softserve

КВАЛІФІКАЦІЙНІ ВИМОГИ ДО ВИПУСКНИКІВ ЗВО

за спеціальностями, орієнтованими на розробку програмного забезпечення

Версія 1.6

10/8/2019

Зміст

Історія змін

Дата	Версія	Опис	Автор
ММ/ДД/РРРР	<X.X>	<деталі>	<Ім’я>

1 Вступ

Даний документ містить кваліфікаційні вимоги з розробки програмного забезпечення та іноземної мови до випускників вищих навчальних закладів, що планують працювати як розробники програмного забезпечення. Кваліфікаційні вимоги побудовані на базі моделі компетенцій компанії СофтСерв при сприянні і з урахуванням специфіки роботи аутсорсинг IT галузі України.

Документ рекомендований для застосування до випускників навчальних спеціальностей:

113 Прикладна математика
121 Інженерія програмного забезпечення
122 Комп'ютерні науки та інформаційні технології
123 Комп'ютерна інженерія

Документ складається з наступних частин:

- **Загальні вимоги**, які включають методологію розробки програмного забезпечення, дизайну, роботи з вимогами, роботи з даними та забезпечення якості розробки, і не залежать від технології спеціалізації. Є необхідним мінімумом знань;
- **Технологія спеціалізації** – основна технологія розробки ПЗ, представлені вимоги до найбільш популярних технологій;
- Вимоги до **практичних навичок** та вмінь;
- **Персональна ефективність** – знання та навики, які включають основи бізнес-комунікації, вміння співпрацювати в команді, вміння вирішувати проблеми та бути ефективним у веденні справ;
- Вимоги з володіння **англійською мовою**, вміння вести комунікацію англійською мовою.

ВАЖЛИВО: Даний документ тільки уточнює вимоги за напрямками розробка ПЗ та іноземної мови. Він ні в якому разі не відміняє та не зменшує вимоги з інших профільних предметів, таких як вища математика, математичний аналіз, дискретна математика, диференціальні рівняння, теорія ймовірностей та математична статистика, чисельні методи та інші.

1.1 Термінологія

Термін	Визначення
ПЗ	Програмне забезпечення
ЗВО	Заклад вищої освіти

1.2 Посилання

#	Версія	Дата	Назва	Ким виданий
1	<X.X>	ММ/ДД/РРРР	<Назва документу>	<Організація, якою виданий>

2 Загальні знання

Даний розділ містить перелік загальних знань, якими повинен володіти кожен інженер розробки та контролю якості програмного забезпечення незалежно від технології спеціалізації.

2.1 Методологія розробки програмного забезпечення

2.1.1 Життєвий цикл розробки програмного забезпечення

- Методології розробки програмного забезпечення (Waterfall, Kanban, Agile/Scrum, Scrumban, Waterfall)
- Етапи розробки програмного забезпечення
- Базові методології оцінки терміну виконання проектів (Scope Concept, Work Break Down Structure)

2.1.2 Інструменти розробки програмного забезпечення

- IntelliJ IDEA (Java)
- Microsoft Visual Studio Express (.Net)
- PyCharm (Python)
- PhpStorm (PHP)
- Sublime, Visual Studio Code, WebStorm (Web Development)
- Firebug, Chrome Web Developer Tool (debugger)
- Sonar
- Jira, BugZilla або Mantis
- MS Project (опціонально)

2.1.3 Робота з системами керування версіями

- Системи керування версіями, призначення, типи
- Використання Git/SVN
- Створення проекту
- Внесення змін у репозиторій
- Обновлення робочої версії
- Обробка конфліктів версій

2.1.4 Автоматична збірка програмного забезпечення

- Концепція автоматичної побудови збірок
- Утиліти для автоматизації процесу збирання програмного продукту: ant, maven
- Сценарії багатофазних процесів збірок
- Створення випускних документів
- Створення планових збірок
- Інтеграція стадій впровадження, переміщення застосувань до релізу (Continuous integration/Continuous Delivery/Continuous Deployment)

2.1.5 Забезпечення якості розробки ПЗ

- Стандарти кодування (code convention)
- Аналіз коду (code review) і засоби для його проведення
- Основні принципи контролю якості
- Методи тестування, основні типи тестів
- Дефекти: класифікація, керування, верифікація
- Автоматизовані засоби керування дефектами

