**Питання на співбесіду на Тестувальника ПЗ:**

1. **Розкажіть про себе**
2. **Чому ви хочете бути тестувальником?**
3. **Що таке тестування?**

Процес, де ми досліджуємо програму чи документацію цієї програми з певною метою (знайти баг, підтвердити, що продукт готовий до релізу тощо).

1. **Назвіть цілі тестування**

* Виявити дефекти до того, як їх виявляють користувачі.
* Надання актуальної інформації про стан продукту зараз.
* Перевірка на відповідність ПЗ всім заявленим вимогам.

1. **Які принципи тестування ви знаєте?**

* Testing shows the presence of defects, not their absence (Тестування демонструє наявність дефектів, але не гарантує їх відсутності).
* Exhaustive testing is not possible (Вичерпне тестування неможливе).
* Early testing saves time and money (Тестування слід починати на ранніх стадіях життєвого циклу розробки програмного забезпечення, щоб знайти дефекти якомога раніше).
* Defects cluster together (Накопичення дефектів. Принцип Паретто: для багатьох явищ 80 відсотків наслідків спричинені 20 відсотками причин).
* Beware of the pesticide paradox (Парадокс пестициду. Якщо повторювати ті самі тести знову і знову, то в певний момент цей набір тестів перестане виявляти нові дефекти).
* Testing is context dependent (Тестування залежить від контексту).
* Absence of errors is a fallacy (Відсутність знайдених дефектів під час тестування не завжди означає готовність продукта до релізу).

1. **Коли слід починати тестування?**

Слід розпочинати тестування на ранніх стадіях життєвого циклу розробки ПЗ, щоб знайти дефекти якомога раніше.

(Пам'ятаємо про принцип №3 – раннє тестування (Early testing)).

1. **Опишіть життєвий цикл Програмного Забезпечення**

Software development lifecycle – період часу, який починається з моменту ухвалення рішення про необхідність створення програмного продукту та закінчується в момент його повного вилучення з експлуатації.

* Планування та аналіз вимог
* Дизайн архітектури продукту
* Розробка продукту (програмування)
* Тестування продукту
* Розгортання продукту (реліз)
* Супровід

1. **Що таке entry criteria та exit criteria?**

Entry criteria (критерії входу в тестування) визначають умови, які мають бути виконані до початку робіт. Наприклад, визначені вимоги, графік, встановлено програму, що тестується.

Exit criteria (критерії виходу з тестування) визначають, які умови мають бути виконано, щоб завершити рівень тестування або набір тестів. Наприклад, закрито всі завдання і баг репорти, досягнуто певного відсотка пройдених тестів.

1. **В чому різниця між QC і QA?**

*Quality Assurance (QA)* – QA забезпечує правильність процесу тестування, підходить до контролю якості глобально, стежить за нормалізацією процесів. (Займається тестуванням програми у всіх фазах життєвого циклу програмного забезпечення. Найбільше уваги в роботі приділяє процесам, але також займається тестуванням продукту.)

Quality Control (QC) – QC передбачає контроль за дотриманням вимог. (Займається тестуванням програми у фазах розробки та тестування. На роботі займається продуктом (додатком, сайтом).

1. **Які методології управління проєктами ви знаєте?**

* Waterfall
* V-модель
* Ітераційна, інкрементальна модель (відмінність: в ітераційній моделі ідея проєкту спочатку маленька, але потім з’являються додаткові функції. В інкрементальній моделі ідея проєкту від самого початку велика)
* Спіральна модель
* Модель хаоса
* Гнучка модель( Agile (еджаїл): Scrum, Kanban)

1. **Що таке Waterfall?**

Waterfall Model – відома послідовним проходженням стадій, кожна з яких має завершитись повністю до початку наступної.

Найчастіше використовується в медичних та військових проєктах.

Переваги: вартість і дедлайни визначені заздалегідь; розробка відбувається швидко; хороший результат лише у проєктах із чітко та заздалегідь визначеними вимогами.

Недоліки: неможливо повернутись на попередню фазу або повторити фазу; неможливо змінити вимоги під час розробки.

