

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЗ „ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА”



Навчально-науковий інститут математики та інформаційних технологій
Кафедра математики та інформатики

О. О. Смагіна, В. Ю. Козуб, Г. О. Козуб

ТЕХНОЛОГІЯ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ
методичні вказівки до лабораторних робіт
для здобувачів вищої освіти першого (бакалаврського) рівня
спеціальності F3 – «Комп'ютерні науки»

Лубни - 2026

УДК 004.272:37

Рекомендовано до друку вченою радою
ДЗ «Луганський національний університет імені Тараса Шевченка»,
(протокол № 7 від 30.01.2026 р.)

Укладачі:

Смагіна О. О. – кандидат педагогічних наук доцент.

Козуб В. Ю. – доктор філософії з комп'ютерних наук;

Козуб Г. О. – кандидат технічних наук, доцент.

Рецензенти:

Гоменюк С. І. – доктор технічних наук, професор, декан математичного факультету
Запорізького національного університету;

Могильний Г. А. – кандидат технічних наук, доцент, директор навчально-наукового
інституту математики та інформаційних технологій ДЗ „Луганський національний
університет імені Тараса Шевченка”.

Смагіна О. О., Козуб В. Ю., Козуб Г. О. Технологія створення програмних
продуктів: методичні вказівки до лабораторних для здобувачів вищої освіти першого
бакалаврського рівня спеціальності F3 – «Комп'ютерні науки». Лубни: ДЗ „ЛНУ імені
Тараса Шевченка”, 2026. 27 с.

Методичні вказівки призначені для здобувачів вищої освіти першого
(бакалаврського) рівня, які навчаються за денною та заочною формами з метою розвитку та
закріплення практичних навичок під час підготовки та виконання лабораторних робіт.
Представлені основні відомості, практичні завдання та контрольні питання до
лабораторних робіт з освітнього компоненту «Технологія створення програмних
продуктів».

ПЕРЕДМОВА

Мета вивчення дисципліни: основною метою дисципліни «Технологія створення програмних продуктів» – це формування для здобувачів сучасних уявлень про методології та практики повного циклу, процесів розробки, тестування та впровадження в умовах вирішення реальних практичних завдань. Також до цілей освоєння дисципліни відносяться формування у здобувачів здібностей з реалізації оптимального підходу під час вибору моделі життєвого циклу розробки, організації командної роботи за гнучкими методологіями та забезпечення якості програмних продуктів.

В результаті освоєння курсу здобувач опанує: сучасні моделі життєвого та критерії їх вибору; принципи та практики Agile-методологій (Scrum, Kanban) та їх застосування в командній розробці; основи DevOps-культури та побудови CI/CD pipeline збору, й аналізу та специфікацію вимог до UX-дизайну; технології управління проєктами в ІТ-сфері та роботи з стейкхолдерами; стратегії та методи тестування програмного забезпечення; практики управління версіями коду та організації code review; методи моніторингу, аналітики та оптимізації програмних продуктів; сучасні інструментальні засоби розробки та автоматизації процесів.

Вміти: аналізувати вимоги до програмного продукту та обирати відповідну модель SDLC; організовувати роботу команди за Agile-методологіями та проводити Scrum-церемонії; налаштовувати CI/CD pipeline для автоматизації збірки, тестування та деплою; збирати та документувати функціональні та нефункціональні вимоги від стейкхолдерів; створювати wireframe, прототипи користувацьких інтерфейсів та проводити юзабіліті-тестування; планувати спринти, управляти product backlog та проводити ретроспективи; розробляти стратегію тестування та створювати автоматизовані тести; працювати з системами контролю версій Git та організовувати ефективний code review; налаштовувати системи моніторингу

та аналізувати метрики продуктивності продукту; створювати технічну документацію, API-специфікації та user documentation.

Володіти: навичками роботи з (Azure DevOps, Asana); методиками проведення Sprint Planning, Daily Standup, Sprint Review та Retrospective; техніками налаштування Jenkins, GitLab CI/CD, GitHub Actions для автоматизації; практиками написання User Stories, Use Cases та створення SRS-документації; інструментами прототипування та дизайну (Figma, Adobe XD, Sketch); засобами управління ресурсами, часом, ризиками та змінами в проєктах; методами автоматизованого тестування (unit, integration, E2E) та використання фреймворків; техніками Git Flow, GitHub Flow, Trunk-Based Development та проведення code review; інструментами моніторингу та аналітики (Prometheus, Grafana, Google Analytics, Mixpanel); навичками створення комплексної проєктної, технічної та користувацької документації.

