2. What is API testing?

API stands for Application Programing Interface. It’s basiclly allow software system to communicate with each other.

Also allows us to make request directly to a server or a data store without interacting with the UI. To secure this conenction, the acces is only avaliable if we use tokens for it. The most common is bearer token that is passed in the http request header. The common API standard is REST API – Representational State Transfer. It is rules that has been the common standard since 2000

API testing is about sending request and veryfing responses. With that we test functionalty, performance, security and reliability so it may work as expected.

Request Methods:

GET – we request data acces and we get the information about it

POST – we create something new

PUT – update an already existing object

DELETE – delete object

. Responses HTTP status codes indicating success or failure  
There is Functional testing, where we test that the API return the expected responses for valid requests. There is performance testing. In this case what we are looking for its respond time, and how it behaves under various loads. Security testing is where we try to very if its secure, manly focusing on acces control and data encryption. There is reliabilty Testing where we test its stability over time. And finaly there is validation of response data. In this we ches if it returns the proper formated data.

**200 OK**: The request was successful, and the response contains the requested resource

**201 Created**: A new resource was successfully created.

**204 No Content: The request was successful, but there is no content to return.**

* **Example: Deleting a resource using**

**400 Bad Request**: The server cannot process the request due to invalid syntax

**401 Unauthorized**: Authentication is required, and the request lacks valid credentials

**403 Forbidden**: The client is authenticated but does not have permission to access the resource.

**404 Not Found**: The server cannot find the requested resource

**405 Method Not Allowed**: The request method (e.g., POST) is not allowed on the endpoint

**500 Internal Server Error**: A generic error indicating the server encountered an issue

Question 2

3. What is the difference between regression and smoke testing?

**Regression testing**: It is used to authenticate a code change in the software does not impact the existing functonality of the product. Test cases are re-executed to check the previous functionality of the application is working fine, and the new changes have not produced any bugs. Can be performed on a new build when there is a significant cahnge in the original functionality. For example:

* new functionality is added
* there is a Change Requirement
* a defect is fixed
* the res is a performance issue to fix
* in case of enviroment change

**Smoke testing:** It checks if the core functionality is working after a new build or update so we can check quickly if the software is stable enough for further testing. It is done in a small set on a critical functionality

Question 13

What is Agile?

Agile is a process of implementing guiding principles that allow teams and businesses to rapidly develop products that have value to their business stakeholders. **It aligns software development with the needs of customers while breaking down barriers between traditionallly isolated teams**. Large emphasis what the client wants and what can be done

Agile manifesto:

* **Individuals and interactions** over processes and tools
* **Working software** over comprehensive documentation

User story is during the consultation process, you parse the work to be done into functional increments. It is written by the perspective of an end user. Each user story contribute to the value of the product and help define its purpose. It is usually written by stakeholder, clients, managers, developers or the users them selves. It is not equal to system requirements

Most common user tempalte is the the 5 W’s:

* who – for example as a user
* what – for example what I want to acces via my mobile
* why – because I might not always have a computer
* where
* want

Acceptance Criteria are **the conditons that must be met for the product to be acceptable from a user perspective.** These will be defined as a set of statements with well-established pass/fail results for both functional and non-functional requirements. Any variations will be taken into account. It helps prevent misscommunication

Well written Acceptance Criteria:

* Acceptance criteria are the basis for the testing systems
* Criteria are written by clients, product owners, and/or development team
* Development team reviews acceptance criteria before writing tests

What is TDD?

Test-driven Development. Tests are written before writing the actual code. Write only enough code to pass the test, which leads to simpler and more reliable code. Key steps are the following:  
Red phase where we write a test specific to the feature or function, before we write the code. It describe the desired behavior and small piece of functionality. It will obviously fail beacuse the code does not exist yet, but it will show the perspective of the test. Green phase where we write just enough code to make the thest pass. It focuses on to pass the test with the simplest solotion, so it stops any unnecessary complexity. The Refactor phase is when the code pass the test the code is improved in structure, readability and perfomance, without changing its behavior. After the refactoring the code should still pass.