1. **Що таке Agile?**

*Філософія Agile* - це набір практик, метою яких є оперативна реакція на зміни під час робочого процесу. Такі підходи допомагають командам швидко реагувати на зворотний зв'язок від клієнтів та замовників, тим самим постійно покращуючи вироблений продукт.

*Цінності Agile:*

* Люди та співпраця важливіші за процеси та інструменти
* Працюючий продукт важливіший за вичерпну документацію
* Співпраця із замовником важливіша за обговорення умов контракту
* Готовність до змін важливіша за дотримання плану

*Суть Agile маніфесту:*

* Вся робота над проєктом розділяється на короткі цикли (ітерації) і ведеться поетапно;
* В кінці кожної ітерації замовник отримує готовий мінімально робочий продукт чи його частину, яку вже можна використовувати;
* Протягом всього робочого процесу команда співпрацює з замовником;
* Будь-які зміни в проєкті вітаються і швидко інтегруються в роботу.

1. **Що ви знаєте про Scrum?**

Підхід передбачає взаємодію кожного етапу розробки ПЗ, як команди розробників (Delivery Team), а й самого замовника (Product owner), і навіть Scrum-майстра.

*Суть даного фреймворку:*

* Робота ділиться на спринти тривалістю 1-4 тижні
* Перед початком спринта команда сама формує список задач на ітерацію
* Під час кожного спринта створюється продукт чи послуга, які можна продемонструвати клієнту
* Після виконання спринту проводиться ретроспектива. Це мітинг, мета якого отримати фідбек від кожного учасника команди, виявити поточні успіхи та проблеми, тобто оцінка та аналіз виконаної роботи.
* Кожен наступний етап нарощуватиме функціонал проєкту, доки всі функції не будуть реалізовані.
* Мітинги, які допомагають в процесі роботи: Щоденний стендап, Планування, Ретроспектива та Sprint Review.
* Обов'язкові ролі: Scrum-master (допомагає вирішити блокери, слідкує за дотриманням скраму) та Product Owner (виставляє пріоритети і цілі спринта)

1. **Розкажіть про Цикл тестування в житті ПЗ?**

*Software testing life cycle* – це послідовність дій, які проводяться в процесі тестування, за допомогою яких гарантується якість ПЗ та його відповідність вимогам.

*Етапи* ***STLC*** *циклу:*

* **Планування тестування (Test Planning):**

*Визначення стратегії та цілей тестування. Розробка тестового плану, який включає опис області тестування, ресурси, розклад і бюджет.*

* **Аналіз вимог (Requirement Analysis):**

В*ивчення вимог до продукту визначення тестових сценаріїв і кейсів. Створення матриці вимог та тестів для забезпечення покриття.*

* **Проектування тестів (Test Design):**

*Розробка тестових випадків та сценаріїв на основі вимог. Визначення даних, необхідних для виконання тестів.*

* **Реалізація тестів (Test Implementation):**

*Підготовка тестових середовищ та даних. Запуск тестів та реєстрація результатів.*

* **Виконання тестів (Test Execution):**

*Запуск тестових кейсів та сценаріїв. Запис результатів та виявлення дефектів.*

* **Звітність та метрики (Test Reporting and Metrics):**

*Підготовка звітів про статус тестування та якість продукту.*

*Збір та аналіз метрик тестування.*

* **Завершення тестування (Test Closure):**

*Підготовка фінальних звітів та аналіз виконаного тестування.*

*Підготовка тестової документації та архівування ресурсів.*

* 1. **Які види тестової документації ви знаєте?**

Тест стратегія, тест план, тест полісі, вимоги, тест кейс, чек ліст, тест сьют, баг репорт, тест репорт (Test Run Result, тест самері репорт).

* 1. **Що таке вимоги і які види вимог ви знаєте?**

Це документ в якому описується що програма має робити і як. Види: software requirements (реквайментс) specification  (спесификейшн), product requirement document, user stories, acceptance criteria.

* 1. **Що таке User Story?**

Це неформальні вимоги, що пишуться в один рядок і складаються з трьох частин: “Як …, я хочу …, щоб …”. Також Юзер сторі - це формат завдання в Джирі, який має підзавдання.

* 1. **Що таке тест кейс?**

Це інструкція з тестування, яка складається з кроків для відтворення та очікуваного результату.

