Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інформаційні системи

Викладач: к.т.н., доц. Саяпіна Інна Олександрівна

План заняття

- 1. Знайомство
- 2.Інформація щодо організації освітнього процесу
- 3. Корисні ресурси
- 4. Навіщо вивчати цей курс?
- 5. Життєвий цикл розробки програмного забезпечення
- 6. Архітектура програмного забезпечення інформаційної системи.

Знайомство

- Викладач: к.т.н., доцент Саяпіна Інна Олександрівна
- Telegram: @Inna_Sai

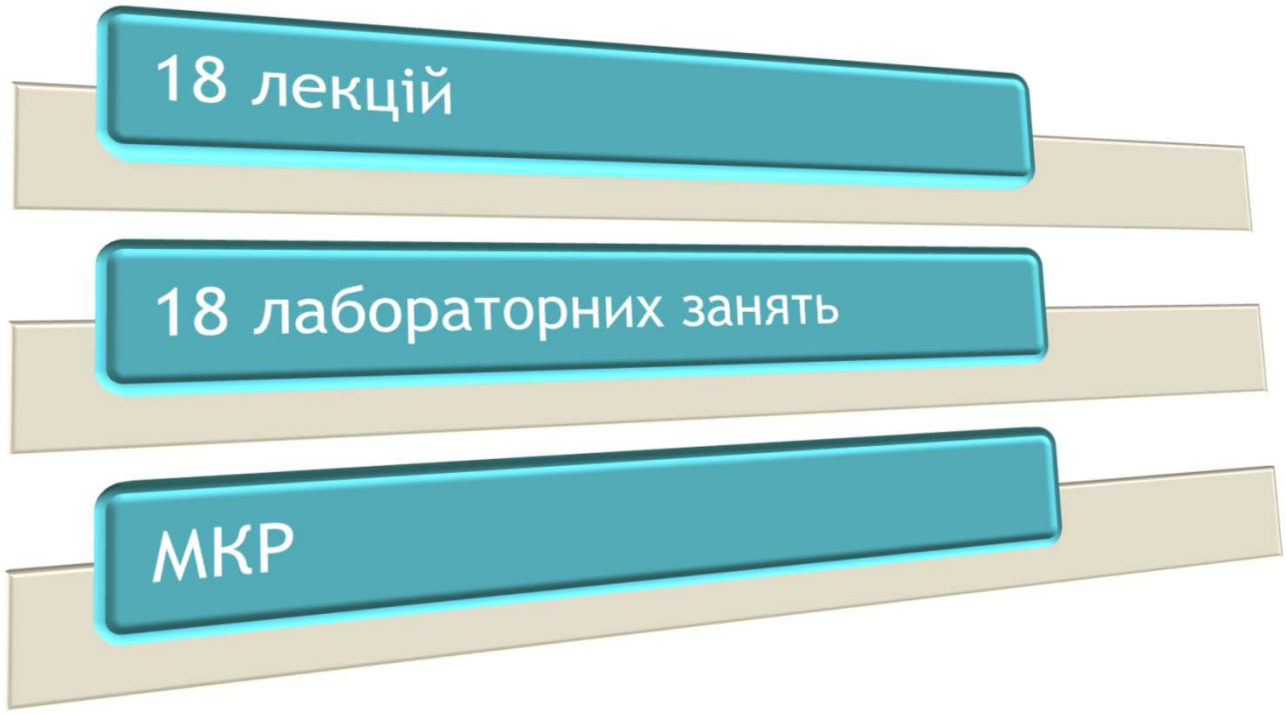
- Дисципліни, що викладаю:
  - Бази даних,
  - Компоненти програмної інженерії,
  - Інформаційні системи.
- Посилання на опитування

Інформація щодо організації освітнього процесу

- Для комунікації та швидкого вирішення питань - чат у Slack

- Матеріали курсу також викладаються у Google Classroom:
  - *https://classroom.google.com/c/NjIwODA4MjYxOTQ5?cjc=qffrb3i*

Склад дисципліни «Інформаційні системи»



Система оцінювання

- 6 Лаб/р по 10 балів кожна
- МКР 40 балів
- + бонусні бали за активність на заняттях

Система оцінювання лабораторних робіт

*Максимальна кількість балів за кожну лабораторну роботу становить **10 балів**. Бали нараховуються за:*

- *якість виконання лабораторної роботи (звіту): **0-4** бали ;*
- *захист лабораторної роботи у вигляді презентації отриманих результатів та відповіді на запитання: **0-4** бали;*
- *за своєчасне представлення до захисту роботи нараховується **0-2** бали: по 1 балу за своєчасно зданий звіт та своєчасно виконаний захист роботи.*

- *Критерії оцінювання якості виконання лабораторної роботи (звіту):*
  - ***4 бали** – робота виконана якісно, в повному обсязі;*
  - ***2-3 бали** – робота виконана не в повному обсязі, або містить незначні помилки;*
  - ***1 бал** – робота містить суттєві прогалини та/або значні помилки;*
- *Критерії оцінювання захисту лабораторної роботи:*
  - ***3-4 бали** – презентація та доповідь повністю розкриває контекст завдання, якісно виконані, відповіді повні, усі висновки добре аргументовані;*
  - ***1-2 бали** – презентація та доповідь мають прогалини, у відповідях є помилки;*
  - ***0 балів** – презентація отриманих результатів відсутня.*

Календарний контроль

- *На першій атестації (8-й тиждень) студент отримує «зараховано», якщо його поточний рейтинг складає не менше **15 балів** (50 % від максимальної кількості балів, яку може отримати студент до першої атестації).*
- *На другій атестації (14-й тиждень) студент отримує «зараховано», якщо його поточний рейтинг складає не менше **25 балів** (50 % від максимальної кількості балів, яку може отримати студент до другої атестації).*

Політика щодо академічної доброчесності.

