1. **Відкрийте веб-сайт Вашої улюбленої соцмережі. Опишіть, які дії виконуються на кожному з рівнів тестування , якби у Вас була повна документація.**

На прикладі Telegram:

* Component Testing

1. Сторінка “Авторизація”, а саме розташування, відклик і функціонування “Поле вибору країни”, “Поле вибору номеру телефона”, check box “Keep me signed in”, кнопку “Log in by R code”, кнопку “Продовжити Українською”.
2. Наступне вікно авторизації розташування, відклик і функціонування “Поля введення коду” надісланого на номер телефону та кнопки редагування “Номеру телефону”.
3. Наступне вікно авторизації розташування, відклик і функціонування “Поля вводу пароля” та кнопки “Приховання/відображення” символів паролю.
4. Тестування “Початкової сторінки” розміщення, відклику і функціонування кнопок: “Меню”, “Логотипу”, “Пошуку”, “Людей” і “Груп”.
5. В кнопці “Меню” розміщення, відклик і функціонування кнопок: “Фотографія”, “Нікнейм”, “Нова група”, “Контакти”, “Виклики”, “Люди поблизу”, “Збережене”, “Налаштування”, “Запросити друзів”, “Можливості телеграм” та “Кнопка зміни теми на нічну”.
6. В кнопці “Пошук” розміщення, відклик, функцію “Пошук” та кнопки “Чати”, “Медіа”, “Завантаження”, “Посилання”, “Файли”, “Музика”, “Голосові”.
7. В копці “Нове повідомлення” розміщення, відклик, функцію кнопки “Пошук”, “Сортування за”, “Нова група”, “Новий секретний чат”, “Новий канал”, “Додати контакт”.
8. В контакті особи розміщення, відклик і функціонування кнопок “Стрілка повернення”, “Виклик”, “Навігаційне меню”, “Сповіщення”, “Відеовиклик”, “Пошук”, “Очистити історію”, “Змінити кольори”, “Видалити чат”.

Наскільки я зрозумів, то на цьому рівні ми будемо тестувати всі кнопки та функції окремо та ізольовано від інших модулів. Тобто їхню наявність, їхній дизайн та функціональність, якщо вона не залежить від інших модулів, або якщо для неї розробники зробити затичку. Тобто у нас на даному етапі є окремі частини коду, які вже можуть щось робити, але ще не можуть обмінюватися даними.

* Integration Testing

1. Перевірка можливості та коректності роботи функції пошуку “Людей”, “Груп”, “Музики”, “Відеофайлів”. Тобто надсилання запиту до сервера за допомогою “Пошуку”, обробки запиту за допомогою “Серверної логіки” (Що теоретично теж модуль, оскільки має певні алгоритми сортування та пошуку), надсилання відповіді, отримання і відображення відповіді.
2. Додавання “Контактів людей” та “Груп” подальше їх відображення на головній сторінці.
3. Надсилання певному “Контакту” фото, музики, посилання та подальше їхнє відображення в функціоналі “Пошуку”.
4. Можливість пересилання посилання, зображення, музики, відео в “Збереженні” та їхнє подальше зберігання та відображення там.
5. Тестування відповідності інтерфейсу та його функцій до заявлених вимог замовника.

На даному рівні тестування ми маємо тестувати роботу декількох окремо розроблених модулів, які б обмінювалися данними, приймали, пересилали або зберігали данні. Тобто на даному рівні тестування відбувається обов”язкова взаємодія двох і більше модулів/компонентів/юнітів.