* 1. **З чого складається тест кейс?**

Тестові дані, пріоритет, тип (категорія), номер, назва, preconditions, steps to reproduce, designer, дата створення, очікуваний результат. Можуть бути скріншоти. Може бути модуль до якого відноситься test case - test suite.

* 1. **Що таке чекліст?**

Це список, який скл з 3 колонок: ID кроку, назва, статус (passed, failed, blocked, in progress, skipped), тестові дані*.*

* 1. **Чим відрізняється чекліст від тест кейса?**

Чек ліст простіший, тест кейс об'ємний і детальніший.

* 1. **Що таке тест план?**

Тестовий документ, який описує як буде відбуватись процес тестування.

* 1. **З чого складається тест план?**

Описуємо як ми будемо тестувати, що будемо/не будемо тестувати, які рівні/техніки будемо використовувати. Описуємо ролі, обов'язки (roles and responsibilities), ризики та способи їх знизити. Entry, exit критерії. Графік, environment.

* 1. **Чим тест план відрізняється від тест стратегії?**

Тест план це документ який описує тестування на маленькому рівні: продукту, релізу, спринту, юніту, фічі. Стратегія описує тестування на рівні проєкта. Тест полісі- документ, який описує тестування на рівні організації.

* 1. **Що таке Traceability matrix?**

Матриця відслідковуваності - табличка, де ми співставляємо 2 будь-яких тестових документи: тест-кейси до вимог, тест-кейси до багів, тест-кейси до тест-кейсів (матриця залежності). Аналіз впливів: фічі до фічей (зазвичай у регресійному тестуванні). Ця матриця робиться щоб переконатися що у нас немає тестів, які нічого не перевіряють.

* 1. **Що таке Impact Analysis?**

Аналіз впливів, один із видів матриці відслідковуваності, де ми співставляємо дві фічі на предмет залежностей, щоб визначити що треба тестувати для регресії.

* 1. **Що таке баг?**

Невідповідність до вимог, недолік в програмі.

* 1. **Чим дефект відрізняється від баг-репорта?**

Баг-репорт-документ, який описує баг. Дефект - формальна назва багу.

* 1. **Що таке баг-репорт?**

Докладна покрокова інструкція для відтворення помилки

* 1. **З чого складається баг-репорт?**

ID, назва, Preconditions, Steps, Actual/Expected result, Severity, Priority, Type, людина, яка зробила цей баг-репорт, людина, яка має його виправити, статус, середовище, скріншоти, відео.

* 1. **Що таке Severity та Priority?**

Пріоритет - це як швидко треба виправити цей баг, і як він впливає на авторитет компанії. Северіті - ступінь впливу, показує наскільки баг ламає програму.

* 1. **Наведіть приклад багу з низьким пріоритетом, але високим ступенем впливу.**   
     Не можливість перейти за посиланням в  інсту на якомусь веб сайті. Зазвичай якщо ми маємо високе severity, то і пріоритет теж буде високим або середнім. Бо такі баги це велика рідкість.
  2. **Опишіть життєвий цикл баг-репорта.**

New, Open, Assigned, In progress. Duplicate - якщо такий баг вже є. Deferred - якщо баг не терміновий, то його можна перенести на наступний спрінт. Fixed, verified (перевірено), reopened- якщо щось невірно пофіксили. Closed.

* 1. **Як ви будете тестувати якщо вимог нема?**

Шукати вимоги в Юзер Сторіз (завдання в Джирі), і там може бути опис задачі. Шукати в конфлюенсі. Пробувати гуглити. Якщо такого нема, спитати Продакт Овнера. Якщо його нема, то лишається здоровий глузд, аналіз конкурентів.

* 1. **Коли QA-інженер повинен звертатися за допомогою?**

В останню чергу після гугла і конфлюенса. Спочатку питати колегу, потім якщо він не знає то ліда, потім продакт овнера.

* 1. **Які види тестування ви знаєте?**

Позитивне\негативне, мануальне\автоматизоване, статичне\динамічне, чорний\білий\сірий ящик, функціональне, нефункціональне і пов’язане зі змінами.