- Згідно з Кодексом честі студента КПІ, при виконанні лабораторних чи контрольних робіт **забороняється** користуватися чужими виконаними лабораторними чи контрольними роботами та/або їх фрагментами.
- У випадку виявлення плагіату лабораторна робота може бути оцінена від 0 до 1/3 сумарної кількості запланованих за неї балів. Під час виконання контрольних робіт забороняється користуватись будь-якими сторонніми джерелами, у тому числі чатом GPT.

Корисні ресурси

coursera

udemy

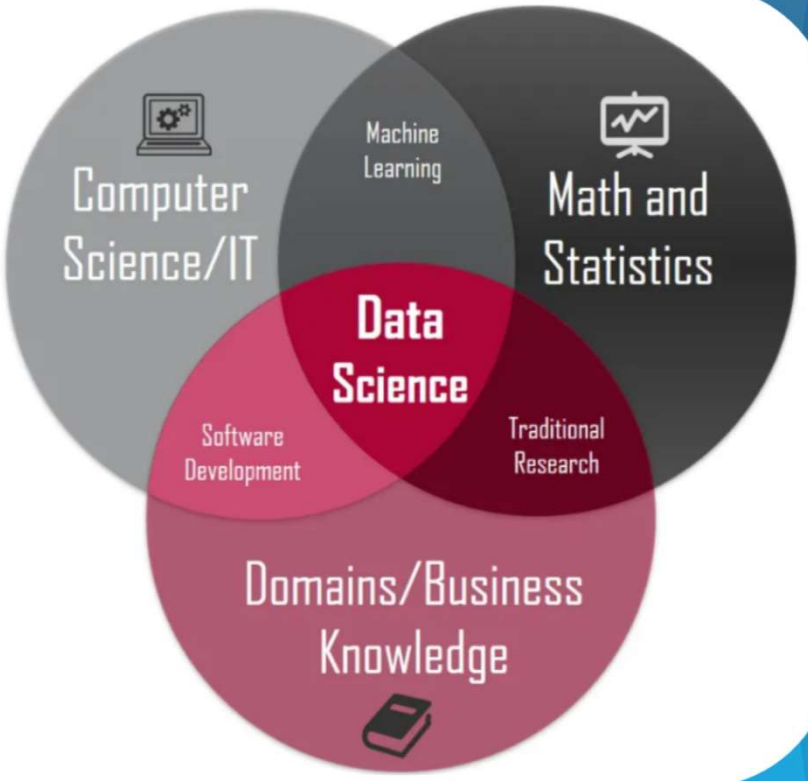
- Треба мати email домену kpi
  (щоб отримати заповніть гугл-форму
  https://docs.google.com/forms/d/e/1FAIpQLSfmzGxLsOKUb5KPurQlfeUasyVgYVarUdLGhFzpEex4R4Z0JA/viewform?embedded=true)
- За посиланням https://mooc4ua.online/ обираєте НТУУ КПІ та реєструєтесь на університетську пошту XXXX@LLL.kpi.ua

Література у бібліотеці КПІ

1. Р.С. Мартін. Чиста архітектура. Фабула, 2019. 368 с.
2. Фрімен Е. , Робсон Е., Сьерра К. , Бейтс Б. Head First Патерни проєктування. Фабула, 2020, 672 с.
3. Кучеров Д.П. Інженерія програмного забезпечення: навчальний посібник. Київ: НАУ, 2017. 386 с.
4. Бородкіна І.Л. Бородкін Г.О. Інженерія програмного забезпечення: навчальний посібник для студентів вищих навчальних закладів. Київ : Видавництво "Центр учбової літератури". 2020. 2004 с.
5. Лавріщева К. М. Програмна інженерія: підручник. К, 2008. (http://csc.knu.ua/uk/library/books/lavrishcheva-6.pdf)
6. Бабенко Л.П., Лавріщева К.М.. Основи програмної інженерії: Навчальний посібник для студ. вищих навч. закл. К.: Знання, 2001.

Навіщо Data Scientist'у вчити Software Development???

- Ефективна обробка даних
- Інтеграція з існуючими системами
- Полегшення комунікації з колегами
- Надійність та масштабованість системи обробки даних
- Full-Stack Data Science



У цьому курсі розглянемо:

- *Процес розробки інформаційної системи*
  - *Життєвий цикл розробки ПЗ інформаційних систем*
  - *Функціональні та нефункціональні вимоги*
  - *Архітектурно значущі вимоги та атрибути якості*
- Моделювання архітектури інформаційної системи та розробка документації
  - Засоби та методи моделювання архітектури інформаційних систем
  - Нотації та засоби документування архітектури інформаційних систем
- Архітектурні стилі та шаблони проєктування інформаційних систем
  - Монолітна архітектура
  - Багатошарова архітектура
  - Керована подіями архітектура
  - Сервіс-орієнтована архітектура
  - REST-архітектура
  - Мікросервісна архітектура