* System Testing

1. Тестування програми на різних типах операційних систем, таких як: “iOS”, “macOS”, “Android”. Та в різних оточеннях, таких як: “Google”, “Safary”, “FireFox”, “Opera” та інші. Проведення тестування на різних типах пристроїв “Телефон”, “Планшет”, “Ноутбук”, “ПК” та інші.
2. Тестування різних User story таких направити повідомлення, видалити повідомлення, знайти користувача, додатись в групу, переслати повідомлення, записати голосове повідомлення, відправити картинку, медіа або музику, і тому подібне.
3. Тестування роботи системи при певному робочому навантаженні, з'ясування ліміту навантаження та межі навантаження за якою настає падіння програми, перевірка швидкодії програми при робочому та стрес навантаженні.
4. Перевірка рівня захисту та безпеки програми, встановлення слабких місць, створення документації про певні проблеми, особливості і правила безпечного користування..
5. Тестування дизайну програми UI/UX та її функціональності, для того щоб встановити, що заявлена функціональність в документації та функціональність розроблена девелоперами працює, відповідає одне одній та відповідає вимогам замовника.

Передфінальний етап проведення тестування, коли у нас уже зібрані всі розроблені елементи в єдину робочу програму. Ми можемо більш змістовно співставити задум і вимоги замовника до продукта, який розроблений. Тестувати, як користувач буде використовувати програму. Які навантаження може вона витримувати. Яку швидкодію вона має на різному обладнанні. Наскільки є безпечним продукт для користувача та наскільки він відповідає вимогам чинного законодавства.

* Acceptance Testing

1. Тестування тестувальниками продукту залученими зі сторони замовника для встановлення незалежної оцінки якості виконання продукту та відповідності замовленим вимогам.
2. Перевірка на відповідність розробленої системи до вимог чинного законодавства на тереторії якої буде відбуватись розгортання даної програми.
3. Альфа та бета-тестування з залученням певного кола користувачів, які зацікавлені в продукті, готові ним користуватися, надавати звіти про помилки, а також свої відгуки про зручність інтерфейсу та користування програмою взагалі.
4. Представлення продукту замовнику, з'ясування чи відповідає розроблений продукт їхнім вимогам і чи на їхню думку продукт відповідає очікуванням замовників.
5. Готовий і перевірений продукт має бути обов'язково перевірений в безпековому плані, як для користувачів так і обладнання на якому такий продукт має знаходитися. Безпек та витік пам'яті.

Останній рівень тестування, на якому продукт готовий, представлений для замовника, підданий аналізу з сторони, на даний для обмеженої кількості користувачів в користування, та перевірений на безпечність користування та відповідності вимогам закону.

**2. У вас є програма-калькулятор (наприклад, стандартний калькулятор Windows) та 5 хвилин на проведення приймального тестування. Що б Ви протестували і чому?**

1. Відкриття програми “Калькулятор” та її коректне відображення в системі.

* Нам необхідно розуміти, що програма взагалі відкривається і не видає помилку при відкритті.

1. Перевірка розташування, відклику, дизайну всіх кнопок “Цифр” і “Знаків”.

* Потрібно розуміти, що всі кнопки “Цифр” від 0 до 9 присутні, а також присутні основні дії, як “Додавання”, “Віднімання”, “Множення”, “Ділення”.

1. Перевірка дій “Додавання”, “Віднімання”, “Множення”, “Ділення”, “Приведення до відсотків”, “Взведення в квадрат”, “Очищення”, “Розділення комою”, “Ділення цілого на х”, “Взведення до степені числа в корені”.

* Основною функціональністю калькулятора є проведення математичних дій, тому потрібно провести декілька тестів, щоб дійсно зрозуміти, що при відніманні числа віднімаються, при додаванні додаються і т.д.

1. Перевірка розгортання та згортання калькулятора “До мінімального” та “До стандартного формату”.

* Оскільки можна припустити, що основній масі користувачів буде достатньо мінімального функціоналу, то інші особливі функції можна тестувати трохи далі і блище, до закінчення 5 хвилинного часу.

1. Перевірка розташування, відклику та функціональності кнопки “Історії розрахунків”.

* Історія розрахунків, може бути цікавою для користувача, а тому слід протестувати це оскільки є висока вірогідність, що користувач хоче скористатися цим функціоналом.

1. Перевірка розташування, відклику та функціональності кнопки “Навігаційне меню”.

