**ПРАКТИЧНА РОБОТА №3 (4 год.)**

**Тема:** Контроль версій

**Мета:** Набуття навичок роботи з Git та GitHub.

Основні завдання:

1. Реєстрація на GitHub
2. Створення локального репозиторію
3. Створення репозиторію на GitHub
4. Команди роботи в терміналі з репозиторієм
5. Відправлення файлів до репозиторію
6. Колаборація

**ЛІТЕРАТУРНІ ДЖЕРЕЛА**

1. Git Book<https://git-scm.com/book/en/v2>
2. GitHowTo<https://githowto.com/>
3. Основи роботи з Git. Базові команди<https://hyperhost.ua/info/uk/osnovi-roboti-z-git-bazovi-komandi?gad_source=1&gclid=CjwKCAjwufq2BhAmEiwAnZqw8tYQJKMQT0BP_MPScCsc10IBE1Mg7cM9Duv_4mapN1SWfFfBEnHilhoC_XMQAvD_BwE>
4. Основи роботи з GitHub для **Unity** <https://youtu.be/z6GM_qb-sUY?si=gne1JKl9cuwDngn6>

**ЗАВДАННЯ ДЛЯ ВИКОНАННЯ**

**Завдання 1. Перевірка встановлення Git.**

1. Перевірити чи встановлено на Ваш комп’ютер систему контролю версій Git. Відкрийте командний рядок (Термінал) та введіть команду **git version.**

При встановленому програмному забезпечення Git Ви отримаєте відповідь про версію Git. Для прикладу **git version 2.44.0.windows.1**

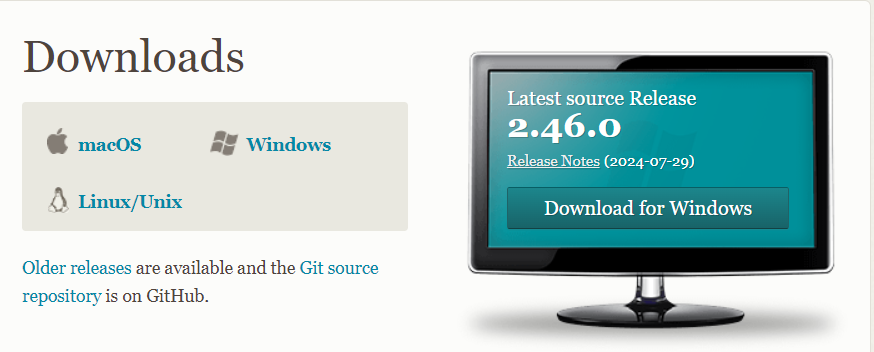
Подальші дії:

* У разі відсутності повідомлення про версію Git необхідно встановити Git для цього виконайте ***Завдання 2***.
* При встановленому Git проведіть оновлення версії до останньої офіційної версії (Latest source Release) – ***Завдання 3***.

**Завдання 2. Встановлення Git (за потреби).**

1. Перейти на офіційний сайт <https://git-scm.com/downloads>

2. Виберіть операційну систему на яку буде встановлено Git (рисунок 1.)



*Рис. 1. Вибір операційної системи.*

3. Завантажте інсталяційний пакет та проведіть встановлення Git.

4. Проведіть перевірку встановлення перевіривши версію програмного забезпечення в терміналі **git version.**

**Завдання 3. Оновлення Git (за потреби).**

Для оновлення версії Git оберіть один з варіантів:

1. Якщо ваша версія Git 2.14.1 або молодша:

Видаліть Git, завантажте останню версію Git і інсталюйте її знову.

1. Версія між 2.14.2 і 2.16.1:

Використовуйте команду **git update**

1. Якщо версія дорівнює або перевищує Git 2.16.1:

Використовуйте команду **git update-git-for-windows**

**Завдання 4. Створення та ініціалізація локального репозиторію.**

1. Створіть на вашому комп’ютері каталог TestGit.