ЗМІСТ

ТЕМА 1. СУЧАСНИЙ ЖИТТЄВИЙ ЦИКЛ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (SDLC).....	6
ТЕМА 2. AGILE-МЕТОДОЛОГІЇ ТА ГНУЧКА РОЗРОБКА.....	8
ТЕМА 3. DEVOPS ТА CI/CD: АВТОМАТИЗАЦІЯ ПРОЦЕСІВ РОЗРОБКИ..	10
ТЕМА 4. LOW-CODE/NO-CODE ПЛАТФОРМИ ТА ШВИДКА РОЗРОБКА.	12
ТЕМА 5. ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ТА ТЕСТУВАННЯ ПЗ ЧАСТИНА 1....	14
ТЕМА 5. ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ТА ТЕСТУВАННЯ ПЗ ЧАСТИНА 2....	16
ТЕМА 6. УПРАВЛІННЯ ВЕРСІЯМИ ТА КОЛАБОРАЦІЯ В КОМАНДІ ЧАСТИНА 1.....	19
ТЕМА 6. УПРАВЛІННЯ ВЕРСІЯМИ ТА КОЛАБОРАЦІЯ В КОМАНДІ ЧАСТИНА 2.....	21
ТЕМА 7. АНАЛІТИКА, МОНІТОРИНГ ТА ОПТИМІЗАЦІЯ ПРОГРАМНИХ ПРОДУКТІВ ЧАСТИНА 1.....	23
ТЕМА 7. АНАЛІТИКА, МОНІТОРИНГ ТА ОПТИМІЗАЦІЯ ПРОГРАМНИХ ПРОДУКТІВ ЧАСТИНА 2.....	26

ТЕМА 1. СУЧАСНИЙ ЖИТТЄВИЙ ЦИКЛ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ (SDLC)

Мета роботи: набути практичних навичок аналізу вимог до програмного проєкту, вибору оптимальної моделі SDLC до специфіки проєкту, освоїти методики планування фаз розробки, навчитися ідентифікувати ключові етапи SDLC та їх взаємозв'язки, оволодіти практичними навичками оцінювання ефективності різних моделей SDLC для конкретних типів проєктів та розробити детальний план реалізації програмного продукту з урахуванням обраної моделі.

Практичне завдання

1. Необхідно провести аналіз та порівняння різних моделей створення додатку онлайн-банкінгу. Додаток має включати функціонал перегляду балансу, переказів між рахунками, оплати послуг, історії транзакцій.

2. Розглянути та детально описати чотири основні моделі SDLC: Waterfall (каскадна модель), Iterative (ітеративна модель), Spiral (спіральна модель) та Agile (гнучка модель). Для кожної моделі необхідно визначити основні фази розробки, їх послідовність та тривалість, встановити точки й ризики та обмеження кожної моделі, а також оцінити придатність для проєкту онлайн-банкінгу з обґрунтуванням переваг та недоліків.

3. Для обраної моделі SDLC створити детальний план реалізації проєкту мобільного додатку онлайн-банкінгу, який повинен включати декомпозицію всіх фаз життєвого циклу (планування, аналіз вимог, проєктування, розробка, тестування, впровадження, підтримка). Необхідно визначити тривалість кожної фази, встановити контрольні точки (milestones), визначити команду проєкту та розподіл ролей, ідентифікувати можливі ризики та стратегії їх мінімізації, а також описати критерії успішного завершення кожної фази.

Контрольні питання

1. Опишіть розуміння сутності SDLC його значення для успішної реалізації проєктів, зокрема необхідно пояснити призначення кожної з семи основних фаз SDLC (а саме планування) навести приклади конкретних

активностей та артефактів, що створюються на кожній фазі, а також описати ролі та відповідальність учасників команди на різних етапах життєвого циклу.

2. Поясніть відмінності між класичними та сучасними моделями SDLC, де необхідно розкрити переваги та недоліки каскадної моделі (Waterfall) порівняно з гнучкими методологіями (Agile), пояснити в яких ситуаціях доцільно використовувати ітеративну або спіральну модель, описати як обирається оптимальна модель SDLC залежно від розміру проєкту, складності системи, рівня визначеності вимог та бюджетних обмежень, а також обґрунтувати чому в сучасній розробці спостерігається тенденція до гібридних підходів.

3. Потрібно описати практичне застосування метрик оцінювання ефективності SDLC, які ключові показники ефективності (KPI) використовуються для моніторингу прогресу, розкрити як результати аналізу ефективності SDLC використовуються для оптимізації процесів розробки в наступних проєктах та формування best practices в організації.

Література

1. Juan de Vicente Mohino, Javier Bermejo Higuera, Juan Ramón Bermejo Higuera, Juan Antonio Sicilia Montalvo. The Application of a New Secure Software Development Life Cycle (S-SDLC) with Agile Methodologies. 2019. URL: <https://www.mdpi.com/2079-9292/8/11/1218>
2. Antonios Saravanos, Matthew X. Curinga. Simulating the Software Development Lifecycle: The Waterfall Model. 2023. URL: <https://www.mdpi.com/2571-5577/6/6/108>

ТЕМА 2. AGILE-МЕТОДОЛОГІЇ ТА ГНУЧКА РОЗРОБКА

Мета роботи: набути практичних навичок організації роботи команди за Agile-методологіями, освоїти принципи та практики Scrum та Kanban, навчитися планувати та проводити основні Scrum-церемонії оволодіти методиками управління product backlog та sprint backlog, розробити навички ефективної комунікації в гнучких командах та навчитися використовувати інструменти для управління Agile-проєктами

Практичне завдання

1. Необхідно організувати роботу команди розробки веб-платформи для онлайн-навчання за методологією Scrum. Проєкт включає розробку функціоналу для здобувачів (перегляд курсів, проходження тестів, відстеження прогресу), викладачів (створення курсів, перевірка завдань, аналітика) та адміністраторів (управління користувачами, модерація контенту). Команда складається з Product Owner, Scrum Master, 4 розробників. Необхідно сформулювати початковий product backlog з 5-10 user stories, визначити пріоритети за методом та спланувати перші три спринти тривалістю 3 дні кожен.