* В навігаційному меню знаходяться інші типи калькуляторів, що є більш специфічним функціоналом і менша кількість користувачів буде зацікавлена в використанні цієї функції, а тому її можна протестувати пізніше, але все ж ці функції є дуже корисними і декому вони точно знадобляться.

1. В “Навігаційному меню” перевірка розташування, відклику та функціональності кнопки “Стандартний”, “Інженерний”, “Побудова графіків”, “Програмний”, “Обчислення дати”, “Валюта”, Об'єм”, “Параметри”.

* Швидко тестуємо відкриття різних типів калькуляторів. Якщо відкриваються - добре. Якщо ще є час і можна пройтись по ним, то взагалі чудово. Оскільки провести вичерпне тестування неможливо. А основні дії показали вірні результати, то є висока вірогідність того, що основні алгоритми в інших типах калькуляторів працюють коректно.

**3. Вам необхідно провести системне тестування банківської програми (наприклад, Приват24/Mono) але у Вас відсутня будь-яка документація. Опишіть - 5 головних, на Ваш погляд, Use кейсів для тестування такої програми**

На прикладі RaiffeisenBank

1. Авторизація

| **Виконавець** | Користувач або QA. |
| --- | --- |
| **Ціль** | Авторизуватися в системі. |
| **Передумова** | Програма відкрита в браузері або на мобільному пристрої. |
| **Успішний сценарій:**   1. QA або користувач вводить логін та пароль. 2. Система здійснює вхід в акаунт користувача. 3. Вихід з програми. | |
| **Результат** | QA або користувач здійснив вхід в свій акаунт. |

2. Перевірка рахунку користувача

| **Виконавець** | QA, Користувач. |
| --- | --- |
| **Ціль** | Перевірити баланс рахунку користувача. |
| **Передумова** | 1. Програма відкрита в браузері або на мобільному пристрої. 2. Користувач авторизувався і отримав доступ до свого облікового запису. |
| **Успішний сценарій:**   1. Користувач натискає на кнопку рахунки. 2. Здійснюється перехід на сторінку / вкладку рахунки. 3. Можна спостерігати картки користувача та баланси на них. 4. Вихід з програми. | |
| **Результат** | QA або користувач може спостерігати за вкладкою рахунки та бачити картки і баланси на них. |

3. Здійснення переказу з картки або рахунку на номер карти.

| **Виконавець** | QA, Користувач. |
| --- | --- |
| **Ціль** | Переказ коштів з картки або рахунку на рахунок за номером карти. |
| **Передумова** | 1. Програма відкрита в браузері або на мобільному пристрої. 2. Користувач авторизувався і отримав доступ до свого облікового запису. 3. Користувач впевнився, що на рахунку присутні кошти. |
| **Успішний сценарій:**   1. Користувач натискає кнопку “Перекази”. 2. Користувача переносить на вкладку перекази і він може спостерігати реквізити “Відправника”, “Отримувача”, “Суму” переказу та кнопку “Продовжити”. 3. Користувач заповнює поля “Відправника”, “Отримувача”, “Суму” переказу та натискає кнопку “Продовжити”. 4. Програма здійснює перехід до вікна підтвердження, яке містить поле в яке потрібно ввести надісланий код. Або система автоматично вводить код при отриманні. 5. Перехід до вікна успішного виконання операції. 6. Вихід з програми. | |
| **Результат** | QA або користувач здійснив транзакцію з рахунку на рахунок. |

