Дoклад 01

По Софтуерно инженерство

Изготвил: Валентин Кьосев 12 ,,а‘‘ клас

1.Какво е use case, за какво служи? Пример.

2. Какво е user story, за какво служи? Пример.

3. Какво е SRS, за какво служи? Пример.

4. Какво е DDS, за какво служи? Пример.

5.Какво е прототипиране на потребителския интерфейс? Опишете процеса.Пример.

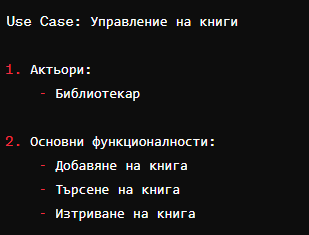
1. **Какво е use case, за какво служи? Пример.**

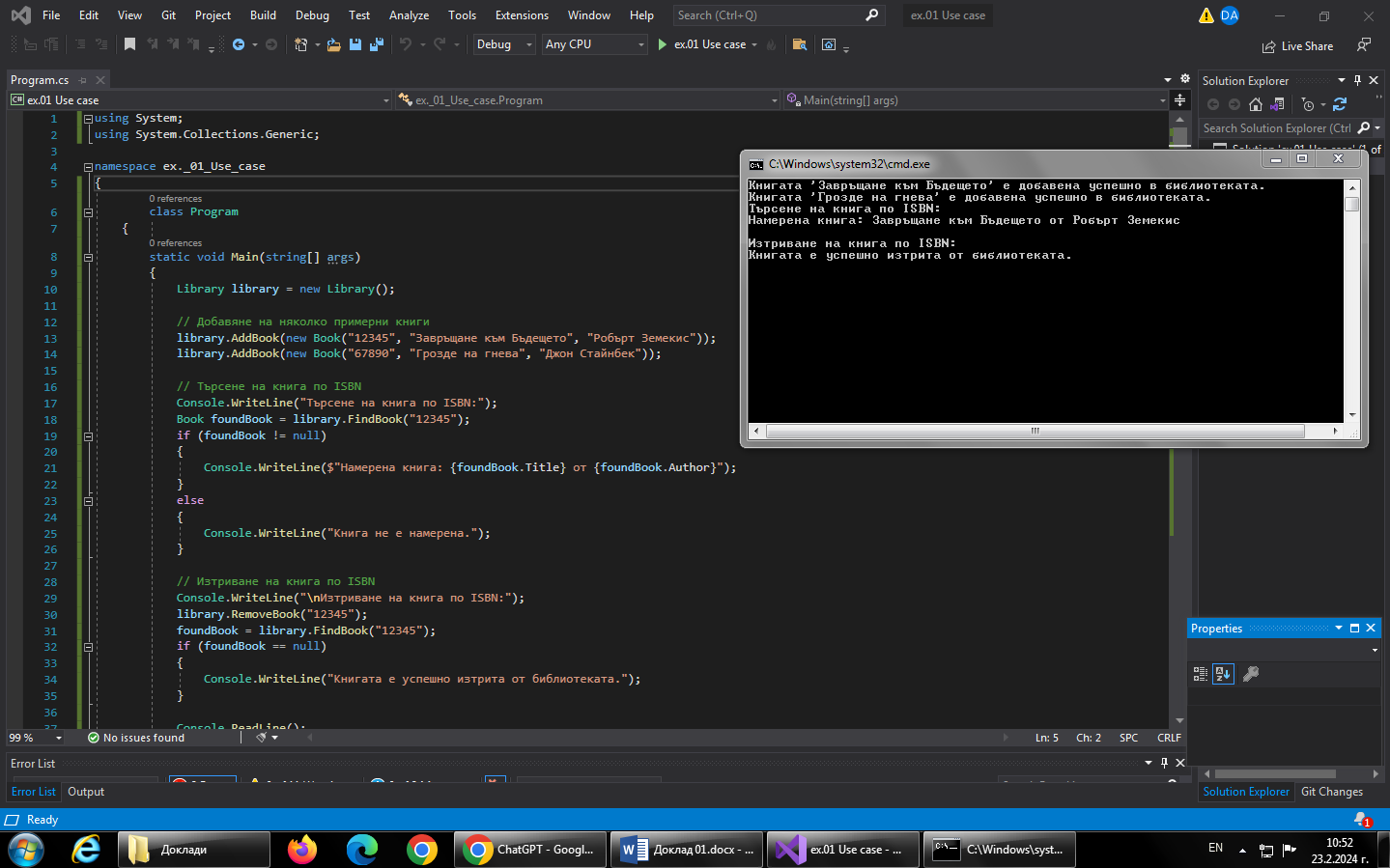
"Use case" е понятие, което се използва в областта на софтуерното проектиране и разработка, за да опише сценарии или ситуации, в които даден продукт или система ще се използва от потребителите си. Това е начин за определяне и документиране на функционалността на системата чрез описване на начина, по който потребителите ще я използват в реалния живот.

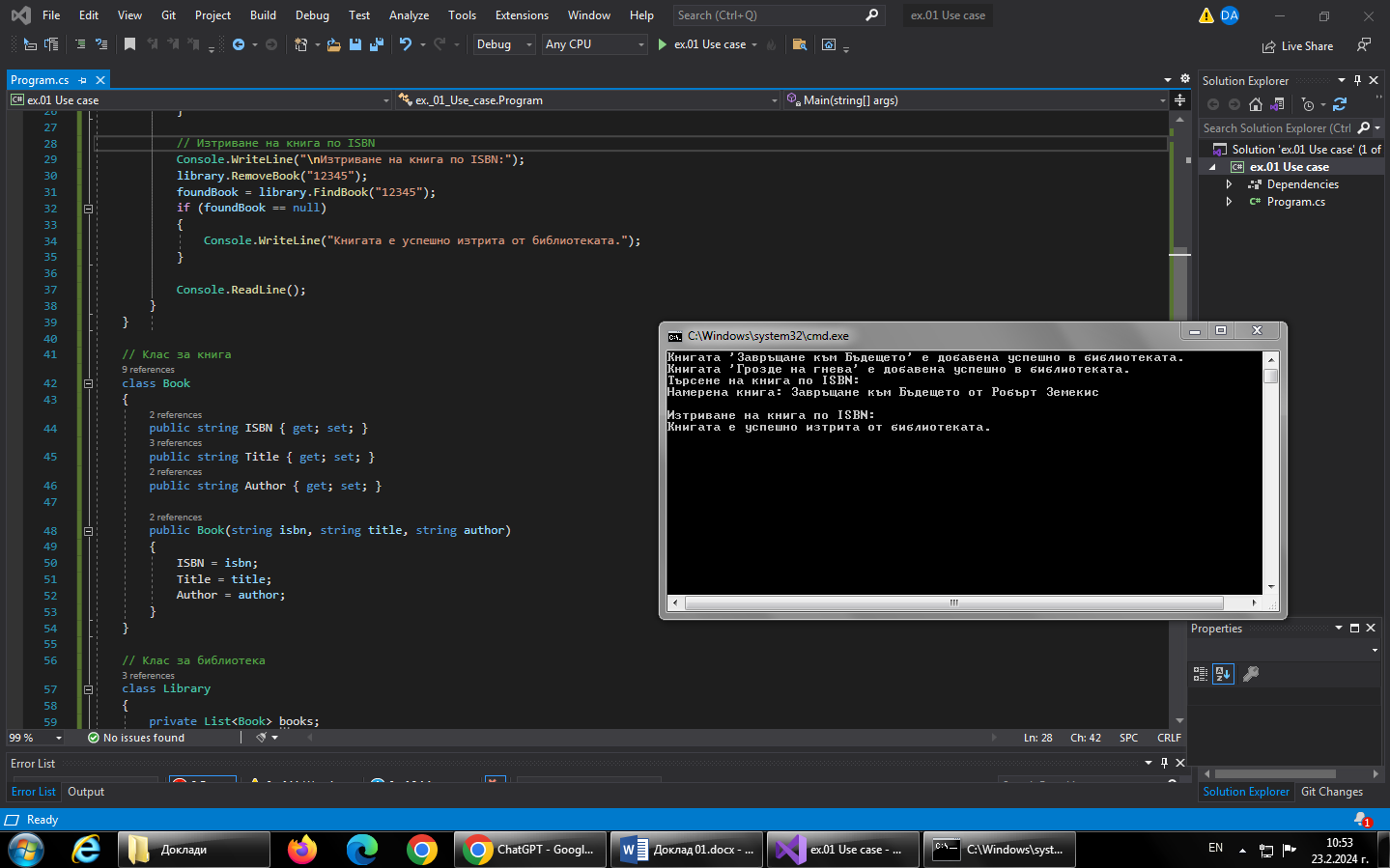
В един use case се описва кой ще използва системата, как ще я използва, какви са стъпките, които трябва да последва, както и как системата ще реагира в различни ситуации.

