продукта

5. Отзывы (карточки с аватарами)
6. Часто задаваемые вопросы (аккордеон)
7. Форма подписки/заявки
8. Футер с контактами и соцсетями

Примерная тематика проектной работы 6 семестр:

"Интерактивное веб-приложение с использованием JavaScript/TypeScript и ML-элементами"

1. Система рекомендаций фильмов/книг на основе предпочтений
2. Приложение для анализа настроения по тексту (sentiment analysis)
3. Платформа для изучения языка с ML-подсказками
4. Финансовый трекер с предсказанием расходов
5. Приложение для классификации изображений (растения/животные/объекты)
6. Генератор контента/текста с помощью AI
7. Система умных заметок с автоматической категоризацией
8. Игра с ML-противником или ML-помощником

Пример задания проектной работы:

Проект: "PlantID — приложение для определения растений по фото"

Контекст: Мобильное веб-приложение для садоводов-любителей, позволяющее сфотографировать растение и получить информацию о нём.

Технический стек:

- React + TypeScript
- TensorFlow.js для ML-модели
- CSS Modules/Styled Components
- Vite для сборки

- Git для контроля версий

Основные требования:

1. ML-компонент (обязательно):

- Интеграция готовой модели классификации изображений (можно использовать Teachable Machine или предобученную модель)
- Загрузка/съёмка фото через интерфейс
- Отображение результата с уверенностью модели (%)
- История предыдущих определений

2. Фронтенд-архитектура:

- Компонентный подход с TypeScript-типизацией
- State management (Context API или Zustand)
- Маршрутизация (минимум 3 страницы)
- Обработка ошибок и loading states

3. Интерфейс:

- Progressive Web App (PWA) возможности
- Оффлайн-работа (кеширование модели и данных)
- Адаптивный дизайн
- Доступность (WCAG 2.1 Level AA)

4. Работа с данными:

- Хранение истории в localStorage/IndexedDB
- Интеграция с API растений (например, Trefle API)
- Кэширование запросов
- Экспорт данных (JSON/CSV)

5. Дополнительные функции (на выбор 3 из 5):

- Система рекомендаций по уходу за растением
- Календарь полива/удобрения
- Сообщество: обмен фотографиями и советами
- AR-режим: наложение информации на камеру
- Умные уведомления (напоминания по уходу)

Структура приложения:

1. Главная страница:

- Камера/загрузка фото
- Быстрые действия
- Последние определения

2. Страница результата:

- Информация о растении
- Галерея фото
- Уход и рекомендации
- Сохранение в "Мой сад"

3. "Мой сад":

- Коллекция растений пользователя
- Календарь ухода
- Прогресс роста

4. Сообщество:

- Лента фотографий
- Обсуждения
- Экспертные советы

5. Настройки:

- Уведомления
- Синхронизация данных
- О приложении

Технические требования:

- Модульные тесты (минимум 70% покрытие)
- Документация компонентов (Storybook или аналоги)
- CI/CD пайплайн (GitHub Actions)
- Оптимизация производительности (Lighthouse score > 90)
- Деплой на бесплатный хостинг (Vercel/Netlify)

Документация (обязательно):

1. Техническое задание (1-2 страницы)
2. Архитектурная документация (диаграмма компонентов)
3. README с инструкциями по запуску
4. API документация (если есть свой backend)
5. Пользовательская инструкция

Командная работа: Рекомендуется в группах по 2-3 человека

Критерии балльной оценки различных форм текущего контроля успеваемости содержатся в соответствующих методических рекомендациях Кафедры информационных технологий Факультета информационных технологий и анализа больших данных.

Дополнительная информация

Различия между проектами:

Критерий	Проект 5 семестр	Проект 6 семестр
Фокус	Вёрстка, адаптивность	Полное приложение с логикой
Технологии	HTML, CSS, базовый JS	React, TypeScript, ML
Сложность	Статичная страница	Динамичное приложение

ML-элементы	Нет	Обязательно
Командная работа	Индивидуально	Групповой проект
Документация	Базовая	Полный комплект
Тестирование	Ручное тестирование	Автоматические тесты
Деплой	GitHub Pages	CI/CD на облачном хостинге

7. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине

Перечень компетенций с указанием индикаторов их достижения в процессе освоения образовательной программы содержится в разделе 2. *Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине.*

Типовые контрольные задания или иные материалы, необходимые для оценки индикаторов достижения компетенций, умений и знаний

ПКП-5 Способность проектировать и реализовывать интеллектуальные информационные системы

1) Демонстрирует знания основных методов машинного обучения и интеллектуального анализа данных, применяет готовые инструменты для создания интеллектуальных алгоритмов

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: основные методы ML (классификация, регрессия, кластеризация) и готовые инструменты (TensorFlow.js, ML5.js) для их реализации в веб-приложениях

Уметь: применять готовые ML-библиотеки на JavaScript для решения типовых задач (распознавание изображений, анализ текста) в веб-интерфейсах

Типовые контрольные задания

"Интеграция модели распознавания изображений в React-приложение".

Использовать TensorFlow.js или готовую модель Teachable Machine.

Создать компонент для загрузки изображения.

Вывести результат классификации (например, "кошка/собака/ничего").

Объяснить в комментариях, какой алгоритм используется.

2) Понимает особенности интеллектуальных информационных систем в части операций разработки, развертывания и сопровождения

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: особенности жизненного цикла ML-приложений (обучение моделей, инференс, A/B тестирование, мониторинг дрейфа данных) в контексте веб-разработки

Уметь: организовывать процесс разработки и деплоя веб-приложений с ML-компонентами, включая версионирование моделей и метрик

Типовые контрольные задания

"План развертывания ML-веб-приложения".