* 1. **Чим верифікація відрізняється від валідації?**

Verification - процес оцінки системи чи її компонентів з метою виявлення чи задовольняє результат поточного етапу розробки вимогам, які були сформовані на початку даного етапу. (процес, коли звіряється чи вірно щось відбувається, працює програма відповідно до вимог)

Validation - визначення відповідності розробленого ПО очікуванням та потребам користувача, вимогам до системи. (перевірка кінцевого результату. Чи вийшов продукт таким яким хотіли?)

* 1. **В чому різниця між статичним та динамічним тестуванням?**

Статичне тестування - це перевірка документації, динамічне - перевірка програми. Статичне може виконуватись без використання комп’ютера, а динамічне тільки з ним.

* 1. **В чому різниця між білим, чорним та сірим ящиками?**

Чорний ящик - неможливо погрузитися в технічну складову продукту, є тільки уявлення як у юзерів. (Доступ тільки до вимог і самої програми)

Білий ящик - протилежність чорному, тобто є  доступ до коду, бази даних та документації АПІ.

Сірий ящик - щось середнє між чорним та білим. Не маємо доступ до коду, але є доступ до інших тех. областей. Яскравий приклад - тестування API використовуючи Postman.

* 1. **В чому різниця між позитивним та негативним тестуванням?**

Позитивне тестування - коли ми вводим валідні дані і очікуємо на хороший результат. Негативне - коли ми вводимо завідома не коректну інформацію і очікуємо на помилку.

* 1. **Скільки негативних перевірок може бути за 1 раз? (Реєстрація, скільки полів можна ввести негативних)?**

Тільки одне поле заповнюємо негативною інформацією, всі інші - позитивними. Так легше локалізувати дефект.

* 1. **В чому різниця між мануальним та автоматизованим тестуванням?**

Мануальне тестування - це тестування з використанням клавіатури і миші, а автоматизоване - це коли інженер написав код, який виконує тестування.

* 1. **Які рівні тестування ви знаєте?**

Component testing, Integration testing, System testing, Acceptance testing

* 1. **Які типи нефункціонального тестування ви знаєте?**

Перформанс група (пов’язані з навантаженням): Load, Performance, Stress, Recovery, Volume, Scalability, Endurance etc.; UI, UX\Usability, Localization, Internationalization, Configuration

* 1. **Які типи тестування, пов’язані зі змінами, ви знаєте?**

Smoke testing, Sanity testing (перекладається як “розсудливість”, на співбесідах не люблять коли кажуть “санітарне тестування” ), Regression testing,  Re-testing або Confirmation testing

* 1. **Чим смоук тестування відрізняється від регресивного?**

Smoke - перевіряємо критично важливий шлях, функціонал.

Regression - перевіряємо старий функціонал, щоб він не зламався з новою доробкою. Зазвичай Смоук проводять безпосередньо перед регресією.

* 1. **В чому різниця між смоук та саніті тестуванням?**

Smoke - перевіряємо критично важливий шлях, функціонал. Зазвичай вважається “поверхневим” тестуванням.

Sanity - тестування конкретної невеликої частини; перевірка частини, яка найчастіше виходить з ладу; перевірка нового функціонала.  Його також називають тестуванням “вглиб”, тобто детальної перевірки.

* 1. **Що таке регресія? Реліз? Code freeze?**
  2. **Якщо тест кейсів багато, а часу на тестування нема, як обрати тести на регресію?**

Потрібно пріоритезувати тест кейси. Необхідно дивитися спочатку на критично важливий функціонал, який є суттю даного ПО. Якщо вистачить часу, то потрібно ще пройтись по самому популярному шляху продукту та перевірити блок, де було більше всього багів.

* 1. **Що таке Monkey testing?**

Monkey testing є методом тестування програмного забезпечення, при якому тестові дані та вхідні параметри генеруються випадковим чином без спеціального планування чи знань про внутрішню структуру програми. Цей метод також відомий як "random testing" або "fuzz testing".

* 1. **Що таке End-to-End testing?**

Наскрізне, E2E, ланцюгове тестування (end-to-end) тестування - форма тестування, що використовується для тестування програмного забезпечення від початку до кінця, а також його інтеграції із зовнішніми інтерфейсами. Метою наскрізного тестування є тестування всього програмного забезпечення на предмет залежностей, цілісності даних та зв'язку з іншими системами, інтерфейсами та базами даних для перевірки успішності повного сценарію виробництва. E2E тестування зазвичай проводиться після функціонального і системного тестування. Він використовує реальні дані та тестове середовище для імітації робочого режиму.