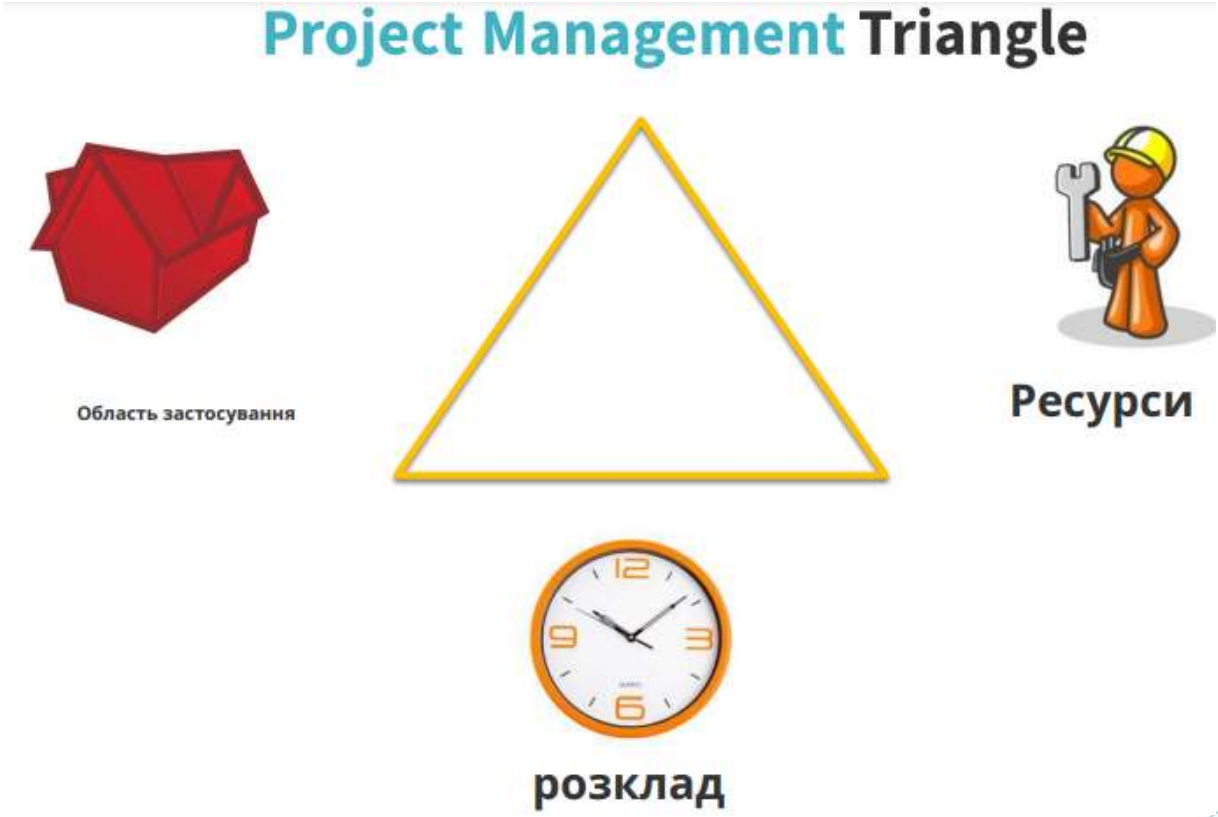
Software Development Life Cicle

Життєвий цикл програмного забезпечення – це сукупність окремих етапів робіт, що проводяться у заданому порядку протягом періоду часу, який починається з вирішення питання про розроблення ПЗ і закінчується припиненням використання програмного забезпечення.

Software Development Life Cicle

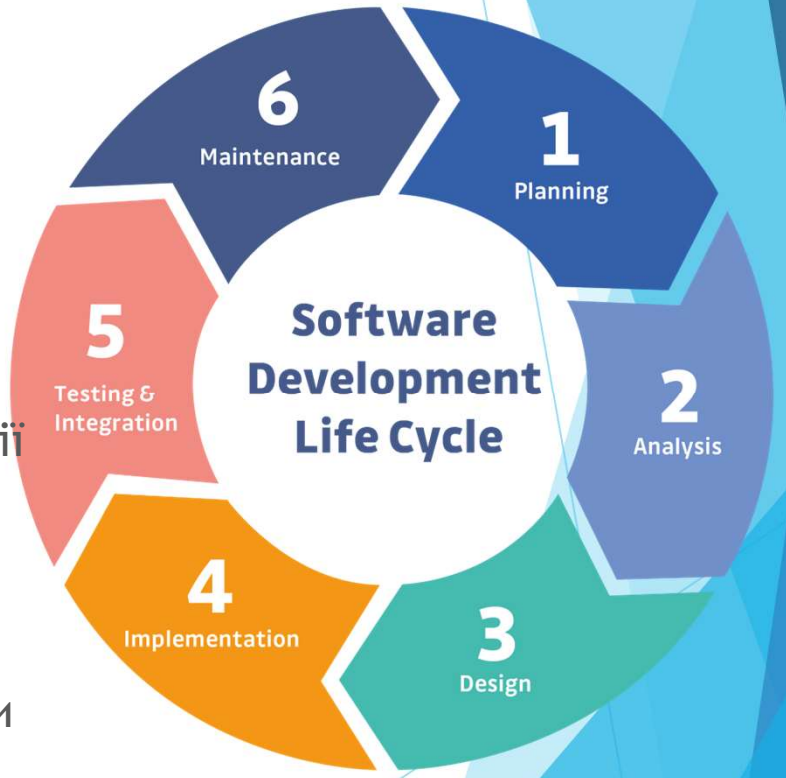
- 1. **Етап планування**:
- точно вирішується що саме потрібно зробити, розробити;
- визначаються проблеми, цілі і ресурси (такі, як персонал і витрати);
- вивчаються можливості альтернативних рішень шляхом зустрічей з клієнтами, постачальниками, консультантами та співробітниками;
- вивчається, як зробити продукт краще, ніж у конкурентів.
- Після аналізу цих даних є три варіанти: розробити нову систему, покращити існуючу або залишити систему як є.
- На цьому етапі надзвичайно важлива комунікація з замовником.

