

# Софтуерно осигуряване на качеството: Въведение

QA, тестване, бъгове, принципи на тестване, процес на тестване



СофтУни

Преподавателски екип



SoftUni



Софтуерен университет

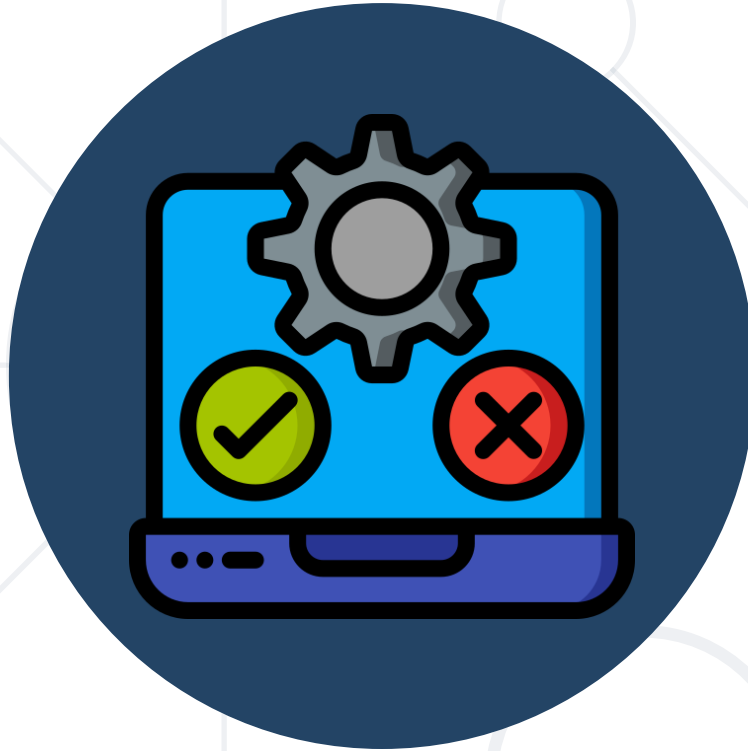
<http://softuni.bg>

1. Какво означава **софтуерно осигуряване на качеството (SQA)**?
2. Какво означава **софтуерно тестване**?
3. Софтуерни **дефекти** (бъгове)
4. Сравнение между **Ръчно** тестване и **Автоматизирано** тестване
5. Седем **принципа** в тестването
6. Тест **сценарии** и тест **случаи** (Test Scenarios and Test Cases)



sli.do

**#QA-Basics**

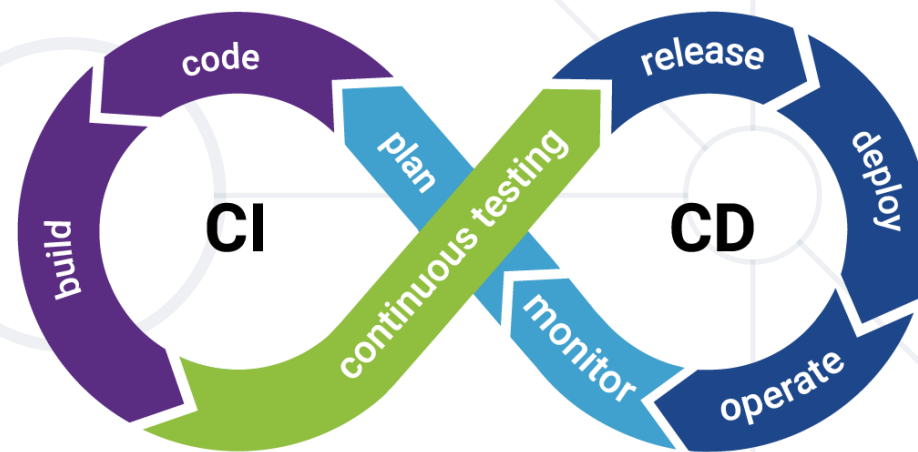


**Какво означава софтуерно осигуряване  
на качеството?**

**Какво означава софтуерно тестване?**

- Какво е **софтуерно осигуряване на качеството (SQA / QA)**?
  - SQA има за цел да **гарантира**, че софтуерът **се държи според очакванията**
  - SQA е **методология** за проверка на **съответствието на софтуера спрямо изискванията**
  - По-голяма част от работата на QA е **софтуерното тестване**: **ръчно** и **автоматизирано**
  - **Софтуерните дефекти (бъгове)** се докладват и проследяват чрез **системи за проследяване на дефекти (bug tracking systems)**
  - Процесът по софтуерно осигуряване на качеството се изпълнява от **QA специалисти**