* 1. **Які ви знаєте техніки тест дизайну?**

Еквівалентне розділення, граничні значення, таблиця прийняття рішень, попарне тестування, діаграма станів і переходів, діаграма сценаріїв використання (юз кейс), вгадування помилок, дослідницьке тестування, чек-ліст бейзд тестінг.

* 1. **Є Input поле, яке приймає цілі значення від 18 до 99 включно. Потрібно протестувати за допомогою технік Boundary Values Analysis і Equivalence Partitioning.**

ЕП: 10, 20, 100

БВ: 17, 18, 99, 100 АБО 17, 18, 19, 98, 99, 100

* 1. **В чому різниця між еквівалентним розділенням та граничними значеннями?**

Еквівалентне розділення - розбиваємо діапазони на класи. Все що входить в клас - всі значення повинні бути еквівалентні один одному за результатом.

Граничні значення - перевірка границь діапазонів та значення поряд з границею.

* 1. **Навіщо потрібна таблиця прийняття рішень?**

Щоб перевірити всі умови, не пропустити важливі перевірки і виділити самі цікаві і основні перевірки.

* 1. **Як створювати тест кейси з ТПР?**

Якщо в ТПР 16 колонок, то не обов’язково створювати 16 тестів. Краще виділити 1 позитивний, по 1 негативному на кожне поле, і всі цікаві (спірні) тести, де треба було уточнити вимоги.

* 1. **Що таке ad-hoc тестування?**

Вільне тестування (ad-hoc testing) – це вид тестування, який виконується без підготовки до тестування продукту, без визначення очікуваних результатів, проєктування тестових сценаріїв. Це неформальне, імпровізаційне тестування.

* 1. **Що таке дослідницьке тестування і чим воно відрізняється від Ad-hoc?**

*Дослідницьке тестування* **(exploratory testing)** – це одночасне вивчення програмного продукту, проєктування тестів і їх виконання. Це неформальний метод проєктування тестів, при якому тестувальник активно контролює проєктування тестів в той час, як ці тести виконуються, і використовує отриману під час тестування інформацію для проєктування нових тестів.

* 1. **Що таке кросс-браузерне тестування?**
  2. **Наведи 5 перевірок які ти б проводив/ла, використовуючи техніку вгадування помилок.**
  3. **Що таке SQL?**

SQL (Structured Query Language) - це мова програмування, яка використовується для взаємодії з реляційними базами даних. Вона дозволяє створювати, змінювати, видаляти та опитувати дані в базах даних.

* 1. **Які бувають бази даних?**

ієрархічна, мережна та реляційна.

* 1. **Що ви знаєте про NoSQL?**

Це сімейство нереляційних баз даних. В них розробники відійшли від використання традиційної табличної моделі подання інформації.

* 1. **Основні команди SQL?**
  2. **Як створити таблицю в SQL?**

За допомогою команди CREATE TABLE

* 1. **Як заповнити таблицю даними?**
  2. **Як змінити дані в таблиці?**

За допомогою команди UPDATE

* 1. **Як видалити дані з таблиці?**

За допомогою команди DELETE

* 1. **Як видалити таблицю?**

За допомогою команди DROP TABLE

* 1. **Дістань всю інформацію з таблиці Люди**

SELECT \* FROM people;

* 1. **Дістань тільки імена і прізвища  з таблиці Люди**

SELECT firstName, lastName FROM people;

* 1. **Дістати все з таблиці Люди, де імя - Анна**

SELECT \* FROM people WHERE firstName = “Анна”;

* 1. **Дістати все з таблиці Люди, де імя починається на А**

SELECT \* FROM people WHERE firstName LIKE “A%”;

* 1. **Дістати все з таблиці Люди, де імя містить букву А**

SELECT \* FROM people WHERE firstName LIKE "%A%";

* 1. **Дістати всіх тестувальників з таблиці Працівники, відсортованих за датою найму**

SELECT \* FROM workers

WHERE  position = “QA engineer”

ORDER BY hireDate DESC;

* 1. **Дістати 5 останніх найнятих тестувальників із таблиці Працівники**

SELECT \* FROM workers

WHERE  position = “QA engineer”

ORDER BY hireDate DESC

LIMIT 5;

* 1. **В чому різниця між where і having?**

Це одне й те саме, але having використовується з командою group by.

