Задача: Разработка на уеб, мобилно или data engineering/data analytics софтуерно приложение, което има следните елементи:

- Back-end API
- Front-end
- Database (SQL or NoSQL)
- Docker, CI/CD

Критерии за оценка:

No.	Критерий	Точки
1	**Цел на проекта и дефиниция на функционалностите**	**10**
1.1	Целта на проекта е ясно и изчерпателно дефинирана	2
1.2	Основните функционалности са описани като user stories https://www.atlassian.com/agile/project-management/user-stories	4
1.3.	Основните потребители и начинът им на работа със системата са описани като use cases https://www.figma.com/resource-library/what-is-a-use-case/ https://medium.com/@queen_shecoder/understanding-use-cases-e72211b1b236	4
2	**Избор на технологии и софтуерна архитектура**	**10**
2.1	Избраните технологии са подходящи за реализиране на проекта	2
2.2	Избран е конкретен модел софтуерна архитектура и е обоснован избора https://www.geeksforgeeks.org/types-of-software-architecture-patterns/	2
2.3	Приложението е разделено на модули/компоненти с ясни връзки и комуникация между тях, описани в архитектурна диаграма	4
2.4	Архитектурата позволява приложението да бъде разширявано и поддържано в бъдеще	2
3	**Среда за работа и управление на кода**	**10**
3.1	Създадено GitHub repository	2
3.2	По време на работа са използвани feature branches и pull requests	4
3.3	Работата е разделена на отделни задачи в GitHub (Issues)	2
3.4	Само необходимите файлове и ресурси са добавени в git	2
4	**Реализация на проекта**	**40**
4.1	Спазена е избраната софтуерна архитектура	2
4.2	Използвани са добри практики и design patterns https://refactoring.guru/design-patterns	2
4.3	Спазени са принципите на ООП за енкапсулация и абстракция при имплементацията на отделните класове/компоненти	2
4.4	Спазен е separation of concerns принципа при имплементацията на различните модули/компоненти от системата https://www.geeksforgeeks.org/separation-of-concerns-soc/ https://nalexn.github.io/separation-of-concerns/	2
4.5	Спазен е single responsibility принципа https://stackify.com/solid-design-principles/	2
4.6	Спазен е dependency inversion принципа https://stackify.com/dependency-inversion-principle/	2

4.7	Публичният интерфейс е документиран чрез Swagger или еквивалентно решение https://swagger.io/	2
4.8	Спазени са REST принципите при дефиниране на публичния интерфейс https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/	2
4.9	Спазени са правилата и добрите практики при дизайна на бази данни	10
4.10	Моделът на данните е добре дефиниран и базата данни е нормализирана (при използване на SQL) или колекциите са добре организирани и ясни връзките между тях (при използване на NoSQL) https://support.microsoft.com/en-us/office/database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5 https://www.guvi.com/blog/database-design-principles-and-best-practices/	4
4.11	Системата е защитена от популярни видове атаки (напр. SQL Injection)	2
4.12	Системата има надежден начин за обработка на грешки и непредвидени събития https://developers.google.com/tech-writing/error-messages/error-handling https://blog.postman.com/best-practices-for-api-error-handling/	2
4.13	Конфигурационни параметри, константи и environment променливи се управляват по надежден начин https://dev.to/khalidk799/environment-variables-its-best-practices-1o1o	2
4.14	Кодът може да се преизползва, няма повторения и няма неизползван код в проекта	2
4.15	Няма прекалено комплексни и сложни методи или функции в кода, които да трудни за разбиране и проследяване или да използват необосновано количество ресурси	2
5	**Тестване**	**10**
5.1	Написани са unit тестове на основните компоненти в системата	5
5.2	Дефинирани са сценарии за тестване на системата от гледна точка на потребител https://www.browserstack.com/guide/user-acceptance-testing-template	5
6	**Качество и чистота на кода**	**10**
6.1	Спазени са еднакви конвенции за стилизация и подреждане на кода в целия проект	2
6.2	Кодът е добре подреден, ясен за четене и има коментари където е нужно	2
6.3	Добавена е автоматична валидация за прилагането на стилизацията https://webapp.io/blog/linting-best-practices/	2
6.4	Добавен е статиен анализ на кода https://www.sonarsource.com/products/sonarcloud/	4
7	**Deployment и DevOps**	**20**
7.1	Приложението е инсталирано на сървър/cloud	6
7.2	Създадена е deployment диаграма за необходимите и използвани ресурси https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/	2
7.3	Приложението е контейнеризирано с Docker	6
7.4	Използвани са GitHub Actions или алтернативен CI/CD pipeline	6
8	**Скалируемост и производителност на системата**	**10**

8.1	Не са налице сериозни проблеми в поддръжката на голям брой потребители или обеми от заявки/данни	6
8.2	Реализацията на системата позволява хоризонтално и вертикално скалиране https://www.digitalocean.com/resources/articles/horizontal-scaling-vs-vertical-scaling	4
9	**Документация, презентация и защита на проекта**	**30**
9.1	Изготвена кратка документация на проекта (5-10 страници)	5
9.2	Степен на завършеност на проекта	5
9.3	Отговори на въпроси и защита на проекта	20

Оценяване:

- Максимален брой точки: 150
- Над 125 точки: Отличен (6)
- Между 110 и 124 точки: Много добър (5)
- Между 85 и 99 точки: Добър (4)
- Между 70 и 84 точки: Среден (3)

Срок: 01.06.2025