УНІВЕРСИТЕТ «КРОК»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТА КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

МЕТОДИЧНІ НАСТАНОВИ
до виконання здобувачами вищої освіти
Проєкту другого рівня

ЗАВТЕРДЖЕНО
на засіданні кафедри
комп’ютерних наук
протокол № 8 від 21.04.2022

Київ, 2022

Методичні настанови до виконання здобувачами вищої освіти Проєкту другого рівня / уклад. Єпік О.М., Тимчук О.С.; Київ, Університет «КРОК». 2022. 32 с.

Укладачі:
Єпік М.О., канд. техн. наук; Тимчук О.С., канд. техн. наук.

Рецензенти:
Пилипенко А.І., канд. техн. наук
Барибін О.І., канд. техн. наук

Проєкт другого рівня виконується здобувачами спеціальності 122 «Комп'ютерні науки» у відповідності до навчального плану 3 курсу у 4 семестрі.

Метою виконання Проєкту є проєктування та реалізація *кросплатформного програмного забезпечення* із застосуванням процедурної та об’єктно-орієнтованої парадигм програмування, методів й алгоритмів обчислень, структур даних і механізмів управління.

Програмне забезпечення обов’язково повинно мати графічний інтерфейс користувача.

Логіка програми має бути реалізована на мові програмування *Python*, а логіка інтерфейсу – на *PyQt* 5.

Проєкт виконується відповідно до рекомендацій щодо створення і керування стартапами (*startups*), розробником яких є Громадська організація «Платформа інноваційного партнерства» (YEP™).

ПЕРЕДМОВА

Положення методичних настанов рекомендовані до застосовування під час навчання здобувачами освітнього ступеня «Бакалавр» на освітній програмі за спеціальністю 122 «Комп'ютерні науки» відповідно до графіку навчального процесу.

Текст методичних настанов містить положення, які розкривають рекомендовані дії здобувачів під час виконання Проєкту.

Для більш глибокого розуміння положень методичних настанов та з'ясування додаткової довідкової інформації здобувачам рекомендовано ознайомитись з текстами релевантних документів, які регламентують організацію освітнього процесу в Університеті «КРОК», регулюють відносини, що виникають між його учасниками, є невід'ємною складовою системи забезпечення якості освітньої діяльності та якості вищої освіти (системи внутрішнього забезпечення якості) Університету.

До таких основних документів належать:

1) Положення про організацію освітнього процесу в Університеті «КРОК» [1];
2) Положення про академічну доброчесність в Університеті «КРОК» [2];
3) Положення про перевірку академічних та наукових текстів на плагіат в університеті «КРОК» [3].

Для якісного написання пояснювальної записки, підготовки до презентації та захисту рекомендовано також ознайомитись з текстами державних стандартів, які встановлюють вимоги до оформлення наукових текстів та бібліографічного опису використаних джерел:

1) ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання [4];
2) ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні вимоги та правила складання [5].

ЗМІСТ

ВСТУП

Проєкт другого рівня здобувач вищої освіти освітньо-професійної програми за спеціальністю 122 «Комп’ютерні науки» [6] реалізує у четвертому семестрі.

Робота над Проєктом передбачає встановлення відповідності засвоєних здобувачами рівня й обсягу знань, умінь, інших компетентностей до вимог стандарту вищої освіти першого (бакалаврського) рівня ступеня «бакалавр» за галуззю знань 12 «Інформаційні технології» спеціальністю 122 «Комп’ютерні науки» (затверджено та введено в дію наказом Міністерства освіти і науки України від 10.07.2019 р. № 962) [6].

У ході виконання Проєкту здобувачі мають продемонструвати свою здатність:

1) до системного мислення, застосування методології системного аналізу для дослідження складних проблем різної природи, методів формалізації та розв’язування системних задач, що мають суперечливі цілі, невизначеності та ризики;
2) проєктувати та розробляти програмне забезпечення із застосуванням різних парадигм програмування: узагальненого, об’єктно-орієнтованого, функціонального, логічного, з відповідними моделями, методами й алгоритмами обчислень, структурами даних і механізмами управління;
3) застосовувати методології, технології та інструментальні засоби для управління процесами життєвого циклу інформаційних і програмних систем, продуктів і сервісів інформаційних технологій відповідно до вимог замовника;
4) до аналізу предметних областей різних галузей знань, збору детальної інформації для формалізації функціональних та нефункціональних вимог до програмного забезпечення, ідентифікації, класифікації та пошуку методів і підходів щодо їх розв'язання;
5) аналізувати, проєктувати та розробляти прототипи людино-машинних інтерфейсів;
6) до командної роботи при розробці програмного забезпечення протягом його життєвого циклу (від збору вимог до впровадження) та різних середовищ;
7) до написання дизайн-документів з використанням UML-нотації;
8) проводити тестування компонентів програмного забезпечення.

У результаті роботи над Проєктом здобувачі вищої освіти мають навчитися:

1) аналізувати предметні області різних галузей знань, збирати детальну інформацію для формалізації функціональних та нефункціональних вимог до програмного забезпечення, ідентифікувати, класифікувати та виконувати пошук методів і підходів щодо їх розв'язання;
2) пояснювати кінцевому користувачу отримані результати у зручний спосіб;
3) застосовувати інструментальні засоби проєктування, розробки, тестування, вимірювання та документування програмного забезпечення;
4) створювати програмне забезпечення з графічним інтерфейсом користувача.

Написання, підготовка до захисту та захист Проєкту базується на таких основних принципах, як усвідомлення важливості академічної доброчесності та відповідальності за її порушення, а також нульової толерантності до порушення академічної доброчесності [2].

Термін виконання та захисту Проєкту визначається графіком освітнього процесу.

Успішно захищені Проєкти здобувачів оприлюднюються на офіційному сайті (або у репозиторії) університету.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 *Scrum-методологія*

Методологія розробки програмного забезпечення (ПЗ) – це шаблон, що визначає взаємодію різних складових процесу розробки ПЗ. Важливо обрати методологію, яка буде найбільш відповідною для проєкту, що дозволить суттєво знизити витрати та підвищити якість результату.