Создать схему пайплайна: данные → обучение → API → фронтенд.

Описать 3 метрики для мониторинга качества модели в продакшене.

Предложить стратегию A/B тестирования новой модели.

Написать checklist деплоя (5 пунктов).

3) Адаптирует практики создания программных продуктов, в том числе командные, для интеллектуальных информационных систем

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: особенности agile-практик (спринты, стендапы, ретроспективы) при разработке ML-проектов и специфику командных ролей (data scientist, ML engineer, frontend developer)

Уметь: адаптировать процессы код-ревью, тестирования и CI/CD для проектов, содержащих как код, так и ML-модели

Типовые контрольные задания

Создать структуру папок для кода, моделей, данных, документации.

Написать gitignore для ML-проекта (.pkl, .h5, большие датасеты).

Создать шаблон PR (pull request) с чек-листом для ML-изменений.

Предложить стратегию ветвления (git flow) для параллельной работы над моделями и интерфейсом.

4) Организует сбор и подготовку данных для систем машинного обучения, в том числе потоковых, онлайн обучения

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: методы сбора, разметки, аугментации и нормализации данных для обучения моделей в браузерной среде

Уметь: создавать инструменты сбора пользовательских данных (аннотации, фидбэк) и подготавливать их для обучения или дообучения моделей

Типовые контрольные задания

Создать интерфейс для разметки изображений (3 класса).

Реализовать сбор и экспорт размеченных данных в JSON.

Добавить валидацию качества разметки (проверка противоречий).

Предложить механизм обратной связи для улучшения модели.

ПКП-6 Способность вести разработку программных систем в команде, вести эффективную коммуникацию

1) Придерживается рекомендованного в команде стиля написания кода, проводит рефакторинг написанного кода, в том числе, другими членами команды, проводит код-ревью

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: стандарты и соглашения по написанию кода на JavaScript/TypeScript (ESLint, Prettier), принципы чистого кода и паттерны рефакторинга

Уметь: проводить код-ревью по чек-листу, предлагать улучшения кода, проводить рефакторинг существующего кода, с сохранением функциональности

Типовые контрольные задания

"Код-ревью компонента React".

Дан грязный компонент с 5+ проблемами (пропсы апу, сложная функция, нет ключей и т.д.).

Найти все проблемы и классифицировать (стиль, производительность, баги).

Написать конструктивные комментарии для коллеги.

Предложить исправленный вариант кода.

2) Документирует код, архитектуру, дизайн, требования к программной системе с использованием принятой в команде системы технических стандартов

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: виды технической документации (JSDoc, архитектурные решения ADR, user stories, диаграммы) и инструменты для их ведения (Swagger, Storybook, Miro)

Уметь: создавать документацию разных уровней: от инлайн-комментариев до архитектурных решений и пользовательских инструкций

Типовые контрольные задания

Написать JSDoc для всех функций и компонентов.

Создать ADR (Architecture Decision Record) о выборе JWT.

Нарисовать схему flow авторизации (можно ASCII-art).

Составить user story для тестировщика.

3) Использует инструментальные средства командной разработки программного обеспечения, контроля версий, отслеживания ошибок, планирования процессов разработки в зависимости от принятой в команде методологии разработки

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: инструменты и практики Git (ветвление, мерж, конфликты), таск-трекеры (Jira, Trello), CI/CD (GitHub Actions), и методологии (Scrum, Kanban)

Уметь: эффективно использовать Git в команде, вести таски в трекере, настраивать базовые CI-пайплайны и участвовать в agile-ритуалах

Типовые контрольные задания

"Настройка проекта для командной работы"

Создать репозиторий с ветками main, develop, feature/*.

Настроить GitHub Actions для линтинга и тестов.

Создать шаблон issues и pull requests.

Составить доску Kanban в проектах GitHub (To Do, In Progress, Review, Done).

Написать git-алманах для команды (5 основных команд).

4) Выстраивает эффективную двустороннюю коммуникацию с нетехническими специалистами по предмету разработки предметной области в целом

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: принципы нетехнической коммуникации: избегание жаргона, визуализация, storytelling, активное слушание, техники вопросов

Уметь: объяснять технические решения бизнесу, формулировать требования, проводить презентации продукта, задавать уточняющие вопросы и резюмировать договоренности

Типовые контрольные задания

"Презентация фичи для продукт-менеджера".

Объяснить техническую фичу (например, "ленивая загрузка") без терминов.

Нарисовать схему на доске (текстовое описание).

Подготовить 3 аргумента "за" для бизнеса (скорость, конверсия, etc.).

Составить 3 вопроса к заказчику для уточнения требований.

Резюмировать договоренности встречи (5 предложений).

5) Коммуницирует задачи разработки между членами команды, оценивает трудоемкость и сложность выполнения задач разработки, ставит задачи и контролирует их выполнение, в том числе в письменной формализованной форме

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: методы оценки задач (story points, часы), техники декомпозиции (user story → tasks), форматы постановки задач (SMART, INVEST), инструменты контроля (daily sync, отчеты)

Уметь: декомпонировать фичи на задачи, оценивать их сложность, ставить задачи коллегам, отслеживать прогресс и проводить демо результатов

Типовые контрольные задания

"Планирование спринта".

Декомпонировать user story "Поиск по каталогу" на 5-7 технических задач.

Оценить каждую задачу в story points (фибоначчи).

Распределить задачи между 3 разработчиками (роли: frontend, backend, ML).

Составить daily standup отчет по шаблону (что сделал, что буду делать, блокеры).

Написать критерии приемки для одной задачи.