2. Для кожної user story необхідно створити детальний опис за шаблоном "Як [роль], я хочу [функціонал], щоб [бізнес-цінність]", визначити критерії прийняття (Acceptance Criteria) у форматі Given-When-Then, оцінити складність за шкалою Story Points (використовуючи послідовність: 1, 2, 3, 4, 5), встановити залежності між user stories та визначити Definition of Done для команди. Також потрібно розбити складні user stories на менші завдання (tasks) та розподілити їх між членами команди враховуючи їхні компетенції та завантаження.

3. Розробити детальний план проведення всіх Scrum-церемоній для одного спринту, включаючи Sprint Planning (де визначається Sprint Goal, обирається набір user stories для спринту, проводиться декомпозиція на задачі), Daily Standup (формат проведення, три ключові питання, тривалість та правила), Sprint Review (демонстрація готового функціоналу стейкхолдерам,

збір фідбеку) та Sprint Retrospective (аналіз того, що пройшло добре, що потребує покращення, action items на наступний спринт). Для кожної церемонії визначити мету, учасників, тривалість, порядок денний та очікувані результати.

Контрольні запитання

1. Опишіть розуміння принципів Agile Manifesto та їх практичного застосування в сучасній розробці програмного забезпечення, зокрема необхідно пояснити умови контракту.

2. Поясніть ролі в Scrum-команді та їхню відповідальність, де необхідно розкрити функції Product Owner (управління product backlog, визначення пріоритетів, комунікація зі стейкхолдерами, прийняття рішень про готовність функціоналу), обов'язки Scrum Master (фасилітація церемоній, усунення перешкод, навчання команди Scrum-практикам, захист команди від зовнішніх втручань).

3. Пояснити роль Development Team (самоорганізація, кросфункціональність, колективна відповідальність за результат), пояснити чому в Scrum не існує ролі проєкт-менеджера в класичному розумінні, а також обґрунтувати важливість чіткого розподілу відповідальності між ролями для успішної роботи команди.

Література

1. Md Athikul Islam, Rizbanul Hasan, Nasir U. Eisty. Documentation Practices in Agile Software Development: A Systematic Literature Review. 2023. URL: <https://arxiv.org/abs/2304.07482>
2. Fernando Almeida. Bibliometric Analysis of Agile Software Development. 2020. URL: <https://arxiv.org/abs/2004.05876>

ТЕМА 3. DEVOPS ТА CI/CD: АВТОМАТИЗАЦІЯ ПРОЦЕСІВ РОЗРОБКИ

Мета роботи: набути практичних навичок побудови CI/CD pipeline для автоматизації процесів збірки, тестування та деплою програмного забезпечення, освоїти принципи DevOps-культури та їх практичне застосування, навчитися налаштовувати інструменти continuous integration та continuous deployment, оволодіти методиками контейнеризації додатків за допомогою Docker, розробити навички автоматизації інфраструктури через Infrastructure as Code (IaC) та навчитися інтегрувати практики DevOps в процес розробки для підвищення швидкості та якості випуску програмних продуктів.

Практичне завдання

1. Спроекувати архітектуру CI/CD pipeline з детальним описом всіх етапів: Source Stage (тригер при push/merge в Git, webhook налаштування), Test Stage (unit тести з coverage звітом, integration тести, Security Stage (SAST аналіз коду, сканування залежностей на вразливості, Docker image security scanning), Package Stage (створення Docker images, тегування версій, публікація в Container Registry) та Deploy Stage (автоматичний деплой в staging, manual approval для production, blue-green або canary deployment стратегії).

2. Документація повинна включати архітектурну діаграму CI/CD pipeline з візуалізацією всіх етапів та їх взаємозв'язків, у вигляді порівняльної таблиці CI/CD інструментів (GitHub Actions, GitLab CI/CD, Jenkins, CircleCI) з оцінкою за критеріями (простота налаштування, вартість, масштабованість, екосистема плагінів).

Контрольні запитання

1. Опишіть розуміння філософії DevOps та її ключових принципів, зокрема необхідно пояснити акронім CAMS (Culture - культура співпраці між dev та ops, Automation - автоматизація повторюваних процесів, Measurement - вимірювання метрик та KPI, Sharing - обмін знаннями та досвідом), навести приклади як DevOps-практики впливають на швидкість доставки програмного

забезпечення та його якість, описати традиційні проблеми взаємодії між командами розробки та експлуатації, які вирішує DevOps, а також розкрити як DevOps-культура змінює організаційну структуру компаній та процеси прийняття рішень.

2. Поясніть різницю між Continuous Integration, Continuous Delivery та Continuous Deployment, де необхідно розкрити концепцію CI (регулярна інтеграція коду в спільний репозиторій, автоматична збірка та тестування при кожному коміті, швидке виявлення проблем інтеграції), визначити CD Delivery (автоматизація процесу до стадії готовності до релізу, manual approval для production деплою, можливість випуску в будь-який момент) та CD Deployment (повна автоматизація включаючи деплой у production без ручного втручання, кожен пройдений commit автоматично потрапляє до користувачів), пояснити переваги та ризики кожного підходу, описати як обирається рівень автоматизації залежно від типу продукту та організаційної зрілості, а також навести приклади компаній що практикують повний CD Deployment.