2.1.6 Командна розробка ПЗ

- Проектна команда і її структура
- Основні ролі і обов'язки в проектній команді
- Робота за Scrum методологією
- Засоби командної розробки ПЗ
- Постановка цілей, концепція SMART
- Звітування

2.2 Керування вимогами

- Основні принципи керування вимогами
- Класифікація вимог
- Методи специфікації вимог: UML

2.3 Дизайн

2.3.1 Об'єктно-орієнтований дизайн

- Абстракція
- Інкапсуляція
- Успадковування та агрегація
- Модульність
- Поліморфізм
- Типи та класи
- Шаблони проектування (GoF Design Patterns та ін.)
- Архітектурні шаблони: багаторівнева архітектура, MVC, SOA, EDA та IoC
- Основи архітектури веб-сервісів (REST, SOAP)

2.3.2 Уніфікована мова моделювання UML

- Основні типи діаграм
- Діаграми прецедентів (Use Case Diagram)
- Діаграми класів (Class Diagram)
- Діаграми відношень (Entity Relationship Diagrams)
- Діаграми послідовності (Sequence Diagrams)

2.4 Робота із базами даних

2.4.1 Теорія реляційних баз даних

- Сутності
- Атрибути
- Кортежі
- Зв'язки, типи зв'язків
- Розуміння та використання ERD
- Нормалізація даних, нормальні форми
- Цілісність даних
- Транзакції: розуміння ACID, блокування, відновлення, рівні ізоляції, паралелізм

2.4.2 Мова запитів SQL

- Робота з об'єктами: таблиці, ключі, індекси, тригери
- Робота з даними: select, insert, update, delete
- Робота зі з'єднаннями: join, union, intersect, except
- Агрегування даних
- Розуміння представленнями (view), функції, процедури та сутності курсору

2.5 Мережеві технології

- Рівні мереж
- TCP/IP
- Основи протоколів прикладного рівня
- Базові знання фізичного рівня протоколів та медіа
- Модель "клієнт-сервер"
- Інтернет стандарти
- XML
- Основи протоколів рівня аплікації (HTTP, FTP, Telnet)
- Розуміння WWW
- MIME
- URL адреси та ідентифікатори URI
- Основні засоби дослідження помилок (ICMP, ping, traceroute)
- Сокети, IP і адресація портів
- Використання проксі-сервера
- Сервіси пошуку імен: DNS, whois
- Сервіси віддаленого доступу: Telnet, SSH, Remote Desktop, VNC

3 Технологія спеціалізації

В даній секції подані на вибір вимоги до двох найбільш популярних технологій. Достатньо набути знань в одній з них.

3.1 Java

3.1.1 Мова програмування

- Ідентифікатори
- Класи, члени класів
- Інтерфейси та їх реалізація
- Стек та купа
- Літерали, змінні
- Передача змінних у методи
- Масиви; оголошення, ініціалізація
- Оператори Java
- If та switch
- Цикли та ітератори
- Винятки та помилки, обробка винятків
- Типи даних: String, StringBuilder та StringBuffer
- Використання Javac та Java команд
- Використання обгортки класів та Boxing
- Збірник сміття
- Використання механізму затвердження
- Навігація, ввід/вивід
- Дати, цифри та валюти
- Аналіз, розмітка, формування
- Перевизначення hashCode() та equals()
- Колекції, використання узагальнених колекцій
- Універсальні типи
- JAR файли
- Статичний імпорт
- Серіалізація
- Внутрішні класи
- Робота з потоками, стани, переходи, синхронізація даних, мінімальні синхронізаційні техніки
- Синхронізація коду
- Рефлексія
- Анотація
- Лямбда вирази. Програмування за допомогою лямбда виразів
- Інтерфейс API потоків вводу-виводу filter(), map(), flatmap()
- Тип даних Optional

3.1.2 Доступ до даних

Загальні знання згідно секції 2.4, а також:

- Робота з підключеннями, пул
- Підготовка та виконання SQL команд
- Робота з транзакціями
- Робота з RowSet та ResultSet
- Робота з Hibernate: класи, об’єкти, анотації, запити, транзакції, паралелізм, кешування, використання інструментів

3.1.3 Сервіси, що працюють на web-сервері

Загальні знання згідно секції 2.5, а також:

