Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Новосибирский государственный технический университет»
Кафедра экономической информатики

**Паспорт**

**Расчетно-графического задания (работы)**

по дисциплине «Разработка программных приложений», семестр 7

## 1. Методика оценки

Выполнение расчетно-графического задания (работы) (далее - РГЗ(Р)) является формой текущей аттестации (контроля) по дисциплине, предусмотренной учебным планом. Цель РГЗ: студент должен закрепить навыки разработки на Flask и настройки CI/CD для автоматизации процессов разработки и обеспечения качества.

Обязательным элементом РГЗ(Р) является разработка программного приложения.

Задание выполняется студентами в группах по два человека. Номер варианта выдается случайным образом из общего списка. После назначения варианта он исключается из списка доступных для следующих групп, чтобы обеспечить уникальность задания для каждой группы.

Количество заданий достаточно для обеспечения каждой группы индивидуальным заданием РГЗ(Р).

Замена задания РГЗ(Р) осуществляется по согласованию с преподавателем из числа резервных (не занятых) заданий.

Преподаватель осуществляет руководство по выполнению задания, оказывает консультационную помощь и принимает отчет по РГЗ(Р).

По результатам выполнения РГЗ(Р) выполняется отчет, который состоит из следующих частей:

1. Титульный лист
2. Задание (по вариантам)
3. Список литературы и источников

*Требования к оформлению:*

Объем РГЗ(З) до 10 страниц машинописного текста формата А4. Шрифт Times New Roman, 12. Формулы набираются в редакторе Math Type. Размещение сканированных формул не допускается. Приложения оформляются сквозная, в нижней части или на периферии арабскими цифрами. Рисунки должны быть пронумерованы и подписаны. Работа должна быть отредактирована, не содержать орфографических, грамматических и стилистических ошибок.

Отчет в установленное сроки сдается на кафедру на проверку. Преподаватель оценивает качество работы, отмечает положительные стороны и недостатки работы и определяет допускается ли она к защите. При необходимости преподаватель возвращает РГЗ(Р) студенту для доработки и устанавливает сроки повторного предоставления для проверки. До защиты работы студентом должны быть сделаны необходимые исправления и дополнения по всем замечаниям преподавателя.

При положительном результате оценивали РГЗ(Р) студент её распечатывает, передает на кафедру и защищает до сессии в назначенное преподавателем время.

Защита РГЗ(Р) состоит в индивидуальном устном собеседовании студента с преподавателем. В процессе защиты выявляются уровень знаний студента, степень его самостоятельности при выполнении работы. По результатам защиты студенту выставляется оценка в соответствии с критериями, приведенными в п. 2 настоящего Паспорта.

## 2. Уровни сформированности компетенций и критерии оценки

РГЗ(Р) считается выполненной **на продвинутом уровне**, если все необходимые расчеты произведены самостоятельно и без ошибок; все разделы РГЗ(Р) выполнены правильно и в полном объеме; работа оформлена в соответствии с требованиями; сдана преподавателю в указанные сроки и не возвращалась для доработки; даны полные и развернутые выводы и рекомендации; на защите студентом даны уверенные и аргументированные ответы, что свидетельствует об уровне сформированности компетенций и соотнесенных с ними индикаторов на продвинутом уровне. Оценка составляет *от 18 до 20 баллов*.

РГЗ(Р) считается выполненной **на базовом уровне**, если все необходимые расчеты произведены самостоятельно и без существенных ошибок, все разделы РГЗ(Р) выполнены правильно, но есть замечания к полноте представления информации, работа оформлена в соответствии с требованиями; сдана преподавателю в указанные сроки и однократно возвращалась студенту для незначительной доработки; в заключении даны выводы и рекомендации; на защите студентом допущены непринципиальные ошибки, что

свидетельствует об уровне сформированности компетенций и соотнесенных с ними индикаторов на базовом уровне. Оценка составляет *от 14 до 17 баллов*.

РГЗ(Р) считается выполненной **на пороговом уровне**, если все необходимые расчеты произведены самостоятельно, но с ошибками, часть из которых носит принципиальный характер; есть замечания к полноте представления информации; работа оформлена в соответствии с требованиями; сдана преподавателю, но неоднократно возвращалась студенту для доработки; в заключении даны краткие выводы; защита РГЗ(Р) вызывает у студента серьезные затруднения, что свидетельствует об уровне сформированности компетенций и соотнесенных с ними индикаторов на пороговом уровне. Оценка составляет *от 10 до 13 баллов*.

РГЗ(Р) считается **не выполненной**, если расчеты произведены с серьезными ошибками; есть замечания к полноте представления информации и оформлению; РГЗ(Р) была сдана преподавателю, но неоднократно возвращалась студенту для доработки, что не привело к улучшению ее качества; РГЗ(Р) не допущена ко защите, что свидетельствует об не сформированности компетенций и соотнесенных с ними индикаторов. Оценка составляет *менее 10 баллов*.

### 3. Шкала оценки

В общей оценке по дисциплине баллы за РГЗ(Р) учитываются в соответствии с правилами балльно-рейтинговой системы, приведенными в рабочей программе дисциплины.