- По-голяма част от QA работата е **софтуерното тестване**
  - **Ръчно** тестване (кликни и провери резултатите)
  - **Автоматизирано тестване** (QA автоматизация посредством скриптове)
- **Непрекъснатата интеграция и непрекъснатото внедряване (CI / CD pipeline)**
  - Автоматизирано изграждане и обновяване в тестова среда (build and deploy)
  - Автоматизирано изпълнение на тестове
  - Известяване (notification) / доклад (report)



- Софтуерното тестване е начин:
  - За оценка на **качеството** на **софтуера**
  - Да се провери дали софтуерът **отговаря** на определени **изисквания** и да се открият бъгове
  - Да се **намали риска** от повреда на софтуера при неговото използване
- **Процесът** по **анализиране** на **софтуерен** продукт включва:
  - Откриване на разликите между **разработения софтуер** и **разписаните спецификации**
  - **Оценка** на функционалностите на софтуерния продукт

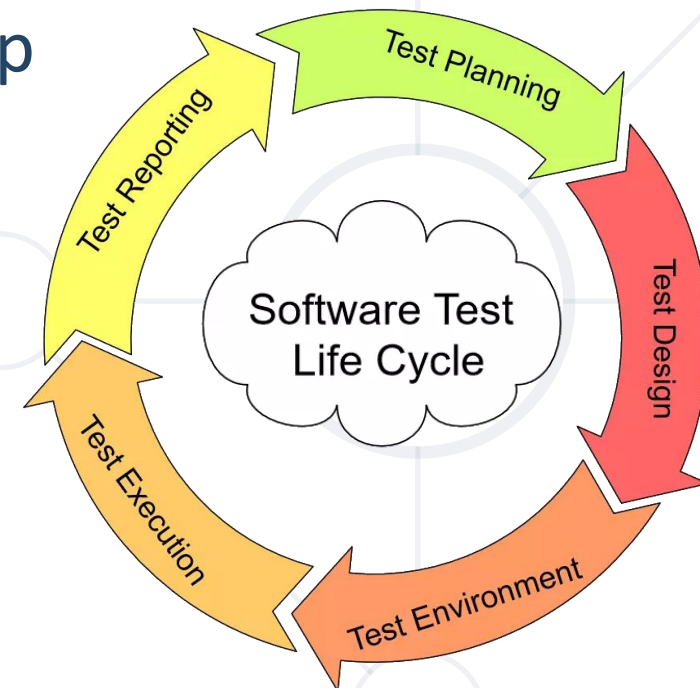
- Основни **цели** на тестването
  - **Предотвратяване** на **дефекти**
  - Верификация на **посочените изисквания**
  - Верификация на **очакваното поведение** на софтуера
  - Да се намали **рискът** от възможен провал на софтуера
  - Да предоставя **информация** на заинтересованите страни
  - Да спомага за спазването на **договорни, законови** или **регулаторни** изисквания





# Процесът на софтуерно тестване

- **Тестването** проверява дали разработеният софтуер **отговаря на изискванията**
- Тестването има за цел **да открие и докладва** дефекти (bugs)
- Процесът по софтуерно тестване включва:
  - **Планиране на тестването**: какво, кога, как?
  - **Дизайн на тестването**: тест **сценарии** и тест **случаи**
  - **Настройка на тестовата среда**: инсталиране, конфигуриране, подготовка на тестови данни, ...
  - **Реализация на тестовете**: изпълнение на тестовете
  - **Отчет на тестването**: регистриране на резултатите от теста и откритите бъгове





# Софтуерни дефекти (Bugs)

Грешки, дефекти, бъгове и неизправности

- Хората допускат **грешки** (пропуски)
- Грешките водят до **дефекти**
  - **Дефектите** = **бъгове** в програмния код или грешки в **изискванията** / **дизайна** / друго
- Ако **бъгът** бъде активиран, това може да доведе до **неизправност**
  - Софтуерът не успява да изпълни това, което се очаква / изпълнява грешни неща
- **QA / софтуерното тестване** цели да намери **бъговете**
  - **Автоматизираното тестване** и **Непрекъснатата интеграция/внедряване (CI / CD)** намаляват бъговете



Грешка

- Програмна **грешка**
  - Функция "събиране", която работи коректно, с изключение на  $5 + 3 = 7$  (скрита/латентна грешка)

Дефект /  
Бъг

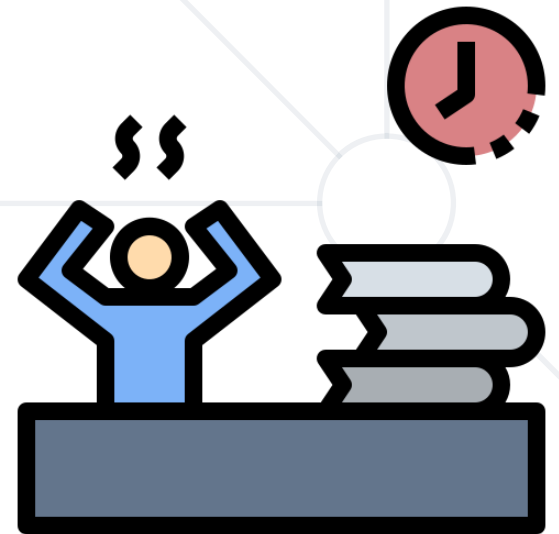
- Активирана грешка -> ефективен **дефект/бъг**  
Извикване на функция "събиране", чрез  $5 + 3$ 
  - Резултат 7 в някаква променлива (вместо 8)