Трикутник PM



Software Development Life Cicle

- ***2. Аналіз системи***:
- визначити і задокументувати вимоги кінцевого користувача системи – в чому його очікування і як їх здійснити.
- для проекту робиться техніко-економічне обґрунтування, яке з'ясовує, чи є проект організаційно, економічно, соціально, технологічно здійсненним.
- дуже важливо підтримувати гарний рівень комунікації з замовниками, щоб переконатися, що у вас є чітке бачення кінцевого продукту і його функцій.
- Адже скільки людей, стільки й думок, важливо дійти до спільного знаменника.
- Після аналізу цих даних буде три варіанти: розробити нову систему, покращити існуючу або залишити систему як є.
- На цьому етапі надзвичайно важлива комунікація з замовником.

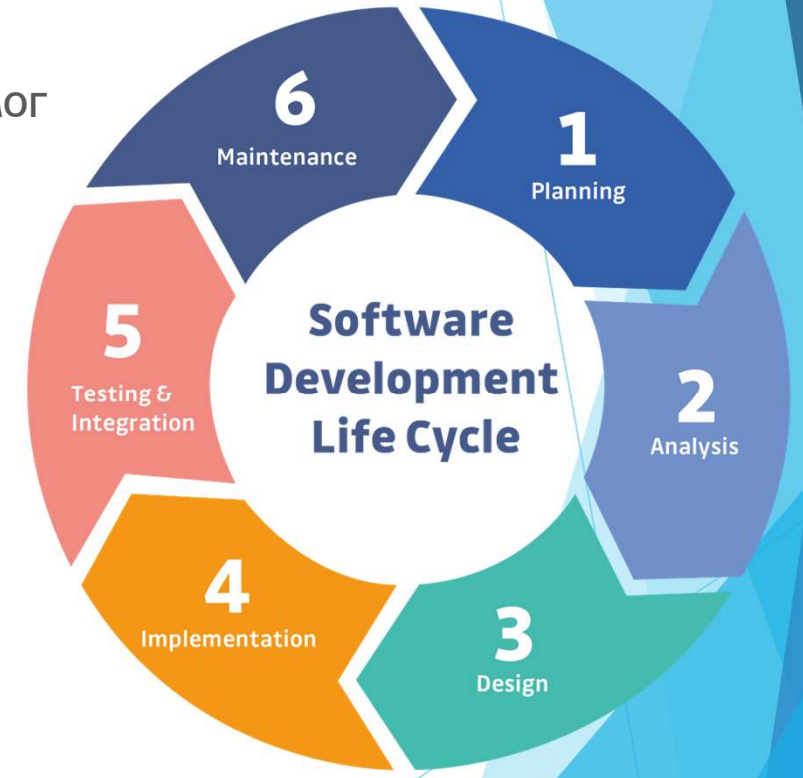


Software Development Life Cicle

- ***3. Проєктування системи***
- Фаза проєктування настає після того, коли досягнуто хорошого розуміння вимог користувача і ви точно знаєте, що саме треба втілити.
- Ця фаза визначає елементи системи, компоненти, рівень безпеки, модулі, архітектуру, різні інтерфейси і типи даних, якими оперує система. Проєкт системи визначає, як система буде виглядати і як функціонувати.
- Потім робиться розширений детальний проєкт, з урахуванням всіх функціональних і технічних вимог.

Software Development Life Cicle

- ***4. Розробка, впровадження і розгортання***
- Ця фаза йде після повного розуміння системних вимог і специфікацій. Це і є власне процес розробки системи, коли її проєкт вже повністю завершено. Саме на цьому етапі пишеться код, а також фаза впровадження може включати в себе конфігурацію і налаштування під певні вимоги і функції.
- На цій стадії система готова до установки у замовника, до запуску в робочий режим. Можливо, кінцевим користувачам буде потрібний тренінг, щоб вони освоїлися з системою і знали, як її використовувати.
- Фаза впровадження може бути дуже довгою – це залежить від складності системи.