- Установка і настройка сервера (Tomcat)
- Сервлети
- Робота з клієнтськими запитами, заголовки HTTP запитів
- Створення відповідей сервера: коди станів, заголовки відповідей
- Відстеження сесій
- Інтеграція сервлетів з JSP: MVC
- Робота з cookies
- Створення web-сервісів: SOAP, REST, WSDL
- Включення файлів і аплетів до JSP сторінок
- Мова розширення JSP 2.0, Expression Language, JSTL
- Web-застосування: використання, розгортання, контроль поведінки за допомогою web.xml
- Події застосування
- Бібліотеки тегів
- Використання основних каскадів MVC та IoC
- Робота з Java Mail

3.1.4 Фреймворки

- Spring Framework (MVC, IoC, WebFlow)
- Apache Axis або Spring WS (web services)

3.1.5 Розподілені об'єкти та компоненти

- Архітектура розподіленого програмного продукту
- Вимоги до розробки розподіленого програмного продукту
- Розподілені об'єкти у Java

3.1.6 Використання JUnit

- Створення тестів
- Робота з тестовими даними
- Виконання тестів
- Аналіз результатів тестів

3.2 .NET

3.2.1 Мова програмування

Рекомендовано орієнтуватись саме на мову програмування C#

- Архітектура платформи .NET
- Бібліотека класів .NET Framework
- Загальна система типів
- Приведення типів
- Елементарні, reference та розмірні типи
- Загальні операції над об'єктами
- Клонування об'єктів
- Члени типів і доступ до них
- Константи і поля
- Методи
- Властивості
- Події
- Обробка тексту
- Перераховуванні типи та бітові поля
- Колекції
- Інтерфейси
- Атрибути
- Делегати
- Обробка виняткових ситуацій
- Ввід та вивід
- Робота з XML
- Автоматичне керування пам'яттю (збірник сміття)
- Хостинг CLR, домени програми
- Багатопотоковість; асинхронність; паралельні обчислення; синхронізація потоків; interlocked функції; дедлоки
- Рефлексія
- Серіалізація
- Регулярні вирази
- Локалізація, глобалізація, інтернаціоналізація

3.2.2 Доступ до даних

Загальні знання згідно секції 2.4, а також:

- Робота з даними
- Транзакції
- ADO.NET (DataReader, DataAdapter, TableAdapter)
- Micro ORM (Dapper)
- LINQ, Entity framework, NHibernate
- Створення та доступ до XML Web Services

3.2.3 Публікація аплікацій

- Публікація веб-сайтів та веб-сервісів

- Створення інсталяційного проекту (десктоп, веб)
- Пререквізити інсталяції
- Реєстрація збірок у GAC
- ClickOnce
- Публікація Windows Service

3.2.4 Сервіси, що працюють на web-сервері

Загальні знання згідно секції 2.5.

3.2.5 Фреймворки

- Web: Service-oriented architecture (SOA), ASP.NET Core MVC, ASP.NET Core
- Основи веб-сервісів (SOAP, WSDL)
- Windows Communication Foundation (WCF)

3.2.6 Використання NUnit/xUnit/MSTest/Unity

- Створення тестів
- Робота з тестовими даними
- Виконання тестів
- Аналіз результатів тестів

3.3 C++

3.3.1 Мова програмування

- C++...++ (C++11, C++14, C++17...)
- Система типів
- Внутрішнє представлення даних
- Оголошення та визначення змінних та констант
- Поняття про ініціалізацію змінних
- Класифікація операторів
- Константні вирази
- Інструкції
- Оголошення та визначення
- Масиви як регулярні типи
- Вказівники – низькорівневий засіб C++
- Спорідненість вказівників та масивів
- Функції, метод декомпозиції
- Поняття про прототип функції
- Особливості оголошення функції, перевантаження
- Класи і структури
- Об’єкти як екземпляри класу
- Оператори доступу до членів класу
- Види та визначення методів
- Конструктори
- Перевантаження конструкторів
- Перевизначення операторів, явне/неявне
- Деструктори
- Перетворення типів, явне/неявне
- Strings
- Константи і поля
- mutable-об’єкти. volatile конструкції
- Механізм контролю назв
- Наслідування
- Синтаксис похідних класів
- Ієрархія областей видимості
- Особливості реалізації конструкторів похідних класів
- Поліморфізм та віртуальні методи
- Взаємозв’язки між класами
- Множинне наслідування
- Синтаксис класів у випадку множинного наслідування
- Порядок виклику конструкторів та деструкторів базових класів
- Віртуальні базові класи
- Оператори генерування винятків у try-блоках
- Перехоплення винятків та catch блоки
- Обробка виняткових ситуацій в конструкторах
- Стандартні винятки C++
- Параметризовані функції і класи
- Ієрархія потокових шаблонів