Неизправност

- Неизправност – **отклонение** в поведението на системата
  - Насрочване на среща в 7:00 сутринта, вместо в 8:00 сутринта

# Какво може да доведе до дефекти / бъгове? (1)

- Причини за грешки на програмиста/тестващия могат да бъдат:
  - Липса на **време**
  - Недостатъчно добро **обучение**
  - Сложен **код**
  - Сложна **инфраструктура**
  - Променящите се **технологии**



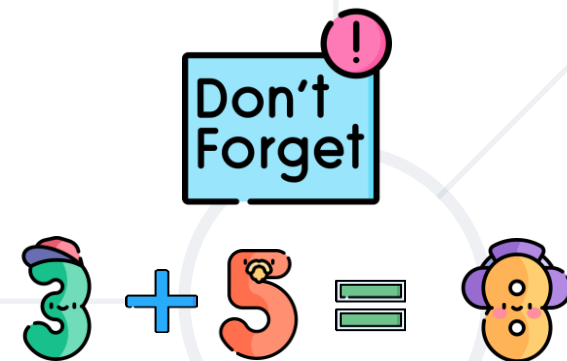
# Примери

- **Примери за липса на време**

- За написването на дадена функция са необходими 10 минути
- Разрешеното време е 2 минути
- По-голяма вероятност да се допусне грешка

- **Примери за недостатъчно добро обучение**

- В софтуер трябва да се имплементира физична формула
- Програмистът не разбира формулата
- Лошо/неправилно изпълнение
- Кодът прави нещо друго



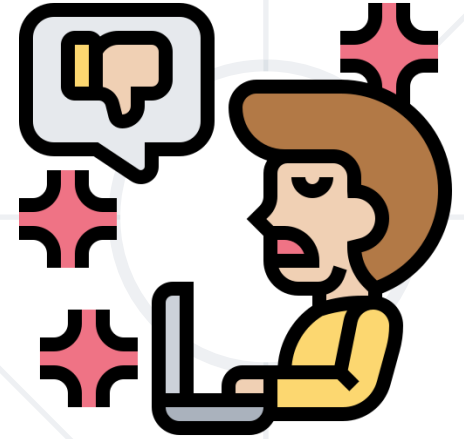
# Какво може да доведе до дефекти / бъгове? (2)

- **Организационни** фактори
  - Неефективна **комуникация**
  - **Неясно** дефинирани изисквания
- **Условия на околната среда**
  - Електронни полета, магнетизъм, радиация, замърсяване и др.
    - Биха могли да повлияят на състоянието на хардуера
  - Неправилна софтуерна среда (напр. грешен IP адрес)
- Пример за **неясно дефинирани изисквания**:
  - "Софтуерът трябва да бъде лесен за използване."



# Какво може да доведе до дефекти / бъгове? (3)

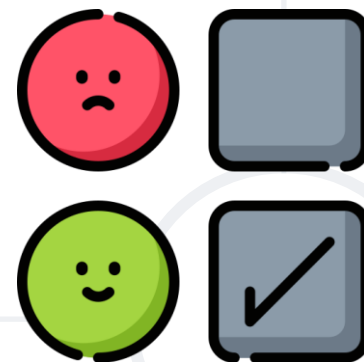
- **Други причини** за бъгове:
  - Неправилна конфигурация или неизправност в **производствената или тестовата среда**
  - **Некоректни** тест данни
    - Правилен тест, който дава **отрицателен резултат**
    - Грешен тест, който дава **положителен резултат**
  - **Некачествени** тестове
  - **Невалидни** очаквани резултати





# Примери

- Пример за **некоректни** тест данни
  - Тестващия регистрира потребителско име "*john123*"
  - Не връща базата данни в първоначалния ѝ вид
  - Втори тестващ се опитва да изпълни същия тест
  - Регистрацията е неуспешна заради **дублиране**, т.е. коректен тест дава **отрицателен резултат**
- **Неправилна конфигурация на производствената среда**
  - Деактивирана функция "изпращане на имейл" на хостинг сървъра
  - Потребител се регистрира, но не получава потвърждаващ имейл