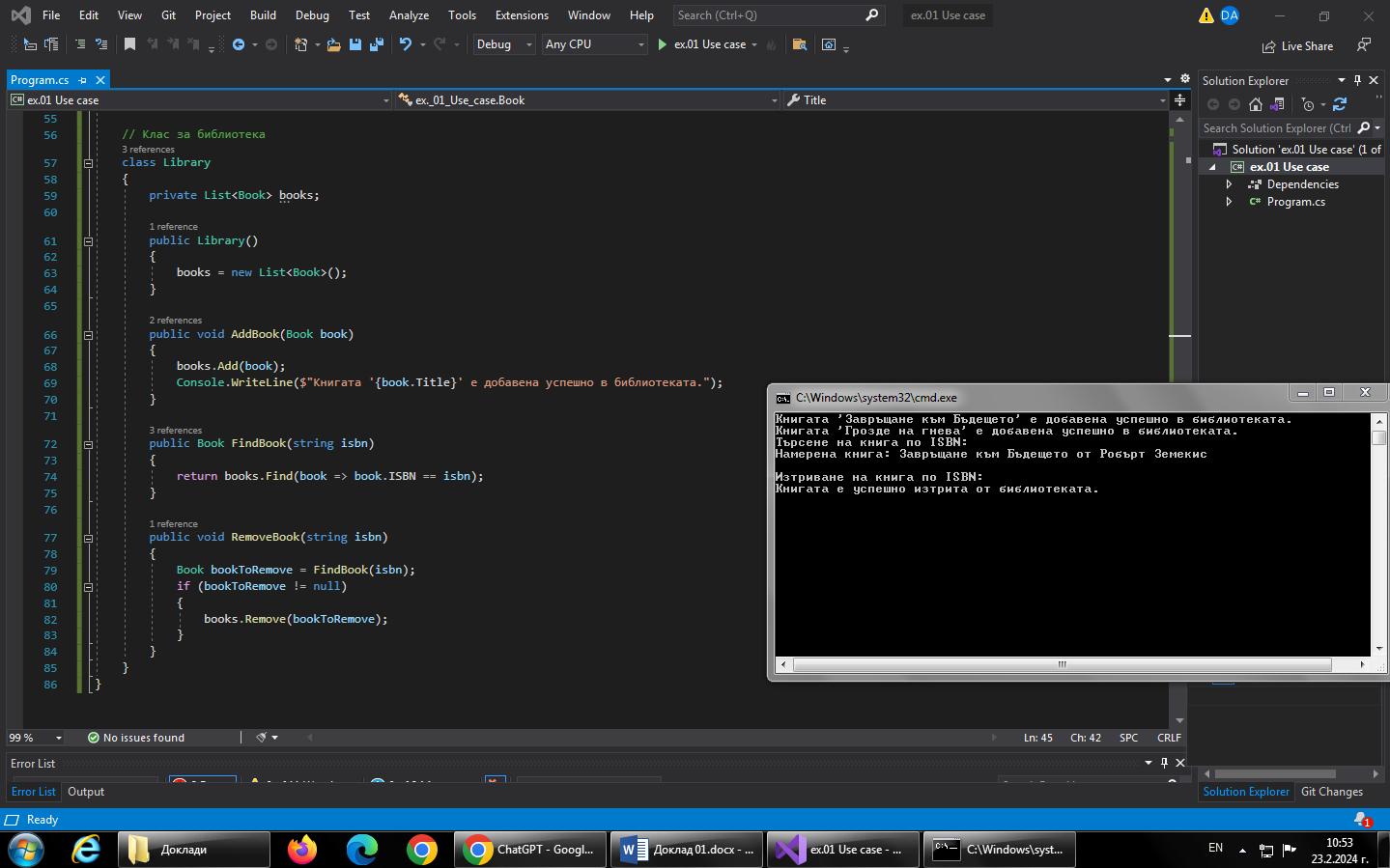
**Пример - use case**

**Изглед на Use Case диаграмата:**

****

****

****

****

Този код демонстрира примерен Use Case за управление на книги в библиотека. Позволява на библиотекаря да добавя нови книги, да търси книги по ISBN и да изтрива книги по същия критерий.

* **Добавяне на книга:**

Създаваме нов обект от тип Book и го добавяме в библиотеката чрез метода AddBook.

* **Търсене на книга:**

Библиотеката има метод FindBook, който търси книга по даден ISBN и връща обекта на намерената книга.

**Изтриване на книга:**

Използваме метода RemoveBook, който търси книга по ISBN и я премахва от списъка с книги в библиотеката.

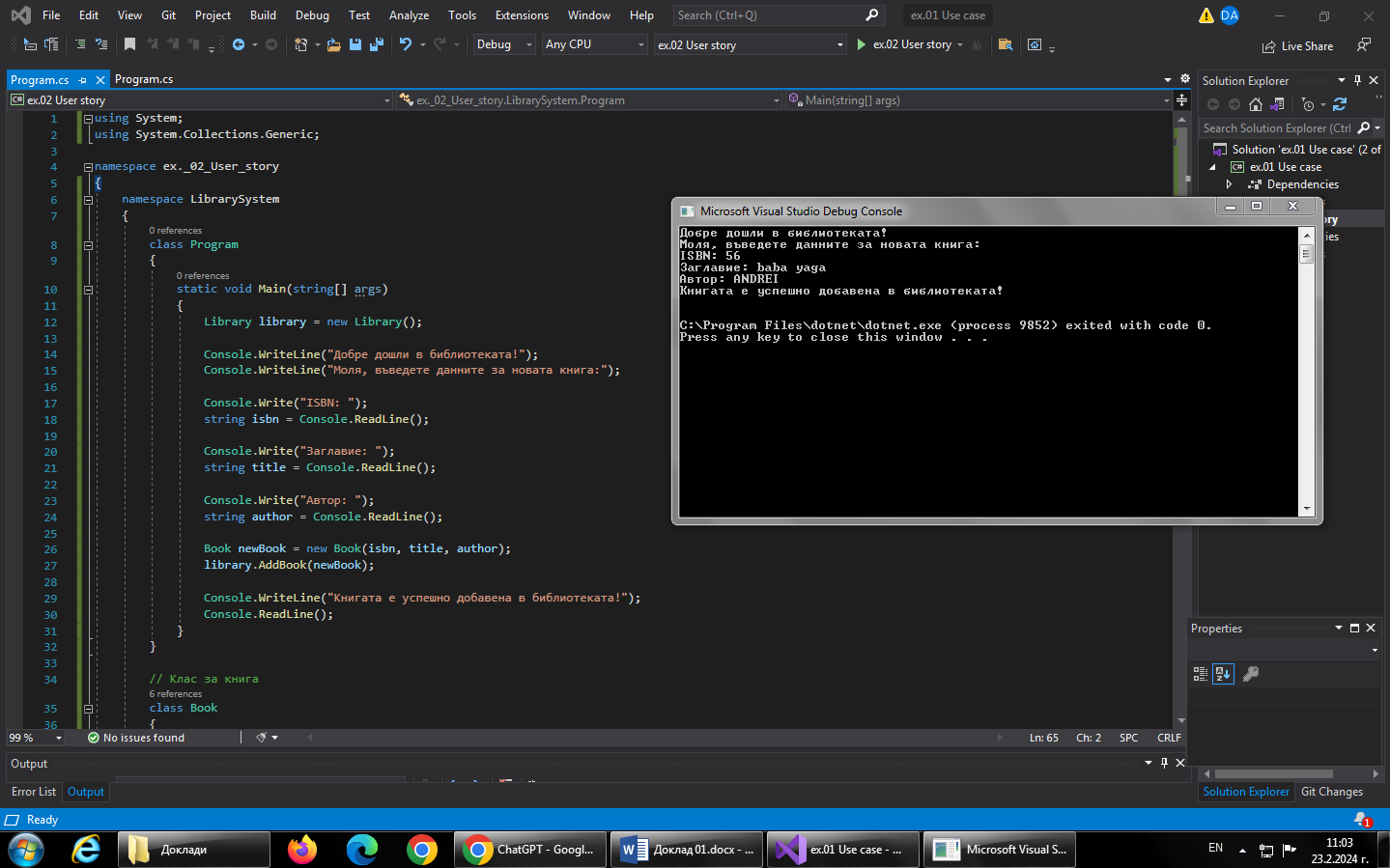
При стартиране на програмата, тя добавя няколко примерни книги, след това търси и изтрива книга по ISBN. Това е само един пример за използване на Use Case, който може да се разшири и модифицира според нуждите на реалния проект.