3. Потрібно описати роль контейнеризації в DevOps-практиках, які проблеми вирішує Docker ("працює на моїй машині", конфлікти залежностей, різниця між середовищами), пояснити архітектуру Docker (images, containers, volumes, networks, registry), розкрити концепцію та її переваги (версіонування інфраструктури, відтворюваність середовищ, code review для змін інфраструктури).

Література

1. Mario Ccallo, Alex Quispe-Quispe. Adoption and Adaptation of CI/CD Practices in Very Small Software Development Entities: A Systematic Literature Review. 2024. URL: <https://arxiv.org/abs/2410.00623>
2. Sowmiya Dhandapani. Enhancing Software Supply Chain Security Through STRIDE-Based Threat Modelling of CI/CD Pipelines. 2025. URL: <https://arxiv.org/abs/2506.06478>

ТЕМА 4. LOW-CODE/NO-CODE ПЛАТФОРМИ ТА ШВИДКА РОЗРОБКА

Мета роботи: набути практичних навичок створення програмних додатків за допомогою Low-Code/No-Code платформ, освоїти принципи швидкої розробки та прототипування бізнес-додатків без написання коду або з мінімальним кодуванням, навчитися обирати відповідну LCNC-платформу залежно від вимог проєкту, оволодіти методиками інтеграції Low-Code рішень з традиційними системами, розробити навички оцінювання переваг та обмежень LCNC-підходів та навчитися визначати випадки коли доцільно використовувати Low-Code розробку замість традиційного програмування.

Практичне завдання

1. Необхідно розробити повнофункціональний додаток для управління замовленнями та інвентаризацією в ресторані за допомогою обраної Low-Code платформи (Microsoft Power Apps, OutSystems, Mendix, Bubble або AppSheet). Додаток має включати функціонал для офіціантів (прийом замовлень за столиками, передача замовлень на кухню, обробка оплат), кухарів (перегляд активних замовлень, відмітка про готовність страв, управління чергою приготування), менеджерів (аналітика продажів, звітність, управління меню та цінами) та адміністраторів складу (облік інгредієнтів, автоматичні сповіщення про критичні залишки, замовлення у постачальників).
2. Спроекувати архітектуру додатку з описом моделі даних (таблиці: Menu Items, Orders, Order Items, Ingredients, Inventory, Users, Payments) з визначенням полів, типів даних та зв'язків між сутностями, дизайну користувацьких інтерфейсів для кожної ролі з врахуванням специфіки використання (мобільні пристрої для офіціантів, планшети на кухні, веб-інтерфейс для менеджерів), бізнес-логіки та workflow (автоматичне списання інгредієнтів при створенні замовлення, розрахунок загальної вартості з врахуванням знижок, генерація alerts при критичних залишках).

3. Провести порівняльний аналіз Low-Code платформ зі створенням таблиці оцінювання мінімум 4 популярних платформ (Microsoft Power Platform, OutSystems, Mendix, Bubble).

Контрольні питання

1. Опишіть розуміння концепції Low-Code/No-Code розробки та її місця в сучасному landscape розробки програмного забезпечення, зокрема необхідно пояснити різницю між Low-Code (платформи що дозволяють створювати додатки переважно візуально але з можливістю додавання custom коду для складних сценаріїв) та No-Code (повністю візуальна розробка без написання коду, орієнтована на business users), навести історичну еволюцію від 4GL мов та RAD-інструментів до сучасних LCNC платформ, описати рушійні сили популярності LCNC (дефіцит розробників, потреба в швидкій цифровізації бізнесу, демократизація розробки), а також пояснити концепцію Citizen Developers та їх роль в сучасних організаціях.
2. Поясніть архітектуру та основні компоненти типової Low-Code платформи, де необхідно розкрити візуальний IDE з drag-and-drop інтерфейсом для проєктування UI та логіки, систему управління даними з візуальним data modeling та автоматичною генерацією CRUD операцій, workflow engine для автоматизації бізнес-процесів та orchestration.
3. Що таке pre-built компоненти та templates для прискорення розробки, систему управління користувачами та безпекою з налаштуванням ролей, integration layer з connectors до популярних систем та API management.

Література

1. Vandana Verma, Supriya Shree. Low-Code/No-Code Platforms: From Concept to Creation. 2025. URL: <https://is.gd/5RVRwq>
2. Giorgia Masili. No-code development platforms: breaking the boundaries between IT and business experts. 2023. URL: <https://is.gd/a8piCg>

ТЕМА 5. ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ТА ТЕСТУВАННЯ ПЗ ЧАСТИНА 1

Мета роботи: набути практичних навичок розробки стратегії тестування програмного забезпечення, освоїти різні рівні та типи тестування (unit, integration, system, acceptance), навчитися створювати тест-кейси та тест-плани відповідно до вимог проєкту, оволодіти методиками Test Pyramid підходу для оптимального розподілу тестів, розробити навички написання unit тестів з використанням сучасних фреймворків, освоїти практики mocking та stubbing для ізоляції компонентів під час тестування.