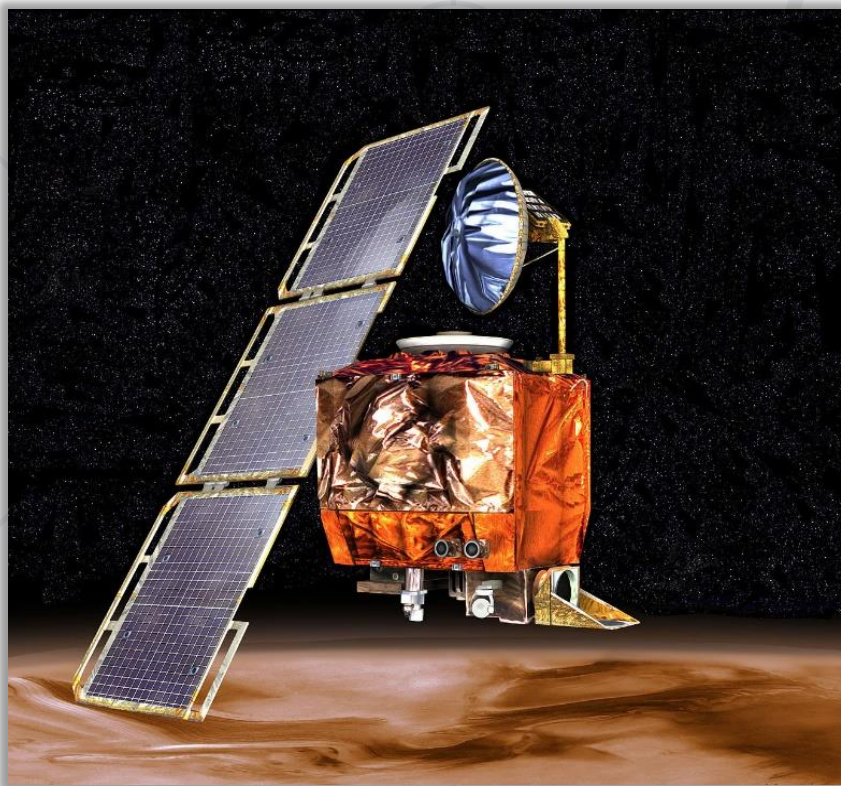




# Фатални софтуерни бъгове

Тежки загуби, причинени от софтуерни дефекти

## Катастрофата на Марс Клаймат Орбитър (1998)



- Предназначен за изучаване на **марсианския климат**, атмосфера и повърхност
- Трябва да поддържа орбита на разстояние **140 – 150 км** от Марс
- Достига **57 км** и бива разрушен от налягането
- Загуба: **\$125 милиона**
- Причини: **използване на грешни мерни единици** (имперски спрямо посочените от НАСА)

## Медицинският ускорител Therac-25 (1985-1987)

- Неизправност в машина за **радиационно облъчване** на раково болни
- Смъртоносни дози радиация са приложени на трима пациенти, други **трима** са тежко **ранени**
- Причина: пропуск в алгоритъма за проверка на грешки



## Космическата сонда Маринър 1 (1962)

- **Маринър 1** е първият космически кораб от американската програма Маринър
- Проектиран да изследва **Венера**
- Грешно функциониране на насочващите **команди**
- Унищожен **5 минути** след изстрелването
- Загуба: **\$18.2 милиона**
- Причина: приликата на горната черта с **тире** ('**┐**' вместо '**-**')





## Еърбъс А300-600R на Китайските Авиолинии (1994)

- Разбива се и се запалва по време на кацане на **летището в Нагоя**
- Загуба: **\$40** милиона + **264 човешки живота**
- Причина: пилотска грешка и липса на препоръчаната актуализация на софтуера (Китайските авиолинии преценяват, че "**не е спешно**")



## Ариана 5, полет 501 (1996)



- Най-новата безпилотна ракета за изстрелване на сателити в Европа
- Унищожена **36,7** секунди след изстрелването
- Загуба: **\$8** милиона. Носи сателит на стойност **\$500** милиона долара
- Причина: софтуерът се опитва да побере **64-битово число в 16-битово пространство**
- Видео: <https://youtu.be/qnHn8W1Em6E>



# Ръчно или автоматизирано тестване

Ръчни кликвания или автоматични скриптове



## ■ Ръчно тестване

- Тип софтуерно тестване, при което тестовете се **изпълняват ръчно**, без използване на **автоматизирани инструменти**
- Човек изпълнява тестовете **стъпка по стъпка**, без тест скриптове
- Тестовете се изпълняват индивидуално, **един по един**

## ■ Автоматизирано тестване

- Тип софтуерно тестване, при което тестовете се **изпълняват автоматично** чрез "структура" за автоматизация на тестове (test automation frameworks)
- Тестващите използват **инструменти** и **скриптове**, за да автоматизират повтарящи се дейности
- Включва **писане на код** и **поддръжка** на тестове