- Шаблони і проектування програм
- Бібліотеки стандартних шаблонів (STL)
- Загальні властивості та основні операції над послідовними контейнерами
- Загальні властивості та основні операції над асоціативними контейнерами
- Категорії ітераторів та допоміжні функції ітераторів

3.3.2 Доступ до даних

Загальні знання згідно секції 2.4

3.3.3 Публікація аплікацій

- Створення інсталяційного проекту (десктоп)
- Пререквізити інсталяції

3.3.4 Сервіси, що працюють на сервері

Загальні знання згідно секції 2.5.

3.3.5 Фреймворки

- QT
- Основи веб-сервісів (SOAP, WSDL)
- Windows Communication Foundation (WCF)

3.3.6 Використання MSTest/QTtest

- Створення тестів
- Робота з тестовими даними
- Виконання тестів
- Аналіз результатів тестів

3.4 Python

3.4.1 Мова програмування

- Використання інтерпретатора мови Python
- Типи даних
- Твердження: assert, pass, return, yield, raise, break, continue, import, global, exec
- Твердження: while, if, for, pass, break, continue
- Визначення функцій
- Лямбда-форми функцій
- Класи та наслідування
- Видимість змінних
- Динамічна типізація (Duck typing)
- Ітератори
- Генератори
- Модулі та пакети
- Рефлексія та Метапрограмування (Reflection & Metaprogramming)
- Оброблення вводу-виводу, читання та запис у файли
- Обробка помилок та винятків
- Робота зі стандартними бібліотеками (Python Standard Library)
- Розгалуження (threading)
- Регулярні вирази
- Розробка мережевих систем: клієнт-сервер
- Розробка з використанням MVC шаблону та його модифікацій
- Відлагодження коду

3.4.2 Тестування коду

- Основні концепції теорії тестування програмного продукту, види тестів
- Інструменти для тестування unittest/pytest
- Концепції fixture, mock і stubs

3.4.3 Робота із базами даних

- Взаємодія з реляційними базами даних (SQLite, MySQL, PostgreSQL)
- CRUD операції з таблицями і рядками таблиць.
- Реалізація механізму транзакції
- ORM (SQLAlchemy, Django ORM)
- Розуміння нереляційних (NoSQL) баз даних

3.4.4 Веб-розробка

- Основи ES6
- AJAX
- Розуміння сучасних тенденцій у розробці із використанням фрейморків
- HTML/HTML5
- CSS

3.5 Ruby

3.5.1 Мова програмування

- Використання інтерпретатора та інтерактивного командного рядка
- Типи даних (Numbers, Strings, Ranges, Regular Expressions, Arrays, Hashes, Symbols, True, False, nil)
- Ключові слова в Ruby
- Керуючі структури
- Функції (методи)
- Класи, об’єкти, модулі
- Видимість змінних
- Duck typing
- Ітератори
- Лямбда-функції
- Рефлексія та Метапрограмування (Reflection & Metaprogramming)
- Оброблення вводу-виводу, читання та запис у файли
- Обробка помилок та виняток
- Gems
- Розгалуження (threading)
- Контроль якості: тестування коду
- Регулярні вирази
- Розробка мережевих систем: клієнт-сервер
- Взаємодія с базою даних, ORM
- Обробка XML та HTML
- Відлагодження коду

3.5.2 Фреймворк Ruby on Rails

- Шаблон MVC
- Маршрутизація і REST архітектура
- Контролери
- Рівень представлення (View layer)
- Робота з ActiveRecord ORM
- Тестування: функціональні та інтеграційні тести

3.6 PHP

3.6.1 Мова програмування

- Типи даних (Integer, Float, String, Array, Boolean, Resource, Null, Object)
- Змінні (local, global, superglobal) та константи (magic constants)
- Вирази та оператори PHP
- Ключові слова в PHP
- Керуючі структури (while, if, else, elseif, for, foreach, break, switch, include, require)
- Класи та об'єкти
- Обробка помилок та виняткових ситуацій
- Функції (user-defined)
- Функції для роботи з рядками. Математичні та інші функції. Робота з датами
- Ввід та вивід, читання та запис у файли
- Групи імен (namespaces)
- Регулярні вирази
- Робота з електронною поштою
- Підтримка UNICODE
- Ітератори
- Робота з мережею
- Робота з XML
- Взаємодія с базою даних, ORM
- Криптографія (Hash, Mcrypt, Mhash, OpenSSL)
- Компресія даних
- Використання PEAR бібліотек
- Розробка з використанням шаблону MVC
- Відлагодження коду
- php.ini директиви та конфігурація періоду виконання