Where does the TDD fall short:

* In the TDD process, developers want to know what to test, how much to test, and how to understand failing tests –what interface should be tested? how much should be tested?
* Without clear guidelines as to what should be tested and how, confusion and misunderstandings are common
* Presenting TDD in a different light through BDD allows the development process to avoid a number of pitfalls

Question 3

4. What is BDD?

So, What is BDD?:

* Behaviour Driven Development
* BDD takes principles from both **TDD and Agile and aims to present the needs of both business analysts and developers into a single framework**
* It emphasizes collaboration between developers, testers, and non-technical stakeholders. It’s a communication bridge between technical team members. and non-technical
* **Business needs are defined in code and are testable**
* Writing test with expressive names, usually in a form of sentance, that declared what a test was cheking

BDD is not for discovering what product to build. Product owners should now what the end product is

BDD is for building the correct, reliable product that end users will love using

Starting points for BDD:

* Product owners already understand what product they desire
* Some, if not most, acceptance criteria have beed defined
* User stories have already been written

Finding our Acceptance Criteria:

* Developers or QA seek clarity from business stakeholders to define acceptance criteria for an application
* This occurs from conversations with stakeholders based on concrete examples dreived from the user story
* The best way to finding is to ask the stakeholder to provide a specific example
* **Names, Context, Situational** awarness what we need

We need a way to translate our acceptance criteria into executable test, that can verify the behaviour of our system and create confidence that we are building the right thing in the beginning even if these requirements might cahnge later on

The specification language for Cucumber, Gherkin is a business-readable, domain-specific language that describes your software’s bahavior, but not how that behavior is being impelemented. Gerkin serves two purposes: to automate testing as well as acting as documentation for the application.

Gherkin Syntax:

* **Feature:** Feature name. – Some terse, yet descriptive, text of what is desired
* **Scenario:** Some determinable business situation. Single concrete example of a user story, about how a system should behave
* **Given:** Describes the context or precondition for the scenario
* **When:** Some event or action by the actor. The person or an event acting on a system
* **Then:** Some testable expected outcome to the scenario

Question 5

6. What is a test pyramid?

Testing pyramid shows the different levels of testing. The base of the pyramid is **Unit test**: Focusing on individual components of functions of the software. Fast, reliable and provide immidiate feedback. The majority of test should consist of this. The next in the line is **Integration tests**: Howdifferent components of the modules work together providing insight of the systems internals. Fewer test than unit test, but still essential. And finally its **End-to-End** testing:

Its about testing from top to bottom every component of the application, to test if it work properly. From the UI to the back-end. It simulates real user scenarios and ensure the system meets the requirements. It’s however slow and and depend on many external factors. It should be the fewest test.

Question 8

9. How do you report defects? what’s the status of a defect report? from new to…? say all states.

If an error occours we shall make a precise and exact documentation about it. First the summary: In wich enviroment the problem came to us. Then what version did we use: what version of the webpage, browser etc.,Steps to reproduce: how did we achived the error. Description of the actual result: what does the screen showes us, we can prove it with screenshots. Expected result: what should have happend, if the everything worked out fine

Question 9

10. What are the most important steps in the manual testing?

Question 10

11. What’s a constructor? an overload? an overwrite? (override - dang it englishj)

Constructor: a special function or method in a class that is called when an object or a class is created to initialize the object’s state. Overloading: Defining multiple functions or methods with the same name but dfifferent parameters. Override: A subclass that has a function or other implemantation that is already defined in its parent class. In thos case the parent class implementation gets overwritten

Question 11

Question 13

14. What is Agile? Difference between Agile and Waterfall.

Waterfall:

The steps to project completion is broken down into different pahses taht is compelted one after another in linear approach. Requirements 🡪 Design 🡪 Development 🡪 Test🡪 Deploy

Project must be done in a fixed budget, fixed scope, fixed time adn good quality. Waterfall doesnt involve much custumer involvement

Agile:

The development involves the product owner opinions about the way the development takes place. In agile the product is built in small increments for example in a scrum. In those increments we go through Design 🡪 Development 🡪 Test 🡪 Deploy. After the development we take feed back and use it to improve the product