Практичне завдання

1. Необхідно розробити комплексну стратегію тестування для веб-платформи соціальної мережі для початківців, яка враховує різні платформи web responsive, навантаження з п'ятьма одночасними користувачами, вимоги до безпеки мінімальні.
2. Створити детальний Test Plan документ який включає розділ Introduction з описом мети тестування scope objectives test approach, розділ Test Items де перераховуються features що будуть тестуватися user registration та authentication profile
3. Реалізувати unit тести з правильним використанням test doubles.

Контрольні питання

1. Опишіть різницю між Quality Assurance та Quality Control пояснити що QA це proactive process фокусований на prevention дефектів через встановлення правильних processes standards guidelines training code reviews, а QC це reactive process фокусований на detection дефектів через testing inspections reviews після того як code написаний, навести приклади QA активностей такі як налагодження парного програмування (pair programs), виконання статичного аналізу коду, налаштування автоматизованих систем тестування, організація навчальних сесій для покращення командних процесів через ретроспективи.
2. Описати QC активності як manual testing exploratory testing automated test execution regression testing user acceptance testing bug tracking та reporting,

пояснити як QA та QC доповнюють один одного в забезпеченні якості продукту де QA забезпечує що команда слідує правильним практикам а QC виявляє дефекти що все таки проникли, розкрити роль QA engineer в agile команді з participation в sprint planning writing acceptance criteria reviewing user stories participating в daily standups sprint reviews retrospectives developing automated tests collaborating з developers на code reviews.

3. Поясніть концепцію Test Pyramid та її важливість для ефективної стратегії тестування, де необхідно розкрити структуру піраміди з широкою основою Unit Tests, описати техніки ізоляції dependencies в unit testing через test doubles з детальним поясненням типів та їх застосування, які відмінності між Dummy objects що передаються але ніколи не використовуються просто для filling parameter lists.

Література

1. Gokul Pandey, Vigneshwaran Jagadeesan Pugazhenth, Aravindhana Murugan. Advances in Software Testing in 2024: Experimental Insights, Frameworks, and Future Directions. 2024. URL: <https://is.gd/WslTfv>
2. Alex Escalante-Viteri, David Mauricio. Artificial Intelligence in Software Testing: A Systematic Review of a Decade of Evolution and Taxonomy. 2025. URL: <https://www.mdpi.com/1999-4893/18/11/717>

ТЕМА 5. ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ТА ТЕСТУВАННЯ ПЗ ЧАСТИНА 2

Мета роботи: набути практичних навичок розробки integration та end to end тестів для комплексної перевірки системи, освоїти методики Test Driven Development (TDD) та Behavior Driven Development (BDD) підходів, навчитися автоматизувати тестування з інтеграцією в CI/CD pipeline, оволодіти техніками performance testing та security testing, розробити навички аналізу метрик якості коду та test coverage, освоїти інструменти для continuous testing та test reporting.

Практичне завдання

1. Розробити Integration Tests двадцять відсотків test suite для тестування взаємодії між компонентами модулями сервісами платформи соціальної мережі з перевіркою API endpoints через real HTTP requests, Створити integration тести для user registration flow, перевірку що user record створений в database, user account status змінюється на active в database.

2. Налаштувати Continuous Testing в CI/CD pipeline з automated test execution при кожному commit push pull request з використанням локальної бази даних. Pipeline configuration включає stages для різних рівнів unit: unit tests stage що запускається вперше з parallel execution для швидкодії використовуючи test sharding splitting.

3. Застосувати Test Driven Development практику для implementation нового feature recommendations engine. TDD цикл починається з написання failing test RED phase що приймає userID та повертає array з suggested users sorted by relevance score: написати test який визиває getConnectionSuggestions з test user ID.

Контрольні питання

1. Поясніть відмінності між різними рівнями тестування integration system acceptance та їх роль в тестувальній стратегії, де необхідно розкрити integration testing як verification що різні сервіси компонентів модулів працюють правильно разом з фокусом на інтерфейсах взаємодії потоку даних між компонентами, описати підходи до інтеграційного тестування такі як big bang integration де всі компоненти інтегровані одночасно, тестування всієї системи одночасно, incremental integration де components додаються та тестуються поступово top(верх) або down(вниз) з незавершеними функціями або bottom up з інтеграційними функціями для higher components, sandwich integration combining top down та bottom up.

2. Пояснити system testing як testing complete integrated system as whole для verify що it meets specified requirements з types functional testing для verify features work correctly, non functional testing для performance security usability

compatibility, описати acceptance testing як validation що system meets business requirements та готова для доставки з User Acceptance (логи) та Testing UAT performed by end users (елемент з бази даних) або business stakeholders в production like environment, Alpha testing performed internally before release, Beta testing з limited external users providing feedback, навести критерії для successful acceptance testing з acceptance criteria met, business goals achieved, stakeholder approval obtained.

3. Опишіть практики Test Driven Development та Behavior Driven Development з їх принципами перевагами та викликами, які етапи TDD cycle включають RED write failing test що описує бажану поведінку перед implementation, GREEN write minimum code to make test pass без worry про quality або optimization, REFACTOR improve code structure readability performance while keeping tests green, пояснити переваги TDD такі як better design через thinking about interfaces та usage перед implementation forces modular loosely coupled code.

