Day 1 – Git Version Control + HTML Basics (6 Hours Detailed Plan)

⏰ Session 1: Concept Clarification

✅ PART 1: Introduction to Version Control (Git)

✉️ **Concepts:**

- What is Version Control?
  - Version Control is a system that **keeps track of changes** in your files (like code, documents, etc.) over time.
    It helps you go **back to an older version** if something goes wrong.
  - 🧠 **Think of it like:**
    Undo/Redo option in MS Word — but for your **entire project**!

- Why Git?
  - Git is the **most popular version control tool**.
    It helps you **save, manage, and share** your code easily.
  - 🛠️ It works locally on your computer and also with online platforms like **GitHub**.
  - 📦 Git keeps your entire project history safe and helps **multiple people work together**.

- Benefits: Track changes, revert mistakes, collaborate easily.
  - ✅ **Track Changes** – See what was changed, when, and by whom.
    ✅ **Revert Mistakes** – Go back to a previous working version anytime.
    ✅ **Collaborate Easily** – Work with teammates without overwriting each other's work.
  - 💡 Using Git saves time, avoids confusion, and makes you a **professional developer**.

⚖️ **Real Life Example:**

"Google Docs version history" ≠ Static Word Doc. Similarly, Git tracks file changes over time.

📃 **Important Git Commands:**

```
git init                    # Initialize git repository
git status                  # Show current status
git add <filename>          # Stage file
git commit -m "message"     # Commit file to repo
git log                     # Show commit history
```

📍 **Step-by-Step Example:**

```
mkdir mca-git-demo
cd mca-git-demo
echo "Hello Git!" > index.txt
git init
git add index.txt
git commit -m "Initial commit"
git status
git log
```

✅ PART 2: Branching, Merging, Conflict Resolution – 25 mins

💼 **Concepts:**

- Create new feature branch
  - A **branch** is like a copy of your project where you can work on **new features** separately.
  - 🧠 **Think of it like:** Making a rough draft without changing the original.
  - 📌 Command:
  - `git checkout -b feature-name`

- Merge to main
  - Once your feature is ready, you **merge** it into the main project (main branch).
  - 🧠 It's like saying: "I've completed the new part — now add it to the main copy."
  - 📌 Command:
  - `git checkout main`
  - `git merge feature-name`

- Handle conflicts
    - If two branches changed the **same line** of code, Git gets confused — this is called a **conflict**.
    - 🛠️ You have to manually **decide which version to keep**, then continue.
    - 📌 After fixing:
    - `git add <file>`
    - `git commit -m "Resolved conflict"`
    - 🧠 **Think of it like:** Two people edited the same sentence — you must choose the final version.

🔧 **Commands:**

```
git branch              # List branches
git checkout -b feature-1   # Create + switch to new branch
# Make some changes
git commit -am "Changes from feature-1"
git checkout main          # Go back to main branch
git merge feature-1        # Merge feature-1 into main
```

❌ **Conflict Example:**

- Modify same line in index.txt on both main and feature-1
- Git will show conflict
- Use VS Code or CLI to resolve manually

```
git add index.txt          # After resolving
git commit -m "Resolved conflict"
```

✅ PART 3: GitHub Repo and Remote – 20 mins

👨‍💼 **Concepts:**

- GitHub = remote version control
    - GitHub is an **online platform** where you can **store your Git projects** safely on the cloud.
    - 🧠 **Think of it like:** Google Drive — but for your code.
      You can **share, view, and manage** your code from anywhere.

- Create repo on GitHub
  - A **repo (repository)** is like a folder for your project on GitHub.
  - 🎯 When you create a new repo on GitHub, you're preparing a **place to upload your code** online.
  - 📌 Go to: `https://github.com` → Click **New** → Name your repo → Click **Create Repository**

- Link with local project
  - After creating the repo on GitHub, you need to **connect it with your local Git project**.
  - 🧠 This link lets you **push (upload)** and **pull (download)** code between your computer and GitHub.
  - 📌 Command Example:
  - `git remote add origin https://github.com/yourname/project-name.git`
  - `git push -u origin main`

🔧 **Commands:**

```
git remote add origin https://github.com/yourname/demo.git
git push -u origin main      # First push
git pull origin main         # Pull changes
git clone https://github.com/yourname/demo.git   # Clone project
```

⏰ Session 2: HTML Basics (1 Hour)

📁 Topics Covered:

- HTML Boilerplate
- Headings, Paragraphs
- Lists: Ordered & Unordered
- Links and Images
- Text formatting

1️⃣ HTML Boilerplate

👉 Boilerplate means the **basic structure** of every HTML page.
It tells the browser: "Hey, this is an HTML document!"

🧱 **Code Example:**

```
<!DOCTYPE html>

<html>

<head>

  <title>My First Page</title>

</head>

<body>

  <!-- Content goes here -->

</body>

</html>
```

🧠 **Quick Notes:**

- `<!DOCTYPE html>` → Declares HTML5
- `<html>` → Root element
- `<head>` → Page setup (title, metadata)
- `<body>` → Visible content (text, images, etc.)