3.6.2 Тестування коду

- Основні концепції теорії тестування програмного продукту, види тестів
- Інструменти для тестування PHPUnit
- Концепції fixture та mock

3.6.3 Робота із базами даних

- Взаємодія з реляційними базами даних (SQLite, MySQL, PostgreSQL)
- CRUD операції з таблицями і рядками таблиць.
- Реалізація механізму транзакції
- ORM (Zend Framework ORM)
- Розуміння нереляційних (NoSQL) баз даних

3.6.4 Веб-розробка

- Основи ES6
- AJAX
- Розуміння сучасних тенденцій у розробці із використанням фрейморків
- HTML/HTML5

- CSS

3.7 Go

3.7.1 Мова програмування

- Пакети, змінні, функції
- Типи
- Оператори управління
- Масиви, зрізи, карти
- Вказівники
- Структури та інтерфейси
- Багатопоточність

3.8 Розробка мобільних аплікацій для iOS

Рекомендовано орієнтуватись на мови програмування Objective-C та Swift.

3.8.1 Мова програмування

- Архітектура iOS аплікації
- Приведення типів
- Управління пам'яттю в Swift
- Колекції
- Властивості в Objective-C/Swift
- Управління потоком, функції
- Структури та перерахування
- Класи
- Категорії, розширення протоколи і делегати
- Замикання в Swift/Objective-C
- Універсальні шаблони і обробка помилок в Swift
- Потоки, черги та механізми роботи з ним

3.8.2 Основи роботи з інтерфейсами і їх побудова

- Application LifeCycle
- Основи побудови інтерфейсів в iOS
- UIView
- AutoLayout
- Особливості побудови користувацьких інтерфейсів
- UIViewController і його нащадки
- Анімації
- Робота з файловою системою
- Робота з мережею
- Testing application Debug with xCode

3.8.3 Доступ до даних

Загальні знання згідно секції 2.4, а також:

- Core Data
- Створення та доступ до XML Web Services

3.8.4 Публікація аплікацій

- Підготовка додатку до публікації в App Store
- Пререквізити інсталяції
- ClickOnce

3.8.5 Сервіси, що працюють на web-сервері

Загальні знання згідно секції 2.5.

3.8.6 Створення тестів із бібліотекою Xcode 10

- Створення тестів
- Робота з тестовими даними
- Виконання тестів
- Аналіз результатів тестів

4 Методологія DevOps

В даній секції подані вимоги по Методології DevOps.

Знання основ методології є обов'язковим.

- DevOps і Agile
- Складові частини DevOps – Development (Software Engineering), Operations, Quality Assurance
- Інтеграція DevOps в процес розробки програмного забезпечення
- DevOps and Digitalization
- Continuous Feedback механізм
- Забезпечення високої доступності сервісів – High Availability

Рекомендовано володіти хоча би однією з технологій описаних нижче.

4.1 Хмарні обчислення (Cloud Computing)

- Основні типи віртуалізації
- Принципи побудови хмарних платформ
- Переваги використання хмарних платформ
- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform

4.2 Мікросервіси та контейнери

- Визначення мікросервісів, їх переваги перед монолітною архітектурою
- Відмінність контейнеризації від звичайної віртуалізації
- Принципи побудови контейнера (Docker)
- Процес створення контейнера, базові образи
- Оптимізація контейнерів
- Система управління контейнерами – Kubernetes
- Kubernetes – основні компоненти і архітектура
- Helm
- Istio
- Service Mesh
- Prometheus

4.3 Моніторинг та Log Management

- Важливість моніторингу в процесі розробки програмного забезпечення
- Системний і функціональний моніторинг
- Моніторинг як сервіс
- Nagios
- Zabbix
- Важливість системи управління журналами (logs) в процесі розробки програмного забезпечення
- Основні формати файлів із журналами (log files)

- Зв'язка Elasticsearch – Logstash – Kibana
- Graylog

4.4 Інфраструктура як код (Infrastructure as a Code)

