Перший рівень — відпрацюй навички на базовому рівні.

1. Склади порівняльну таблицю найбільш поширених методологій:

Методологія	Сильні сторони	Слабкі сторони	Галузь застосування
Waterfall	1) **Більш зрозуміліша для замовника система**. Замовник приходить до компанії підрядка (для прикладу до аутсорс компанії), проговорює з ними нюанси продукту який буде вироблятись і за який певний домовлений час отримує готовий результат по суті не вникаючи в процес розробки. 2) **Плинність кадрів на проекті ніяк не впливає на довготривалість проекту.** Розробники/тестувальники/дизайнери можуть проходити ротацію наа проекті, але за рахунок того, що з замовником спершу були складені вимоги і прописана вся документація - така ротація майже не вплине на тривалість проекту 3) **Прозорість** Завдяки тому, що вся документація вже готова на початку і вона не передбачає змін (зазвичай), а також завдяки поетапності в цій моделі всі знають хто	1) **Дороговизна і бюрократична волокита при змінах на проекті** Якщо врахувати, що вся документація яка була чітко прописана на самому початку не підлягає зміні, то ця модель не дуже підходить для замовників які ведуть бізнес, що передбачає підлаштування під час і умови, бо якщо цей час і умови диктують правила ведення бізнесу, то замовнику дуже довго обійдеться постійне внесення правок в цей проект. 2) **Замовник бачить результат тільки в кінці проекту після здачі.** До моменту здачі проекту замовник абсолютно не залучений до розробки і отримує лише інформацію по типу “Проект готовий на 20%, ми зараз на етапі кодингу” 3) **Тестування проходить в кінці проекту.** Коли в кінці починають тестити проект - можуть знайти дуже багато багів, що на пряму впливатиме на строк	1) **Проекти пов’язані з медичною і фінансовою сферою.** На цих проектах зазвичай для більшої безпеки в майбутньому потрібне тестування коли всі компоненти ПЗ вже готові. Саме методологія Waterfall дозволяє таким чином розробити проект. 2) **Невеличкі проекти.** Іноді проекти бувають не супер масштабні і ця методика також ідеально підійде для них. Замовники в таких випадках одразу бачуть чітку картину свого проекту і цю картину буде не важко виложити в документації. Також ці проекти не будуть тривати довго, тому немає сенсу розбивати все на спринти як в Скрамі

	чим і коли буде займатись.	здачі проекту, тобто якби почали тестити ще на етапі кодингу - це би зекономило дуже багато часу.	
Scrum	1) **Залученість клієнта до всіх етапів розробки** Клієнт залучений в всіх процесах розробки і у випадку якщо йому щось не підходить - можна швидко внести правки до проекту 2) **Зміни в проекті менш дороговартісні.** Якщо замовник вирішив переробити проект, то ввін не буде платити якусь фіксовану суму яка передбачена за зміну в документації як в випадку з Waterfall, по суті він орендує тімку і поки вони перерообляють все або вносять зміни клієнт платить лише за їх погодинний рейт. 3) **Тестування проходить віід самого початку проекту** Це дає можливість швидше виявляти помилки і їх виправлення буде для клієнта менш дороговартісним і швидшим в реалізації 4) **Гнучкість методології** В будь який момент можна внести	1) **Може бути misscommunication з замовником**, бо якщо в випадку з waterfall клієнт приходить з чітким запитом, то в випадку scrum клієнт може сам не знати що він хоче 2) **Постійна комунікація всередині команди (суб'єктивна думка)** Якщо працівник по своїй натурі інтроверт, то щоденні стендапи і часта комунікая можуть суттєво вплинути на його вигорання під час роботи.	1) **Scrum методологія може бути корисна** для проектів де немає чіткої картини як має виглядати фінальна версія продукту, не зрозуміло що потрібно користувачам і що вони будуть юзати в продукті, а також немає чіткої картини з чим і коли вийде щось у найближчих конкурентів (можливо буде можливість щось підглянути у них після релізу і швиденько допилити в найбличому спринті) 2) **Гарний варіант для проектів які запускаються** Чіткість і прозорість поставлення задач дасть можливість якнайкраще комунікувати з клієнтом і закрити проект в найкоротші терміни

	корективи. Клієнт бачить результат пророблені роботи і може контролювати дедлайни		
Kanban	1) **Відсутність спринтів.** Є можливість додавати нові задачі в будь який час, основна ціль - закрити задачу, а не спринт 2) **Краще підходить для підтримки вже існуючого проекту** Коли проект вже виконаний і немає сенсу все розділяти на спринти - найкращий варіант це Kanban, тут є можливість додавати задачі по мірі їх поступлення, оновлення ПЗ або виявлення дефектів 3) **Менше комунікації (суб'єктивна думка)** Ця методика краще підійде для працівників інтровертів які не хочуть спілкуватись щодня. Але також це може бути мінусом, бо немає такої чіткої картини по етапу розробки як в скрамі	1) **Відсутність чіткої картини для працівників.** Якщо у випадку зі Scrum кожен знає чим буде займатись наступного спринта і який він отримає результат, то в Kanban потрібно просто чекати коли скинуть якусь таску, це може повпливати на вигорання 2) **Відсутність чітних строків** Якщо це проект по сапорту, то він може тривати безкінечно, аж поки він не приноситиме грошей кастомеру, це також може повпливати на вигорання працівників, не всі люблять працювати на підтримці чогось існуючого	1) **По суті підходить для будь якого готового проекту який потребує підтримки**
Scrumban	1) **Може взяти всі найкращі сторони з кожної з методологій (Scrum і Kanban)** Це дозволяє робробити методологію повністю під потреби команди і проекту	1) **Відносно нова методологія і не настільки вивчена як попередні** Читав статтю, що ця методолгія ще не набула піку популярності і відповідно вона ще не докінця вивчена на	**1) Гарно підходить до великих і довготривалих проектів** Методологія дуже гнучка і дає можливість корегувати тривалість кожної ітерації окремо в