4. Перевірка здійснених операцій в банківській системі

| **Виконавець** | QA, Користувач. |
| --- | --- |
| **Ціль** | Перевірка здійснених операцій в банківській системі |
| **Передумова** | 1. Програма відкрита в браузері або на мобільному пристрої. 2. Користувач авторизувався і отримав доступ до свого облікового запису. 3. З акаунту користувача здіснено хоча б одну транзакцію. |
| **Успішний сценарій:**   1. Користувач натискає кнопку “Операції в Raiffeisen Online”. 2. Випливає вікно “Операції” і можна спостерігати “Фільтр операцій”, “Дату операції”, “Час операції”, “Рахунок на який здійснено транзакцію”, “Суму транзакції”. 3. Користувач натискає на блок певної транзакції інформація про яку йому потрібна. 4. Випливає вікно з інформацією про транзакцію. КОристувач спостерігає “Вид переказу та номер картки або реквізит”, “Дату та час операції”, “Суму”, “Комісію”, активні кнопки “Повторити”, “Поділитися pdf’, “Надіслати на електронну пошту”, “Створити шаблон”, та строки “Номер транзакції”, “Статус”, “Дата та час виконання”, “Платник” (Номер рахунку відправника), “Отримувач” (Номер рахунку отримувача). | |
| **Результат** | QA або користувач перевірив здійснення транзакції з рахунку на рахунок. |

1. Здійснення комунального платежу.

| **Виконавець** | QA, Користувач. |
| --- | --- |
| **Ціль** | Проведення оплати по комунальному платежу за допомогою банкінгу. |
| **Передумова** | 1. Програма відкрита в браузері або на мобільному пристрої. 2. Користувач авторизувався і отримав доступ до свого облікового запису. 3. Користувач впевнився, що на рахунку присутні кошти. |
| **Успішний сценарій:**   1. **Користувач натиснув кнопку “Платежі”. Відкрилося вікно платежі де користувач може спостерігати кнопку “Комунальні платежі”.** 2. **Користувач натискає кнопку “Комунальні платежі”. Відкрилося вікно з кнопками надавачів комунальних послуг. Обираємо (до прикладу) КИЇВТЕПЛОЕНЕРГО. Нас переносить на вікно вибору картки або рахунка.** 3. **Обираємо картку з позитивним балансом. Натискаємо кнопку продовжити. Відкривається вікно рахунку обслуговування. Вводимо рахунок. Натискаємо кнопку продовжити.** 4. **Обираємо суму до сплати. Натискаємо протовжити. Відкривається вікно з строкою вводу коду підтвердження. Вводимо надісланий код підтвердження або програма автоматично заповнює дане поле при надходженні коду. Натискаємо кнопку “Продовжити”. Вспливає вікно “Готово”.** 5. **Виходимо до головного меню. Здійснюємо перевірку транзакції.** 6. **Виходимо з програми.** | |
| **Результат** | QA або користувач здійснив перерахування комунального платежу та перевірив його здійснення. |

**4. Опишіть усі підходи інтеграційного тестування у разі тестування соціальної мережі Facebook.**

1. **Знизу вгору (Bottom-up)**

Спочатку ми тестуємо логіку роботи програми, яку не бачить користувач. На мою думку високорівневі і низькорівневі модулі, це відсилка до високорівневих та низькорівневих мов програмування. Якщо це так, то їх можна порівняти з FrontEnd та BackEnd програмуванням.   
Тобто спочатку ми тестуємо речі, які пов'язані з роботою логіки програми. Наприклад як “Пошук”, де з бази даних, за запитом користувача, яка знаходиться на сервері ми отримуємо, відповідь з інформацією, яка відповідає цьому запиту.

Алгоритми підбору реклами, які невідомі користувачу, але логіка, яких впливає на вибір реклами для користувача, скоріш за все має тестуватися через whitebox testing.

Підбір друзів та людей, яких ви можете знати, за різними характеристиками зазначеними користувачами в їхніх профілях.

Збереження інформації про користувача на сервері, та коректність відображення цих даних при наступному користуванні програмою.

Враховуючи, що алгоритми роботи низькорівневих модулів та їхньої взаємодії з іншими низькорівневими модулями можуть бути протестовані на рівні коду, то спочатку слід здійснити таке тестування.

Після проведення таких тестів, ми можемо почати підключати низькорівневі модулі, до модулів високорівневих (Скоріше за все ці модулі вже доступні користувачу). Тобто приєднувати логіку роботи програми до інтерфейсу програми.