Існують наступні види методологій розробки ПЗ:

1) *каскадна або поетапна розробка*: модель «Водоспад» (*Waterfall*), каскадна модель з проміжним контролем, V-образна модель (розробка через тестування) – процес створення ПЗ, який є потоком, що послідовно проходить фази аналізу, проєктування, реалізацій, тестування, інтеграції та підтримки (зазвичай використовується для розробки крупних проєктів з тривалим терміном впровадження, на даний час майже не використовується);
2) *ітеративна інкрементна* (еволюційна) модель: дана методологія дозволяє паралельно виконувати ряд завдань з безперервним аналізом результатів і коректуванням попередніх етапів роботи (робота для великого штату кваліфікованих програмістів): RUP (*Rational Unified Process*);
3) *спіральна методика* (*Spiral*) характеризується проходженням проєктом циклу, що повторюється в кожній фазі розвитку: планування-реалізація-перевірка-оцінка (*plan-do-check-act cycle*) (використовується для створення проєктів, які не мають остаточно сформованого результату або вимагають термінового впровадження по етапах);
4) *гнучкі технології або Agile-розробка*: Scrum, XP (*Extreme Programming*), Crystal-методології, DSDM (*Dynamic Systems Development Model*), FDD (*Feature Driven Development*) – швидка розробка без шкоди якості, коли головним є працюючий продукт, а не його документація; сучасний неформалізований підхід до створення ПЗ в процесі якого реагування на зміни цінується вище строго дотримання плану (для проєктів, що швидко розвиваються і з кожною ітерацією ПЗ готови до його релізу);
5) *модель, що орієнтована на користувача*: JAD-модель (*Joint Application Development*);
6) *модель швидкої розробки*: RAD (*Rapid Application Development*).

Гнучка методологія розробки або *Agile-методологія* – серія підходів до розробки ПЗ, що орієнтовані на використання інтерактивної розробки, динамічного формування вимог і забезпечення їх реалізації у результаті

постійної взаємодії всередині робочих груп, що самоорганізовуються і складаються із спеціалістів різного профілю. Існує декілька методик, що відносяться до класу гнучких методологій розробки, а саме Scrum, XP (*Extreme Programming*), Crystal-методології, DSDM (*Dynamic Systems Development Model*), FDD (*Feature Driven Development*).

Більшість гнучких методологій націлено на мінімізацію ризиків шляхом зведення розробки всього проєкту до серії стислих циклів, що є ітераціями, які зазвичай тривають 2-3 тижні. Кожна ітерація є програмним проєктом в мініатюрі і включає усі завдання, що необхідні для видачі міні-приросту по функціональності: планування, аналіз вимог, проєктування, програмування, тестування і документування. Хоча окрема ітерація, як правило, недостатня для випуску нової версії продукту, мається на увазі, що гнучкий програмний проєкт готовий до випуску в кінці кожної ітерації. Після закінчення кожної ітерації команда виконує переоцінку пріоритетів розробки.

Практично всі Agile-команди сконцентровані в одному офісі (*bullpen*). Офіс включає *Product owner* – замовника, який і визначає вимоги до продукту. В якості замовника може виступати бізнес-аналітик, менеджер проєкту або клієнт. Крім того, в офіс можуть входити також дизайнери інтерфейсу, тестувальники, технічні письменники. Тобто Agile-методи націлені в першу чергу на безпосереднє спілкування.

Основні ідеї Agile-методології:

1) люди і взаємодія важливіше процесів і інструментів;
2) працюючий продукт важливіше вичерпної документації;
3) співпраця із замовником важливіша за узгодження умов контракту;
4) готовність до змін важливіше дотримання початкового плану.

Принципи Agile-методології:

1) задоволення клієнта за рахунок раннього і безперебійного постачання цінного програмного забезпечення;
2) вітання змін вимог навіть наприкінці розробки (це може підвищити конкурентоспроможність отриманого продукту);
3) часте постачання робочого програмного забезпечення (кожного місяця або тижня або ще частіше);
4) тісне, щоденне спілкування замовника з розробниками впродовж всього проєкту;
5) проєктом займаються мотивовані особи, які забезпечені потрібними умовами роботи, підтримкою і довірою;

6) метод передачі інформації, що рекомендується, – особиста розмова (обличчям до обличчя);
7) працююче програмне забезпечення – кращий вимірювач прогресу;
8) спонсори, розробники і користувачі повинні мати можливість підтримувати постійний темп на невизначений термін;
9) постійна увага поліпшенню технічної майстерності і зручному дизайну;
10) простота – мистецтво не робити зайвої роботи;
11) кращі технічні вимоги, дизайн і архітектура виходять в команди, що самоорганізовується;
12) постійна адаптація до обставин, що змінюються.

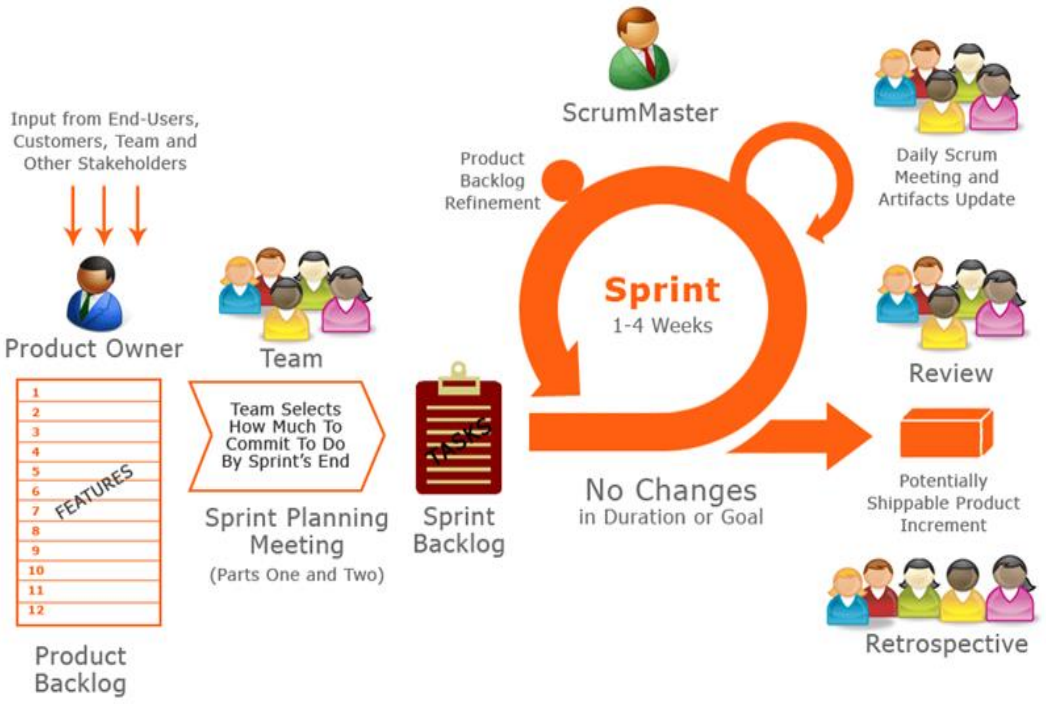
Переваги Agile-методології:

1) якість продукту (залучення замовника до процесу кожної ітерації дає можливість коректувати процес, що незмінно підвищує якість);
2) висока швидкість розробки (ітерація триває не більш 3-х тижнів, до кінця цього терміну обов'язково є результат);
3) мінімізація ризиків (крупний проєкт дає можливість замовникові сплатити декілька ітерацій і в ході роботи зрозуміти, що він вчасно отримає саме те, що хоче і за прийнятну ціну; замовник завжди має можливість спостерігати за ходом розробки, коректувати функціональність проєкту, тестувати або запускати його, навіть може зупинити його у будь-яку мить).