Примеры практико-ориентированных заданий

5 семестр

1. Создать карточку товара (HTML + CSS)

Сверстайте карточку товара: изображение, название, описание, цена и кнопка «Купить». Используйте семантические теги (`<article>`, `<img>`, `<button>`). Стилизируйте: карточку шириной 300px, тень, скруглённые углы, кнопка с цветным фоном и hover-эффектом.

2. Переключение темы (CSS + JS)

Создайте кнопку «Тёмная тема». При клике:

Добавляется класс `dark-theme` к тегу `<body>`.

В CSS заданы переменные для светлой и тёмной тем (цвет фона, текста).

Тема сохраняется в `localStorage` и восстанавливается при загрузке страницы.

3. Валидация пароля (JS)

Поле ввода пароля с кнопкой «Проверить». Пароль считается валидным, если:

Длина ≥ 8 символов

Есть хотя бы одна цифра

Есть хотя бы одна заглавная буква

При проверке выводите под полем сообщение: «Слабый», «Средний» или «Надёжный» (в зависимости от выполнения условий).

4. Адаптивная сетка изображений (CSS Grid)

Создайте сетку из 6 изображений (используйте `placeholder`, например <https://picsum.photos/300/200?random=1>). На десктопе — 3 колонки, на планшете — 2, на мобильном — 1. Добавьте плавный hover-эффект (увеличение изображения на 10% и тень).

6 семестр

1. Счётчик с ограничением (JS)

Создайте счётчик: кнопки «+» и «-», отображение текущего значения. Диапазон: от 0 до 10. При достижении минимума/максимума соответствующая кнопка должна становиться неактивной (disabled). Значение сохранять в localStorage.

2. Поиск по массиву (JS)

Дан массив объектов книг: [{title: 'Война и мир', author: 'Толстой'}, ...]. Поле ввода для поиска. При вводе текста фильтруйте массив по совпадению в названии или авторе и динамически выводите результат списком (). Если ничего не найдено — покажите «Ничего не найдено».

3. Анимация загрузки (CSS)

Создайте индикатор загрузки: три пульсирующих кружка разного цвета, которые последовательно увеличиваются и уменьшаются с задержкой. Анимация бесконечная. Используйте @keyframes и animation-delay.

Примерные вопросы для подготовки к зачету, экзамену

Примерные вопросы для подготовки к зачету

1. Что такое HTML и какова его основная роль?
2. Что такое CSS и для чего он используется?
3. Как связать HTML и CSS через тег <style>?
4. Как связать HTML и CSS через внешний CSS-файл?
5. Как подключить JavaScript к HTML-документу?
6. Опишите базовую структуру HTML-документа.
7. Что такое блочная модель CSS (блоки, отступы)?
8. Как выбрать элемент страницы с помощью JavaScript?
9. Какие способы реакции на действия пользователя (события) вы знаете?
10. Что такое булевы величины и как они используются в условиях?
11. Что такое массив в JavaScript и как с ним работать?

12. Что такое функция, аргументы и возвращаемое значение?
13. Что такое объект в JavaScript? Приведите пример.
14. Как подключить внешнюю JavaScript-библиотеку?
15. Какие циклы существуют в JavaScript (for, while)?
16. Что такое «релиз» в контексте веб-разработки?
17. Каковы основные принципы организации структуры файлов и путей в проекте?
18. Каковы основные возможности Инструментов разработчика (DevTools) в браузере?
19. Что такое семантическая вёрстка? Приведите примеры семантических тегов.
20. Что такое глобальные атрибуты в HTML? Приведите примеры.
21. Какие CSS-свойства используются для работы с типографикой и шрифтами?
22. В чём суть флексбокс-вёрстки (Flexbox)? Опишите основные свойства.
23. Что такое позиционирование элементов в CSS? Перечислите значения свойства position.
24. В чём суть Grid Layout? Чем он отличается от Flexbox?
25. Что такое кодстайл и зачем он нужен?
26. Каковы базовые команды Bash для навигации по файловой системе?
27. Что такое Git? Опишите базовый рабочий процесс (инициализация, коммит).
28. Какую информацию содержат мета-теги в <head> и зачем они нужны?
29. Как описать сайт для поисковых машин (SEO-базовые принципы)?
30. Как встроить видео или аудио на страницу с помощью HTML5 тегов?
31. Какие типы CSS-селекторов вы знаете? (Приведите примеры сложных селекторов).
32. Как стилизовать поля ввода, кнопки и ярлыки (<label>)?
33. Как стилизовать выпадающие списки (<select>) и применить псевдоклассы валидации?
34. Что такое фавикон и как его подключить?
35. Что такое разметка OpenGraph и для чего она используется?

36. В чём разница между псевдоклассами (например, `:hover`) и псевдоэлементами (например, `::before`)?
37. Какие существуют типы полей ввода (`<input>`)? Опишите их назначение.
38. Как задать минимальные и максимальные значения для числового поля?
39. Как работают поля для загрузки файлов, сброса и отправки данных формы?
40. Что такое `<iframe>` и каковы основные сферы его применения?
41. Какие свойства дополняют блочную модель (например, `box-sizing`)?
42. Что такое доступность (`a11y`) интерфейсов? Какие основные принципы и атрибуты (например, `aria-*`) вы знаете?
43. Опишите базовую разметку HTML-формы.
44. В чём разница между резиновой и адаптивной вёрсткой?
45. Что такое растровая графика? Какие основные форматы вы знаете?
46. Как оптимизировать растровые изображения для веба?
47. Что такое плотность пикселей (DPI/PPI) и как она влияет на отображение картинок?
48. Что такое CSS-переменные (`custom properties`) и как их использовать?
49. Как переопределить CSS-переменную внутри другой функции/селектора?
50. Что такое медиазапросы (`@media`)? Опишите их синтаксис.
51. Какие характеристики устройств можно использовать в медиазапросах?
52. Что такое контейнерные запросы (`@container`)? Чем они отличаются от медиазапросов?
53. Что такое кросс-браузерность и как её добиваться?
54. Как работают ветки (`branches`) в Git?
55. Как использовать свойство `image-set()` для адаптивной загрузки фоновых изображений?
56. Что такое ленивая загрузка изображений (`loading="lazy"`) и зачем она нужна?
57. Как реализовать тёмную тему на сайте с помощью CSS-переменных?