# Ръчно или автоматизирано тестване

| Аспект на тестването  | Ръчно   | Автоматизирано  |
|-----------------------|---|---|
| Изпълнение на теста   | Изпълнява се <b>ръчно</b> от QA специалисти                         | Изпълнява се <b>автоматично</b> с помощта на инструменти и скриптове за автоматизация               |
| Ефективност на теста  | <b>Много време, по-ниска ефективност</b>                            | Повече тестове за <b>по-малко време и по-висока ефективност</b>                                     |
| Видове дейности       | Изцяло <b>ръчни</b> дейности  | Повечето дейности могат да бъдат <b>автоматизирани</b> , включително реални потребителски симулации |
| Покритие на тестовете | <b>Трудно е да се гарантира</b> задоволително покритие на тестовете | <b>Лесно се осигурява</b> по-голямо покритие на тестовете   |

- <http://softuni-qa-amazonaws.com/manual-qa-demo>

| Ръчен тест  |   |
|---|---|
| <p><b>CLICK THE BUTTON</b></p> <p>A "Button clicked" message should appear.</p> <p>Click me</p> | <p><b>CLICK THE BUTTON</b></p> <p>A "Button clicked" message should appear.</p> <p>Click me      Button clicked</p> |

- <https://replit.com/@SoftUniQA/AutomationDemo>

## QA-Automation-Demo.py

```
driver = webdriver.Chrome(options=chrome_options)
driver.get("https://manual-qa-demo.softuniqa.repl.co")

sleep(3)
button = driver.find_element(By.ID, "button")
button.click()

msg = driver.find_element(By.ID, "msg")
assert msg.text == 'Button clicked'
```



# Седемте принципа в тестването

Философия на софтуерното тестване

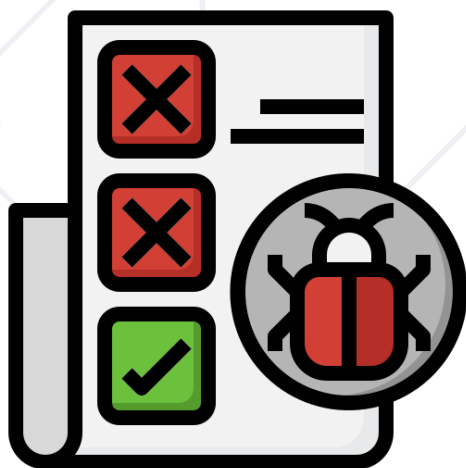
"Софтуерното тестване може да покаже наличието на дефекти, но не и отсъствието им"



- Тестването може да покаже **наличието** на **дефекти**
- Не може да докаже **липсата** на **дефекти**
- Подходящото тестване **намалява вероятността** за наличие дефекти



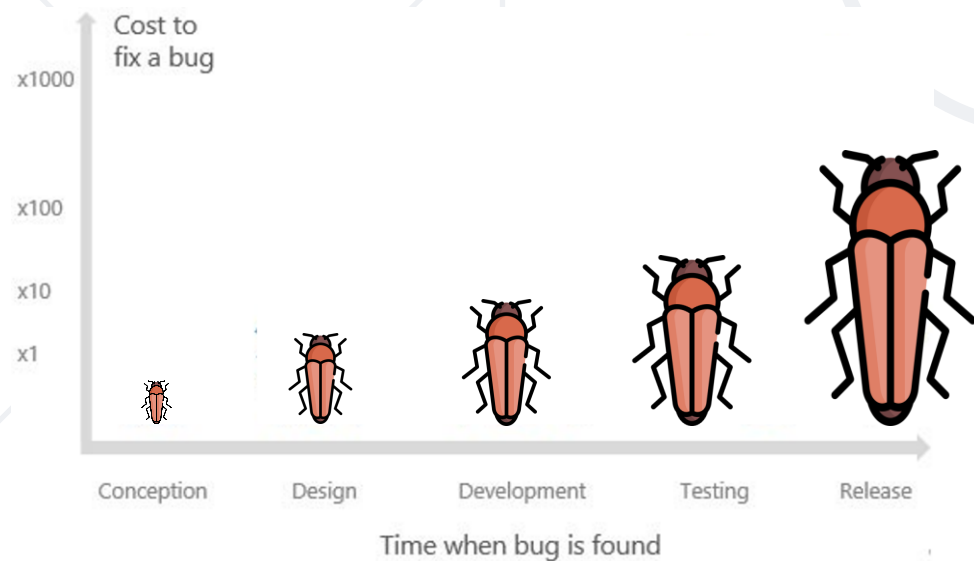
## "Изчерпателното тестване е невъзможно"



- **Комбинациите** от входни данни и тестови условия са безкрайни
- Да се тества всичко е **невъзможно**
- След направена **оценка на риска, приоритет** имат тестовете с най-висок за системата риск