Література

1. Gokul Pandey, Vigneshwaran Jagadeesan Pugazhenthir, Aravindhana Murugan. Advances in Software Testing in 2024: Experimental Insights, Frameworks, and Future Directions. 2024. URL: <https://is.gd/WslTfv>
2. Alex Escalante-Viteri, David Mauricio. Artificial Intelligence in Software Testing: A Systematic Review of a Decade of Evolution and Taxonomy. 2025. URL: <https://www.mdpi.com/1999-4893/18/11/717>

ТЕМА 6. УПРАВЛІННЯ ВЕРСІЯМИ ТА КОЛАБОРАЦІЯ В КОМАНДІ

ЧАСТИНА 1

Мета роботи: набути практичних навичок роботи з системою контролю версій Git для управління кодом проєкту, освоїти основні команди Git для створення commits, гілок та злиття змін, навчитися організовувати структуру репозиторію та використовувати різні branching стратегії, оволодіти методиками вирішення конфліктів при злитті коду, розробити навички роботи з GitHub або GitLab для командної співпраці, опанувати техніки написання зрозумілих commit повідомлень та створення Pull Requests.

Практичне завдання

1. Необхідно створити Git репозиторій для проєкту веб-додатку онлайн бібліотеки, який включає frontend частину з компонентами для catalog book details user profile shopping cart, backend API на Node.js Express з endpoints для books users orders reviews, базу даних PostgreSQL (або аналоги) з таблицями books users orders reviews. Потрібно обрати підхід до організації коду: один великий репозиторій monorepo де frontend backend та scripts в одній папці з перевагами простіше координувати зміни спільні бібліотеки легше refactoring, або окремі репозиторії де frontend та backend в різних repo з перевагами незалежне версіонування кожен сервіс розгортається окремо.

2. Створити структуру папок для monorepo підходу з frontend папкою що містить src components pages styles, backend папкою що містить routes controllers models middleware, shared папкою з common utilities constants types, docs папкою з документацією README API описом, scripts папкою з автоматизацією build deploy test скриптами. Ініціалізувати Git репозиторій командою git init в корені проєкту, створити файл .gitignore щоб ігнорувати node_modules папку dist build папки .env файл з паролями logs папку coverage звіти IDE файли .vscode .idea, створити README.md файл з описом проєкту як встановити як запустити як користуватись.

Контрольні питання

1. Опишіть що таке система контролю версій Git та навіщо вона потрібна в розробці, зокрема необхідно пояснити основні проблеми які Git вирішує такі як збереження історії всіх змін в коді можливість повернутися до попередньої версії, робота декількох розробників над одним проєктом без перезапису змін один одного, можливість експериментувати з новими features в окремих гілках без ризику зламати робочий код, backup коду на віддалених серверах GitHub GitLab, navigation по історії змін хто що коли змінив та чому.

2. Пояснити базові концепції Git такі як repository сховище де зберігається код та вся історія змін, commit збережений знімок стану проєкту в певний момент часу з описом що змінилось, branch гілка паралельна версія коду для розробки окремих features, merge злиття змін з однієї гілки в іншу, remote віддалений репозиторій на сервері для синхронізації коду між розробниками. Описати базовий workflow роботи з Git: клонування репозиторію git clone, створення гілки git branch, внесення змін в файли, додавання змін git add, створення commit git commit, відправка на сервер git push, отримання змін від інших git pull.

3. Пояснити роботу над features з серією commits де кожен commit логічна одиниця змін з clear message, pushing branch на remote server git push origin feature-name, створення Pull Request з заповненням title description пояснення що зроблено. Які issues fixes, code review process де teammates переглядають код залишають comments suggestions approval або request changes.

Література

1. Shane McIntosh, Yasutaka Kamei, Bram Adams, Ahmed E. Hassan. An empirical study of the impact of modern code review practices on software quality. 2015. URL: <https://is.gd/Z7N7V9>

2. Siva Karthik Devineni. Version Control Systems (VCS) the Pillars of Modern Software Development: Analyzing the Past, Present, and Anticipating Future Trends. 2020. URL: <https://is.gd/adYTX8>

ТЕМА 6. УПРАВЛІННЯ ВЕРСІЯМИ ТА КОЛАБОРАЦІЯ В КОМАНДІ

ЧАСТИНА 2

Мета роботи: набути практичних навичок проведення code review та роботи з Pull Requests, освоїти техніки вирішення merge конфліктів при одночасній роботі над одними файлами, навчитися використовувати Git для командної колаборації з best practices, оволодіти методиками організації документації проєкту через README contributing guides, розробити навички налаштування CI/CD інтеграції з Git workflow, освоїти стратегії versioning та tagging releases.

Практичне завдання

1. Необхідно організувати process code review для команди розробки веб-додатку онлайн бібліотеки з встановленням правил та guidelines для reviewing коду. Створити Pull Request template файл .github/pull_request_template.md з секціями: Description що змінено та навіщо, Type of change feature bug fix refactor documentation.

2. Checklist з пунктами код протестований locally, tests написані та passing, documentation оновлена якщо потрібно, no console logs або debug code, code follows style guide, breaking changes documented. Встановити branch protection rules на main та develop гілки в GitHub settings з вимогами: require pull request перед merging не можна push directly, require code review мінімум один approval від teammate, require status checks CI tests мають пройти перед merge, require branches be up to date з base branch перед merge, restrict who can push тільки maintainers або leads.