РГЗ(Р) как форма текущей аттестации (контроля) по дисциплине считается успешно выполненной, если количество баллов, полученное за выполненное задание, составляет от 10 до 20 баллов включительно.

### 4. Примерный перечень РГЗ(Р)

1. Разработать Flask-приложение (API) для управления телефонными контактами, а также настроить автоматический workflow в GitHub Actions для генерации и публикации документации на GitHub Pages.

2. Разработать RESTful API на Flask для анализа текстов, выполняющее операции: подсчёт общего количества слов в переданном тексте и определение самых частотных слов в переданном тексте. Реализовать unit-тесты для проверки функциональности. Настроить балансировку нагрузки с помощью Nginx. Настроить workflow для CI/CD в GitHub Actions.

3. Разработать RESTful API на Flask для ведения дневника расходов, которое позволяет пользователю добавлять, редактировать, удалять и просматривать свои записи о расходах.

Доступ к данным должен быть защищен через авторизацию с использованием Flask-Login, а все действия пользователей должны регистрироваться в системе аудита. Настроить workflow для CI/CD в GitHub Actions.

4. Разработать RESTful API на Flask для управления финансовыми подписками пользователя. Приложение должно позволять пользователям добавлять, редактировать, удалять и просматривать свои подписки. Дополнительно нужно разработать bash-хелпер, который автоматизирует настройку и запуск приложения. Настроить workflow для CI/CD в GitHub Actions.

5. Разработать RESTful API на Flask для управления финансовыми подписками пользователя. Реализовать мигратор на Python для Flask-приложения, который запускается автоматически при старте приложения. Мигратор должен использовать changelog-файл для отслеживания миграций. Настроить workflow для CI/CD в GitHub Actions.

6. Разработать RESTful API на Flask для управления финансовыми подписками пользователя. Реализовать bash скрипт формирования для формирования changelog из текста коммитов для версии. Настроить workflow для CI/CD в GitHub Actions.

7. Реализовать приложение, которое позволяет пользователям создавать заявки на техподдержку, просматривать их статус и обновления, а администраторам — управлять заявками. Система должна поддерживать роли «пользователь» и «администратор». Настроить workflow для CI/CD в GitHub Actions.

8. Реализовать Flask приложение, которое предоставляет пользователям возможность публиковать статьи, просматривать статьи других авторов и комментировать публикации. Для защиты от несанкционированного доступа в приложение необходимо встроить систему авторизации с ограничением количества попыток входа, предотвращающая brute-force атаки. Настроить workflow для CI/CD в GitHub Actions.

9. Создать RESTful API на Flask для управления запросами информации о погоде с ограничением частоты запросов и кэшированием. Приложение должно ограничивать количество запросов от каждого пользователя и кэшировать результаты, чтобы снизить нагрузку на API и ускорить время отклика для часто запрашиваемых данных. Настроить workflow для CI/CD в GitHub Actions.

10. Разработать RESTful API на Flask, которое предоставляет возможность создавать короткие ссылки, перенаправлять пользователей на исходный URL и отслеживать статистику их использования. Для оптимизации работы необходимо внедрить кэширование (с помощью Flask-Caching) и ограничение количества созданных ссылок на одного пользователя (Flask-Limiter). Настроить workflow для CI/CD в GitHub Actions.

11. Разработать RESTful API на Flask для управления доступом к обучающим материалам на основе атрибутов пользователя, объекта и контекста. Приложение должно учитывать не только роли пользователей, но и их атрибуты, такие как время доступа, статус учетной записи, уровень подписки и т.д. Настроить workflow для CI/CD в GitHub Actions.

12. Разработать RESTful API на Flask для управления запасами товаров (инвентарем) на складе. Приложение должно поддерживать операции по добавлению, обновлению и удалению товаров, а также автоматическую генерацию отчетов о состоянии инвентаря. Настроить workflow для CI/CD в GitHub Actions.

13. Разработать RESTful API на Flask для управления задачами с учетом их приоритетов и автоматической отправкой уведомлений о приближающихся дедлайнах. Настроить workflow для CI/CD в GitHub Actions с отправкой оповещения об успешности выполнения пайплайна.

14. Разработать RESTful API на Flask для управления рейтингами фильмов с функцией рекомендаций и оптимизацией производительности. Настроить workflow для CI/CD в GitHub Actions.

15. Разработать Flask-приложение для управления персональными заметками с поддержкой авторизации, кэширования популярных заметок и отображения данных на веб-странице с использованием Jinja-шаблонов. Настроить workflow для CI/CD в GitHub Actions.

16. Разработать Flask-приложение для управления бюджетом команды проекта. Приложение должно позволять пользователям (менеджерам) добавлять расходы, распределять бюджет между отделами и отслеживать общие затраты проекта. Реализовать веб-интерфейс для работы с данными бюджета и анализа затрат с помощью Jinja шаблонов. Настроить workflow для CI/CD в GitHub Actions.

17. Разработать Flask-приложение для управления выплатами заработной платы с учетом принципов объектно-ориентированного программирования, функционального подхода и использования баз данных. Приложение должно позволять рассчитывать заработную плату сотрудников, учитывать премии и налоги, а также визуализировать данные через веб-интерфейс с использованием Jinja. Настроить workflow для CI/CD в GitHub Actions.