## "Ранното тестване спестява време и пари"



- Дейностите по тестването трябва да започнат **възможно най-рано**
  - Те трябва да са фокусирани върху предварително определени цели
- Колкото **по-късно** се открие един бъг – толкова **по-висока е цената!**





## "Струпване на дефекти"



- **Тестването** трябва да бъде **правилно** насочено
  - 80% от проблемите са породени от 20% от модулите в системата (Принцип на Парето)
- **Фокусът** пада върху 20%, от които идват повечето проблеми



## "Парадокс на пестицидите"



- **Повтарянето** на едни и същи тестове води до **намаляване на ефективността** им
  - **Неоткритите по-рано** бъгове си остават **неоткрити**
- Необходима е разработка на **нови** и/или **модифицирани** тестове



## "Тестването зависи от контекста"



- Тестовете трябва да са **съобразени и подбрани** в зависимост от приложението, което ще се тества
- Софтуер изискващ високо ниво за безопасност, се тества по различен начин от този за електронна търговия



## Заблудата "Липса на дефекти"



- Схващането, че софтуер с **малък брой бъгове** е успешен продукт е **погрешно**
- Самото намиране и отстраняване на бъгове е безсмислено, ако:
  - Изградената система е **неизползваема**
  - Не отговаря на **нуждите** и **очакванията** на потребителите





# Тест сценарии (Test Scenarios)

Истории за тестване

- Какво е "тест сценарий"?

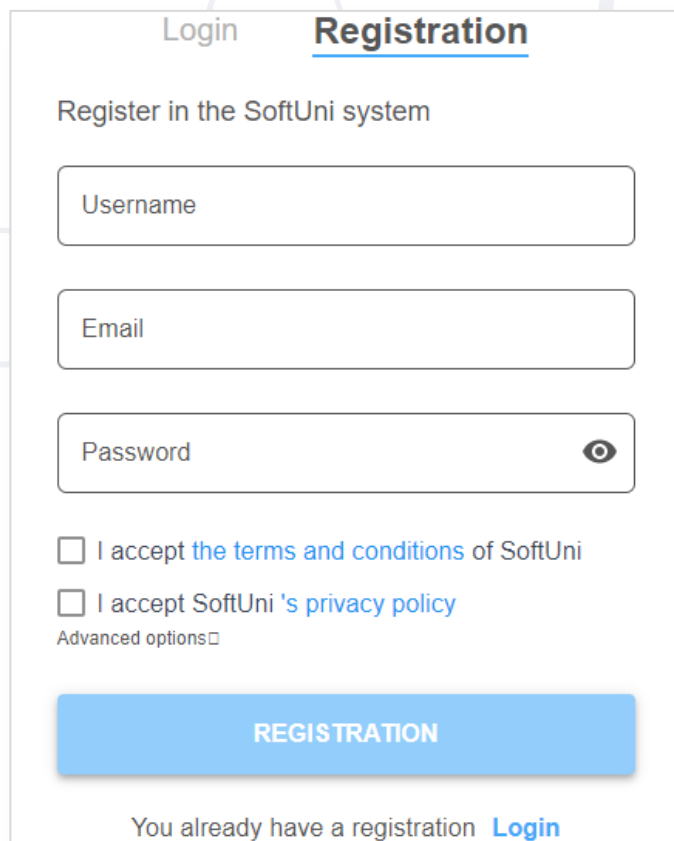
- Всяка **функционалност** / **свойство** / **потребителска история**, която може да бъде тествана
- Нарича се още "**story under test**" или "**feature under test**"
- Пример: *тествай формата за вход*

- Защо ни е необходим?

- **Сложните системи** могат да бъдат разделени на няколко **тест сценария**
- Задава **посоката**, в която ще се тества
- За изучаване на **функционалността** на програмата **от край до край** (end-to-end functioning)

## ■ Тест сценарий 1:

- Регистрирай се в платформа



Registration

Register in the SoftUni system

Username

Email

Password

☐ I accept [the terms and conditions](#) of SoftUni

☐ I accept SoftUni 's [privacy policy](#)