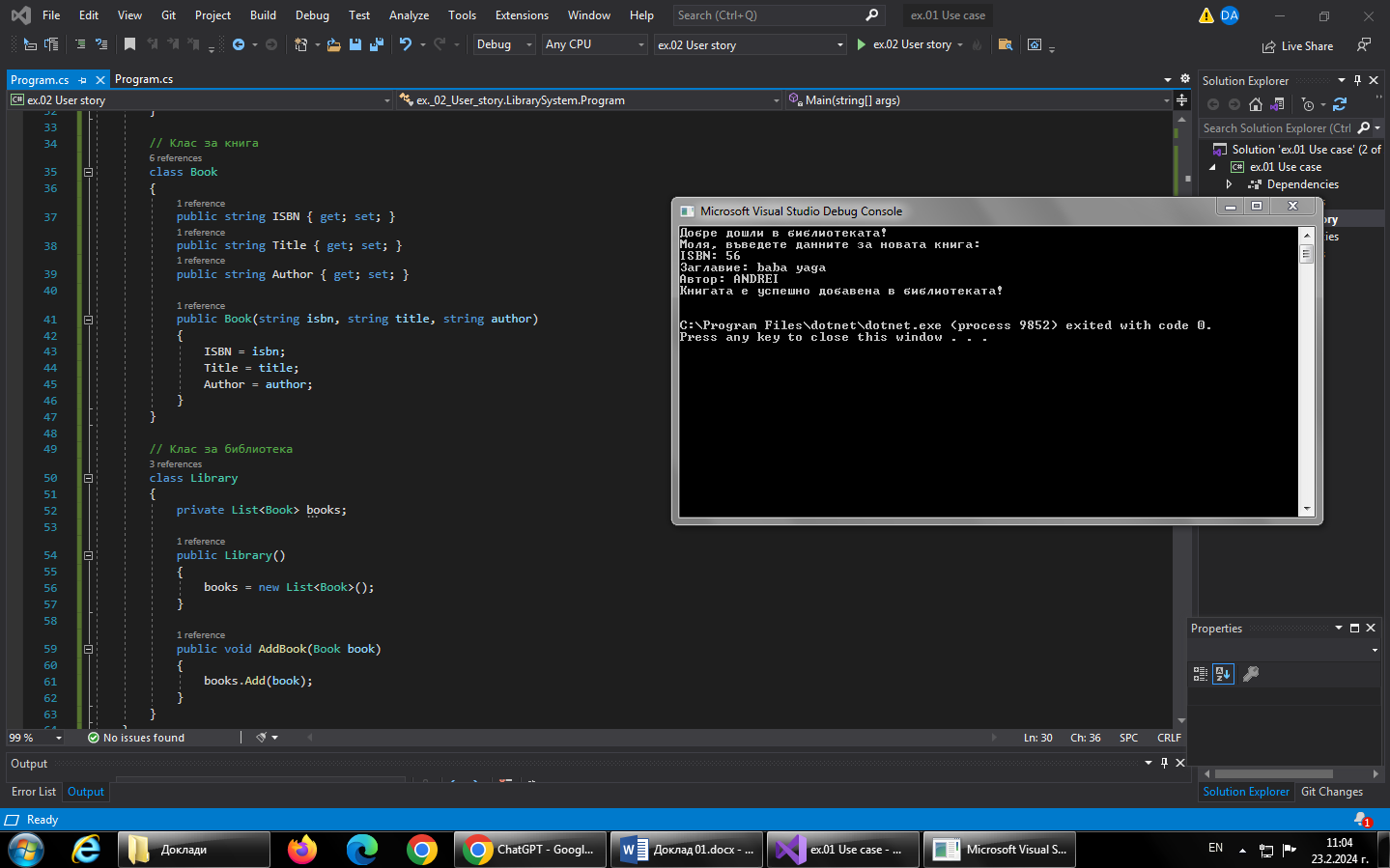
1. **Какво е user story, за какво служи? Пример.**

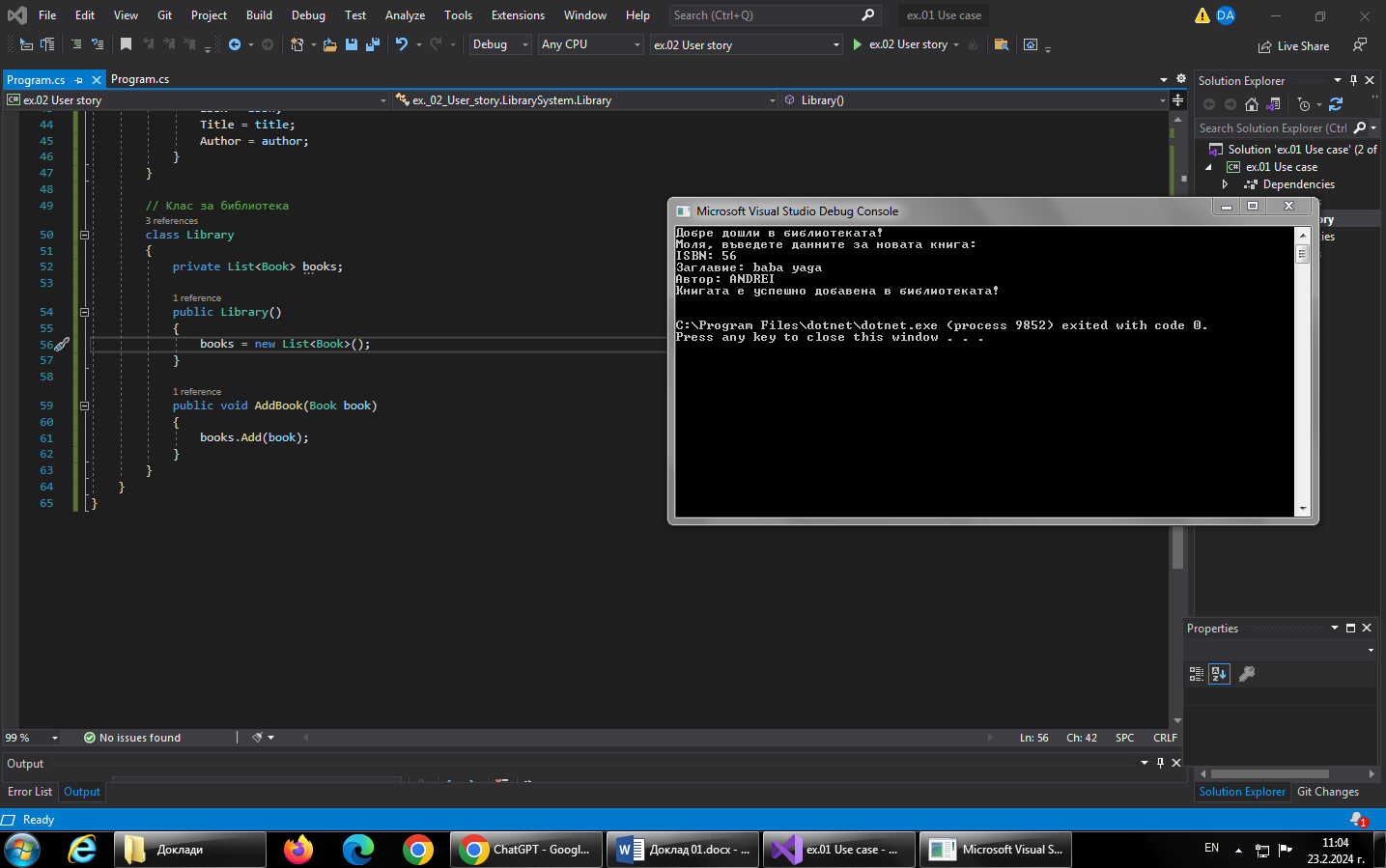
"User story" е концепция, която се използва в Agile методологията за разработка на софтуер, като Scrum и Kanban. Това е кратко описание на функционалността на софтуерна система от гледна точка на потребителя, което служи като начин за комуникация между членовете на екипа за разработка и за уточняване на изискванията за продукта.