Scrum – це гнучкий метод керування проєктами, метою якого є підвищення продуктивності праці в командах розробників. Дана методологія також використовується коли є недолік в наявності досвідчених фахівців на ключових позиціях або чітко сформульованих вимог до продукту.

Своєму терміну «Scrum» зобов'язаний регбі, в якому це слово означає метод командної гри у вигляді побудови трьох ліній кожним з суперників і спроби захопити м'яч. Для успішного перехоплення потрібна не лише хороша фізична підготовка, але і злагодженість кожного учасника сутички і чітке розуміння мети.

Scrum – це методологія керування проєктами, яка побудована на принципах тайм-менеджменту. Основною її особливістю є залученість до процесу усіх учасників, кожен з яких має певну роль. Над розв'язанням завдання працює не лише команда розробників, але і всі ті, кого цікавить вирішення даного завдання [11]:



Основні терміни Scrum-методології:

1) *Власник продукту* (*Product Owner*) – людина, яка має безпосередній інтерес у якісному кінцевому продукті, вона розуміє як цей продукт повинен виглядати та працювати; ця людина не працює у команді розробників, вона працює на боці замовника або клієнта (наприклад, інша компанія, інший відділ і т. д.), але вона є членом команди; дана людина розставляє пріоритети для завдань;
2) *Scrum-майстер* (*Scrum Master*) – керівник проєкту, який стежить за дотриманням всіх принципів;
3) *Scrum-команда* (*Scrum Team*) – команда, яка приймає усі принципи *Scrum* і готова з ними працювати;
4) *Спринт* (*Sprint*) – відрізок часу, що береться для виконання визначеного (обмеженого) списку завдань (рекомендований термін – 2-4 тижні, тривалість визначається командою один раз);
5) *Беклог* (*Backlog*) – список усіх робіт (щоденник загального користування); поділяється на 2 види: Product-беклог (повний список усіх робіт, використовуючи які отримуємо кінцевий продукт) та Спринт-беклог (список робіт, які визначила команда і узгодила з Власником продукту на найближчий звітний період (спринт), завдання беруться із Product-беклог);
6) *Планування спринту* (*Sprint Planning Meeting*) – це нарада, де присутні усі (команда, Scrum-майстер, Власник продукту); впродовж цієї наради Власник продукту визначає пріоритетні завдання, які він би хотів побачити виконаними після закінчення терміну; команда оцінює час,

який знадобиться для виконання поставлених завдань; Результатом наради є список вимог, який не може змінюватися протягом спринту і до кінця спринту повинен бути виконаний.

Життєвий цикл спринту складається з:

1) планування (*Sprint Planning*);
2) зупинки (*Sprint Abnormal Termination*);
3) щоденна зустріч (*Daily Scrum Meeting*);
4) ретроспектива (*Retrospective*).

Планування спринту (*Sprint Planning*): на початку кожного спринту проводиться планування спринту. У плануванні беруть участь замовники, користувачі, менеджмент, Власник продукту, Scrum-майстер і Scrum-команда. Планування спринту складається з 2 послідовних зустрічей.

У першій зустрічі приймають участь замовники, користувачі, менеджмент, Власник продукту, Scrum-майстер і Scrum-команда. Мета першої зустрічі: визначити мету (*Sprint Goal*) і *Sprint Backlog* – функціональність, яка буде розроблена протягом наступного спринту для досягнення мети спринту. Артефакт (результат зустрічі): *Sprint Backlog*.

У другій зустрічі приймають участь Scrum-майстер і Scrum-команда. Мета другої зустрічі: визначити, як саме буде розроблятись визначена функціональність для того, щоб досягнути мети спринту. Для кожного елементу *Sprint Backlog* визначається список завдань і оцінюється їхня тривалість. Артефакт (результат зустрічі): у *Sprint Backlog* з'являються завдання.

Якщо протягом спринту виявляється, що команда не може встигнути зробити заплановані на спринт завдання, то Scrum-майстер, Власник продукту і Scrum-команда знову зустрічаються і з'ясовують, як можна скоротити обсяг робіт і при цьому досягнути мети спринту.

Зупинка спринту (*Sprint Abnormal Termination*) проводиться у виняткових ситуаціях. Спринт може бути зупинений до того, як закінчаться відведені 30 днів. Спринт може зупинити команда, якщо вона розуміє, що не може досягнути мети спринту в зазначені терміни. Спринт може зупинити Власник продукту, якщо необхідність у досягненні мети спринту зникла. Після зупинки спринту проводиться зустріч з командою, де обговорюються причини зупинки спринту. Після цього починається новий спринт: проводиться його планування і починаються роботи.

Щоденна зустріч (*Daily Scrum Meeting*) проходить кожний ранок на початку дня. Він призначений для того, щоб всі члени команди знали, хто і чим