3. Створити code review checklist документ з питаннями для reviewer: чи код робить те що описано в PR, чи логіка зрозуміла та maintainable, чи є potential bugs або edge cases не covered, чи tests adequate та cover main scenarios, чи performance considerations addressed, чи security vulnerabilities potential, чи naming conventions followed, чи comments adequate не надмірні.

Контрольні питання

1. Опишіть process та best practices для code review в командній розробці, зокрема необхідно пояснити мету code review такі як catching bugs та potential issues перед merge в основному коді, де reviewers вчать з коду інших розробників, ensuring consistency з coding standards та architectural patterns, через constructive feedback. Описати що reviewer має перевіряти: чи код робить що має робити без помилок, чи код зрозумілий легко читається, simplicity чи немає комплексних проблем у тестуванні, чи всі методи тестування покривають основну роботу коду. Чи немає vulnerabilities по типу: «SQL injection» «XSS authentication issues».

2. Пояснити how to give good feedback: be respectful та constructive фокусуйтеся на код не на person, be specific вказуйте exact lines та що можна покращити з suggestions, explain why ваші suggestions покращать код, ask questions замість demands encouraging discussion, praise good code також визнавайте хорошу роботу не тільки критикуйте. Описати how to receive feedback: don't take it personally це про код не про вас, ask for clarification якщо не розумієте comment, be open to learning можливо є кращі підходи які ви не знали, push back respectfully якщо disagree з обґрунтуванням вашого approach.

3. Потрібно описати документування проєкту через README contributing guides та best practices для project documentation, які секції мають бути в good README: project title та short description що робить додаток, badges для build status test coverage version license, table of contents для navigation великих README, installation instructions покрокові команди.

Література

1. Shane McIntosh, Yasutaka Kamei, Bram Adams, Ahmed E. Hassan. An empirical study of the impact of modern code review practices on software quality. 2015. URL: <https://is.gd/Z7N7V9>

2. Siva Karthik Devineni. Version Control Systems (VCS) the Pillars of Modern Software Development: Analyzing the Past, Present, and Anticipating Future Trends. 2020. URL: <https://is.gd/adYTX8>

ТЕМА 7. АНАЛІТИКА, МОНІТОРИНГ ТА ОПТИМІЗАЦІЯ ПРОГРАМНИХ ПРОДУКТІВ ЧАСТИНА 1

Мета роботи: набути практичних навичок налаштування системи моніторингу для веб-додатків, освоїти методи збору та аналізу логів помилок та подій користувачів, навчитися створювати дашборди для відстеження ключових метрик продуктивності додатку, оволодіти техніками виявлення проблем через моніторинг часу відгуку сервера кількості помилок завантаження процесора, розробити навички налаштування сповіщень про критичні проблеми через електронну пошту або месенджери, освоїти базові інструменти для моніторингу стану серверів та додатків.

Практичне завдання

1. Необхідно налаштувати систему моніторингу для інтернет-магазину електроніки, який складається з головної сторінки з каталогом товарів, сторінки з детальною інформацією про товар, кошика для покупок, сторінки оформлення замовлення з формою даних клієнта та вибором доставки, сторінки оплати з інтеграцією платіжної системи, особистого кабінету користувача з історією замовлень.

2. Встановити та налаштувати систему збору логів для запису всіх важливих подій в додатку. Логи мають записувати інформацію про запити користувачів який користувач до якої сторінки звернувся в який час, помилки які виникають під час роботи з описом що пішло не так де саме виникла помилка який код помилки, дії користувачів додавання товару в кошик оформлення замовлення успішна оплата, роботу бази даних які запити виконуються скільки часу займають чи є повільні запити, стан сервера використання ресурсів процесора оперативної пам'яті вільного місця на диску.

3. Налаштувати рівні логування для різних типів подій: рівень помилка для критичних проблем коли щось зламалось і не працює, рівень попередження для підозрілих ситуацій які можуть стати проблемою наприклад повільний запит до бази, рівень інформація для звичайних подій користувач зайшов на сайт переглянув товар, рівень налагодження для

детальної технічної інформації потрібної програмістам під час розробки. Створити структуру для збереження логів з окремими файлами для кожного дня або тижня щоб легше знаходити потрібну інформацію, автоматичним видаленням старих логів через місяць щоб не займали багато місця, форматуванням логів щоб легко читались з міткою часу рівнем подією описом.

Контрольні запитання

1. Опишіть що таке моніторинг додатків та чому він важливий для успішної роботи сервісу, зокрема необхідно пояснити основні цілі моніторингу такі як виявлення проблем до того як користувачі їх помітять наприклад повільна робота або помилки, розуміння як користувачі взаємодіють з додатком які функції використовують де виникають труднощі, відстеження продуктивності чи сайт працює швидко чи сервери справляються з навантаженням, планування ресурсів коли потрібно додавати більше серверів або збільшувати потужність, прийняття рішень на основі даних які покращення принесуть найбільшу користь.

2. Пояснити різницю між моніторингом та логуванням де логування це запис подій що сталися з детальною інформацією для розслідування проблем після їх виникнення, а моніторинг це активне відстеження поточного стану системи в реальному часі з метою раннього виявлення відхилень від норми.