	2) **Інтуїтивно зрозумілий** Не потребує скрам майстра чи продукт овнера, макимально гнучка методологія	практиці. 2) **Відсутність безпосереднього контролю за працівниками.** Оскільки кожен сам обирає ті ззадачі над якими працюватиме - важко відстежити зусилля кожного члена кооманди до розробки продукту	залежності від скоупу задач які висуває замовник на той чи інший період часу

2 рівень. Напиши розгорнуті відповіді (0,5 - 1 сторінки тексту) на такі два питання:

- На твою думку, чому з’явився Agile-маніфест?
- Які проблеми він мав вирішити і чи це вдалося?

На мою думку Agile маніфест з’явився через те, що нові компанії вимагали нових рішень, власники бізнесу хотіли бути більш залучені до процесу розробки свого продукту, а не просто чекати готового результату як було в випадку з методологією Waterfall. Якщо до цього маніфесту бізнес і IT були досить розділені, то після нього з’явились більш гнучкі методології які дозволяли бізнесу напряму впливати на результат роботи і вносити корективи. Сам принцип роботи змінився кардинально і тепер діалог між бізнесом і IT проходить майже на щоденній основі, це дозволяє кастомеру краще і глибше донести цінності своєї компанії і очікування по продукту до представників IT компанії які цей продукт по суті і розробляють, а IT в свою чергу вкладається в розуміння бізнес частини і це по суті дозволяє їм говорити на одній мові.

Центральною особою в Agile методологіях став не замовник, а юзер, тобто все, що робиться - робиться для кінцевого користувача, важливим стало те, що потрібно зрозуміти, що саме потрібно користувачу, в якому об’ємі і чому це треба робити, а також що з цього всього потрібно робити найперше. В Agile виділяється дуже багато часу і дуже багато зусиль саме на хорошу технічну якість. Тобто не робиться завідома не якісний продукт, який потім 100% потрібно буде переробляти. Це можливо завдяки тому, що методології гнучкі і є можливість паралельно з розробкою нових фіч - фіксити старіші. Клієнт часто отримує більш готовий і свіжіший продукт, але звичайно все в залежності від функціоналу який закладався в спринт чи інший відрізок часу. Саме за рахунок того, що клієнт часто бачить результат і робиться величезний акцент на якість, згрубша “ми не хочемо показувати те, що нам стидно показати” :)

Я вважаю, що цей маніфест мав вирішити проблему комунікації між бізнесом і IT і успішно з цим справляється. Дякуючи новим методологіям в кастомерів розв’язались руки в плані впливу на розробку їхнього ПЗ. З’явлась можливість вносити корективи в залежності від правил які диктує їх бізнес, а також

слідкувати за нововведенням які вносять їх основні конкуренти і почерпнувши з них все найкраще - додавати до свого ПЗ в найкоротші строки. Також Agile дав можливість нам як користувачам безпосередньо впливати на кінцевий продукт, тобто замовник бачить відгуки і спираючись на них також можна розробити більш якісне і usability friendly програмне забезпечення.

Третій рівень. Ти – засновник/ця стартапу і плануєш випустити на ринок мобільний застосунок для обміну світлинами котиків.
Яку методологію ти обереш для процесу розробки і чому? Відповідь текстово обґрунтуй.

Для стартапу я би обрав методологію Waterfall. Зазвичай стартапи дуже обмежені в фінансуванні на перших етапах і відповідно немає можливості наймати людей на якісь менеджерські позиції як цього зазвичай вимагають гнучкі методології, тому скрам і канбан не дуже підійдуть. Також waterfall дає можливіть поетапної розробки ПЗ, тобто можна наймати людей по мірі потреби - на кожен етап відповідний спеціаліст, це дозволить на перших парах зекономити велику кількість грошей. Так як засновники стартапів максимально вмотивовані і заряджені люди, ще частіше це вихідці з IT компаній - зазвичай вони впевнені в своїх вимогах до кінцевого продукту і це зменшить появу коректівів які потрібно буде вносити на завершальних етапах. У випадку якщо стартап вистрілить - я би сконцентувався на більш гнучких методологіях. Якщо з’явились гроші і потрібно кардинально змінити продукт - це Scrum, якщо просто потрібна підтримка вже існуючого продукту - Kanban