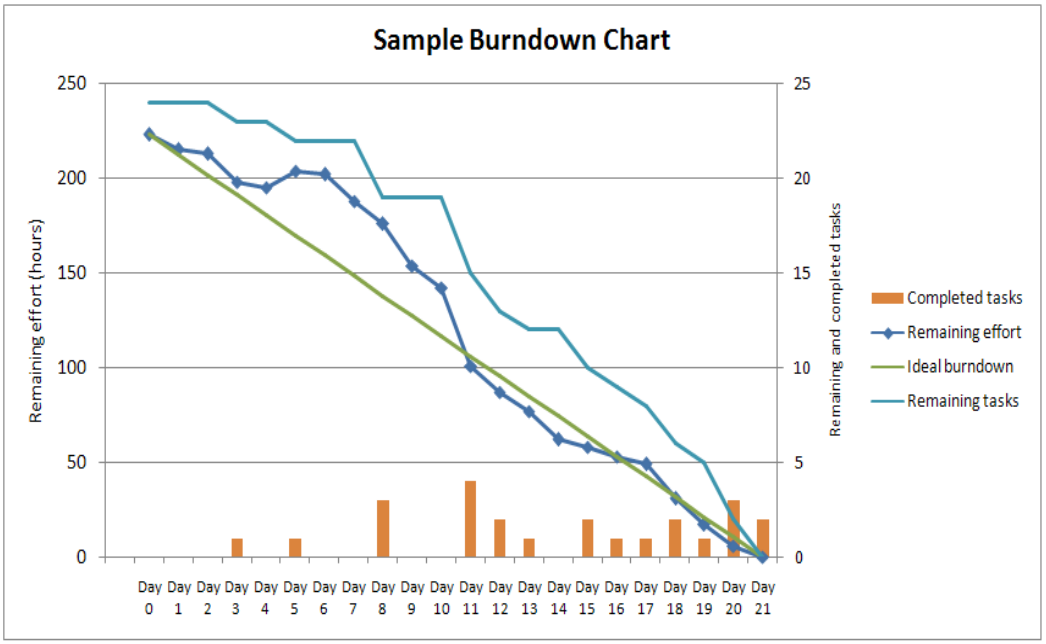
займається. Тривалість зустрічі строго обмежена і не повинна перевищувати 15 хвилин. Мета зустрічі – поділитися інформацією. Дана зустріч не призначена для розв'язання проблем, що виникли у проєкті. Усі спеціальні питання на цій зустрічі не розглядаються. Зустріч проводить Scrum-майстер. Він по колу задає питання кожному члену команди:

1) що зроблено вчора?
2) що буде зроблено сьогодні?
3) з якими проблемами зіткнувся?

Scrum-майстер збирає усі відкриті для обговорення питання у вигляді *Action Items*, наприклад у форматі що/хто/коли.

Щоденно оновлюється *Burndown chart* – діаграма, що ілюструє кількість роботи, що зроблена та залишилась. Існують різні види діаграми [10]:

1) діаграма спалювання робіт для спринту – показує кількість завдань, що зроблені, і кількість завдань, що треба зробити у наступному спринті;
2) діаграма спалювання робіт для випуску проєкту – показує кількість завдань, що зроблені, і кількість завдань, що треба зробити до випуску продукту (зазвичай будується на основі декількох спринтів).



Ретроспектива (*Sprint Retrospective*): наприкінці кожного спринту команда збирається на Ретроспективу, мета якої – переглянути якість існуючих процесів, взаємовідношення членів команди і інструменти, що застосовуються. Команда визначає, що зроблено добре, що зроблено не дуже

добре, а також виявляє потенційні можливості для поліпшень і складає план поліпшень на майбутнє.

Демонстрація робочої версії продукту Product Owner (*Sprint Review*): для обговорення, отримання вражень і побажань.

1.2 *Кросплатформність*

Кросплатформність (багатоплатформність, мультиплатформність) – властивість програмного забезпечення працювати більш ніж на одній програмній (у тому числі – операційній системі) або апаратній платформі, та технології, що дозволяють досягти цієї властивості.

Кросплатформність дозволяє суттєво скоротити витрати на розробку нового або адаптацію існуючого програмного забезпечення. Залежно від засобів реалізації кросплатформність поділяється на

1) кросплатформність на рівні мов програмування (а також інструментів таких мов: компіляторів та редакторів зв'язків);
2) кросплатформність на рівні середовища виконання;
3) кросплатформність на рівні операційної системи;
4) кросплатформність на рівні апаратної платформи.

Кросплатформне програмне забезпечення (ПЗ) може бути поділено на дві категорії:

1) ПЗ, яке потребує індивідуальну компіляцію для кожної платформи, під яку воно розроблено;
2) ПЗ, яке може виконуватися на довільній платформі без спеціальної підготовки, тобто, ПЗ написане на інтерпретуючій мові (яка може підтримувати попереднє створення переносимого байт-коду), інтерпретатори або середовища виконання якої є стандартними компонентами або підтримуються більшістю сучасних платформ.

Кросплатформне програмування — практика написання програм, які можуть працювати більше ніж на одній платформі.

1.3 *Тестування*

Модульне тестування (*Unit testing*) – це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми.

Модулем називають найменшу частину програми, яка може бути протестованою.

Check-list (контрольний список, що містить ряд необхідних перевірок для тестування) для тестування програмного продукту складається з:

1) тестування зручності використання;
2) функціонального тестування;
3) тестування сумісності;
4) тестування бази даних;
5) тестування безпеки;
6) тестування продуктивності.

Тестування зручності використання (юзабіліті) – це тестування доброзичливості додатку для користувача. При тестуванні зручності використання перевіряється, чи легко новому користувачеві розібратися в додатку і тестується системна навігація.

Мета тестування зручності використання: впевнитися у простоті і ефективності використання продукту при використанні стандартних практик тестування зручності використання.
Сценарії тестування *зручності використання*:

1) вміст форм вірний, без граматичних і орфографічних помилок;
2) всі шрифти відповідають вимогам;
3) всі тексти правильно вирівняні;
4) всі повідомлення про помилки вірні, без орфографічних і граматичних помилок, і відповідають заголовку вікна;
5) підказки існують для всіх полів;
6) всі поля правильно вирівняні;
7) між полями, колонками, рядами і повідомленнями про помилки залишено досить вільного місця;
8) всі кнопки повинні мати стандартний формат і розмір;
9) посилання на домашню сторінку має бути на кожній сторінці сайту;
10) неактивні поля мають бути сірими;
11) підтверджуючі повідомлення повинні відображатися для всіх операцій оновлення і видалення;

12) перевірте застосунок при різних роздільних здатностях екрану (640 x 480, 600x800 і т. д.);
13) перевірте, що користувач може користуватися системою без роздратування;
14) перевірте, що *Tab* правильно працює;
15) панель скролу повинна з'являтися лише тоді, коли вона потрібна;
16) всі поля (текстові, випадаючі меню, радіо-кнопки і т. д.) і кнопки мають бути доступні з клавіатури, і користувач має бути в змозі користуватися сайтом, використовуючи лише клавіатуру;
17) переконайтеся, що дані у випадаючих списках не обрізуються із-за розмірів поля, і перевірте, чи зашиті дані в код або керуються адміністратором.

Функціональне тестування – тестування функціональностей і операційної поведінки продукту з метою переконатися, що вони відповідають специфікаціям. Даний вид тестування ігнорує внутрішні механізми системи або компоненту і концентрується виключно на вихідних даних, отриманих у відповідь на призначене для користувача введення і умови виконання сценаріїв.

Мета функціонального тестування: переконатися, що продукт відповідає потрібній функціональній специфікації, згаданій у документації з розробки.