3. Описати основні типи метрик для моніторингу: метрики продуктивності час відгуку сервера швидкість завантаження сторінок пропускна здатність, метрики доступності чи сайт працює відсоток часу доступності кількість збоїв, метрики використання ресурсів завантаження процесора використання пам'яті помилок, кількість та типи помилок які виникають де саме в коді, бізнес метрики кількість користувачів замовлень.

Література

1. Jack Marquez, Oscar H. Mondragon. Modelling the Impact of Cloud Storage Heterogeneity on HPC Application Performance. 2024. URL: <https://is.gd/KYvovK>

2. Nane Kratzke. Cloud-Native Observability: The Many-Faceted Benefits of Structured and Unified Logging—A Multi-Case Study. 2022. URL: <https://www.mdpi.com/1999-5903/14/10/274>

ТЕМА 7. АНАЛІТИКА, МОНІТОРИНГ ТА ОПТИМІЗАЦІЯ ПРОГРАМНИХ ПРОДУКТІВ ЧАСТИНА 2

Мета роботи: набути практичних навичок аналізу поведінки користувачів через систему аналітики, освоїти методи виявлення вузьких місць у продуктивності додатку та їх усунення, навчитися проводити тестування навантаження для перевірки роботи під великою кількістю користувачів, оволодіти техніками оптимізації швидкості завантаження сторінок та зменшення часу відгуку сервера, розробити навички створення звітів про стан системи та рекомендацій для покращення, освоїти базові методи масштабування додатків при зростанні навантаження.

Практичне завдання

1. Необхідно налаштувати систему аналітики користувацької поведінки для інтернет-магазину щоб розуміти як відвідувачі використовують сайт та де виникають проблеми. Встановити систему відстеження Google Analytics або інший аналогічний сервіс з налаштуванням відстеження «яким шляхом користувачі рухаються по сайту», подій натискання на кнопки додавання товару в кошик початок оформлення замовлення, конверсій скільки відвідувачів завершують покупку співвідношення відвідувань до продажів. Виокремити та описати джерела трафіку та введення адреси реклами.

2. Створити воронку продажів для аналізу етапів покупки з кроками: відвідування головної сторінки, перегляд каталогу товарів, відкриття сторінки конкретного товару, додавання товару в кошик, перехід до оформлення замовлення, заповнення форми з даними доставки, вибір способу оплати, завершення оплати.

3. Для кожного кроку відстежувати скільки користувачів доходить скільки відсіюється на якому кроці відсів наймасштабніший, що може бути причиною складна форма незрозумілий інтерфейс технічні помилки високі ціни доставки.

Контрольні запитання

1. Опишіть що таке аналітика користувацької поведінки та як вона допомагає покращити продукт, зокрема необхідно пояснити основні метрики аналітики веб-трафіку кількість відвідувачів скільки людей заходить на сайт за день унікальних та загальних відвідувань, перегляди сторінок скільки сторінок переглядає в середньому один користувач, час на сайті скільки часу проводять користувачі, зробити ключовий показник проблем у вигляді метрики.

2. Пояснити концепцію воронки продажів як послідовність кроків від першого відвідування до цільової дії покупка реєстрація підписка з аналізом конверсії на кожному кроці скільки відсотків переходить далі скільки відсіюється де найбільші втрати. Описати когортний аналіз групування користувачів за датою першого відвідування для порівняння поведінки користувачі які прийшли в січні versus лютому чи є різниця в утриманні та активності. Пояснити А В тестування коли показують різні версії сторінки різним групам користувачів щоб визначити яка версія краще працює наприклад два варіанти кнопки різних кольорів розмірів текстів вимірюємо на якій більше натискань.

3. Розкрити як використовувати дані аналітики для прийняття рішень якщо багато користувачів залишають кошик на етапі оплати можливо проблема зі складною формою або недовірою до безпеки потрібно спростити процес додати значки безпеки, якщо певні товари часто переглядають але рідко купують можливо ціна зavelika або опис незрозумілий потрібно покращити.

Література

1. Jack Marquez, Oscar H. Mondragon. Modelling the Impact of Cloud Storage Heterogeneity on HPC Application Performance. 2024. URL: <https://is.gd/KYvovK>

2. Nane Kratzke. Cloud-Native Observability: The Many-Faceted Benefits of Structured and Unified Logging—A Multi-Case Study. 2022. URL: <https://www.mdpi.com/1999-5903/14/10/274>

Навчальне видання

**Ольга Олександрівна Смагіна
Владислав Юрійович Козуб
Галина Олександрівна Козуб**

**ТЕХНОЛОГІЯ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ
методичні вказівки до лабораторних робіт
для здобувачів вищої освіти першого (бакалаврського) рівня
спеціальності F3 – «Комп'ютерні науки»**

Комп'ютерне макетування – О. О. Смагіна
Коректор – Ю. Г. Козуб

Здано до склад. 23.01.2026 р. Підп. до друку 30.05.2026 р. Формат 60x84 1/16.
Папір офсет. Гарнітура Times New Roman. Друк ризографічний.
Ум. друк. арк. 1,0. Наклад 100 прим. Зам. № 134
Видавництво Державний заклад «Луганський національний університет імені Тараса
Шевченка»
м. Полтава, вул. Коваля, 3
e-mail: mail@luguniv.edu.ua www.luguniv.edu.ua
Свідоцтво суб'єкта видавничої справи ДК № 3459 від 09.04.2009 р.