Question 14

15. One developer doesn’t agree with you, how do you handle it? He says the defect it’s not a defect, but it’s a feature. How do you handle this situation?

read more

Question 15

16. What are your three strengths/weakness points?

Question 16

17. Have you worked with Jira, Junit?

Question 17

18. How would you write a manual test case? which are the stages?

Question 18

19. SQL questions. Do you have experience in database testing?

Question 23

24. What is CI/CD?

Continous Integration: Frequently integrating small changes into the main branch. The code is commited, the automated tests are run. Helps with detecting early bugs, making it easier to fix.  
Continious Deployment: If code cahnges pass the automated testing, it will be automatically deployed tp production. No need for manual intervention wich yields faster delivery.

Question 24

25. What is try/catch method?

It’s about hadling exceptions or errors that might happend durning the code execution. It allows to try a block of code and catch any errors that may occur, handeling it without crashing the entire program. There is three blocks in this scenario. **Try** block that contaisn the code that my error out. The next is the **catch** block. It will be executed if an error happens in the try block, allowing the user to respond to it. And aftert that the **finaly** block that is optional. It will run regadles if and error occurred or not. Mainly used for cleaning up task, like closing files and connections

Question 25

26. Did you have asserts in your latest tests? more asserts in one test or just one assert?

Question 26

27. Severity and priority. What is the difference and who should handle it?

Severity is about the impact of a defect on the application. It can be Trivial – Minor – Major – Critical  
Priority refers to the urgency of fixing the defect. It descibes how quicly the issue needs to be resolved. Its levels goes by low – medium and high. The severity is determinded by the testers and devs. The priority is determined by the owners and stakeholders

Question 27

28. What is static testing?

No code execution just examining the code, the documents for potential issues. It helps with early bug detections. It involves formal inspections where a team reviews the requirements and its completion. Walkthroughs where the code author presents the work for a team. And requriemets reviews for where the documentation is examined to ensure clarity, compeltness and feasibility

Question 28

29. What is functional/non-functional testing? give examples

**Functional**: Functional testing ensures that the application works according to the functional requirements or specifications. It focuses on verifying that every feature behaves as expected.

Key Features

 **Focus:** Tests what the software **should do**.

 **Goal:** Verify that the software’s features (as outlined in requirements) work correctly.

 **User Perspective:** Ensures that users can perform their desired actions without issues

**Process**

1. **Understand Requirements:** Review functional requirements or user stories to identify test cases.
2. **Test Case Design:** Write detailed test cases for each function or feature.
3. **Execute Tests:** Run the test cases manually or through automation tools.
4. **Compare Results:** Compare actual results with expected results.
5. **Log and Fix Bugs:** Report issues if the application doesn’t work as expected.

**Examples of Functional Testing**

1. **Login Functionality:**
   * Enter valid username and password → Verify login is successful.
   * Enter invalid credentials → Verify login fails with an appropriate error message.
2. **Shopping Cart Functionality:**
   * Add an item to the cart → Verify the item appears in the cart.
   * Remove an item from the cart → Verify it’s removed.
3. **Payment Gateway:**
   * Process a payment using a credit card → Verify the transaction succeeds.
   * Process payment with invalid card details → Verify it fails with an error.

### ****Types of Functional Testing****

* **Unit Testing:** Tests individual components or modules of code (e.g., a function or method).
* **Integration Testing:** Tests the interaction between different modules or components.
* **System Testing:** Tests the entire system as a whole.
* **User Acceptance Testing (UAT):** Ensures the system meets the user’s expectations.

**Non-functional testing**: Non-functional testing evaluates the **quality attributes** of a system. It checks how well the system performs under different conditions rather than testing specific features

### ****Key Features****

* **Focus:** Tests how the system works.
* **Goal:** Assess the system’s performance, scalability, reliability, and other quality metrics.
* **System Perspective:** Ensures the system operates smoothly and efficiently under various conditions.