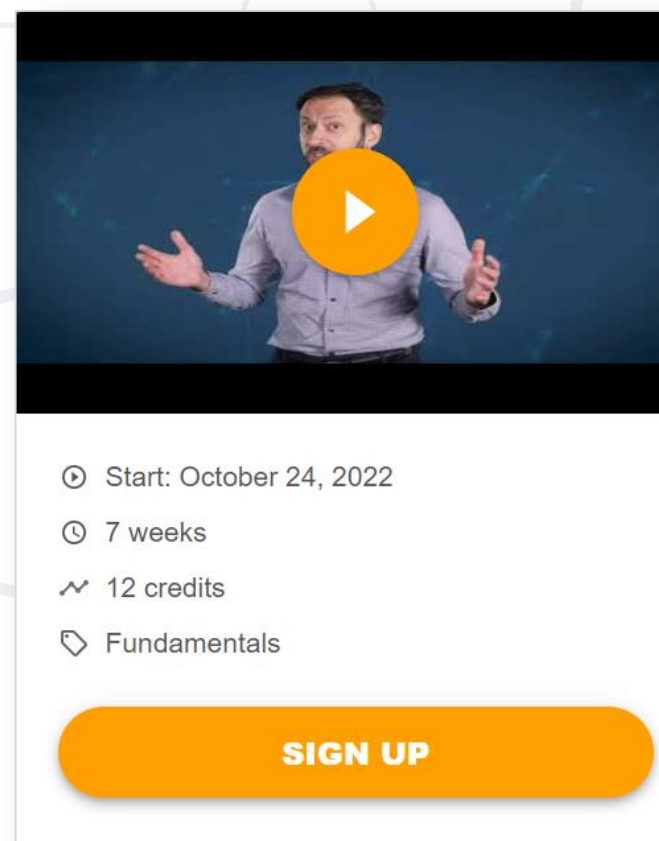
Advanced options ☐

REGISTRATION

You already have a registration [Login](#)

## ■ Тест сценарий 2:

- Запиши се за курс



Start: October 24, 2022

7 weeks

12 credits

Fundamentals

SIGN UP

- Един **тест сценарий** включва няколко **тест случая** (test cases)



- Пример:
  - Потребителска история:  
*Потребителите трябва да могат да "влязат" с потребителските си данни*
  - Тест сценарий : *Вход с потребител + парола*
  - Тест кейс:
    - Вход с валиден потребител + парола → успешно влизане
    - Вход с невалиден потребител + парола → грешка



# Как се изготвя тест сценарий?

- Запознайте се с **документите с изискванията** (requirements)
- Помислете за възможните **потребителски действия** за всяко едно изискване
- **Всяка функционалност** трябва да има собствен тест сценарий
  - Създайте тест кейсове, които покриват **очакваното потребителско поведение**
  - Създайте тест кейсове, които покриват **неочакваното потребителско поведение**
- Уверете се, че сте **покрили всички изисквания**
- Предайте сценариите за **преглед**





# Тест Случаи (Test Cases)

Тестове на единична, конкретна функция

- Какво представляват **тест кейсове**?
  - Поредица от **действия**, изпълнявани с цел да **проверят конкретна пътека** на изпълнение
  - Могат да включват специфични входни и изходни **условия**
- **Защо са ни необходими?**
  - За сравнение на **очакваните** с **действителните** резултати
  - За проучване на начина на **функциониране** на даден софтуерен компонент с определен **вход** и при определени **входни условия**

# Тест кейс

- Поредица от **стъпки** за проверка на **правилното** поведение
- **Поне два тест случая** за тестване на определен сценарий
  - Положителен тест
  - Отрицателен тест
- Тест случаите се състоят от:
  - Заглавие (+ незадължително описание)
  - Стъпки за изпълнение
  - Очакван резултат



**Заглавие:** <заглавие на теста>

**Описание:** <кратък преглед>

**Стъпки:** <начин на действие>

1. ...
2. ...
3. ...

**Очаквани резултати:**

- ...
- ...
- ...

**Заглавие:** Пригответе късо кафе

**Описание:** стартирайте кафе машината, сипете вода и смляно кафе и сварете чаша кафе.

**Стъпки:**

1. Включете машината.
2. Поставете смес от смляно кафе в отвора за кафе.
3. Напълнете контейнера с вода до максимум....

**Очаквани резултати:**

- Процесът на варене трябва да завърши за < 50 секунди.
- Чашата за кафе трябва да побира късо горещо кафе (60 мл)....

- Примерен **тест сценарий**:
  - Регистрирайте се в дадена платформа
- **Тест кейс**, част от този сценарий :
  - Несъществуващо, валидно потребителско име → успех
  - Дублирано потребителско име → грешка
  - Празно потребителско име или парола → грешка
  - Твърде дълго потребителско име или парола → грешка
  - Невалидни знаци в потребителското име или парола → грешка