2️⃣ Headings & Paragraphs

👉 Headings are used to show **titles or section names**
👉 Paragraphs are used for **regular text content**

🧱 **Code Example:**

```
<h1>Main Title</h1>

<h2>Subheading</h2>

<p>This is a paragraph. It gives information about the topic.</p>
```

🧠 `<h1>` to `<h6>` – H1 is biggest, H6 is smallest
🧠 `<p>` – Wraps a block of text

3️⃣ Lists: Ordered & Unordered

👉 Lists are used to show multiple items.

🔷 **Unordered List (bullets)**

```
<ul>

  <li>HTML</li>

  <li>CSS</li>

  <li>JavaScript</li>

</ul>
```

🔷 **Ordered List (numbers)**

```
<ol>

  <li>Download VS Code</li>

  <li>Install Git</li>

  <li>Create GitHub Account</li>

</ol>
```

🧠 `<ul>` = unordered
🧠 `<ol>` = ordered
🧠 `<li>` = list item

4️⃣ Links and Images

👉 Links let users click and go to other websites or pages
👉 Images are used to display pictures

🧱 **Link Example:**

```
<a href="https://github.com">Visit GitHub</a>
```

🧱 **Image Example:**

```
<img src="https://via.placeholder.com/150" alt="Placeholder Image">
```

🧠 `<a>` = Anchor tag
🧠 `href=""` = Destination URL
🧠 `<img>` = Image tag
🧠 `src=""` = Image link
🧠 `alt=""` = Text shown if image doesn't load

5️⃣ Text Formatting

👉 These tags **change the look** of the text.

🧱 **Examples:**

```
<strong>This is bold text</strong><br>

<em>This is italic text</em><br>

<u>This is underlined</u><br>
```

🧠 `<strong>` → Bold
🧠 `<em>` → Italic
🧠 `<u>` → Underline
🧠 `<br>` → Line break (no closing tag)

🔧 Sample Code:

```
<!DOCTYPE html>
<html>
<head>
  <title>My HTML Page</title>
```

```
</head>
<body>
  <h1>Hello, MCA Students!</h1>
  <p>This is a paragraph.</p>

  <ul>
    <li>HTML</li>
    <li>CSS</li>
  </ul>

  <ol>
    <li>Step One</li>
    <li>Step Two</li>
  </ol>

  <img src="https://via.placeholder.com/150" alt="Placeholder Image">
  <a href="https://github.com">Visit GitHub</a>
</body>
</html>
```

✉️ Quick Explanation:

- `<!DOCTYPE html>`: Declares HTML5
- `<h1>` to `<h6>`: Headings
- `<ul>` / `<ol>`: Lists
- `<a>`: Link tag
- `<img>`: Image tag

⏰ Session 3: Hands-on Tasks (3 Hours)

✅ Task 1: Git Practice (1 Hour)

1. Create folder `mca-html-demo`
2. Create README.md file
3. Initialize Git
4. Create new branch and merge
5. Commit all steps

```
echo "# My HTML Project" > README.md
git init
```

```
git add README.md
git commit -m "Added README"
git checkout -b html-page
```

✅ Task 2: HTML Page Development (1.5 Hours)

- Create a simple profile page:
  - Heading: Your Name
  - Paragraph: About you
  - Skills: List (ul)
  - Image: Your photo or placeholder
  - Link: GitHub profile

✅ Task 3: Push to GitHub (30 Mins)

1. Create GitHub Repo
2. Add remote origin
3. Push code
4. Share public repo link

```
git remote add origin https://github.com/yourname/html-demo.git
git push -u origin main
```

⏰ Session 4: Quiz (30 Minutes)

✏ Quiz Format:

- 10 MCQs
- 5 Practical Questions

💡 Sample MCQs:

1. What is the use of git add?
2. Which tag is used for an image?
3. What is git status used for?
4. Which tag makes text bold?
5. Command to push code to GitHub?

📑 Sample Practical:

- Create a page with heading, image, and link
- Show git commit history
- Create a new branch
- Resolve a conflict
- Push to GitHub

📅 Preparation Checklist:

- Install Git & VS Code
- Create GitHub Account
- Internet Access
- Placeholder images or use online URLs

📅 Preparation Checklist:

✅ VS Code Installed

- Download from: https://code.visualstudio.com
- Install with default settings
- Install **Live Server Extension** for HTML testing

✅ Git Installed

- Download from: https://git-scm.com
- Install with Git Bash and GUI option checked
- Configure with:

```
git config --global user.name "Your Name"

git config --global user.email "your@email.com"
```

✅ GitHub Account Ready

- Sign up at: https://github.com
- Use verified email ID
- Create your first test repo before class

✅ Placeholder Images

- Use: https://via.placeholder.com
- Example: `<img src="https://via.placeholder.com/150" alt="Sample">`
- Or download images in advance and keep in project folder

✅ Sample Repo Structure

```
project-folder/
├── index.html
├── images/
│   └── profile.jpg
├── README.md
└── .git/
```

- Keep simple and clean folder setup
- Ensure folder is Git initialized before pushing