58. Опишите продвинутые возможности Grid Layout: `grid-auto-flow`, `minmax()`, `fit-content()`.
59. В чём разница между параметрами `auto-fill` и `auto-fit` в Grid Layout?
60. Какие современные единицы измерения (например, `vh`, `vw`, `vmin`, `clamp()`) вы знаете?
61. Что такое SVG и в чём его преимущества перед растровой графикой?
62. Как экспортировать и оптимизировать SVG для веба?
63. Как стилизовать элементы внутри SVG?
64. Что такое CSS-переходы (`transition`)? Приведите пример.
65. Что такое 2D-трансформации (`transform`)? Какие функции вы знаете?
66. Как создать CSS-анимацию с помощью `@keyframes`?
67. Что такое вариативные шрифты (`variable fonts`) и как их использовать?
68. Какие CSS-свойства используются для декорирования (например, `box-shadow`, `border-radius`, `gradients`)?
69. Что такое 3D-трансформации в CSS? Приведите пример.
70. Какие современные интерактивные элементы (например, `<details>`, `<dialog>`) вы знаете?

Примерные вопросы для подготовки к экзамену

1. Что такое HTML и какова его основная роль?
2. Что такое CSS и для чего он используется?
3. Как связать HTML и CSS через тег `<style>`?
4. Как связать HTML и CSS через внешний CSS-файл?
5. Как подключить JavaScript к HTML-документу?
6. Опишите базовую структуру HTML-документа.
7. Что такое блочная модель CSS (блоки, отступы)?
8. Как выбрать элемент страницы с помощью JavaScript?
9. Какие способы реакции на действия пользователя (события) вы знаете?

10. Что такое булевы величины и как они используются в условиях?
11. Что такое массив в JavaScript и как с ним работать?
12. Что такое функция, аргументы и возвращаемое значение?
13. Что такое объект в JavaScript? Приведите пример.
14. Как подключить внешнюю JavaScript-библиотеку?
15. Какие циклы существуют в JavaScript (for, while)?
16. Что такое «релиз» в контексте веб-разработки?
17. Каковы основные принципы организации структуры файлов и путей в проекте?
18. Каковы основные возможности Инструментов разработчика (DevTools) в браузере?
19. Что такое семантическая вёрстка? Приведите примеры семантических тегов.
20. Что такое глобальные атрибуты в HTML? Приведите примеры.
21. Какие CSS-свойства используются для работы с типографикой и шрифтами?
22. В чём суть флексбокс-вёрстки (Flexbox)? Опишите основные свойства.
23. Что такое позиционирование элементов в CSS? Перечислите значения свойства position.
24. В чём суть Grid Layout? Чем он отличается от Flexbox?
25. Что такое кодстайл и зачем он нужен?
26. Каковы базовые команды Bash для навигации по файловой системе?
27. Что такое Git? Опишите базовый рабочий процесс (инициализация, коммит).
28. Какую информацию содержат мета-теги в <head> и зачем они нужны?
29. Как описать сайт для поисковых машин (SEO-базовые принципы)?
30. Как встроить видео или аудио на страницу с помощью HTML5 тегов?
31. Какие типы CSS-селекторов вы знаете? (Приведите примеры сложных селекторов).
32. Как стилизовать поля ввода, кнопки и ярлыки (<label>)?
33. Как стилизовать выпадающие списки (<select>) и применить псевдоклассы валидации?

34. Что такое фавикон и как его подключить?
35. Что такое разметка OpenGraph и для чего она используется?
36. В чём разница между псевдоклассами (например, :hover) и псевдоэлементами (например, ::before)?
37. Какие существуют типы полей ввода (<input>)? Опишите их назначение.
38. Как задать минимальные и максимальные значения для числового поля?
39. Как работают поля для загрузки файлов, сброса и отправки данных формы?
40. Что такое <iframe> и каковы основные сферы его применения?
41. Какие свойства дополняют блочную модель (например, box-sizing)?
42. Что такое доступность (a11y) интерфейсов? Какие основные принципы и атрибуты (например, aria-*) вы знаете?
43. Опишите базовую разметку HTML-формы.
44. В чём разница между резиновой и адаптивной вёрсткой?
45. Что такое растровая графика? Какие основные форматы вы знаете?
46. Как оптимизировать растровые изображения для веба?
47. Что такое плотность пикселей (DPI/PPI) и как она влияет на отображение картинок?
48. Что такое CSS-переменные (custom properties) и как их использовать?
49. Как переопределить CSS-переменную внутри другой функции/селектора?
50. Что такое медиазапросы (@media)? Опишите их синтаксис.
51. Какие характеристики устройств можно использовать в медиазапросах?
52. Что такое контейнерные запросы (@container)? Чем они отличаются от медиазапросов?
53. Что такое кросс-браузерность и как её добиваться?
54. Как работают ветки (branches) в Git?
55. Как использовать свойство image-set() для адаптивной загрузки фоновых изображений?