Сценарії *функціонального тестування*:

1) протестуйте валідацію всіх обов'язкових полів;
2) переконайтеся, що знак зірочки відображається у всіх обов'язкових полях;
3) переконайтеся, що система не відображає вікно помилки при незаповнених необов'язкових полях;
4) переконайтеся, що коди коректно валідуються і не викликають помилок в розрахунках;
5) протестуйте числові поля: вони не повинні приймати літери, у випадку введення літери повинне відображатися відповідне повідомлення про помилку;
6) протестуйте від'ємні значення в числових полях, якщо вони дозволені;
7) протестуйте, що ділення на нуль вірно обраховується;
8) протестуйте максимальну довжину кожного поля, щоб переконатися, що дані не обрізуються;

9) протестуйте спливаюче повідомлення («Це поле обмежене 500 знаками»), яке повинне відображатися, якщо введені дані перевищують дозволений розмір поля;
10) переконайтеся, що підтверджуюче повідомлення відображається для операцій оновлення і видалення;
11) переконайтеся, що значення вартості відображуються в потрібній валюті;
12) протестуйте всі поля введення на спецсимволи;
13) протестуйте функціональність сортування;
14) протестуйте функціональність доступних кнопок;
15) протестуйте всі дані у випадаючих списках: вони мають бути розташовані в хронологічному порядку.

Тестування сумісності перевіряє сумісність додатку з іншими елементами системи, в якій воно працює, – наприклад, браузерами, операційними системами або залізом.

Мета тестування сумісності: оцінка того, наскільки добре ПЗ працює під певною ОС, з іншим ПЗ або залізом.

Тестування продуктивності – проводиться для оцінки відповідності системи або компоненту специфічним вимогам до продуктивності.

Сценарії *тестування продуктивності*:

1) визначення продуктивності, стабільності і масштабованості додатку під різним навантаженням;
2) визначення, чи може актуальна архітектура підтримувати додаток при пікових навантаженнях;
3) визначення, яка конфігурація призводить до найкращих показників продуктивності;
4) визначення, чи не змінився час відгуку в нової версії додатку;
5) визначення пляшкового горла додатку і інфраструктури;
6) оцінка продукту і заліза з метою упевнитися, що вони витримають прогнозовані обсяги навантаження.

Неможливо провести *тестування продуктивності вручну* з ряду причин:

1) знадобиться велика кількість ресурсів;
2) неможливо одночасно здійснювати ряд дій;
3) відсутній відповідний спосіб відстежування поведінки системи;
4) складність виконання завдань, що повторюються.

2 СТРУКТУРА ПРОЄКТУ

Обсяг	240 годин / 8 кредитів ECTS
Виконання	командне (можливе індивідуальне виконання при попередньому погодженні з Керівником Проєкту)
Управління	*Scrum*-методологія
Команда	- Власник Продукту (*Product Owner*), Керівник (*Scrum Master*), Команда розробників (*Scrum Team*): від 3 до 5 здобувачів - роль *Product Owner* виконує Керівник Проєкту - роль *Scrum Master* поділяють Керівник Проєкту і Капітан Команди
Кількість спринтів	12
Обов’язкові церемонії	- планування спринту (обов'язково за участю *Product Owner*) - щоденні наради - демонстрації - ретроспектива (обов'язково за участю *Product Owner*)
Обов’язкові складові результату виконання проєкту	- реліз програмного забезпечення - пояснювальна записка - презентація продукту + текст докладу - спланований сценарій демонстрації роботи програмного забезпечення

3 ПОРЯДОК ВИКОНАННЯ ПРОЄКТУ

3.1 *Етапи виконання Проєкту*

№	Спринт	Дедлайн	Результат
1	Формування проєктної команди	до 12.01.202X	Затверджений склад проєктної команди
2	Дослідження ринку	до 18.01.202X	Документ з пропозиціями щодо нового продукту
3	Аналіз вимог	до 22.01.202X	Документ з вимогами до нового продукту
4	Проєктування взаємодії з користувачем	до 03.05.202X	Візуалізовані приклади використання, спрощені та деталізовані прототипи продукту
5	Візуальний дизайн	до 03.05.202X	Графічні файли з кінцевим дизайном продукту
6	Проєктування програмного забезпечення	до 18.05.202X	Файли з діаграмами та схемами компонент програмного забезпечення
7	Розробка програмного забезпечення	до 21.05.202X	Розроблений програмний продукт
8	Тестування програмного забезпечення	до 25.05.202X	Перелік помилок до виправлення
9	Підготовка пояснювальної записки	до 28.05.202X	Пояснювальна записка
10	Підготовка презентації	до 31.05.202X	Презентація продукту та спланований сценарій демонстрації роботи програмного забезпечення
11	Захист	до 04.06.202X	Підсумкова оцінка за виконання Проєкту

3.2 *Реліз програмного забезпечення*

Реліз програмного забезпечення повинен відповідати перерахованим нижче вимогам:

1) мати графічний інтерфейс користувача;
2) мати часткове покриття *unit*-тестами;
3) вихідний код програми відповідати стандарту оформлення коду;
4) мати відповідні коментарі або *doc-strings* до програмних елементів;
5) проєкт розміщено в репозиторії на GitHub.

3.3 *Структура та вимоги до пояснювальної записки*

Під час підготовки пояснювальної записки треба дотримуватися основних вимог до її змісту.

Пояснювальна записка умовно поділяється на:

1) вступну частину;
2) основну частину;
3) додатки.

Вступна частина містить такі структурні елементи:

1) титульний аркуш;
2) реферат.

Основна частина містить такі структурні елементи:

1) вступ;
2) змістовну частину звіту (суть пояснювальної записки);
3) висновки;
4) перелік джерел посилання.

Рекомендований обсяг пояснювальної записки не повинен перевищувати 50 сторінок.

Короткі настанови до обов'язкових структурних елементів пояснювальної записки наведені у таблиці 1.