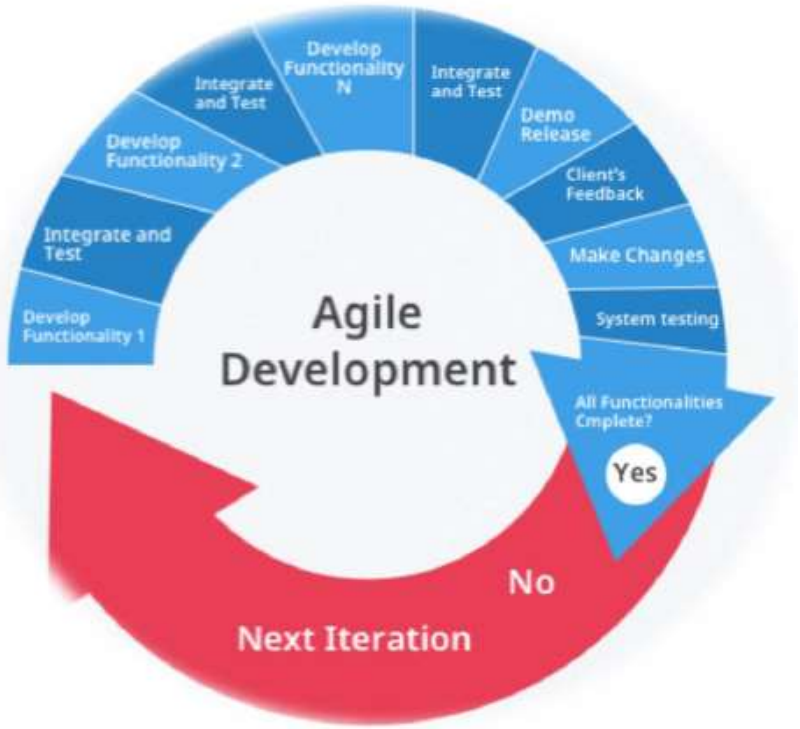
Software Development Life Cicle

- ***5. Тестування та інтеграція***

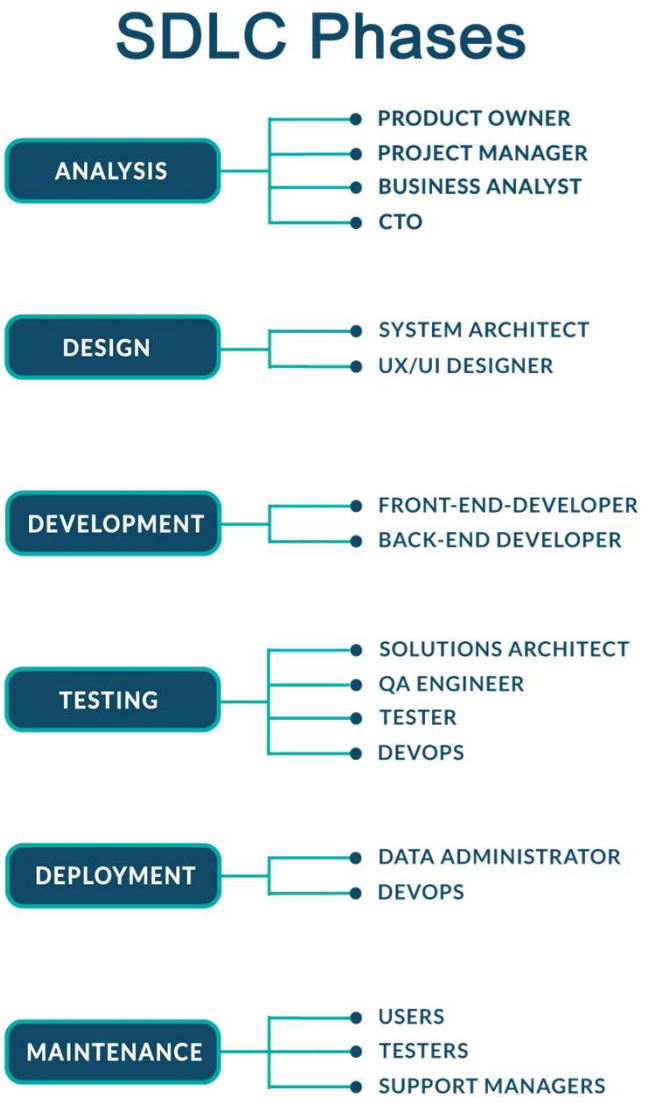
- Тут відбувається об'єднання різних компонентів і підсистем в єдину цілісну систему. В систему подаються різні вхідні дані і аналіз виходу, поведінки і функціонування.
- Роль тестування все збільшується, при цьому воно не вимагає знань коду, конфігурації обладнання чи дизайну.
- Тестування може виконуватися самими користувачами або спеціальною командою тестувальників. Воно може бути систематичним і автоматизованим, з тим, щоб упевнитися, що актуальні результати роботи системи збігаються з передбаченими і бажаними

Software Development Life Cicle

- ***6. Підтримка системи***
- На цій фазі здійснюється періодична технічна підтримка системи, щоб переконатися, що вона не застаріла.
- Сюди входить:
- заміна старого обладнання і постійна оцінка продуктивності
- здійснюються апдейти певних компонентів, щоб упевнитися, що система відповідає потрібним стандартам і новітнім технологіям, і не схильна до загроз безпеки.