56. Что такое ленивая загрузка изображений (`loading="lazy"`) и зачем она нужна?
57. Как реализовать тёмную тему на сайте с помощью CSS-переменных?
58. Опишите продвинутые возможности Grid Layout: `grid-auto-flow`, `minmax()`, `fit-content()`.
59. В чём разница между параметрами `auto-fill` и `auto-fit` в Grid Layout?
60. Какие современные единицы измерения (например, `vh`, `vw`, `vmin`, `clamp()`) вы знаете?
61. Что такое SVG и в чём его преимущества перед растровой графикой?
62. Как экспортировать и оптимизировать SVG для веба?
63. Как стилизовать элементы внутри SVG?
64. Что такое CSS-переходы (`transition`)? Приведите пример.
65. Что такое 2D-трансформации (`transform`)? Какие функции вы знаете?
66. Как создать CSS-анимацию с помощью `@keyframes`?
67. Что такое вариативные шрифты (`variable fonts`) и как их использовать?
68. Какие CSS-свойства используются для декорирования (например, `box-shadow`, `border-radius`, `gradients`)?
69. Что такое 3D-трансформации в CSS? Приведите пример.
70. Какие современные интерактивные элементы (например, `<details>`, `<dialog>`) вы знаете?
71. Что такое JavaScript и какова его основная роль в веб-разработке?
72. Назовите основные примитивные типы данных в JavaScript.
73. Что такое DOM (Document Object Model)?
74. Как выбрать элемент из DOM с помощью JavaScript?
75. В чём разница между свойствами `.innerHTML` и `.textContent`?
76. Как создать новый элемент и добавить его в DOM?
77. Как удалить элемент из DOM?
78. Как работают условные операторы (`if...else`, `switch`) в JavaScript?

79. Что такое массив? Как получить доступ к его элементам и узнать длину?
80. Опишите работу циклов `for`, `while`, `do...while`.
81. Что такое функция? Как её объявить и вызвать?
82. Какие операции можно производить со строками и числами?
83. Какие методы работы с массивами вы знаете (например, для добавления/удаления элементов)?
84. Что такое `localStorage` и `sessionStorage`? Чем они отличаются?
85. Опишите структуру и логику простого To-do приложения.
86. Что такое дебаггинг и какие основные типы ошибок в JavaScript вы знаете?
87. Как использовать `debugger` и консоль разработчика для отладки?
88. Что такое модули в JavaScript (`import/export`)?
89. Что такое сборщик кода (на примере Vite)? Зачем он нужен?
90. Как подключить и настроить Vite в проекте?
91. Опишите методы массивов: `forEach`, `map`, `filter`, `reduce`, `find`, `sort`. Чем они отличаются?
92. Что такое стрелочные функции? Чем они отличаются от обычных?
93. Как создать объект, добавить, удалить (`delete`) или перебрать его свойства?
94. Что такое обработка событий (Event Handling) в JavaScript?
95. Что такое валидация форм? Как связать JavaScript-логику валидации с DOM-элементами формы?
96. Что такое регулярные выражения (RegExp) и для чего они используются при валидации?
97. В чём разница между синхронным и асинхронным кодом?
98. Что такое Event Loop в JavaScript?
99. Что такое Promise («Обещание»)? Как обработать его с помощью `.then()/catch()` или `async/await`?
100. Что такое API? Приведите примеры.

101. Что такое HTTP-протокол? Назовите основные методы запросов (GET, POST и др.).
102. Как отправить HTTP-запрос из JavaScript (например, с помощью fetch)?
103. Что такое формат JSON? Как преобразовать объект JavaScript в JSON и обратно?
104. Как анализировать сетевые запросы во вкладке Network инструментов разработчика?
105. Как происходит стандартная отправка HTML-формы?
106. Что такое TypeScript и какие проблемы он решает по сравнению с JavaScript?
107. В чём разница между статической (TypeScript) и динамической (JavaScript) типизацией?
108. Как добавить TypeScript в существующий проект?
109. Что такое файл tsconfig.json и для чего он нужен?
110. Что такое дженерики (Generics) в TypeScript? Приведите пример.
111. Как типизировать DOM-элементы и их события в TypeScript?
112. Что такое файлы деклараций (*.d.ts) и зачем они нужны?
113. Как с помощью JSDoc сделать код самодокументированным?
114. Какие инструменты для работы с нейросетями в помощь разработчику вы знаете?
115. Что такое промт-инжиниринг? Сформулируйте базовые принципы составления промтов.
116. Как можно использовать ИИ для поиска информации и объяснения сложных концепций?
117. Как ИИ может помочь в планировании этапов работы над проектом?
118. Какие есть риски и лучшие практики при генерации фрагментов кода с помощью ИИ?
119. Как можно использовать ИИ для отладки и объяснения ошибок в коде?
120. Как ИИ может помочь в документировании кода или проекта?
121. Как можно использовать ИИ для генерации тестовых данных или написания автотестов?

**Промежуточная аттестация обучающихся проводится в формате тестирования*

8. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины

Основная литература:

1. Шаталова, А.Ю. Основы веб-разработки : Учебное пособие / А.Ю. Шаталова, М.В. Коротеев Электрон. дан. Москва : КноРус, 2025 282 с. Режим доступа: book.ru Internet access <https://book.ru/book/957272> ISBN 978-5-406-14458-9. [БИК ID: RU\bookru\bibl\957272]
2. Диков, А. В. Клиентские технологии веб-дизайна. HTML5 и CSS3 [Электронный ресурс] : учебное пособие для вузов / Диков А. В. 2-е изд., стер. Санкт-Петербург : Лань, 2023 188 с. Книга из коллекции Лань - Информатика <https://e.lanbook.com/book/318443> ISBN 978-5-507-46740-2. [БИК ID: RU-LAN-BOOK-318443]