Таблиця 1 – Настанови до обов'язкових структурних елементів пояснювальної записки до Проєкту другого рівня

Назва розділу	**Назва підрозділу**	**Вимоги та обов'язкові елементи**
Титульний лист	-	Див. у Додаток А
Лист завдання	-	-
Календарний план виконання проєкту	-	-
Реферат	-	- не більше 1 сторінки - українською та англійською мовами

Зміст	-	-
Вступ	-	- актуальність теми - мета і задачі проєкту - практична цінність проєкту - структура та обсяг пояснювальної записки
Аналіз предметної області і постановка задачі	Проблеми розвитку інформаційних технологій	- відомі теоретичні і практичні результати, які отримані іншими вітчизняними та іноземними авторами за напрямом роботи, із посиланнями на їх праці
	Аналіз існуючих інформаційних технологій	- можливості, переваги і недоліки відомих інформаційних технологій, які в повній мірі або частково вирішують задачі, які будуть закладені у новий продукт
	Висновки за розділом	- результати аналізу предметної області - постановка задачі
Проєктування застосунку	Аналіз варіантів використання	- функціональні та нефункціональні вимоги до нового продукту - діаграма прецедентів (*Use case diagram*)
	Моделювання процесів	- діаграми діяльності (*Activity diagram*)
	Моделювання даних	- опис ідентифікованих даних - формат зберігання даних
	Проєктування архітектури	- ідентифіковані програмні (наприклад, сховище даних, рівень доступу до даних, рівень програми та рівень інтерфейсу) та апаратні (наприклад, клієнтські комп'ютери, сервери та мережеве середовище) компоненти - схема розподілу програмних компонентів між апаратними компонентами - вимоги: експлуатаційні, до виконання, безпеки, культурні та політичні
	Проєктування графічного інтерфейсу користувача	- діаграма навігації по екранах (*User flow diagram*) - діаграма структури інтерфейсу (*Interface structure diagram*) - вайрфрейми (*Wireframe*) - прототип інтерфейсу (*UI prototype*)
	Проєктування структури	- діаграма класів (*Class diagram*) - діаграма об'єктів (*Object diagram*) - діаграма послідовності (*Sequence diagram*)
	Висновки за розділом	- результати етапу проєктування

Реалізація застосунку	Особливості реалізації	- опис засобів реалізації програмного забезпечення - опис деталей реалізації програмного забезпечення
	Конструювання	- діаграма компонентів (*Component diagram*) - діаграма пакетів (*Package diagram*)
	Реалізація графічного інтерфейсу користувача	- опис головного вікна і дочірніх вікон - опис діалогових вікон - опис використаних віджетів - опис використаних механізмів компонування віджетів - схема сигналів і слотів
	Документування	- схема організації довідкових технічних документів - схема організації користувацьких документів
	Тестування	- модульне тестування - функціональне тестування
	Висновки за розділом	- результати етапу реалізації
Висновки	-	- стислий виклад результатів вирішення поставленого завдання, що отримані в процесі аналізу предметної області, проєктування та реалізації програмного забезпечення - відомості про достовірність отриманих результатів - не більше 1 сторінки
Список посилань	-	- перелік використаних джерел у порядку згадування по тексту мовою оригіналу джерел відповідно до вимог ДСТУ 8302:2015 - не менше 20 найменувань
Додатки	-	- екрани застосунку - фрагменти лістингу

Вимоги до оформлення пояснювальної записки:

1. Формат аркушу - А4.
2. Обсяг основного тексту пояснювальної записки не повинен перевищувати 50 сторінок; загальний обсяг не повинен перевищувати 70 сторінок.
3. Оформлення пояснювальної записки повинно відповідати вимогам стандарту ДСТУ 3008:2015 «Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання» [4].

4. Оформлення списку посилань повинно відповідати вимогам стандарту ДСТУ 8302:2015 «Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання» [5].
5. Вимоги до оформлення основного тексту:
    a. параметри сторінки (поля): ліве – 2,5 см, праве – 1,5 см, верхнє і нижнє – 2 см;
    b. основний шрифт – Times New Roman, 14 пт;
    c. міжрядковий інтервал – 1,5;
    d. інтервал до та після рядків – 0;
    e. абзацний відступ – 1,25 см;
    f. вирівнювання тексту - по ширині.
6. Кожний розділ починається з нової сторінки.
7. Назва розділу оформлюється у верхньому регістрі.
8. Номер рисунків та таблиць складається з двох цифр, перша – номер розділу, в якому розташований рисунок або таблиця, друга – порядковий номер рисунку або таблиці у розділі.
9. Рисунки та таблиці розміщуються після першого посилання на них у тексті (рис. 1.1 або табл. 1.1). Після підпису рисунку або таблиці обов'язково наводяться дані про джерело, звідки вони взяті (розроблено автором (-ами) / розроблено автором (-ами) за даними [номер посилання із зазначенням номеру сторінки] / [номер посилання із зазначенням номеру сторінки]).
10. Нумерація сторінок – внизу по центру.
11. Вимоги до оформлення тексту додатків:
    a. параметри сторінки (поля): ліве – 2,5 см, праве – 1,5 см, верхнє і нижнє – 2 см;
    b. основний шрифт – Courier New, 12 пт;
    c. міжрядковий інтервал – 1;
    d. інтервал до та після рядків – 0;
    e. абзацний відступ – 0;
    f. вирівнювання тексту – по лівому краю.

3.4 *Структура та вимоги до презентації*

Презентація має містити наступні слайди:

1) титульний слайд;
2) аналіз предметної області;
3) постановка задачі;
4) аналіз варіантів використання;
5) діаграми діяльності;

6) опис даних;
7) архітектура;
8) візуальний дизайн;
9) об’єктно-орієнтований дизайн;
10) засоби реалізації;
11) діаграми компонентів і пакетів;
12) екрани застосунку;
13) висновки.

Короткі настанови до обов’язкових структурних елементів презентації наведені у таблиці 2.

Таблиця 2 – Настанови до обов’язкових структурних елементів презентації до Проєкту другого рівня

Назва слайду (ів)	**Вимоги та обов’язкові елементи**
Титульний слайд	- назва продукту - члени команди із зазначення ролей
Аналіз предметної області	- актуальність теми - мета і задачі проєкту
Постановка задачі	- стисло перелік основних задач - опис вхідних і вихідних даних
Аналіз варіантів використання	- список функціональних та нефункціональних вимоги до нового продукту - діаграма прецедентів (*Use case diagram*)
Діаграми діяльності	- діаграма діяльності (*Activity diagram*)
Опис даних	- діаграми відношень (*Entity relationship diagram*)
Архітектура	- схема розподілу програмних компонентів між апаратними компонентами
Візуальний дизайн	- діаграма навігації по екранах (*User flow diagram*) - діаграма структури інтерфейсу (*Interface structure diagram*) - прототип інтерфейсу (*UI prototype*)
Об’єктно-орієнтований дизайн	- діаграма класів
Засоби реалізації	- перелік засобів реалізації програмного забезпечення - перелік засобів управління проєктом
Діаграми компонентів і пакетів	- діаграма компонентів (*Component diagram*) - діаграма пакетів (*Package diagram*)
Екрани застосунку	- головне вікно застосунку - дочірні вікна застосунку
Висновки	- найбільш важливі практичні результати, одержані під час виконання проєкту - якісні та кількісні показники здобутих результатів