Една "user story" обикновено е във формата на просто изречение или кратък текст, който описва нуждите на потребителя и предоставя информация за това какво трябва да бъде направено за да се удовлетворят тези нужди.

**Пример - user story**

****

****

****

Този код демонстрира използването на User Story за добавяне на книга в библиотеката. Позволява на библиотекаря да въведе ISBN, заглавие и автор на нова книга, след което да я добави в библиотеката.

* Потребителят стартира програмата и въвежда данните за новата книга - ISBN, заглавие и автор.
* Създава се нов обект от тип Book с тези данни.
* Този обект се подава към метода AddBook на обекта Library, който го добавя към списъка с книгите в библиотеката.

При стартиране на програмата, библиотекарят може да въведе данните за новата книга, като последва стъпките на потребителската история. След успешното добавяне, тя получава съобщение за успех.

Този пример може да се разшири с допълнителни функционалности, като търсене на книга, изтриване на книга и други, в зависимост от изискванията на проекта или потребителските истории.

1. **Какво е SRS, за какво служи? Пример.**

SRS (Software Requirements Specification) е документ, който подробно описва изискванията към софтуерната система, която трябва да бъде разработена. Този документ служи като основа за разработчиците, проектантите и всички заинтересовани страни, за да разберат какви функционалности, характеристики и ограничения трябва да има системата.

Създаването на SRS е ключова стъпка в процеса на софтуерно проектиране и разработка, тъй като той помага за уточняване на изискванията, предотвратява недоразумения и осигурява ясност за всички заинтересовани страни относно това какво трябва да се изгради.

**Ето някои от основните цели на SRS:**

• Да определи функционалните и нефункционалните изисквания към системата.

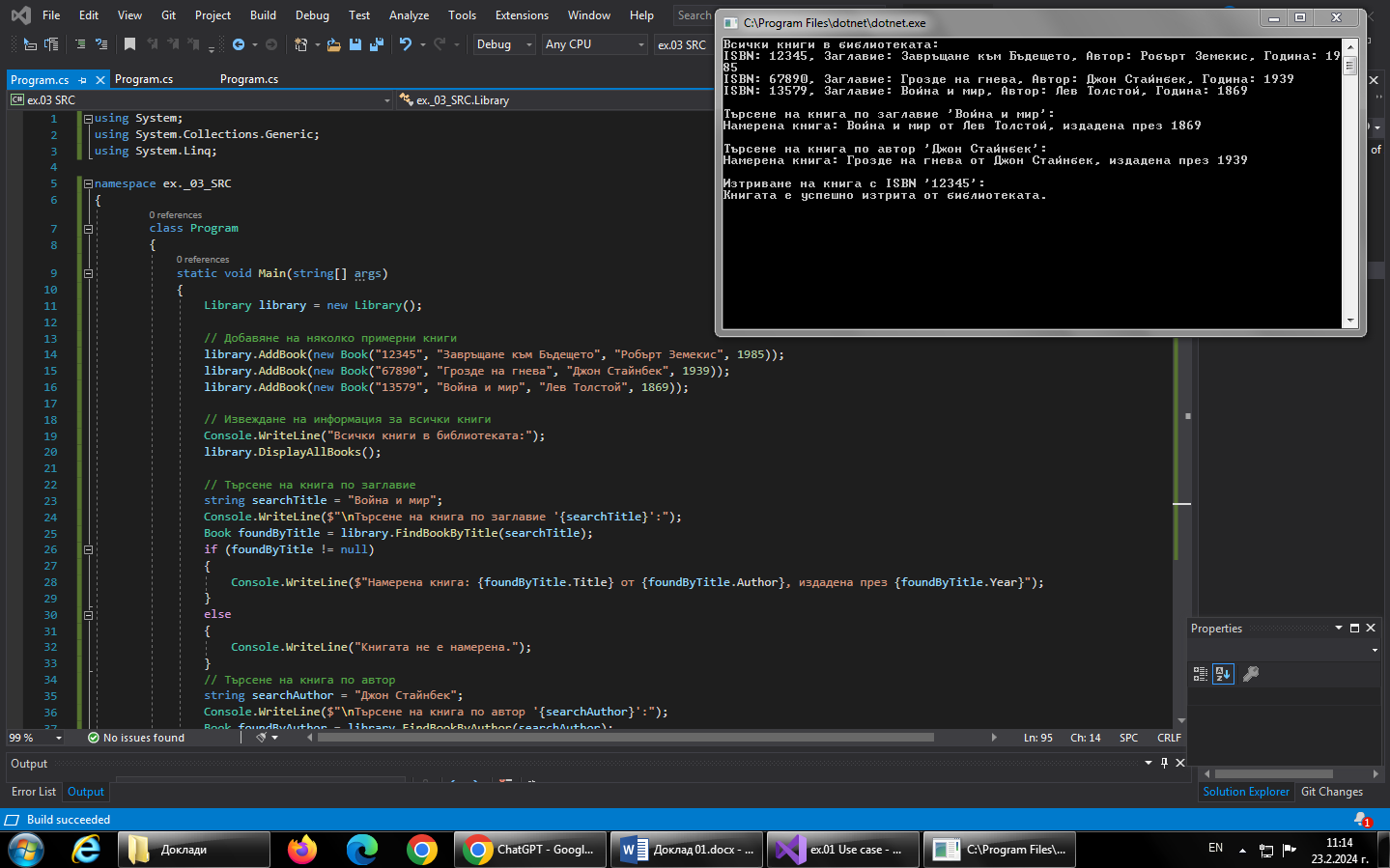
• Да уточни поведението на системата в различни ситуации.