2. В терміналі (GitBash, Visual Studio Code або терміналі Windows (cmd.exe)) за допомогою команди операційної системи перейдіть у дану папку.

3. Використовуючи команду **git init** ініціалізуйте каталог як локальний репозиторій.

4. Виконайте команду **git status** для перевірки стану файлів репозиторію. Оскільки файли в каталозі відсутні то ви отримаєте відповідне повідомлення.

**Завдання 5. Конфігурування репозиторію.**

1. Задайте ім'я автора, яке використовуватиметься для всіх комітів у локальному репозиторії. В терміналі задайте команду

**git config --local user.name <name>**

, де **<name> -** ваше ім’я.

Для прикладу: **git config --local user.name "John Doe"**

2. Задайте електронну пошту автора, яка використовуватиметься для всіх комітів у локальному репозиторії. В терміналі задайте команду

**git config --local user.email <email>**

, де **<email> -** ваша електронна пошта.

Для прикладу: **git config --global user.email johndoe@example.com**

3. Перевірте конфігурування репозиторію командою

**git config --list**

Серед списку параметрів відшукайте задані вами параметри.

**Завдання 6. Створення файлів та визначення статусу.**

1. Створіть файл **index.html** в каталозі TestGit та збережіть файл.

*Вміст файлу наступній:*

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet" href="css/default.css" type="text/css">

</head>

<body>

<header id="head\_main">

<h1>САМОРОЗВИТОК</h1>

<nav>

<ul>

<li><a href="#">Мова</a></li>

<li><a href="#">Наука</a></li>

<li><a href="#">Культура</a></li>

</ul>

</nav>

</header>

<main>

<article>

<header>

<h2>Система контролю версій Git</h2>

<h3>Git Book</h3>

</header>

<section>

<p>Про систему контролю версій</p>

<p> Що таке “система контролю версій”, і чому це важливо? Система контролю версій - це система, що записує зміни у файл або набір файлів протягом деякого часу, так що ви зможете повернутися до певної версії пізнише. Як приклад, в цій книзі, для файлів, що знаходяться під контролем версій, буде використовуватися код програмного забезпечення, хоча насправді ви можете використовувати контроль версій практично для будь-яких типів файлів. Якщо ви графічний або веб-дизайнер і хочете зберегти кожну версію зображення або макета (швидше за все, захочете), система контролю версій (далі СКВ) якраз те, що потрібно. Вона дозволяє повернути вибрані файли до попереднього стану, повернути весь проект до попереднього стану, побачити зміни, побачити, хто останній міняв щось і спровокував проблему, хто вказав на проблему і коли, та багато іншого. Використання СКВ також в цілому означає, що, якщо ви зламали щось або втратили файли, ви просто можете все виправити. Крім того, ви отримаєте все це за дуже невеликі накладні витрати.

</p>

</section>

<footer>

<p>Link: <a href="https://git-scm.com/book/en/v2">

https://git-scm.com/book/en/v2</a></p>

</footer>

</article>

<aside>

</aside>

</main>

<footer>

</footer>

</body>

</html>

2. Задайте команду **git status** для перевірки стану репозиторію. Ознайомтеся із повідомлення системи. Зробіть скріншот вікна повідомлення.

3. Додати файл index.html до відслідкування Git командою **git add index.html**.

4. Задайте команду **git status** для перевірки стану репозиторію. Ознайомтеся із повідомлення системи. Зробіть скріншот вікна повідомлення.

5. Порівняйте два повідомлення.

6. Збереження відтиску (**commit**). Задайте команду **git commit** повідомленням (опція -m) «Перша версія проекту»: **git commit** **-m «Перша версія проекту».**

7. Задайте команду **git status** для перевірки стану репозиторію. Ознайомтеся із повідомлення системи. Зробіть скріншот вікна повідомлення.

8. Створіть каталог CSS в TestGit. В цьому каталозі створіть стильовий файл **default.css** (css/default.css).