Вимоги до оформлення презентації:

1. Для підготовки презентації рекомендовано використовувати корпоративний шаблон презентації, який доступний за адресою https://www.krok.edu.ua/download/brand-book/krok-template-presentation.pptx.
2. Всі слайди (крім першого) повинні містити порядковий номер, розташований у правому нижньому кутку.
3. Кожен слайд (крім першого) повинен мати коротку назву (заголовок, без крапки в кінці).
4. Вимоги до оформлення тексту презентації:
    a. шрифт заголовків – Calibri, 36 пт, напівжирний;
    b. шрифт порядкових номерів слайдів - Calibri, 20 пт, напівжирний;
    c. шрифт основного тексту - Calibri, 22 пт;
    d. міжрядковий інтервал – 1;
    e. інтервал до та після рядків – 0;
    f. абзацний відступ – 0 см;
    g. вирівнювання тексту - по лівому краю.

4 КРІТЕРІЇ ОЦІНЮВАННЯ ПРОЄКТУ

Підсумкове оцінювання за виконання Проєкту, яке відображає рівень досягнення запланованих результатів навчання, проводиться за 100-бальною накопичувальною шкалою.

Підсумкова оцінка розраховується за формулою:

$$G = 0{,}5 \cdot G_{team} + 0{,}3 \cdot G_{report} + 0{,}2 \cdot G_{presentation}$$

де
G_{team} – середня оцінка членів команди, у т.ч. *Product Owner*,
G_{report} – оцінка за пояснювальну записку,
$G_{presentation}$ – оцінка за презентацію продукту.

Для оцінювання рівня з G_{team}, G_{report}, та $G_{presentation}$ використовується 12-бальна оцінна шкала з подальшим переведенням у 100-бальну шкалу:

Рівні оцінної шкали	Відмітки	Інтерпретація рівня досягнення результатів навчання курсу
IV високий	12	чудово
	11*	відмінно
	10	майже відмінно
III середній	9	більш ніж добре
	8*	добре
	7	майже добре
II достатній	6	більш ніж задовільно
	5*	задовільно
	4	майже задовільно
I початковий	3	мало задовільно
	2*	незадовільно
	1	знання майже відсутні

* – базова оцінка рівня оцінної шкали

5 ДОТРИМАННЯ ВИМОГ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ ПРИ ВИКОНАННІ ПРОЄКТУ

До початку виконання Проєкту необхідно уважно ознайомитися з Положенням про академічну доброчесність в Університеті КРОК [2], Положенням про перевірку академічних та наукових текстів на плагіат в університеті «КРОК» [3], Рекомендаціями МОН України щодо запобігання академічному плагіату та його виявлення в наукових роботах (авторефератах, дисертаціях, монографіях, наукових доповідях, статтях тощо) [9].

Основні види порушень академічної доброчесності, яких слід уникати під час виконання Проєкту, написання пояснювальної записки, підготовки презентації [2]:

плагіат – оприлюднення (опублікування), повністю або частково, чужого твору під іменем особи, яка не є автором цього твору (пункт в ст. 50 Закону України «Про авторське право і суміжні права»);

академічний плагіат – оприлюднення (частково або повністю) наукових (творчих) результатів, отриманих іншими особами, як результатів власного дослідження (творчості) та/або відтворення опублікованих текстів (оприлюднених творів мистецтва) інших авторів без зазначення авторства (ч. 4 ст. 42 Закону України «Про освіту»);

самоплагіат – оприлюднення (частково або повністю) власних раніше опублікованих наукових результатів як нових наукових результатів;

фабрикація – вигадування даних чи фактів, що використовуються в освітньому процесі або наукових дослідженнях;

фальсифікація – свідома зміна чи модифікація вже наявних даних, що стосуються освітнього процесу чи наукових досліджень.

При використанні фрагментів, запозичених з інших джерел, необхідно прийняти до уваги, що вони будуть *ідентифіковані як плагіат* за умови, якщо будуть відтворені у тексті роботи [9]:

1) без змін, з незначними змінами, або в перекладі, обсягом від речення і більше, без посилання на автора (авторів) відтвореного тексту;
2) повністю або частково, через перефразування чи довільний переказ без посилання на автора (авторів) відтвореного тексту;
3) як цитати з третіх джерел без вказування, за яким саме безпосереднім джерелом наведена цитата;
4) як наведена в іншому джерелі науково-технічна інформація (крім загальновідомої) без вказування на те, з якого джерела взята ця інформація;

5) як оприлюднені твори мистецтва без зазначення авторства цих творів мистецтва.

У контексті технічної (на текстові збіги) та експертної перевірки (керівниками) повного тексту роботи плагіатом буде вважатись наступне [3]:

1) видання виконаної іншим автором роботи за свою без внесення в неї жодних змін;
2) дослівне копіювання фрагментів тексту (від фрази до набору речень) без належного оформлення цитування;
3) внесення незначних правок у скопійований матеріал (переформулювання речень, зміна порядку слів у них тощо) та без належного оформлення цитування;
4) представлення суміші власних і запозичених аргументів без належного цитування;
5) *парафраза* – переказ своїми словами чужих думок, ідей або тексту; сутність парафрази полягає в заміні слів (знаків), фразеологічних зворотів або пропозицій при використанні будь-якої авторської наукової праці (збереженої на електронних або паперових носіях, у тому числі розміщеної в мережі Інтернет);
6) *компіляція* – створення значного масиву тексту без поглибленого вивчення проблеми шляхом копіювання тексту із низки джерел без внесення в нього правок, з посиланням на авторів та «маскуванням» шляхом написання перехідних речень між скопійованими частинами тексту.

Положенням про перевірку академічних та наукових текстів на плагіат в університеті «КРОК» передбачено виявлення рівня оригінальності тексту представленої роботи у відсотках, визначених за допомогою програмно-технічних засобів перевірки на текстові збіги. За результатами такої перевірки передбачені наступні дії:

1) при виявленні високого рівня оригінальності роботи (понад 75%) текст вважається оригінальним, додаткові дії щодо запобігання неправомірним запозиченням не проводитимуться;
2) при виявленні задовільного рівня оригінальності роботи (55-75%) вважається, що в тексті роботи наявні окремі ознаки академічного плагіату. Додатково електронний звіт з результатами антиплагіатної перевірки з гіперпосиланнями на запозичені джерела буде експертно перевірений (науковим керівником) на наявність посилань на першоджерела для цитованих фрагментів, в разі необхідності буде проведена повторна технічна перевірка;

3) при виявленні низького рівня оригінальності твору (35-55%) вважається, що в тексті роботи наявні певні ознаки академічного плагіату, проте робота може бути прийнята за умови доопрацювання з обов’язковою наступною перевіркою на оригінальність;
4) при виявленні рівня оригінальності роботи нижче за 35%, вважається, що в тексті роботи наявні істотні ознаки плагіату. Робота до розгляду (захисту) не приймається.