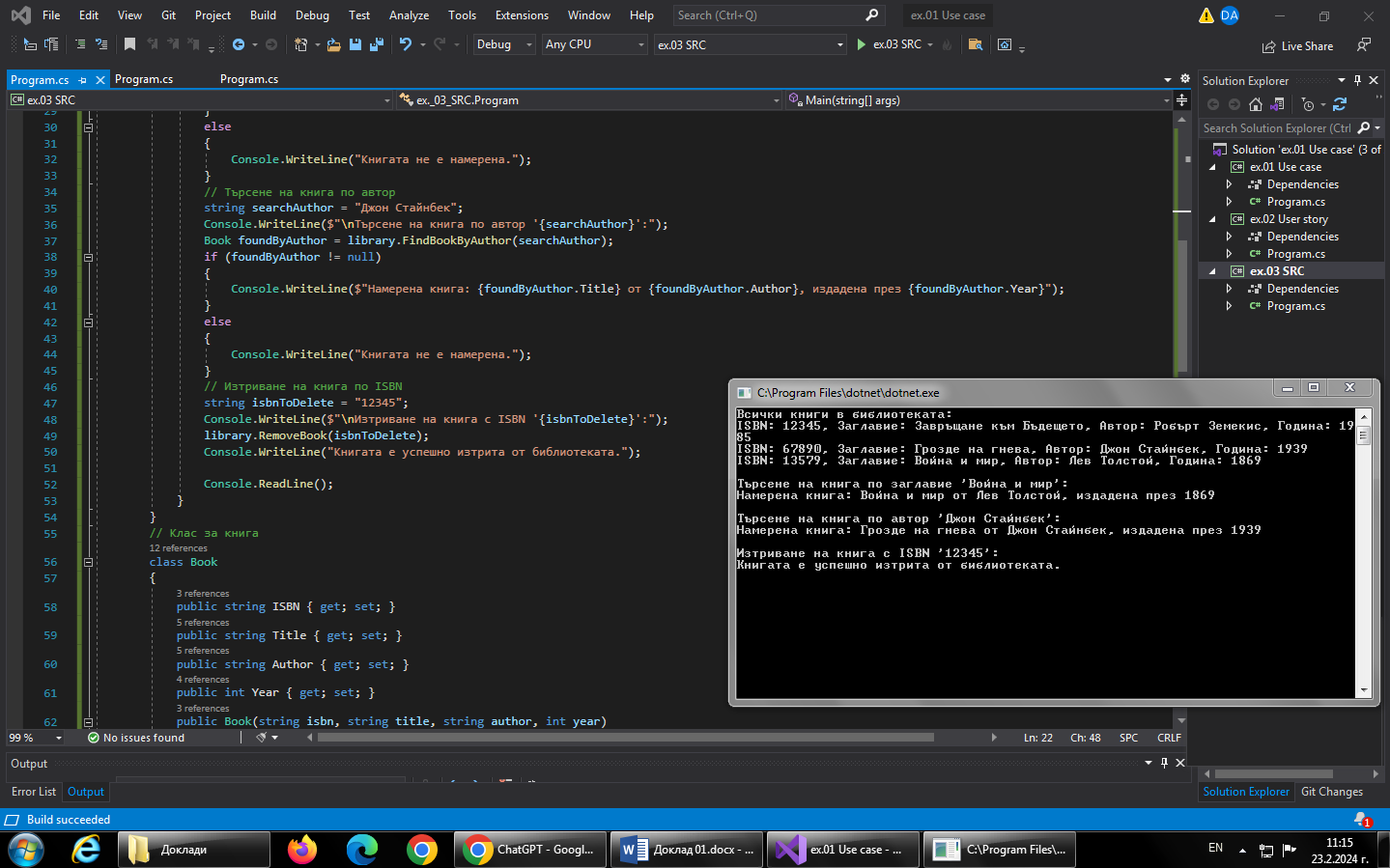
• Да предостави основа за оценка на разходите и времето за разработка.

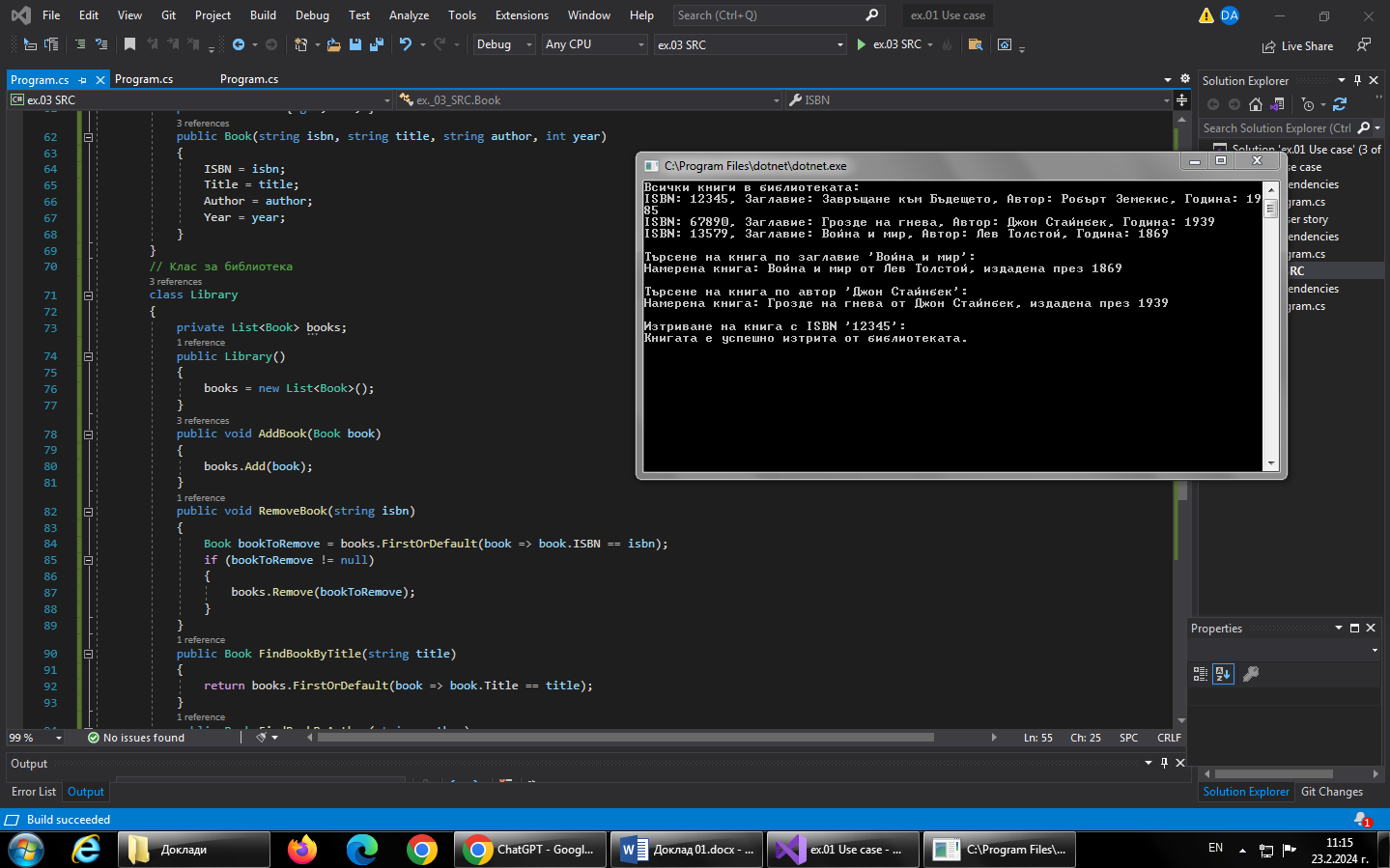
• Да служи като основа за проверка и тестване на системата.