* 1. **Що ви знаєте про Join в базах даних?**

операція з'єднання таблиць в SQL, яка сполучає дві таблиці в реляційній базі даних, утворюючи нову тимчасову таблицю, яку інколи називають «з'єднаною таблицею»

* 1. **Як створювати тест кейси з БД?**
  2. **Особливості тестування мобільних додатків.**
  3. **Нативні, гібридні веб-додатки.**

Зазвичай мобільні додатки створюються для отримання переваги від конкретного мобільного пристрою. Нативні та гібридні додатки вважаються найбільш популярними. Нативні додатки створюються для конкретного пристрою в той час як гібридні будуються на нативних та веб додатках.

* 1. **Що таке емулятор та симулятор?**

Різниця між симулятором та емулятором полягає в тому, що симулятор працює повністю з віртуальними об'єктами, у той час як емулятор дає змогу працювати з окремими реальними елементами. Ключова перевага симуляторів полягає в тому, що вони, як правило, не дуже вимогливі до ресурсів.

* 1. **Що не можна протестувати на емуляторі, а лише на реальному пристрої?**
  2. **Які бувають переривання?**

Переривання - це сигнал, що повідомляє процесор про настання якої-небудь події, яка потребує невідкладної уваги. При цьому виконання поточної послідовності команд призупиняється і керування передається обробнику переривання, який реагує на подію та обслуговує її, після чого повертає управління в перерваний код.

* 1. **Що таке Dev tools і навіщо ним користуватись?**

Developer Tools – інструмент за замовчуванням у браузері Google Chrome та інших, який призначений для розробників та тестувальників з метою відладки коду. З його допомогою можна переглядати код сторінки, логи, завантаження сторінки, відображення в різних дозволах, помилки коду і багато іншого.

* 1. **В чому різниця між frontend та backend?**

Фронтенд-розробка – це як красива обкладинка книги або доглянутий фасад будинку. Вона займається всім тим, що ви бачите і з чим взаємодієте на веб-сайтах і веб-додатках.

Основна мета бекенда – забезпечити, щоб усе працювало. Тут немає красивих кнопок і стильних кольорів.

* 1. **Що таке JSON та XML?**

JSON – це текстовий формат обміну даними, що базується на мові JavaScript. XML – це мова розмітки, що розширюється.

* 1. **Що таке сніффер і навіщо він потрібний?**

Сніффер – це аналізатор трафіку. У свою чергу, трафік – це вся інформація, яка проходить через комп’ютерні мережі. Цей аналізатор дивиться, яка інформація передається. Для цього її необхідно перехопити. Прикладом сніфферу є Fiddler, Charles Proxy.

* 1. **Що таке кеш та кукі? Чим відрізняються?**

Файли cookie використовуються для зберігання інформації про різні аспекти роботи користувача, а кеш використовується для прискорення завантаження вебсторінок

* 1. **Що таке ідентифікація, аутентифікація і авторизація?**

ідентифікація - це називання особою себе системі;

аутентифікація - це встановлення відповідності особи, призначеному ним самим ідентифікатором;

авторизація - надання цій особі можливостей у відповідність до покладених йому правами

* 1. **Навіщо потрібно чистити кеш перед тестуванням веб-застосунків?**
  2. **Чим HTTP відрізняється від HTTPS?**

по HTTPs безпечно передавати особисті дані, а по HTTP - ні.

* 1. **Опишіть, що таке клієнт-серверна архітектура?**

Архітектура, при якій мережеве навантаження розподіляється між клієнтом та сервером.