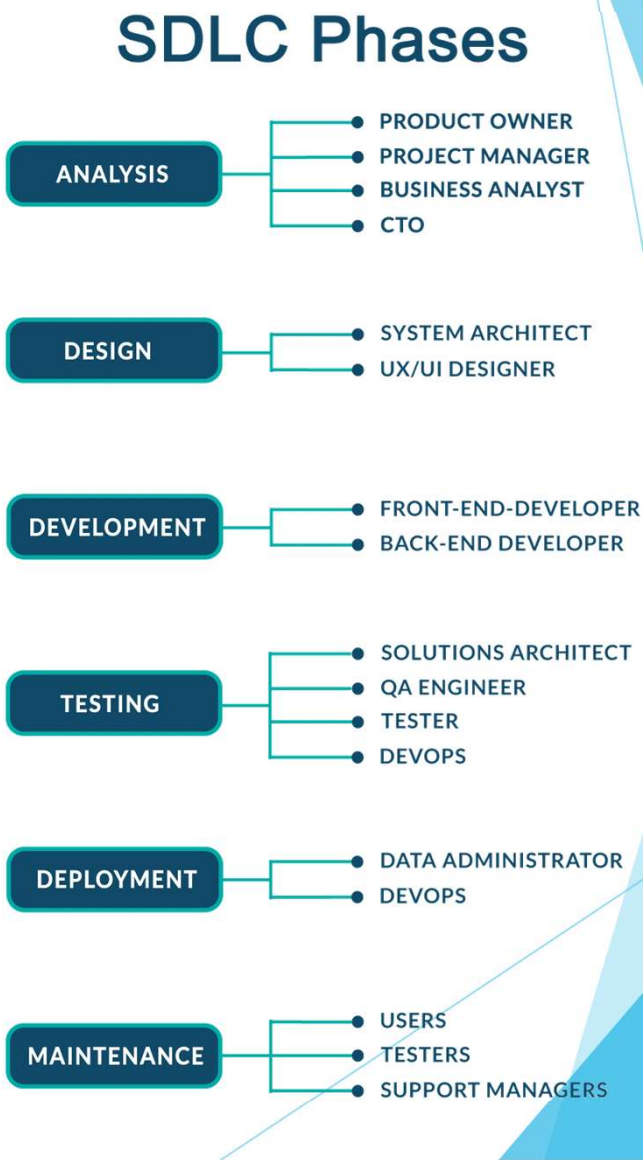

Develop Functionality 1
Integrate and Test
Develop Functionality 2
Integrate and Test
Develop Functionality N
Integrate and Test
Demo Release
Client's Feedback
Make Changes
System testing
All Functionalities Cmplete?
Yes
No
Next Iteration
Agile Development

Ролі та задачі



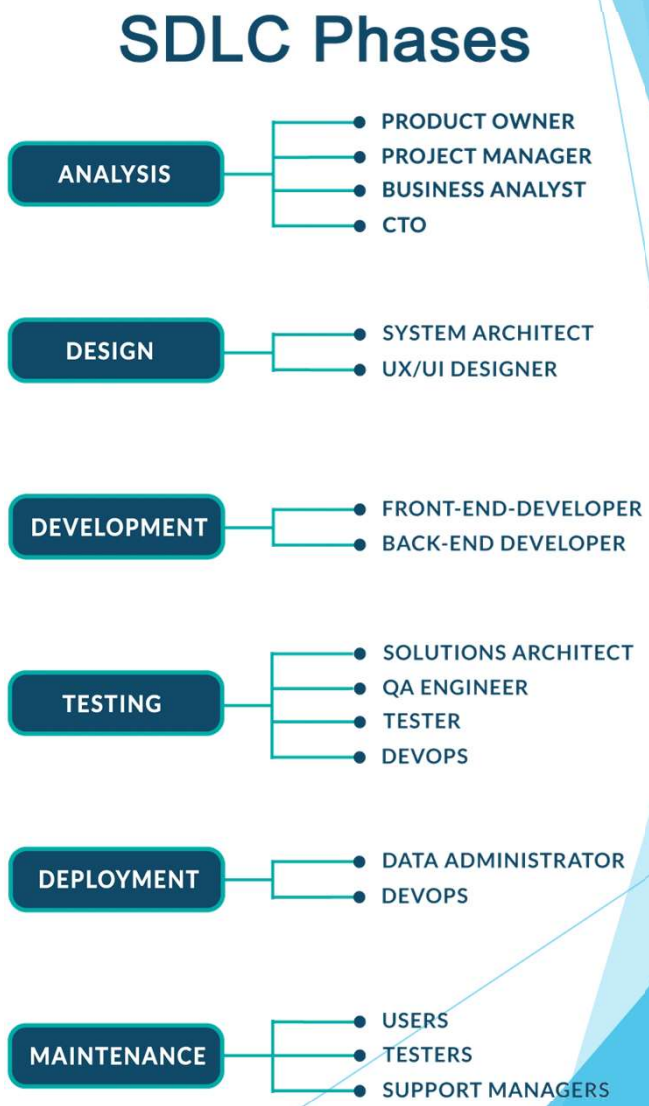
Розробник

- Загальна відповідальність за розробку, дизайн і впровадження нових або модифікованих програмних продуктів або поточні бізнес проєкти
- Різні мови програмування: Java, C#, C++, Ruby, Python тощо.
- Конкретні сфери діяльності: банківська справа, IoT, охорона здоров’я тощо.
- Робота над back-end або front-end частиною (full stack як варіант)