Дополнительная литература:

1. Полуэктова, Наталия Робертовна Разработка веб-приложений : учебник для вузов / Н. Р. Полуэктова. 2-е изд. Электрон. дан. Москва : Юрайт, 2025 204 с (Высшее образование) URL: <https://urait.ru/bcode/567610> (дата обращения: 01.09.2025). Режим доступа: Электронно-библиотечная система Юрайт, для авториз. пользователей <https://urait.ru/bcode/567610> ISBN 978-5-534-18645-1 : 889.00. [БИК ID: RU2fURAIT2f567610]
2. Никулова, Г. А. Web-технологии: Введение в программирование на JavaScript: защита контента средствами JS и CSS [Электронный ресурс] : учебно-методическое пособие / Никулова Г. А., Терлецкий А. С. Липецк : Липецкий ГПУ, 2023 76 с. Книга из коллекции Липецкий ГПУ - Информатика <https://e.lanbook.com/book/403757> ISBN 978-5-907792-00-5. [БИК ID: RU-LAN-BOOK-403757]

9. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. •Электронно-библиотечная система BOOK.RU <http://www.book.ru>
2. •Электронная библиотека Финансового университета (ЭБ) <http://elib.fa.ru/>
3. •Электронно-библиотечная система «Университетская библиотека ОНЛАЙН» <http://biblioclub.ru/>

4. •Электронно-библиотечная система издательства «ЮРАЙТ» <https://urait.ru/>
5. •Электронно-библиотечная система издательства Проспект <http://ebs.prospekt.org/books>
6. •Электронно-библиотечная система издательства Лань <https://e.lanbook.com/>
7. •Деловая онлайн-библиотека Alpina Digital <http://lib.alpinadigital.ru/>
8. •Научная электронная библиотека eLibrary.ru <http://elibrary.ru>
9. •Национальная электронная библиотека <http://нэб.рф/>

10. Методические указания для обучающихся по освоению дисциплины

При изучении теоретического материала необходимо опираться на рабочую программу дисциплины, материалы лекций и литературу из основного списка. Кроме этого, необходимо активно работать с Интернет-источниками и пособиями других авторов, помогающими усвоить материал отдельных разделов программы.

Необходимо конспектировать лекции, пометая сложные и непонятные моменты с тем, чтобы задать вопросы лектору в конце лекции или же на консультации.

При подготовке к семинарским занятиям необходимо изучить вопросы, вынесенные на самостоятельное изучение, так как семинарские занятия предполагают их обсуждение и дискуссию по теме; кроме того, задания для самостоятельной работы необходимы для того, чтобы успешно выполнить самостоятельные задания на семинарах.

Индивидуальные задания для работы на компьютере, файлы с выполненными заданиями необходимо хранить в личной сетевой папке в компьютерной сети вуза.

Рекомендации по выполнению проектных работ:

Общие рекомендации:

1. Планирование и организация markdown

****Недельный план для любого проекта:****

Неделя 1: Анализ и планирование

- День 1-2: Изучение ТЗ, составление вопросов
- День 3: Исследование аналогичных проектов
- День 4: Создание структуры проекта
- День 5: Планирование этапов
- День 6-7: Создание макета/прототипа

Неделя 2: Базовая реализация

- День 1-3: Верстка/кодинг основных компонентов
- День 4: Интеграция частей
- День 5: Базовая стилизация

- День 6: Первое тестирование
- День 7: Анализ прогресса, корректировка плана

Неделя 3: Доработка и оптимизация

- День 1-2: Дополнительные функции
- День 3: Адаптивность и кроссбраузерность
- День 4: Оптимизация производительности
- День 5: Тестирование на разных устройствах
- День 6: Исправление багов
- День 7: Подготовка к сдаче

Неделя 4: Финальная полировка

- День 1: Проверка всех требований ТЗ
- День 2: Оптимизация загрузки
- День 3: Написание документации
- День 4: Подготовка презентации
- День 5: Финальное тестирование
- День 6-7: Буфер на непредвиденное

2. Инструменты и окружение markdown

****Обязательный инструментарий:****

1. Редактор кода: VS Code с расширениями:

- Live Server / Live Preview
- ESLint / Prettier
- Auto Rename Tag
- Color Highlight
- GitLens

2. Браузерные инструменты:

- DevTools Chrome/Firefox

- Lighthouse для аудита

- Responsive Design Mode

3. Дизайн и прототипирование:

- Figma (бесплатно для образования)

- Coolors для цветовых палитр

- Google Fonts для шрифтов

4. Оптимизация:

- TinyPNG для изображений

- SVGO для SVG

- Can I Use для проверки поддержки

5. Документация:

- Markdown редактор

- Draw.io для диаграмм

- ScreenToGif для записи демо

3. Работа с Git (обязательно для обоих проектов) markdown

****Рекомендуемый Git Workflow:****

1. Инициализация:

```
git init
```

```
git add .
```

```
git commit -m "Initial commit"
```

2. Структура веток:

main (только стабильные версии)

develop (основная разработка)

feature/* (для новых функций)

bugfix/* (для исправлений)

3. Коммиты:

- Делайте маленькие коммиты (одна задача = один коммит)
- Пишите понятные сообщения на английском/русском
- Используйте конвенцию: тип(область): сообщение

Примеры: feat(auth): add login form

fix(styles): correct mobile menu

docs(readme): update installation

4. .gitignore обязательно должен содержать:

node_modules/

.DS_Store

*.log

build/

dist/

.env

Рекомендации для проектной работы 5 семестра

1. Пошаговый подход к вёрстке markdown

****Порядок работы:****

ШАГ 1: Подготовка (1 день)