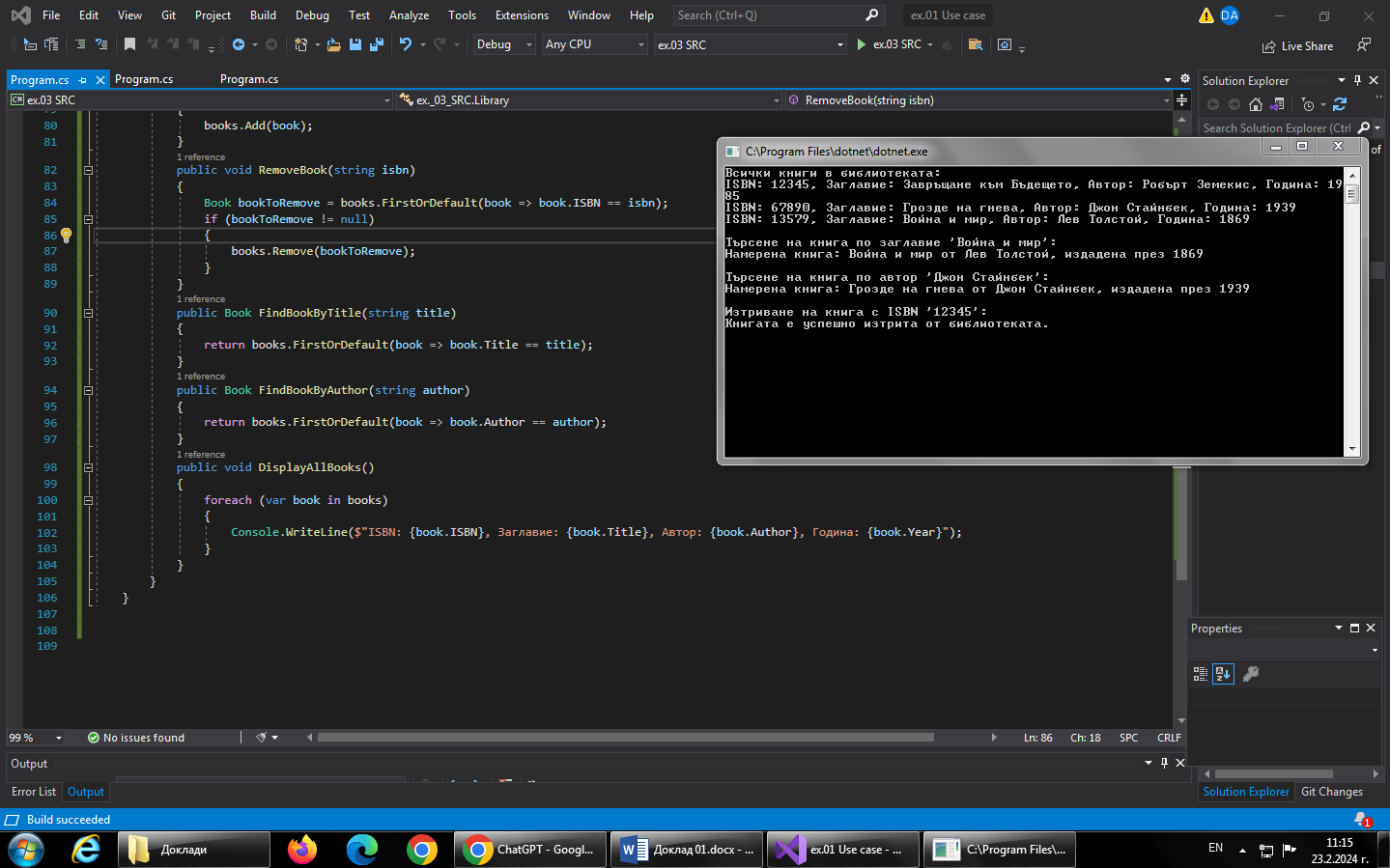
**Пример – SRC**

Този пример включва по-широк спектър от функционалности, като добавяне на книги, търсене на книги по различни критерии, извеждане на информация за книгите и други.

****

****

****

****

Този код добавя още функционалности към библиотечната система:

* Добавяне на книги със зададени ISBN, заглавие, автор и година на издаване.
* Търсене на книга по заглавие и по автор.
* Извеждане на информация за всички книги в библиотеката.
* Изтриване на книга по зададен ISBN.

Този код демонстрира някои от основните принципи на програмиране в C# и как те могат да бъдат използвани за изграждане на функционална библиотечна система, отговаряща на специфични изисквания, дефинирани в SRS документ.

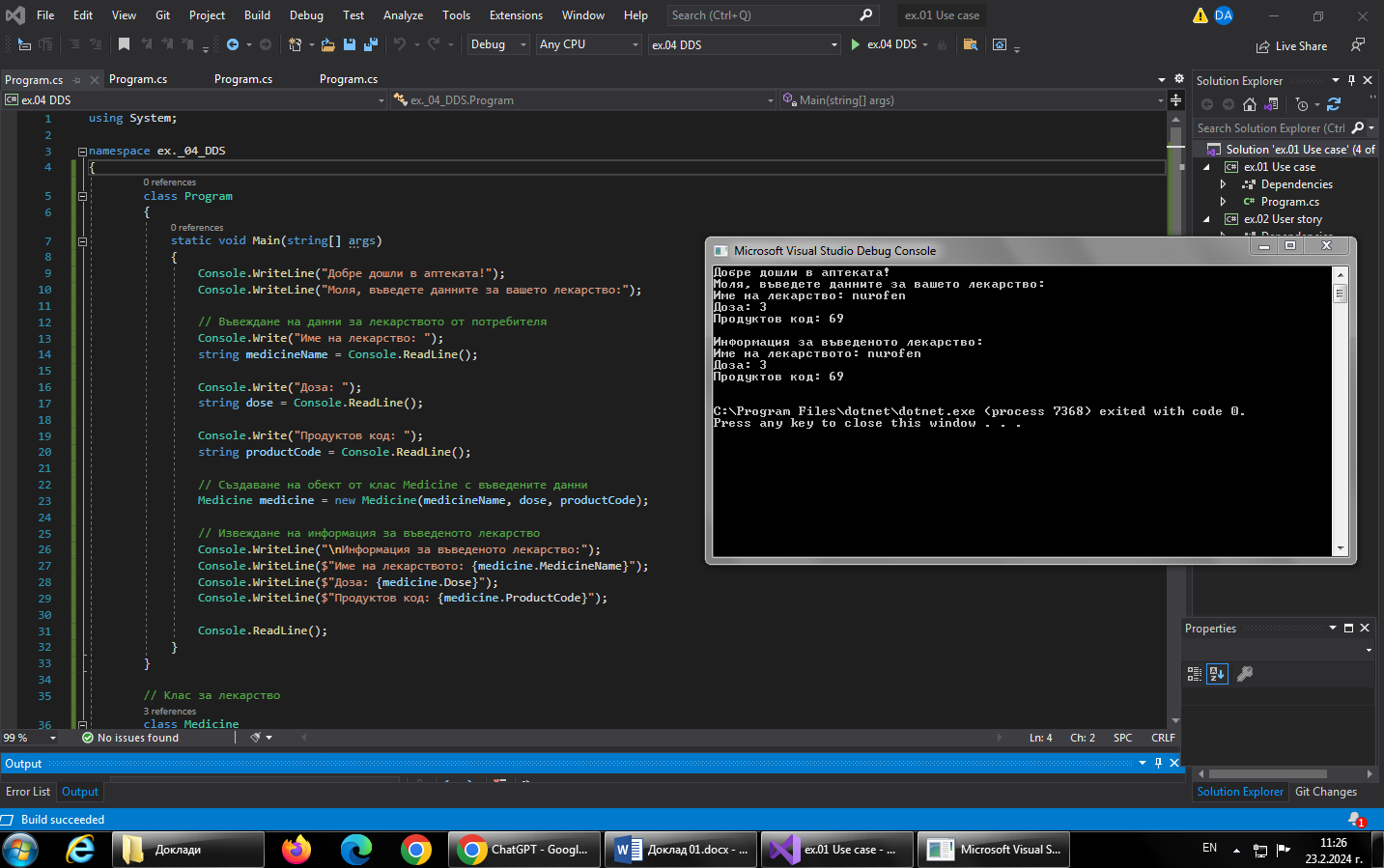
1. **Какво е DDS, за какво служи? Пример.**