- Парадигма Infrastructure as Code, складові частини, важливість
- Vagrant
- Ansible
- Chef
- Puppet
- AWS CloudFormation
- Terraform

4.5 Автоматизація процесу тестування програмного забезпечення

- Важливість автоматизації процесу тестування програмного забезпечення
- Основні типи автоматичного тестування: Unit Tests, Integration Tests, Functional Tests, Acceptance tests, Load and Performance Tests.
- Selenium
- Parasoft
- TestComplete
- Microsoft Azure Test Plans

4.6 Continues Integration та Continuous Delivery

- Важливість і основні переваги застосування Continuous Integration/Delivery в процесі створення програмного забезпечення
- Відмінність Continuous Integration від Continuous Delivery
- Основні компоненти процесу Continuous Integration
- Jenkins
- GitLab CI
- Travis CI
- Spinnaker

5 Контроль якості програмного продукту

5.1 Основи тестування програмного продукту

5.1.1 Основи процесу тестування програмного продукту

- Що таке тестування та контроль якості (QC)
- Для чого необхідне тестування
- Основні принципи тестування
- Ключові вирази в Основах процесу тестування відповідно до стандарту ISTQB

5.1.2 Ролі та обов'язки членів проектної команди

- Розуміння типової структури проектної команди та ролей у проектній команді
- Розуміння типових ролей QC команди та обов'язків
- Розуміння різниці рівня QC та ролі QC

5.1.3 Вимоги до аналізу вимог

- Типи документів, відповідно до яких можуть подаватися вимоги
- Призначення та важливість перегляду вимог
- Важливість та мета відстеження вимог
- Призначення відстеження та залежність матриць

5.2 Test Design

5.2.1 Процеси розробки тестування

- Призначення та різні типи тестової документації
- Підходи до тестового проектування на основі специфікацій (black-box)
- Основні методики тестування black-box

5.3 Виконання тесту та управління дефектами

5.3.1 Виконання тесту

- Рівні тесту та типи тесту
- Різниця між статичним та динамічним тестування
- Типи звітів про виконання тесту

5.3.2 Відстеження дефектів

- Для чого потрібні звіти про дефекти та приклад укладення хорошого зразку
- Важкість і пріоритетність
- Життєвий цикл дефектів

5.4 Інструменти підтримки проектів та тестування

Необхідно знати хоча б один з інструментів:

- Інструменти управління проектами та відстеження дефектів (н-д, Jira)
- Інструменти керування сценаріями тестування
- Інструменти колективної роботи (н-д, Confluence)
- Необхідність Тест та Дефект менеджмент, їх використання
- Необхідність інструментів для співпраці та їх застосовування

5.5 Суміжні області

5.5.1 Бази даних

Загальні знання згідно секції 2.4.

5.5.2 Мережі

Загальні знання згідно секції 2.5.

5.5.3 Unix/Linux

- Що таке Unix and Linux?
- Різниця між Unix and Linux
- Командний рядок Linux

5.5.4 Адміністрування OS MS Windows

- Адміністрація локальної мережі
- Управління обліковими записами користувачів
- Управління дозволами

6 Практичні навики

Нижче перераховані практичні навики, які випускник має набути під час виконання лабораторних, курсових та дипломної робіт:

- навики командного виконання проектів по розробці ПЗ з використанням усіх фаз життєвого циклу (від збору вимог до впровадження) та різних середовищ (тестове середовище, середовище впровадження, промислове середовище)
- навики виконання проектів з використанням різних методологій розробки ПЗ (Waterfall, Scrum)
- навики виконання проекту на різних ролях (розробник, тестувальник, аналітик, керівник)
- навики використання засобів командної розробки:
  - сховище коду (repository)
  - засоби побудови та інтеграції
  - засоби керування дефектами
- навики написання, коментування та відлагодження програмного коду в об'ємі поданому в секції 3
- навики створення простої бази даних та написання SQL-запитів згідно обсягу поданого в секції 2.4
- навики проведення аналізу чужого коду (code review)
- навики написання дизайн-документів з використанням UML-нотації
- можливість створення діаграми UML на основі бізнес аналізу продукту
- навики використання англійської мови для комунікації в межах проектної команди (оформлення коментарів, документації, протоколів нарад, переписка з членами команди, проведення презентацій роботи)
- знання всіх можливих типів програм (таких як web, desktop, тощо) та специфіки їх тестування
- знання концепції клієнт-сервер
- знання типів веб-додатків та їх роботи на базовому рівні
- знання стандартних елементів керування, їх назви і властивості
- досвід тестування інструментів та середовищ, а також можливість їх налаштування на необхідному рівні
- базові знання про тестові проектні дії для покриття набору вимог до продукту шляхом ефективного тестового проектування
- знання методик тестування: результати тестування повинні бути надійними і не потребувати додаткового підтвердження
- можливість проведення аналізу дефектів та написання детальних та корисних звітів про дефекти з очікуваними результатами для виправлення дефектів
- розуміння того, що означають процеси тестування та контролю якості, щоб гарантувати необхідну якість продукту