* 1. **З чого складається HTTP-запит?**

Посилання, Тип запиту, тіло або параметри, заголовки

* 1. **З чого складається HTTP response?**

тіло повідомлення, що містить дані запиту, запитаний ресурс або опис проблеми, якщо запит не виконано + статус код

* 1. **Які ви знаєте методи HTTP?**
  2. **Чим GET відрізняється від POST?**
  3. **Які ви знаєте статус коди?**
  4. **Що таке REST API?**
  5. **Відмінності від SOAP?**
  6. **Що таке IP?**
  7. **Що таке домен?**
  8. **Що таке логи?**
  9. **Як тестувати АПІ? Які є тести для АПІ?**
  10. **Як створювати тест кейси для АПІ?**
  11. **Що таке моки, стаби, драйвери?**

Мок — це об'єкт-заглушка, який імітує поведінку реальних об'єктів у контрольованих умовах. Моки не просто підставляють значення, а й можуть перевіряти, чи були викликані певні методи з очікуваними параметрами. Це дозволяє контролювати поведінку тестованого коду та перевіряти, як він взаємодіє з іншими компонентами.

* Приклад: Якщо ви тестуєте функцію, яка зберігає дані в базі даних, ви можете використовувати мок, який імітує виклики до бази даних і перевіряє, чи викликана функція "insert()" з правильними аргументами.
* Застосування: Часто використовується у юніт-тестуванні для перевірки взаємодії між об'єктами.

Стаб — це простий об'єкт-заглушка, який підставляє заздалегідь визначені значення або відповіді на виклики методів. На відміну від моків, стаби не перевіряють взаємодію з тестованим кодом, а лише надають фіксовані дані, щоб ізолювати тестовану частину коду від реальних залежностей.

* Приклад: Якщо ви тестуєте функцію, яка отримує дані від зовнішнього сервісу, ви можете створити стаб, який повертає певний набір даних незалежно від того, що відбувається на цьому сервісі.
* Застосування: Стаби корисні, коли потрібно підставити передбачувані відповіді або дані, щоб зосередитися на тестуванні іншої логіки.

Драйвери в контексті тестування — це компоненти, які запускають або викликають інші компоненти чи модулі. Вони є протилежністю стабів і використовуються для тестування модулів, які знаходяться на нижчих рівнях (тобто не самодостатні і потребують ініціації або виклику іншими компонентами).

* Приклад: Якщо ви тестуєте модуль обробки даних, який зазвичай викликається користувацьким інтерфейсом, драйвер може забезпечити вхідні дані для цього модуля і ініціювати його виконання.
* Застосування: Використовуються у тестуванні модулів низького рівня, які залежать від викликів від інших частин системи.
  1. **Що ти знаєш про оцінювання задач\естімейт? В чому оцінюють задачі?**  
     Методики оцінювання:

1. Експертна оцінка (Expert Judgment):
   * Досвідчені тестувальники або керівники проектів надають свою оцінку задачі, базуючись на попередньому досвіді з аналогічними задачами.
   * Наприклад, якщо команда вже тестувала подібний функціонал, вони можуть спиратися на попередній досвід для оцінювання часу на нову задачу.
2. Аналогія (Analogous Estimation):
   * Оцінка базується на порівнянні з попередніми задачами. Якщо завдання схоже на те, що вже було виконано раніше, можна взяти аналогічний естімейт.
   * Наприклад, якщо попередній аналогічний тестування зайняв 3 дні, нову задачу можна оцінити в той самий час.
3. Метод трьох точок (Three-Point Estimation):
   * Передбачає обчислення середньої оцінки на основі трьох сценаріїв: оптимістичного (O), песимістичного (P) та реалістичного (R).
   * Формула: Естімейт = (O + 4R + P) / 6
   * Наприклад, якщо оптимістичний варіант оцінює задачу в 2 години, реалістичний — у 4 години, а песимістичний — у 8 годин, то естімейт буде дорівнювати (2 + 4×4 + 8) / 6 = 4,33 години.
4. Planning Poker:
   * Метод, використовуваний у команді для колективної оцінки задач. Кожен учасник вибирає картку зі своєю оцінкою, а потім проводиться обговорення, щоб дійти згоди.
   * Це допомагає уникнути упереджень та дозволяє врахувати думки всіх членів команди.
5. **Які метрики, пов'язані з тестуванням, ти знаєш?**

1. Test Coverage (Покриття тестуванням) - Вимірює відсоток покриття тестами коду або функціоналу системи.

2. Defect Density (Щільність дефектів) - Визначає кількість дефектів, виявлених на одиницю коду (наприклад, на 1000 рядків коду або на модуль).