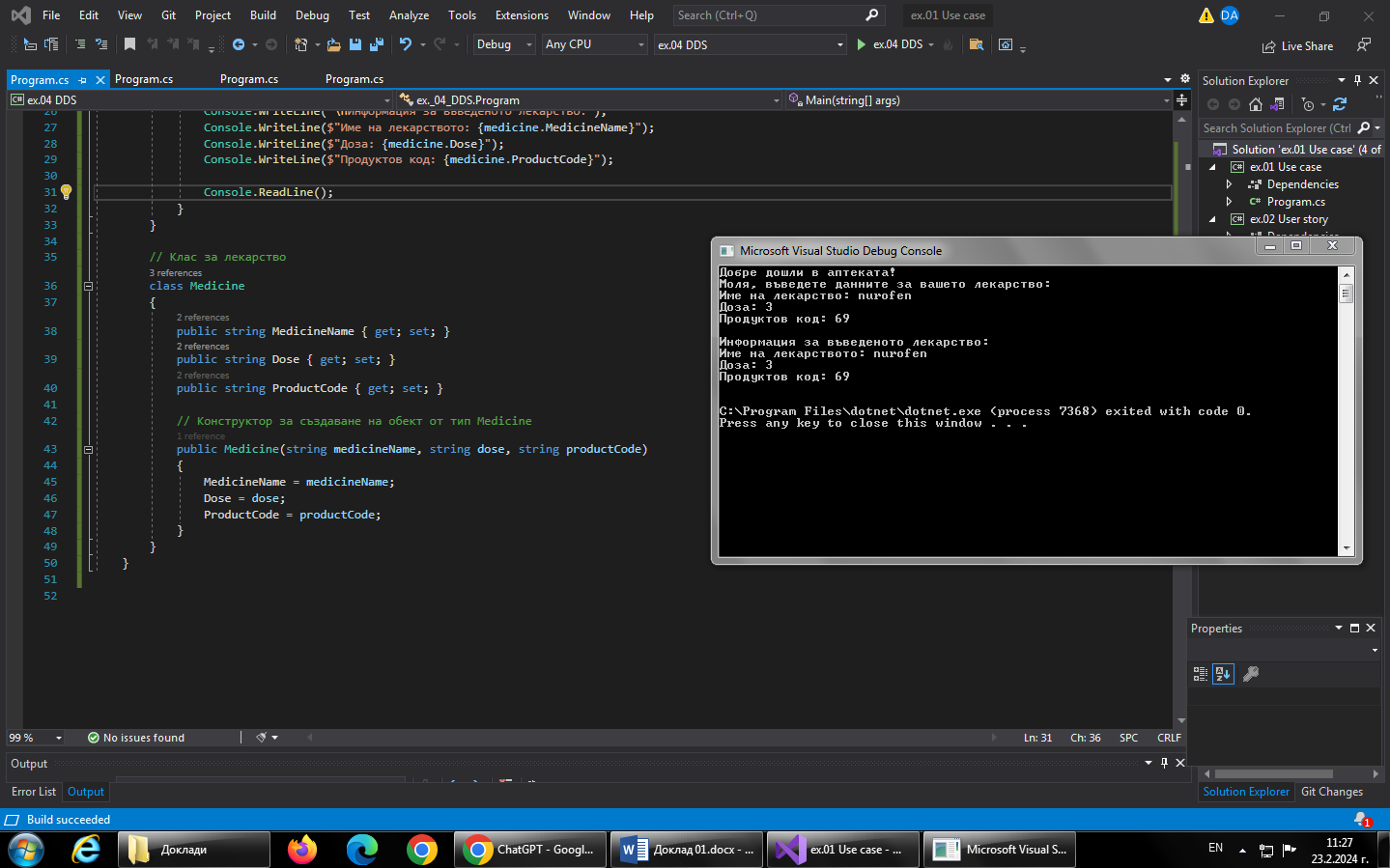
DDS е съкращение от "Direct Digital Synthesis" (Пряко цифрово синтезиране). Това е техника за генериране на сигнали, която се използва в електрониката и комуникационните системи. Целта на DDS е да създаде прецизен и стабилен електрически сигнал, който обикновено е синусоидален, но може да бъде и друг тип сигнал (като правоъгълен, триъгъленен и др.).

DDS работи като комбинира малък брой основни стъпки на фазата или честотата на сигнала, за да създаде желания сигнал. Този процес се извършва чрез управление на цифровите стъпки и фази на основния източник на сигнали. Поради този начин на работа, DDS предлага много голяма прецизност и гъвкавост при генерирането на сигнали.

**Пример – DDS**

Програма на C#, която демонстрира използването на класове, методи, променливи и входни данни от потребителя:





Този код създава конзолно приложение, което представя аптека, където потребителят може да въведе данни за лекарство - име на лекарството, доза и продуктов код. След това програмата създава обект от клас Medicine, който съхранява въведените данни, и ги извежда на екрана.

Когато стартирате програмата, тя пита потребителя за името на лекарството, дозата и продуктовия код. След това създава обект от тип Medicine с въведените данни и ги извежда на екрана.

Този пример демонстрира основните концепции на програмирането на C#, като класове, обекти, методи, входни данни от потребителя и извеждане на информация в конзолата. Той е много опростен, но може да бъде разширен и подобрен с добавяне на валидация на данните, обработка на грешки и други функционалности според нуждите на приложението.

1. **Какво е прототипиране на потребителския интерфейс? Опишете процеса.Пример.**

Прототипирането на потребителския интерфейс (GUI) е процесът на създаване на предварителни модели или прототипи на софтуерния интерфейс, който потребителите ще използват. Този процес има за цел да се изследва и тества функционалността, дизайна и потребителското изживяване преди финалната разработка на продукта.

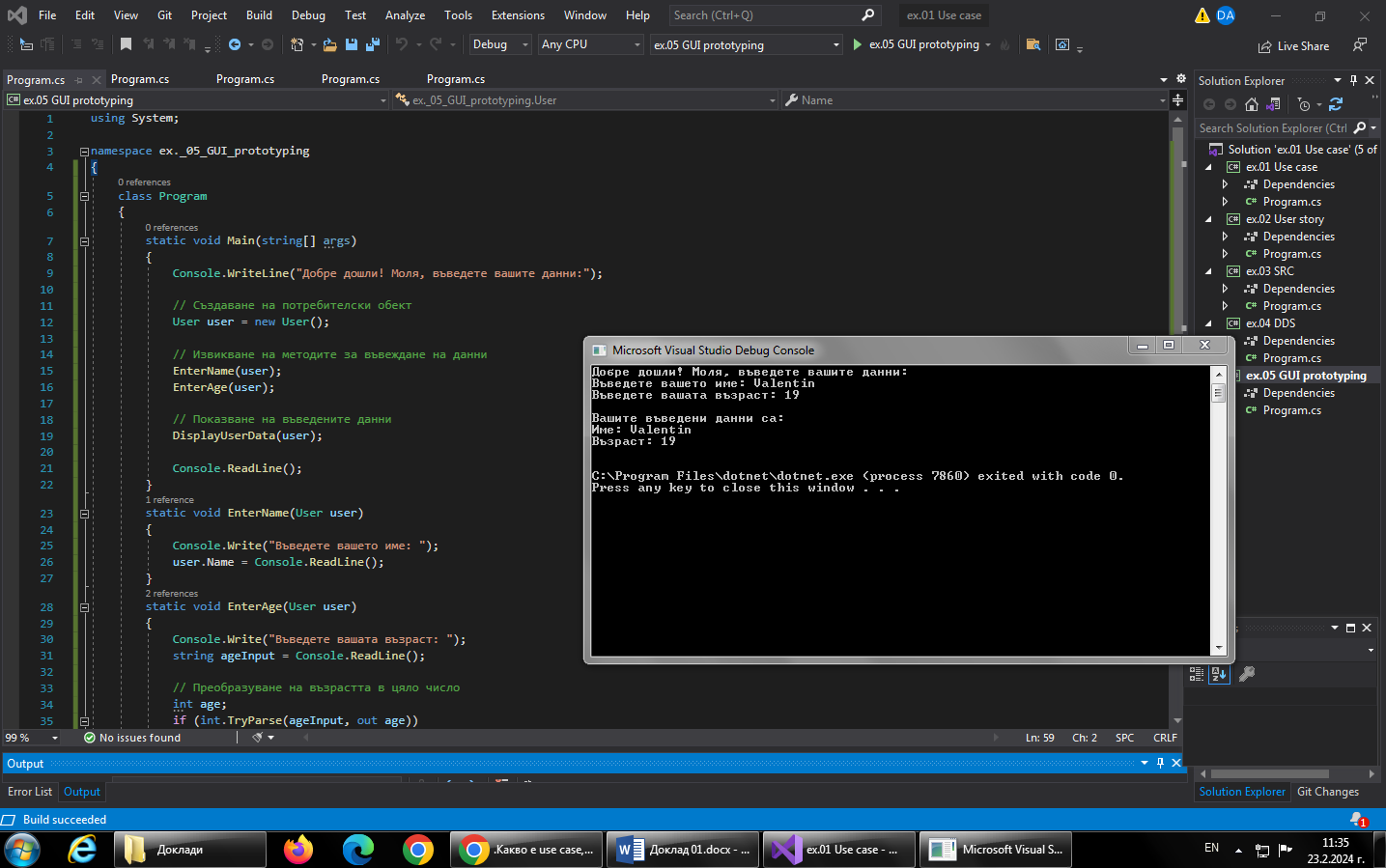
**Основните стъпки в процеса на прототипиране на потребителския интерфейс:**