Додайте наступний зміст

#head\_main{

background:red;

height:90px;

border-bottom:10px solid #cc0b0b;

}

#head\_main h1{

color:#fff;

float:left;

display: inline-block;

}

#head\_main nav>ul>li{

display:inline-block;

float:right;

margin-right:30px;

width:100px;

text-align:center;

}

#head\_main nav>ul>li>a{

color:#fff;

background:#cc0b0b;

font-size:20px;

text-decoration: none;

line-height:90px;

display: block;

}

#head\_main nav>ul>li>a:hover{

background:#730303;

}

main{

width:1000px;

margin: auto;

}

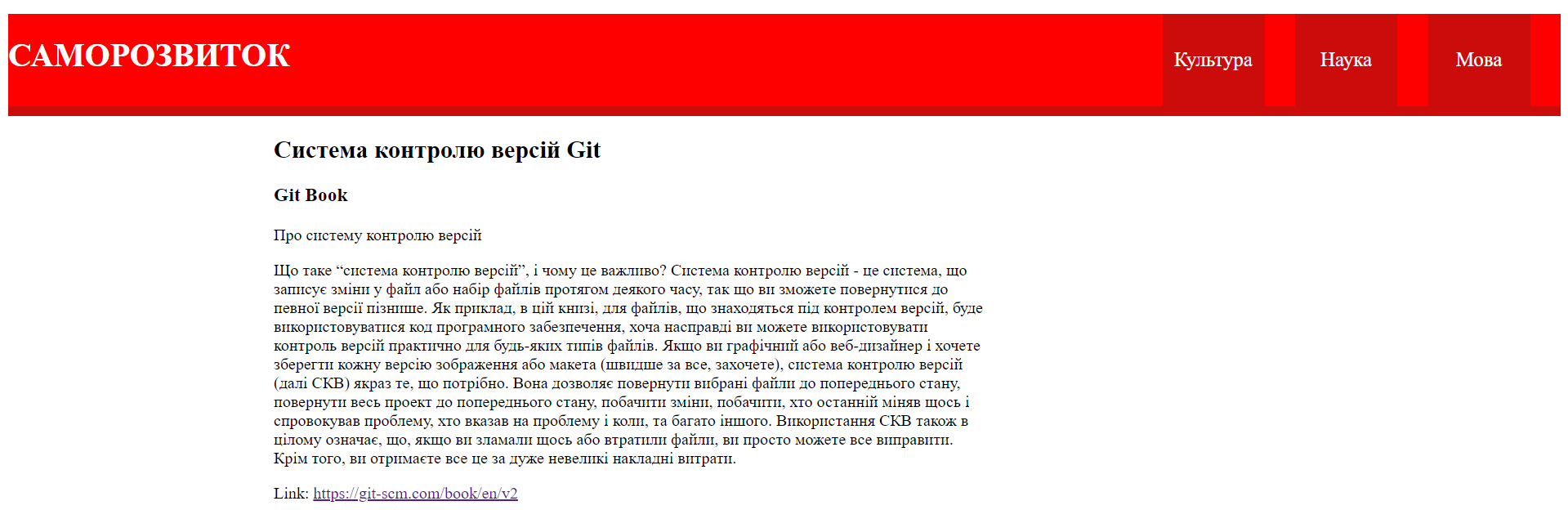
article{

width:700px;

float:left;

}

9. Відкрийте файл **index.html** у браузері та порівняйте відображення сторінки з еталонним (рисунок 2.)



*Рис.2. Еталонне відображення сторінки.*

10. Задайте команду **git status** для перевірки стану репозиторію. Ознайомтеся із повідомлення системи. Зробіть скріншот вікна повідомлення.

11. Додайте файл **css/default.**css до відслідкування та зробіть відповідний коміт.

**Завдання 7. Модифікування файлів та визначення статусу.**

1. Змініть файл **index.html** додавши CDN підключення Bootsrap.

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>

Збережіть зміни.

2. Задайте команду **git status** для перевірки стану репозиторію. Ознайомтеся із повідомлення системи. Зробіть скріншот вікна повідомлення.