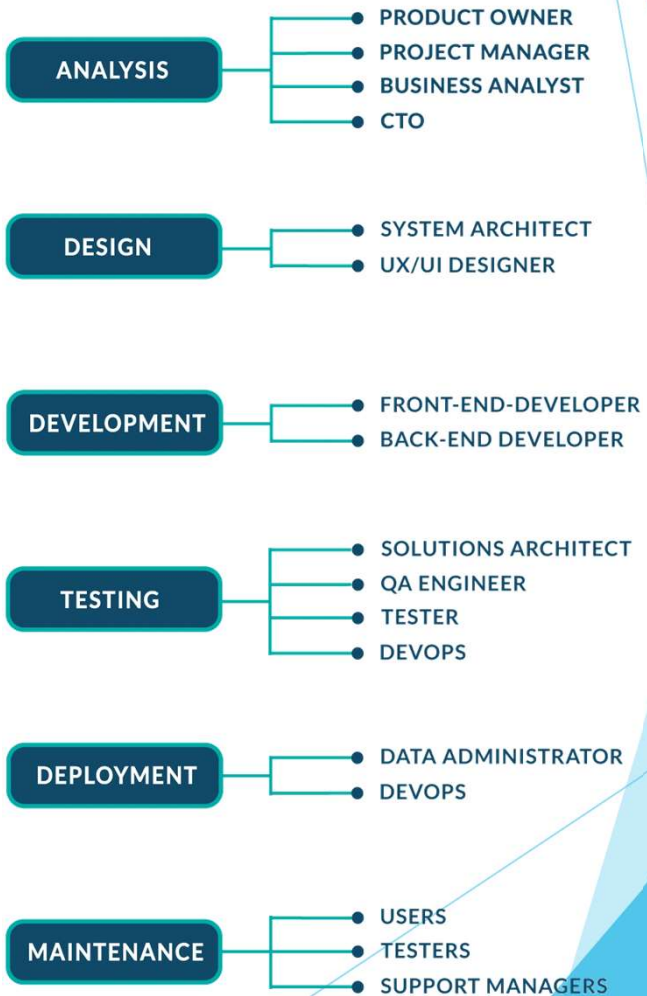
Менеджер проєкту

- Забезпечує бачення проєкту
- Виконує організацію роботи
- Слідкує за послідовністю та порядком виконанням процесів
- Робить так, щоб речі, які ніхто не любить робити, все одно виконувалися
- Усуває перешкоди
- Забезпечує інтеграцію



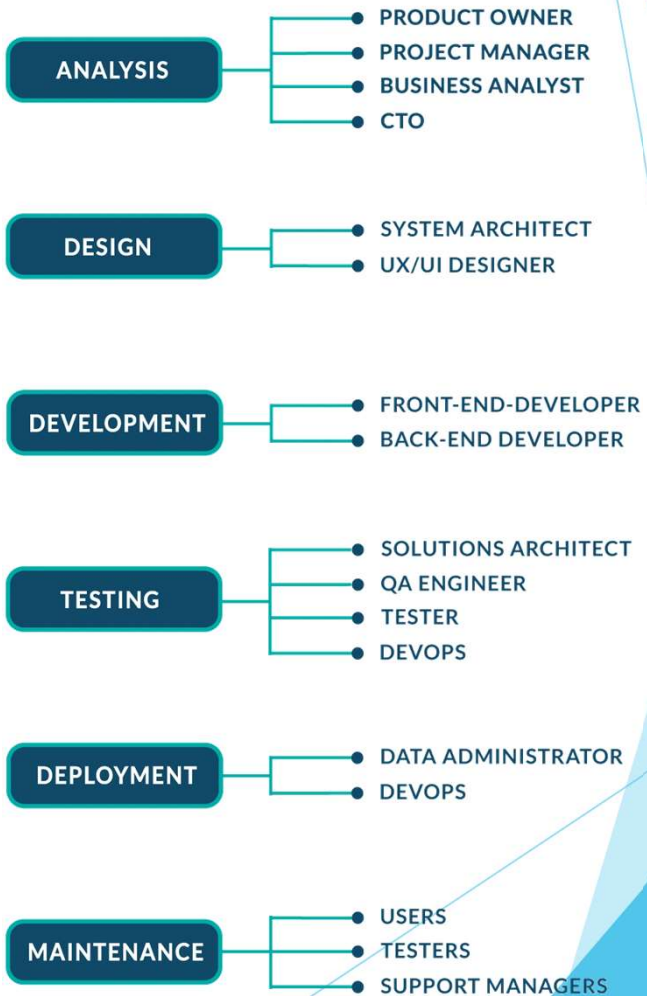
Тестувальник QA

- Виконує тестові кейси різних обставин
- Документує та оцінює результати тестування
- Виявляє, реєструє та повідомляє про помилки та збої програми
- Відстежує дефекти та допомагає усунути помилки
- Переглядає тестові процедури та розробляє тестові скрипти
- Співпрацює з інженерами, щоб стимулювати роботу з контролю якості
- Може спеціалізуватись на ручному або автоматизованому ттестуванні