7 Персональна ефективність

Для успішного виконання своїх професійних обов'язків на посаді розробника ПЗ випускники ЗВО повинні володіти так званими "м'якими навичками" (soft skills).

Такі навики включають в себе:

- бізнес-комунікацію (наради, презентації, активне слухання, письмова комунікація, проходження співбесіди тощо)
- роботу в команді (надавання зворотнього зв'язку, ділення досвідом, вирішування конфліктних ситуацій тощо)
- управління часом
- вміння вирішувати проблемні ситуації
- висока увага до деталей
- навички пошуку, збору та обробки даних та інформації, необхідних для реалізації завдання

8 Знання та вміння із англійської мови

8.1 Conference calls

A graduate should have enough skills to be able to

- have a small talk
- provide information/explain/suggest/convince/apologize
- fluently describe responsibilities and duties on the project
- inform about current issues
- propose solutions providing general reasoning
- ask questions to clarify the information
- agree or disagree providing reasons on very general level
- comprehend key information of the calls / make conclusions and analysis
- understand a written document (meeting minutes) summarizing the talk with actions needed to be done
- write general summary after the meeting (meeting minutes) using needed vocabulary

8.2 Socializing

A graduate should have enough skills to be able to

- communicate thoughts and ideas in both formal and informal environments
- chat via messengers using casual/slang phrases and abbreviations with ease
- talk in person without long pauses in your natural pace
- have a small talk
- describe background and overall life experience/educational plans
- tell a joke without losing its initial sense after translation
- express admiration/sympathy toward interlocutor
- ask for clarification/provide clarification if necessary politely
- make a presentation/short pitch in public

8.3 Technical interviews, on-site visits, team meetings

A graduate should have enough skills to be able to

- provide the needed information articulating the message clearly
- fluently talk about duties and responsibilities on the project
- inform about current issues and propose solutions
- ask questions to clarify information
- agree or disagree providing reasons
- comprehend the information on the sufficient level to make conclusions and analysis
- extract the key information from technical documentation while preparing for the meeting
- write meeting minutes/action items after the meeting

8.4 Presenting information

A graduate should have enough skills to be able to

- present ideas in a clear way by following presentation structure appropriate for the context and using linking phrases effectively
- answer follow up questions
- fully comprehend key information presented
- create a well-structured presentation plan outlining key points

8.5 Technical documentation

A graduate can understand the main concept of the following documentation:

- articles
- emails
- reports
- policies
- internal documentation
- project requirements, estimations, specifications

8.6 Reporting results

8.6.1 Reading

A graduate can understand the standards of reporting in the company (policies, templates and procedures).

8.6.2 Writing

A graduate can describe the executed tasks/work done according to the company standards (conforming to policies and procedures, using templates).

8.7 Self-development

A graduate should have enough skills to be able to

- distinguish the desirable and useful type of online/offline course among many available ones and complete it
- attend and participate in conferences/webinars based on current educational request
- apply gained knowledge to cover your needs in practice
- share gained knowledge within making a presentation

8.8 Written communication

A graduate should have enough skills to be able to

- write letters on familiar or predictable matters knowing the structure of e-mails
- write formal and informal letters, reports, summaries, instructions, business letters and memos, CV, e-mails, instant messages

- use the grammar and vocabulary appropriately to begin and end letters, to organize ideas, to describe, to show attitude, to draft and edit texts, to style and build phrases avoiding repetition and using punctuation appropriately
- make comments and analyze the executed task/work
- reply to the requestor providing the exact information accurately
- document testing procedures using templates and assist senior staff in creating test documentation such as test strategy, test plan, test policy, etc.
- document test cases, test scenarios, other test artifacts according to existing standards effectively and accurately.
- document software product, documentation discrepancies using existing standards and tools effectively