3. Додати файл index.html до відслідкування Git командою **git add index.html**

4. Збереження відтиску (**commit**). Задайте команду **git commit** повідомленням (опція -m) «Друга версія проекту»: **git commit** **-m «Add CDN Bootstrap index.html».**

5. Відкрийте файл **index.html** у браузері та порівняйте із зразком (рисунок 3.)



Рис. 3. *Зразок відображення сторінки з підключенням Bootstrap.*

**Завдання 8. Перегляд історії комітів.**

1. Переглянути історію комітів які ви здійснили командою **git log**

Примітка: для завершення перегляду натисніть клавішу Q.

2. Переглянути скорочену історію комітів додавши параметр **--stat**.

Виконайте команду **git log –stat** та проаналізуйте вивід.

3. Перегляд останніх двох комітів. Додайте до команди число -2.

Виконайте команду **git log --stat -2** та проаналізуйте вивід.

4. Зміна виводу інформації про коміти при застосуванні параметру **--pretty**. Опція oneline друкує кожен коміт в один рядок, що корисно, якщо ви дивитесь на багато комітів.

Виконайте команду **git log --pretty=oneline** тапроаналізуйте вивід.

5. Задайте команду для виводу лише трьох останніх комітів в один рядок.

**Завдання 9. Створення облікового запису на GitHub.**

Створити обліковий запис на GitHub. Запам’ятайте логін та пароль до нього. Налаштуйте зображення аватарки.

**Завдання 10. Підключення віддаленого репозиторію з навчальними матеріалами.**

1. Створіть каталог EDU\_Botstrap на вашому комп’ютері.

2. Перейдіть в термінал (Visual Studio Code або GitBash). Активізуйте даний каталог.

3. Ініціалізуйте локальний каталог в системі Git: **git init**

4. Створіть клан віддаленого репозиторії з вашим локальним.

Команда: **git clone** [**https://github.com/Volodymyr-Kachurivskyi/Learn\_BS.git**](https://github.com/Volodymyr-Kachurivskyi/Learn_BS.git)

Ви отримаєте усі файли які є репозиторії. Перегляньте репозиторій на вашому комп’ютері.

Примітка: Для відстеження змін віддаленого репозиторію виконайте команду **git fetch** та **git pull**

**Завдання 11. Створення репозиторію на GitHub для зберігання виконаних лабораторних робіт та самостійної роботи.**

1. Створіть новий ***приватний*** репозиторій на під таким комбінованим іменем: Lab\_BS\_<ВАШЕ ПРІЗВИЩЕ латиницею>. В цьому репозиторію ви будете розміщувати файли виконання завдань до лабораторних та самостійних робіт дисципліни

2. Створіть у себе на комп’ютері аналогічний каталог Lab\_BS\_<ВАШЕ ПРІЗВИЩЕ латиницею>.

3. Перейдіть в термінал (Visual Studio Code або GitBash). Активізуйте даний каталог.

4. Ініціалізуйте локальний каталог в системі Git: **git init**

5. Клонуйте віддалений репозиторій на GitHub Lab\_BS\_<ВАШЕ ПРІЗВИЩЕ латиницею> з локальним репозиторієм.

6. Задайте локальні параметри репозиторію:

**git config --local user.name <name>**

**git config --local user.email <email>**

7. Проведіть редагування файлу README.md який розміщений в локальному репозиторії. Додайте довідкову інформацію про вас.

8. Виконайте коменти додавання файлу. Створить коміт.

9. Відправите файли на віддалений репозиторій **git push**

10. Перейдіть до віддаленого репозиторію та перевірте наявність змін у ньому.

**Завдання 12. Підключення до репозиторію GitHub інших користувачів для колаборативної роботи**.

1. Підключення викладача до віддаленого репозиторію Lab\_BS\_”name” на GitHuB.

Зайдіть в Setting → Collaborations. Додайте в команду проекту викладача **Volodymyr-Kachurivskyi**.

2. Підключення колеги до власного репозиторію.

Підключіть одного колегу до репозиторію.