### User Registration

Username:  
**maria**

Password:  
●●●●●●●●

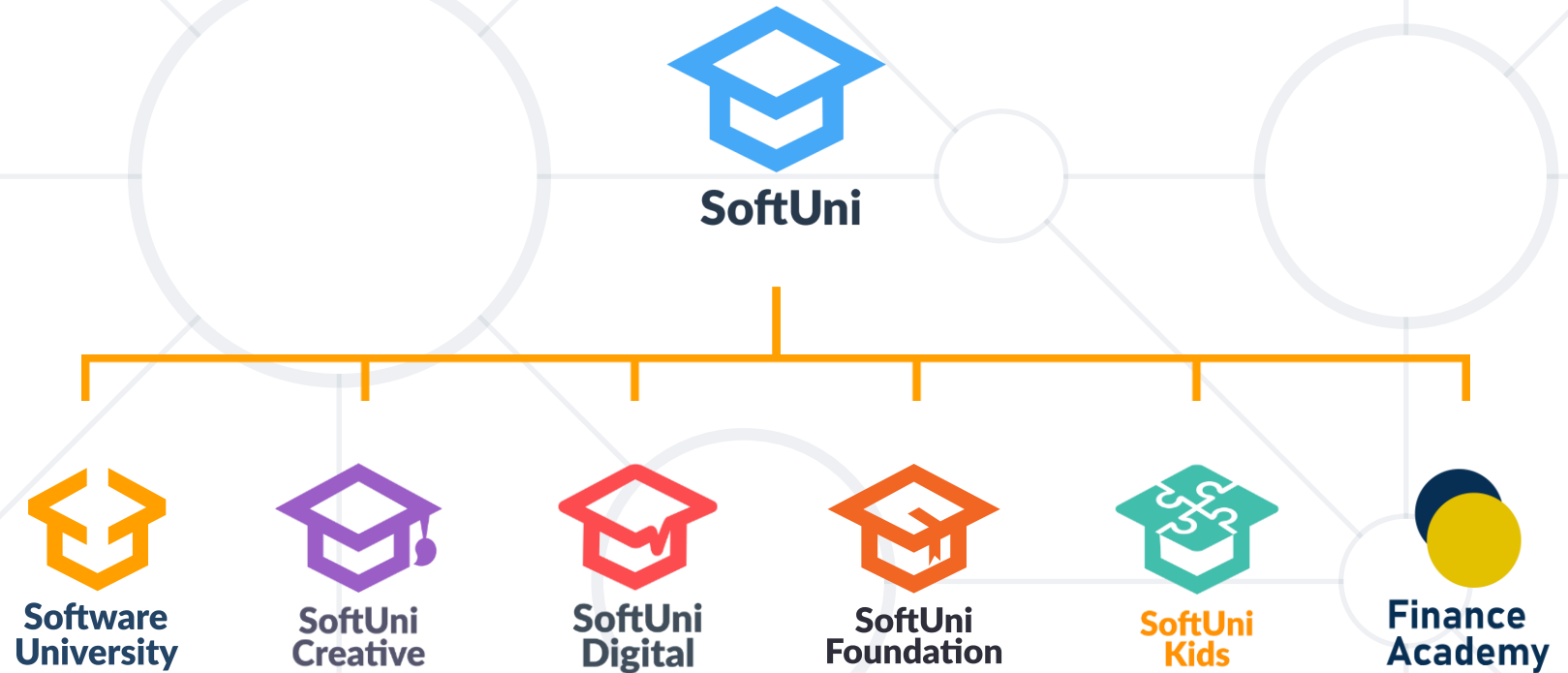
Full Name:  
**Maria Steward**

**Register**

- Дефиниция за **SQA**
- Дефиниция за **Софтуерно тестване**
- **Софтуерни дефекти / Bugs**
- **Ръчно** и **Автоматизирано** тестване
- **Седемте принципа в тестването**
- **Тест сценарий**: тестване на определена функционалност (форма за регистрация)
- **Тест кейс**: различните начини, по които се тества функционалността



# Въпроси?





# Диамантени партньори на СофтУни

**SUPER  
HOSTING  
.BG**



**Coca-Cola HBC  
Bulgaria**

 **Flutter**<sup>TM</sup>  
International

**INDEAVR**  
Serving the high achievers



 **DRAFT  
KINGS**



**BOSCH**



**DXC**  
TECHNOLOGY



- Официален уеб сайт:

<https://softuni.bg/trainings/4357/qa-basics-november-2023>

- Официален дискуссионен форум:

<https://softuni.bg/forum>

- Официална фейсбук група:

<https://www.facebook.com/groups/qabasicsnovember2023>

- Този курс (презентации, примери, демонстрационен код, упражнения, домашни, видео и други активи) представлява **защитено авторско съдържание**
- Нерегламентирано копиране, разпространение или използване е незаконно
- © СофтУни – <https://softuni.org>
- © Софтуерен университет – <https://softuni.bg>



- Софтуерен университет – качествено образование, професия и работа за софтуерни инженери
  - [softuni.bg](http://softuni.bg)
- Фондация "Софтуерен университет"
  - [softuni.foundation](http://softuni.foundation)
- Софтуерен университет @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)
- Дискуссионни форуми на СофтУни
  - [forum.softuni.bg](http://forum.softuni.bg)



Software University