- Создать структуру папок:

project/

├── index.html

├── css/

| ├── style.css

| └── reset.css

```
|   └── variables.css
|── js/
|   └── main.js
|── images/ (оптимизированные!)
|── fonts/
└── README.md
```

ШАГ 2: HTML-структура (1-2 дня)

- Начинайте с семантических тегов
- Сначала desktop-версия
- Добавляйте комментарии к секциям
- Проверяйте через validator.w3.org

ШАГ 3: Базовая стилизация (2 дня)

- Сначала сброс стилей (`reset.css`)
- CSS-переменные для цветов/шрифтов
- Mobile-first media queries
- Flexbox для основных блоков

ШАГ 4: Адаптивность (1 день)

- Тестируйте на 3+ размерах экрана
- Используйте относительные единицы (`%`, `rem`)
- Скрывайте/преобразовывайте элементы для mobile

ШАГ 5: Детализация (2 дня)

- Анимации и переходы
- Оптимизация изображений
- Подключение шрифтов
- Формы и интерактивность

ШАГ 6: Оптимизация (1 день)

- Проверка Lighthouse
- Минимизация CSS/JS
- Ленивая загрузка изображений
- Финальное тестирование

2. Чек-лист качества markdown

****Что проверить перед сдачей:****

- ☐ HTML валиден (validator.w3.org)
- ☐ Все изображения оптимизированы (<200KB каждая)
- ☐ Сайт работает без JavaScript (прогрессивное улучшение)
- ☐ Контрастность текста соответствует WCAG
- ☐ Все интерактивные элементы доступны с клавиатуры
- ☐ Нет горизонтального скrolла на mobile
- ☐ Шрифты загружаются правильно
- ☐ Favicon подключен
- ☐ Meta-теги заполнены (title, description, viewport)
- ☐ Сайт работает в последних версиях Chrome, Firefox, Safari
- ☐ Страница загружается за <3 секунд на 3G
- ☐ Кнопки имеют достаточную область клика (min 44px)
- ☐ Формы имеют правильные типы полей и placeholder
- ☐ Все внешние ссылки открываются в новой вкладке
- ☐ Нет консольных ошибок
- ☐ README.md заполнен

3. Частые ошибки и как их избежать markdown

****Типичные проблемы новичков:****

1. "Блок съезжает на маленьких экранах"

Решение: Используйте `max-width: 100%` для изображений и медиа-контейнеров

2. "Не работают media queries"

Проверьте: `<meta name="viewport" content="width=device-width, initial-scale=1">`

3. "Шрифты тормозят загрузку"

Оптимизация: Используйте `font-display: swap`; и подключайте только нужные начертания

4. "Цвета выглядят по-разному в браузерах"

Решение: Используйте современные цветовые пространства (`oklab`, `display-p3`)

5. "Анимации лагают"

Оптимизация: Используйте `transform` и `opacity` вместо `top/left`

Рекомендации по выполнению проекта 6 семестра:

1. Архитектурный подход markdown

****Структура React + TypeScript проекта:****

project/

```
├── public/
|   ├── index.html
|   ├── favicon.ico
|   └── manifest.json
├── src/
|   ├── components/
|   |   ├── common/ (кнопки, инпуты)
|   |   ├── layout/ (хедер, футер)
|   |   └── features/ (специфичные компоненты)
|   ├── pages/ (страницы приложения)
|   └── hooks/ (кастомные хуки)
```



```
| |— utils/ (вспомогательные функции)
| |— services/ (работа с API)
| |— types/ (TypeScript типы)
| |— styles/ (глобальные стили)
| |— assets/ (изображения, шрифты)
| |— App.tsx
| |— main.tsx
|— tests/
|— docs/
|— .eslintrc.js
|— .prettierrc
|— tsconfig.json
|— vite.config.ts
|— package.json
```

2. Работа с ML-компонентами markdown

****Интеграция машинного обучения:****

ШАГ 1: Выбор подхода

- Вариант А: Готовые ML-API (OpenAI, Google Vision) — проще, требует интернет
- Вариант В: TensorFlow.js в браузере — сложнее, работает оффлайн
- Вариант С: Комбинированный — базовый функционал оффлайн + улучшения онлайн

ШАГ 2: Оптимизация модели

1. Квантование модели (уменьшение размера)
2. Ленивая загрузка (не грузить пока не нужно)
3. Web Workers для тяжёлых вычислений
4. Кэширование результатов

ШАГ 3: UX для ML-функций

- Показывайте индикатор загрузки
- Отображайте уверенность модели (в %)
- Предусмотрите fallback-варианты
- Добавьте возможность корректировки результатов

ШАГ 4: Этические аспекты

- Информируйте пользователя об использовании AI
- Не собирайте личные данные без согласия
- Проверяйте bias модели
- Давайте возможность отключить ML-функции

3. Чек-лист сложного проекта

markdown

****Расширенный чек-лист для итогового проекта:****

[] Архитектура:

- [] Проект использует feature-based структуру
- [] TypeScript типы покрывают $\geq 80\%$ кода
- [] Нет циклических зависимостей

[] Производительность:

- [] First Contentful Paint < 1.5s
- [] Largest Contentful Paint < 2.5s
- [] Cumulative Layout Shift < 0.1
- [] Bundle size < 500KB (gzipped)
- [] Используется code splitting
- [] Оптимизированные изображения (WebP)

[] ML-компонент:

- [] Модель загружается асинхронно
- [] Показывается индикатор загрузки
- [] Результаты кэшируются
- [] Есть обработка ошибок
- [] Модель работает оффлайн (если требуется)

[] Тестирование:

- [] Написаны unit-тесты (минимум 5)
- [] Есть интеграционные тесты
- [] Проверена доступность (axe-core)
- [] Тестирование на разных устройствах