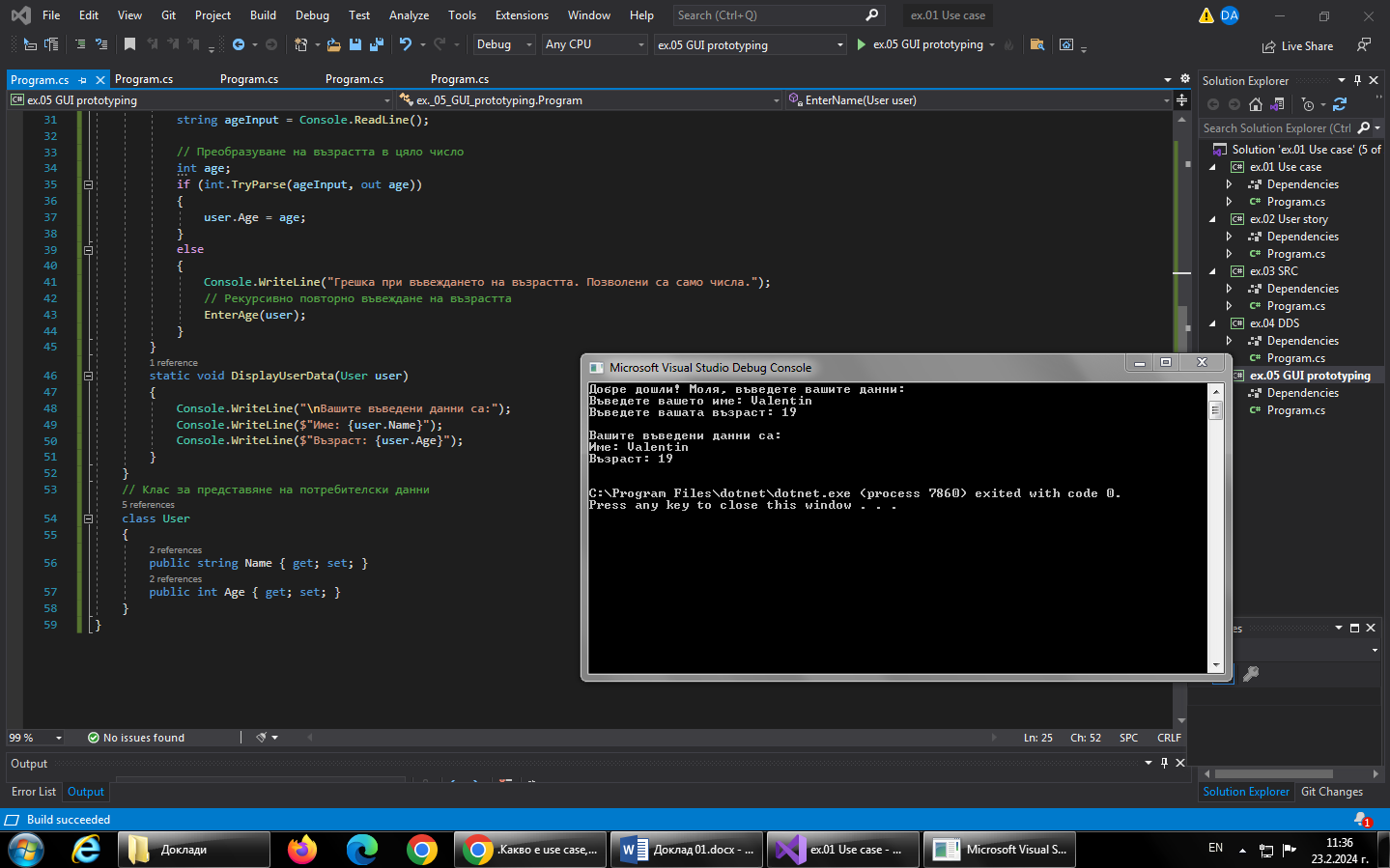
* Събиране на изисквания: Първата стъпка е да се разбере целта на приложението или уеб сайта, както и нуждите на потребителите му. Тук се определят функциите, които трябва да бъдат включени в интерфейса.
* Създаване на концепция: На база на събраната информация се разработва концепция за интерфейса. Това включва начертаване на основните елементи като бутони, полета за въвеждане на текст, менюта и други.
* Low-Fidelity прототипи: На тази стъпка се създават нискофиделитетни прототипи, които са обикновено бързи, скицирани версии на интерфейса. Те служат за изследване на различни възможности за организация на елементите и навигацията.
* Тест и оценка: Нискофиделитетните прототипи се използват за тестване с потребители или за получаване на обратна връзка от клиентите и екипа по дизайн. Този процес позволява да се идентифицират проблеми и да се внесат корекции преди да се премине към по-нататъшни стъпки.
* High-Fidelity прототипи: След като основните проблеми са идентифицирани и решени, се създават по-детайлни прототипи с по-реалистичен вид. Тук вече се включва цветове, шрифтове, изображения и други дизайнерски елементи.
* Проверка и корекции: Високофиделитетните прототипи се използват за допълнително тестване и оценка. Потребителите, клиентите и други заинтересовани страни имат възможност да изразят своето мнение и да се правят необходимите корекции.
* Подготовка за разработка: Когато прототипът е одобрен, той служи като основа за разработката на истинския потребителски интерфейс. Разработчиците използват прототипа като референция за визуалния дизайн и функционалността на приложението или уеб сайта.

Този процес на прототипиране на потребителския интерфейс помага да се увери, че крайният продукт отговаря на нуждите и очакванията на потребителите, като също така помага да се намалят разходите и времето, изразходвани за разработката чрез идентифициране на проблемите предварително и търсене на решения за тях.

**Пример - Прототипиране на потребителския интерфейс**

В следващия пример ще създадем проста конзолна програма, която прототипира интерфейс за въвеждане на данни за потребител и тяхната възраст. Това ще включва използването на класове за представяне на данните и няколко метода за взаимодействие с потребителя. Не е включена валидация на входните данни за целите на простота на примера.





Този пример създава прост конзолен потребителски интерфейс, който позволява на потребителя да въведе своето име и възраст. Той използва клас User за представяне на данните на потребителя и има три метода:

* EnterName(User user): Този метод изисква от потребителя да въведе името си и го записва в свойството Name на обекта user.
* EnterAge(User user): Този метод изисква от потребителя да въведе възрастта си, като валидира входа за цяло число и го записва в свойството Age на обекта user.
* DisplayUserData(User user): Този метод просто извежда в конзолата въведените от потребителя данни за име и възраст.

Този пример е много опростен и представлява базово прототипиране на потребителския интерфейс в конзолна среда. В реални приложения прототипирането би включвало по-сложни елементи на интерфейса, валидация на данните, обработка на грешки и други функционалности.

**Използвана литература/Източници:**

<https://ff.tu-sofia.bg/~bogi/StopFak/SoftEngr/uml_lab1.pdf>

<https://www.visual-paradigm.com/guide/agile-software-development/what-is-user-story/#:~:text=A%20user%20story%20is%20a,simplified%20description%20of%20a%20requirement>.

<https://en.wikipedia.org/wiki/SRC>

<https://www.src.org/>

<https://bg.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D1%80%D0%B5%D0%B1%D0%B8%D1%82%D0%B5%D0%BB%D1%81%D0%BA%D0%B8_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81>