### ****Process****

1. **Define Non-Functional Requirements:** Understand the performance, security, usability, and other system expectations.
2. **Plan Tests:** Identify scenarios that simulate expected and extreme conditions (e.g., heavy user load).
3. **Set Up Testing Environment:** Simulate real-world conditions (e.g., multiple users, high network traffic).
4. **Execute Tests:** Use specialized tools to test and measure system behavior.
5. **Analyze Results:** Identify bottlenecks, vulnerabilities, or areas for improvement.

### ****Examples of Non-Functional Testing****

1. **Performance Testing:**
   * Check if a website loads within 2 seconds under normal load.
2. **Load Testing:**
   * Simulate 1,000 concurrent users accessing an e-commerce site and verify the system remains stable.
3. **Stress Testing:**
   * Push the system beyond normal limits (e.g., 10,000 users) to determine its breaking point.
4. **Security Testing:**
   * Test if sensitive data (e.g., passwords, credit card details) is protected.
5. **Usability Testing:**
   * Evaluate how easy and intuitive the application is to use.

### ****Types of Non-Functional Testing****

* **Performance Testing:** Measure speed, response time, and scalability.
* **Load Testing:** Assess how the system handles expected loads.
* **Stress Testing:** Identify system limits under extreme conditions.
* **Security Testing:** Check for vulnerabilities and data protection.
* **Compatibility Testing:** Ensure the system works across different browsers, devices, or platforms.
* **Accessibility Testing:** Verify that the application is usable by people with disabilities.

### ****Types of Non-Functional Testing****

* **Performance Testing:** Measure speed, response time, and scalability.
* **Load Testing:** Assess how the system handles expected loads.
* **Stress Testing:** Identify system limits under extreme conditions.
* **Security Testing:** Check for vulnerabilities and data protection.
* **Compatibility Testing:** Ensure the system works across different browsers, devices, or platforms.
* **Accessibility Testing:** Verify that the application is usable by people with disabilities.

Question 29

30. What is a test plan?

Test Plan Id:Unique id for the doc. Introduction:Overview of the project, of its purpose, the scope, backgorund info. Scope of Testing: description of features, functions, and modules that will be tested. Test objectives: Specific goal to ensuring functionality, performance, security and usability. Test strategy:Plan the approach to the testing, and determining the types of testing.Test Criteria:Entry criteria – specific conditions that must be met before testing begins, Exit criteria – conditions that must be met before testing is finsihed. Test Enviroment: details about hardware, software, network configurations and other reosources. Test Schedule:timelines and milestones.Resource Requirements: Persons and its roles and responsibilites, as well as any tools that needed. Risk Assessment: Analysis of potential risks and strategies for mitigating those. Test Deliverable: List of documents and artifacts that will be produced durning the testing process. Approval and Sign-off: Approve of the test plan

test types

Unit testing: for individual components

Integration Testing: how different modules or components for together

Functional Testing: The application behaves according to requirements

End to End : test the entire application as a real suer

Regression testing: Testing if recent changes affected existing functionality

Performance Test: Responsivnes, stability, scalability, application under heavy load

Security testing: Identify vulnerablitis and security flaws

UI testing:Test about he graphical user interface  
API testing: test that focus on interfaces between different software components

Smole testing: check the basic functionality

Acceptance testing: test about the application meets business requirements

OOP principles, testing approaches

OOP Principles:

### ****Encapsulation:****

### ****Definition****

Encapsulation is the concept of bundling data (attributes) and methods (functions) that operate on the data into a single unit (class). It also restricts direct access to some of the object's components, which is done through **private/protected access modifiers** and allows interaction only through defined methods (**getters/setters**).

### ****Purpose****

* Protect the internal state of the object.
* Control how data is accessed or modified.

**Abstraction:**

### ****Definition****

Abstraction is the process of hiding implementation details and exposing only the functionality to the user. It helps reduce complexity by showing only the essential features of an object.

### ****Purpose****

* Focus on "what" an object does rather than "how" it does it.
* Simplify complex systems.