У разі виявлення академічного плагіату у Проєкті, він знімається із захисту без права повторного захисту.

Відсоток рівня оригінальності роботи може бути збільшений за рахунок правомірних запозичень, до яких належать:

1) власні назви (індивідуальні найменування окремих одиничних об’єктів, у т.ч. найменування установ, назви праць, які досліджувалися в роботі, бібліографічні посилання на джерела та ін.);
2) усталені словосполучення, що характерні для сфери ІТ, комп’ютерних наук;
3) належним чином оформлені цитування;
4) самоцитування (фрагменти текстів, що належать автору роботи, опубліковані або оприлюднені в електронній формі ним у інших творах).

Під час виконання Проєкту та написання пояснювальної записки, необхідно переконатися, що посилання у тексті пояснювальної записки зроблені правильно.

Посилання у тексті робиться при цитуванні джерела чи думки дослідника, при вказівці на певне важливе свідчення джерела, при запозиченні положень, використанні фактичного матеріалу, результатів досліджень інших авторів, посилань на досвід.

Кожну цитату обов'язково супроводжувати посиланням на джерело у тексті. Позначається таке посилання порядковим номером за переліком джерел посилання у клямрах (наприклад, «... у працях [1–7]...»).

Якщо використовуються відомості, матеріали з монографій, оглядових статей, інших джерел з великою кількістю сторінок, тоді у посиланні необхідно точно вказати номери сторінок, ілюстрацій, таблиць, формул з джерела, на які є посилання (наприклад, [3, с. 29]).

Якщо текст цитується не за першоджерелом, а за іншим виданням чи документом, то необхідно розпочинати посилання словами: «Цит. за:».

При непрямому цитуванні (переказі, викладенні думок інших авторів своїми словами), що дає значну економію тексту, треба бути гранично точними, коректними.

Необхідно уникати як надмірного, так і недостатнього цитування, оскільки це знижує рівень: надмірне цитування створює враження компілятивності праці, а недостатнє − знижує цінність викладеного матеріалу.

Якщо наводиться цитата і треба виділити у ній деякі слова, необхідно зробити спеціальний припис (після тексту, який пояснює виділення, треба поставити крапку, далі дефіс і вказати ініціали автора роботи, а весь текст припису помістити у дужки) (наприклад, (курсив наш – А.А.), (підкреслено мною – А.А.), (доповнено мною – А.А.)).

Бібліографічний опис використаних джерел оформлюйте відповідно до вимог ДСТУ 8302:2015 [5].

Відповідальність за перевірку Проєкту у встановлені терміни, прийняття рішення щодо доопрацювання та повторну перевірку на плагіат, про допуск до захисту несе Керівник Проєкту (Product Owner).

6 ТЕХНІЧНЕ ОБЛАДНАННЯ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Технічне обладнання: комп’ютер/ноутбук.

Програмне забезпечення:

1) Python 3.x (https://www.python.org/downloads/);
2) PyQt5 (https://pypi.org/project/PyQt5/);
3) IDE PyCharm (https://www.jetbrains.com/pycharm/);
4) VCS GIT (https://git-scm.com/).

7 ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Положення про організацію освітнього процесу в Університеті «КРОК». URL:https://library.krok.edu.ua/media/library/category/publichna-informatsiya/3-3_osvitnij-protses-2020-11-02.pdf
2. Положення про академічну доброчесність в Університеті «КРОК». URL: https://www.krok.edu.ua/download/nakazi/2018-10-18_kodeks-akademichnoi-dobrochesnosti.pdf
3. Положення про перевірку академічних та наукових текстів на плагіат в Університеті «КРОК» URL:https://library.krok.edu.ua/media/library/category/publichna-informatsiya/2020-04-10_polozhennya-pro-perevirku-tekstiv-na-plagiat.pdf
4. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання. URL:http://www.knmu.kharkov.ua/attachments/3659_3008-2015.pdf
5. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні вимоги та правила складання. URL:http://pdf.lib.vntu.edu.ua/books/2018/%D0%94%D0%A1%D0%A2%D0%A3%208302%20%D0%BF%D0%BE%D0%B2%D0%BD%D0%B8%D0%B9.pdf
6. Освітньо-професійна програма «Комп'ютерні науки» Університету «КРОК». URL:https://library.krok.edu.ua/media/library/category/litsenzuvannia-ta-akredytatsiia/osvitni-prohramy/komp-yuterni-nauki-bakalavr-2021-04-29.pdf
7. Стандарт вищої освіти першого (бакалаврського) рівня ступеня «бакалавр» за галуззю знань 12 «Інформаційні технології» спеціальністю 122 «Комп'ютерні науки». URL:https://mon.gov.ua/storage/app/media/vishcha-osvita/zatverdzeni%20standarty/2019/07/12/122-kompyuterni-nauki-bakalavr.pdf
8. Положення про політику та процедури врегулювання конфліктних ситуацій в Університеті «КРОК». URL:https://library.krok.edu.ua/media/library/category/publichna-informatsiya/2021-07-07_polozhennya-pro-protseduri-vregulyuvannya-konfliktnikh-situatsij.pdf
9. Рекомендації МОН України щодо запобігання академічному плагіату та його виявлення в наукових роботах (авторефератах, дисертаціях, монографіях, наукових доповідях, статтях тощо). URL:https://zakon.rada.gov.ua/rada/show/v8681729-18#Text
10. Burndown chart. URL: https://en.wikipedia.org/wiki/Burndown_chart
11. INTRO ON AGILE SCRUM. URL: https://agileevangelists.wordpress.com/2014/01/23/intro-on-agile-scrum/

ДОДАТОК А
Форма титульного аркушу пояснювальної записки Проєкту

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ПРОЄКТ ДРУГОГО РІВНЯ

Тема: «……»

Ступінь вищої освіти – бакалавр

Спеціальність – 122 «Комп’ютерні науки»

Освітня програма «Комп’ютерні науки»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав (-ла): здобувач (ка) 2 курсу

групи _____

Прізвище, ініціали

Керівник: *науковий ступінь, посада, ПІБ керівника*

м. Київ – 202_ рік