3. Defect Leakage (Витік дефектів) - Вимірює відсоток дефектів, які були виявлені клієнтом або на пізніх етапах (у продакшн середовищі), порівняно з тими, що були знайдені під час тестування.

4. Test Case Execution Rate (Швидкість виконання тест-кейсів) - Вимірює кількість тест-кейсів, виконаних за певний проміжок часу.

5. Pass/Fail Rate (Співвідношення пройдених/непройдених тестів) - Відсоткове співвідношення між кількістю тест-кейсів, які пройшли успішно, та тих, що не пройшли.

6. Defect Removal Efficiency (Ефективність усунення дефектів) - Визначає ефективність тестування у виявленні дефектів до релізу продукту.

7. Test Case Effectiveness (Ефективність тест-кейсів) - Визначає, наскільки ефективними були тест-кейси у виявленні дефектів. Якщо з 100 виконаних тест-кейсів 20 виявили дефекти, ефективність складає 20%.

8. Mean Time to Detect (MTTD) та Mean Time to Repair (MTTR)

* MTTD: Середній час, який потрібен для виявлення дефекту після його появи.
* MTTR: Середній час, який потрібен для виправлення дефекту після його виявлення.

9. Automation Coverage (Покриття автоматизацією) - Відсоток тест-кейсів, що автоматизовані, від загальної кількості тест-кейсів. Якщо автоматизовано 200 тест-кейсів із 500, покриття автоматизацією складає 40%.

10. Requirements Coverage (Покриття вимог) - Вимірює відсоток покриття вимог тест-кейсами. Якщо 80 з 100 вимог мають тест-кейси, покриття вимог складає 80%.

1. **У вас відкривається веб-сайт. Білий екран. Більше нічого. Як тестуватимете?**
2. **Вам прийшло завдання без опису. Як ви себе поведете?**
3. **Як тестуватимете кнопку?**
4. **Як тестуватимете поле?**
5. **Протестуйте маску в інстаграм.  5 позитивних та 5 негативних перевірок.**
6. **Накидайте перевірок на робот-пилосос. Позитивних та негативних.**
   1. **Ви виявили помилку під час тестування мобільної програми та повідомили про неї.** **Розробник неспроможний її відтворити. Що ви можете пропустити в цьому випадку?**
   * Надати детальний звіт з прикріпленими відео чи скрінами.
   * Переконатися, що розробник використовує ті самі налаштування, середовище та версії програмного забезпечення, що й ви під час тестування.   
     Запропонувати мітинг, де розробник покаже екран і покроково відтворить баг.
   1. **Як QA-інженер ви читаєте вимоги для створення тестових випадків та виявили серед них деяку двозначність. Що ви можете зробити у такому разі?**
7. **В чому різниця між мікросервісом і монолітом?**
8. **В чому різниця  HTML, XML, XHTML?**
   * HTML (HyperText Markup Language) Мова розмітки гіпертексту, стандартна мова розмітки, яка використовується для створення веб-сторінок, використовується для форматування та відображення даних
   * XML （eXtensible Markup Language: Відноситься до мови розмітки, що розширюється, XML призначений для передачі та зберігання даних, а не їх відображення. Теги XML не визначені. Нам потрібно самим визначити теги: XML розроблений для самоописання, а XML – рекомендований стандарт W3C.
   * XML доповнює HTML, а XML не замінює HTML, це важливо розуміти.
   * XHTML (eXtensible HyperText Markup Language) Розширювана мова розмітки гіпертексту - це мова розмітки, яка поводиться аналогічно мові розмітки гіпертексту (HTML), але має більш строгий синтаксис.
   * HTML5 використовує бази даних SQL і кеш додатків для тимчасового зберігання даних, тоді як HTML для цього використовується тільки кеш браузера.
   1. **Що таке CSS?**

CSS розшифровується як Cascading Style Sheets, що в перекладі

означає «Каскадні Таблиці Стилів». Мова CSS – це мова розмітки, яка використовується для візуального оформлення веб-сайтів, вона розширює можливості HTML. CSS дозволяє швидко змінити візуальне оформлення сайту, не вдаючись до більш складних мов програмування. З його допомогою можна змінити кольори, шрифти, фон, взагалі займатися красою сайту.