Тобто ми можемо використати графу “Пошук” для введення туди імен, назв груп, символів для тестування надсилання запиту з більш високорівневого модулю та надання відповіді з більш низькорівневого модулю, серверу.

Додавання друзів як приклад роботи високорівневих модулів та низькорівневих модулів. Ми натискаємо кнопку “Друзі” (Високорівневий модуль доступний користувачу), бачимо там кнопку “Рекомендації” алгоритм пошуку людей, які можуть бути цікавими для користувача. Робота модуля була здійснена на рівні сервера, відповідь відображена на більш високому модулі, інтерфейсі користувача. Кнопка “Додати друга” посилає запит на сервер, він його обробляє, і відображає у іншого користувача запит на додавання до друзів.  
 Тобто при тестуванні Знизу вгору (Bottom-up) ми спочатку протестуємо всі модулі робота яких залежить від серверу, з подальшим підключенням частин інтерфейсу та можливістю взаємодії цих частин між собою.

1. **Зверху донизу (Top-down integration)**

При даному виді інтеграційного тестування спочатку ми протестуємо роботу всіх кнопок, їх місця розташування, їх дизайну, та враховуючи, що у нас є затички їхній відклик.

Тобто спочатку будеть протестовані такі кнопки, як “Головна”, “Люди/друзі”, “Відео”, “Ігри”, “Повідомлення”, “Меню”, “Історії”, “Групи”, Прямі ефіри” та вміст кожної з кнопок.

На початковому етапі інтеграційного тестування нам потрібно встановити, чи відповідає дизайн програми заявленій функціональності та дизайну, що описується в документації. Якщо є певні невідповідності, то складаються баг репорти.

Наступним етапом буде підключення високорівневих модулів та низькорівневих модулів прямої взаємодії. Тобто коли через “Пошук” на боці користувача ми можемо отримати інформацію яку шукаємо. Додавання друзів з підключенням низькорівневого модуля буде направляти запит на сервер та відображати у особи, яку запрошують запит на долучення до друзів.

В даному виді інтеграційного тестування спочатку буде досліджено все що бачить користувач, а тільки потім буде підключена логіка роботи програми. Оскільки модульність передбачає не тільки просте приєднання модуля до модуля, а й більш складну сітку взаємодії, то до високорівневого і низькорівневого модуля простої взаємодії можуть підключатися і інші модулі, які будуть впливати на роботу модулів, які вже були підключені між собою.

1. **Великий вибух ("Big Bang" Integration)**

При даному типі інтеграційного тестування продукт і всі його модулі вже приєднані один до одного і здійснюють покладені на них функції.   
Тобто модуль “Пошук” буде надсилати запит на сервер, сервер обробить запит та надасть відповідь з найбільш підходящими варіантами по критерію пошуку, а користувач зможе побачити дані відповіді в своєму інтерфейсі.

В при гортанні стрічки модуль підбору інформації автоматично буде надсилати запити до серверу та показувати фото, відео, новини інших користувачів на яких підписаний наший користувач. Оскільки модуль стрічки автоматично подасть запит на модуль який відповідає за підбір інформації (сервер) і цей модуль надасть відповідь з підходящою для користувача інформацією (Фото викладені друзями, новини, які цікавлять користувача, таргетована реклама тощо). При даному типі тестування не потрібно тестувати високорівневі та низькорівневі модулі окремо один від одного. Що значно зменшує час на розробку тест кейсів та саме тестування, але збільшує складність виявлення місцезнаходження багу, оскільки більш складна модульна система робить неочевидни місце його знаходження. Тобто теоретично якщо при модульному тестуванні ми виявили баги та виравили їх і модулі працювали коректно, то відсутні гарантії того, що при поєднанні декількох модулів не зявляться нові баги, які б можна було більш просто виявити при підключенні модуля до модуля і перевірки їхньої роботи.