[] Документация:

- [] README с инструкциями по запуску
- [] Комментарии к сложной логике
- [] JSDoc для публичных функций
- [] Описаны архитектурные решения
- [] Скриншоты/гифки демо

[] Деплой:

- [] Работает на выбранном хостинге
- [] Настроен CI/CD
- [] Доменное имя (опционально)
- [] SSL сертификат

4. Командная работа (если проект групповой) markdown

****Рекомендации для команд:****

Распределение ролей (на 3 человека):

- ****Архитектор****: TypeScript, структура проекта, code review

- ****Дизайнер/Верстальщик****: UI/UX, адаптивность, анимации
- ****ML-инженер****: интеграция моделей, оптимизация, API

Ежедневные практики:

1. Daily Standup (15 минут):

- Что сделал вчера?
- Что буду делать сегодня?
- Какие есть блокеры?

2. Code Review процесс:

- Каждый PR проверяется минимум одним человеком
- Используется чек-лист для ревью
- Комментарии должны быть конструктивными

3. Ведение проекта:

- Используйте проектную доску (GitHub Projects, Trello)
- Ведите changelog
- Документируйте важные решения

Инструменты для коммуникации:

- Max/Telegram для ежедневного общения
- Google Meet для еженедельных созвонов
- Figma для совместного дизайна
- Notion для документации

5. Подготовка к защите проекта markdown

****Как подготовить презентацию:****

Структура выступления (7-10 минут):

1. Вступление (30 сек):

- Название проекта

- Проблема, которую решает

- Целевая аудитория

2. Демонстрация (3-4 мин):

- Live-демо основных функций

- Показ работы на разных устройствах

- Демонстрация ML-компонента

3. Техническая часть (2-3 мин):

- Стек технологий

- Архитектурные решения

- Интеграция ML

- Самые интересные технические моменты

4. Результаты (1-2 мин):

- Метрики производительности

- Что получилось хорошо

- Что можно улучшить

- Планы на развитие

Подготовка материалов:

- Презентация (5-7 слайдов)

- Видео-демо (1-2 минуты, про запас)

- Ссылка на живое приложение

- Ссылка на репозиторий

- QR-код для быстрого доступа

Что показать в первую очередь:

1. Уникальная фишка проекта

2. Работа ML-компонента

3. Адаптивность

4. Производительность (Lighthouse метрики)

ЧТО ДЕЛАТЬ ЕСЛИ ВОЗНИКЛИ ПРОБЛЕМЫ

Типичные проблемы и решения:

markdown

****Проблема: "Не могу разобраться с..."****

1. Ищите решение в порядке:

- Документация технологии (официальная)
- Stack Overflow (похожие вопросы)
- GitHub Issues проекта
- Сообщества (Discord, Telegram)
- Задайте вопрос ментору

2. Формулируйте вопрос правильно:

Не правильно - "У меня ничего не работает"

Правильно - "При попытке X происходит Y, хотя ожидается Z. Уже пробовал A и B"

3. Используйте отладку:

- console.log() в ключевых точках
- Debugger в DevTools
- React DevTools для компонентов
- Network tab для запросов

Ресурсы для помощи:

markdown

****Полезные ресурсы:****

Для вёрстки:

- CSS-Tricks (гайды и примеры)

- MDN Web Docs (документация)
- Can I Use (поддержка фич)

Для JavaScript/React:

- React Official Docs
- TypeScript Handbook
- You Don't Know JS (книга)

Для ML в вебе:

- TensorFlow.js Official Guide
- Teachable Machine (Google)
- ML5.js Library

Инструменты:

- Figma Community (готовые UI-kit)
- Coolors (генератор палитр)
- Undraw (бесплатные иллюстрации)
- Font Awesome (иконки)

КРИТЕРИИ УСПЕХА

Что отличает хороший проект от отличного:

markdown

****Отличный проект имеет:****

1. ****Продуманный UX:****

- Интуитивная навигация
- Полезные сообщения об ошибках
- Прогрессивная загрузка
- Оффлайн-работа (где уместно)

2. ****Качественный код:****

- Читаемость и понятные названия
- Отсутствие дублирования
- Модульность и переиспользование
- Комментарии к сложной логике

3. ****Внимание к деталям:****

- Микро-анимации
- Состояния загрузки/ошибок
- Accessibility features
- Согласованность дизайна

4. ****Документация:****

- Понятный README
- Инструкция по запуску
- Описание архитектуры
- Скриншоты/демо

Совет от опытного разработчика:

markdown

****Золотое правило:****

"Лучше сделать меньше, но качественно, чем много, но плохо.

Один полностью рабочий и отполированный функционал ценится выше десяти сырых фич."

11. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем

Комплект лицензионного программного обеспечения:

1. Docker
2. Редактор кода VS CODE
3. Web сервер node.js версии не ниже 18 LTS
4. Windows
5. Figma
6. Kaspersky

Современные профессиональные базы данных и информационные справочные системы:

1. Yandex DataLens
2. Yougile
3. Яндекс 360
4. Электронно-библиотечная система BOOK.RU <http://www.book.ru>
5. Электронно-библиотечная система Znanium <http://www.znanium.com>
6. Электронная библиотека Финансового университета (ЭБ) <http://elib.fa.ru/>

Сертифицированные программные и аппаратные средства защиты информации:

1. не предусмотрены

12. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

1. **Учебная аудитория** для проведения учебных занятий, предусмотренных программой, в том числе групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, оснащенная оборудованием и техническими средствами обучения: мебель аудиторная (столы, стулья, доска аудиторная), набор демонстрационного оборудования (проектор, экран)