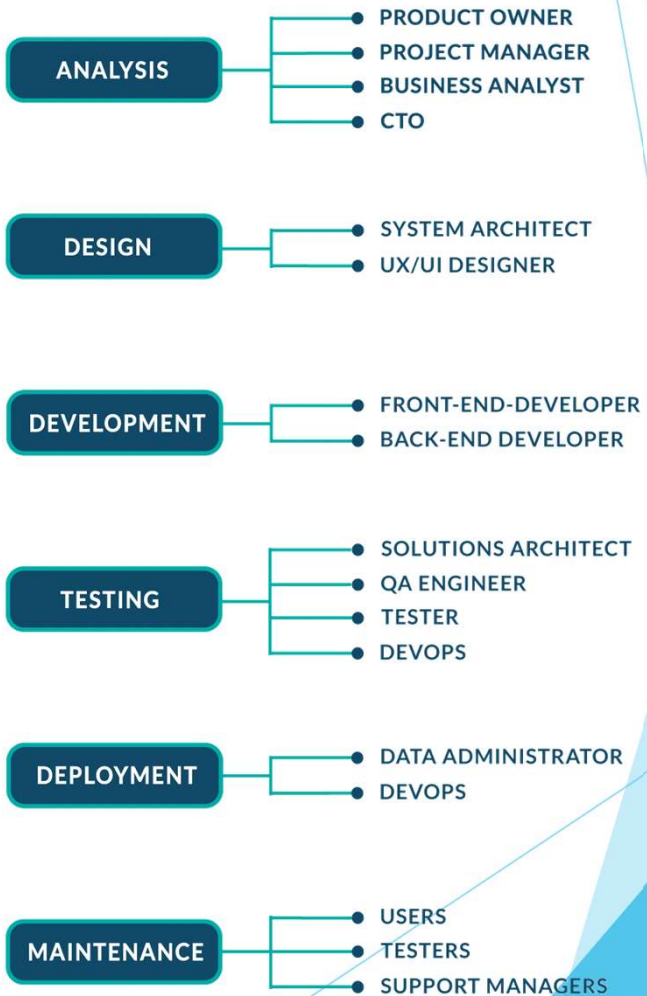
Бізнес Аналітик BA

- Уточнення вимог шляхом постійної співпраці із зацікавленими сторонами клієнта
- Документування вимог за допомогою User stories
- Керування переліком задач (backlog) продукту
- Тісно співпрацює із зацікавленими сторонами
- Збирає і формалізує функціональні та нефункціональні вимоги
- Несе відповідальність за підписування вимог



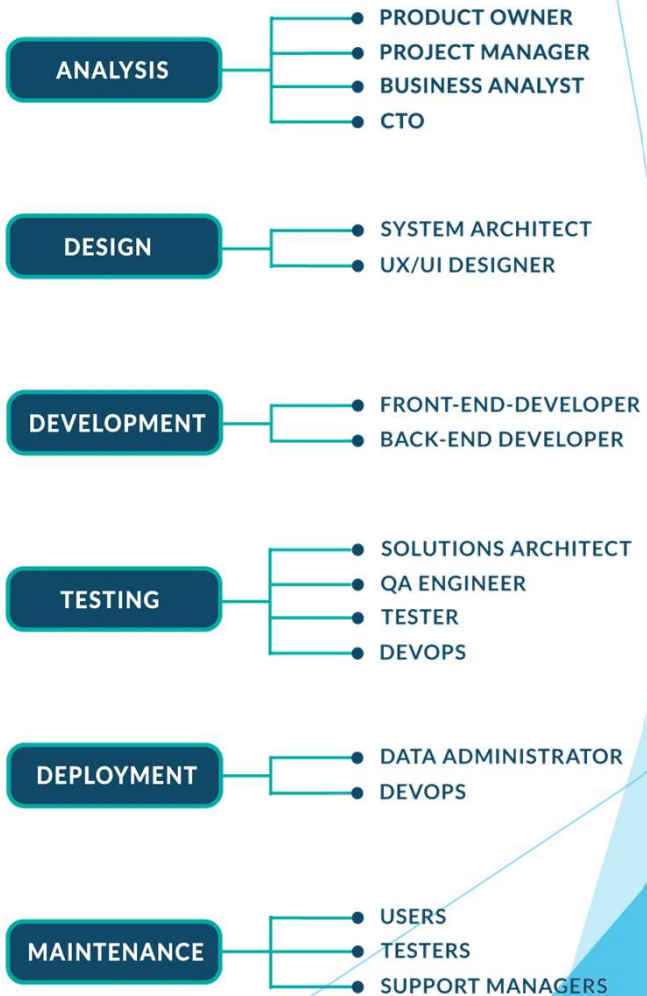
DevOps Інженер

- Відповідає за налаштування, розгортання, масштабування та моніторинг для підтримки хмарної інфраструктури та конвейерів випуску(release pipelines)
- Автоматизує та оптимізує інфраструктуру, операції та процеси розгортання
- Моніторить засоби і автоматизовані середовища для забезпечення проактивних сповіщень про стан і надійність системи
- Усуває несправності і вирішує основні проблеми у розробці, тестуванні та продуктивності хмарних середовищ



UI/UX дизайнер

- Проводить консультації починаючі з ідей до перевірки та розробки основної концепції та впровадження побажань та ідей
- Більш широке планування досліджень і кроків користувача, розробка/створення креслень
- Створення та реалізація артефактів UX (наприклад, персонажів, сценаріїв, карти подорожей, потоків користувачів, карт сайту, каркаси та прототипів), що відповідає потребам проекту



Архітектура програмного забезпечення

- **Архітектура програми або обчислювальної системи** – це фундаментальна конструкція її структур, тобто викладення її програмних елементів, їх зовнішніх властивостей та встановлених між ними відношень.
- Це загальна картина або загальна структура всієї системи — як все працює разом. Звідси випливає, що для розробки системи програмного забезпечення архітектор програмного забезпечення повинен брати до уваги багато факторів:
- • призначення системи,
- • аудиторія або користувачі системи,
- • властивості, які є найбільш важливими для користувачів, і
- • де буде працювати система.

Архітектура програмного забезпечення також залежить від

- мов реалізації,
- середовища розгортання,
- структури команди, що її розробляє
- потреб бізнесу,
- циклу розвитку,
- можливостей для ліній продуктів.