**Inheritance:**

### ****Definition****

**Inheritance allows a class (child class) to derive properties and methods from another class (parent class). It helps in reusing code and creating a hierarchy.**

### ****Purpose****

* **Reduce redundancy by reusing existing code.**
* **Promote code organization and hierarchy.**

**Polymorphism:**

### ****Definition****

Polymorphism allows objects to take on multiple forms. In practical terms, it means that a child class can override a method from its parent class, and the correct method is called based on the object's type at runtime.

### ****Purpose****

* Add flexibility and extensibility.
* Allow methods to behave differently for different object types.

OOP and Testing

Unit Testing and Encapsulation:Unit testing ensures each calss or method behaves as expacted, no data leakage

Inheritance and Testing: Making sure that inherited methods work as expected, especially if overriden

Polymorphism and Testing:Objects with different parent classes exhibit the expected behavior

Variables:

* Three types of variables: var, let and const
* We need to understand waht scope is. Scope is visibility and accessibility of variables, functions and objects at various parts of the code:
* Global scope – variables can be accessed from anywhere in js code
* Function scope – variables are created when the function starts and deleted when the fucntion is completed
* Block scope – variables declared inside curly brackets lock and can’t be accessed from outside of the block.
* Var – global variable, recommended to aboid
* Let – use let when you need to reassing a bariable or when a variable is only needed within a block
* Const - use cosnt for vaiables taht should not chane their value after being assigned

**Happy Flow**

* **Definition**: The "happy flow" (or "happy path") refers to the **ideal scenario** where everything works as expected. This represents the most common and successful use of a feature or application.
* **Purpose**: To verify that the functionality works correctly under normal conditions.
* **Example**:
  + For a login page:
    - Input valid username and password.
    - Click "Login."
    - Verify the user is redirected to the dashboard.
* **Test Case Characteristics**:
  + Uses valid and expected inputs.
  + Ensures the application behaves as intended.

**Negative Tests**

* **Definition**: These tests validate the system's behavior when it receives **invalid, incorrect, or unexpected inputs**. They ensure the system fails gracefully without crashing.
* **Purpose**: To verify error handling, validation, and robustness.
* **Example**:
  + For a login page:
    - Input an invalid username or password.
    - Leave the username or password field blank.
    - Enter a SQL injection payload like "' OR 1=1 --" and ensure the system resists it.
  + Validate the error messages, e.g., "Invalid username or password."
* **Test Case Characteristics**:
  + Uses invalid data or interactions.
  + Focuses on edge scenarios and error handling.

**Edge Cases**

* **Definition**: These tests target the **extreme boundaries** or unusual scenarios that might break the application. They ensure the system can handle unexpected or borderline inputs and behaviors.
* **Purpose**: To verify the application works properly in uncommon or extreme situations.
* **Example**:
  + For a login page:
    - Enter a username or password that is very long (e.g., 255+ characters).
    - Enter special characters or emojis as input (e.g., 😎🤖).
    - Use the maximum or minimum allowed date in a date picker.
* **Test Case Characteristics**:
  + Uses data at the edges of validity (e.g., upper/lower limits of input fields, extreme scenarios).
  + Tests scenarios that are rare but possible.

What is scrum:

First we need to describe the dvelopment cycle. There is the product owner defines what the product should look like via user stories for example. With the description the developer team makes the product that the user will use. The users give feedback about the said product that the product owner creates a priorited list what should be done named as product backlog. This is when scrum and the scrumteam comes into the picture. The scrum team works in sprints. Sprint is a 2-4 week period. The scrum team consists of the product owner, the developer team and the scrummaster. Each sprint starts with a sprint meeting where the team members select the highest priority tasks from the prdocut backlog in away that it can be done a single sprins session.The list of selected tasks are the sprint backlog. During the scrum, a daily meeting is held where each particitpants answers three questions: waht did you do yesterday? what will you do today? what are the obstacles ahve you faced? The scrummaster makes sure that the scrum practices are followed. The outcome of this is a potentially shippable product. It depends on the product owner if it meets his criteria. A **Sprint Review** is conducted at the end which is the demo of what have been done during a sprint session. **Sprint Retrospective**, where the team reflects on their process and makes improvements.