



# Welcome to this **CoGrammar** Lecture: Git Workflow and Collaboration

The session will start shortly...

Questions? Drop them in the chat.  
We'll have dedicated moderators  
answering questions.



# Software Engineering Session Housekeeping

---

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.

## **(Fundamental British Values: Mutual Respect and Tolerance)**

- No question is daft or silly - **ask them!**
- There are **Q&A sessions** throughout this session, should you wish to ask any follow-up questions.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: [Questions](#)

## Software Engineering Session Housekeeping cont.

---

- For all **non-academic questions**, please submit a query: [www.hyperiondev.com/support](https://www.hyperiondev.com/support)
- Report a **safeguarding** incident: [www.hyperiondev.com/safeguardreporting](https://www.hyperiondev.com/safeguardreporting)
- We would love your **feedback** on lectures: [Feedback on Lectures](#)

# Enhancing Accessibility: Activate Browser Captions

---

## Why Enable Browser Captions?

- Captions provide **real-time text for spoken content**, ensuring inclusivity.
- Ideal for individuals in noisy or quiet environments or for those with **hearing impairments**.

## How to Activate Captions:

### 1. YouTube or Video Players:

- Look for the CC (Closed Captions) icon and click to enable.

### 2. Browser Settings:

- Google Chrome: Go to *Settings > Accessibility > Live Captions* and toggle ON.
- Edge: Enable captions in *Settings > Accessibility*.

# Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:



Ian Wyles  
Designated Safeguarding  
Lead



Simone Botes



Nurhaan Snyman



Rafiq Manan



Ronald Munodawafa



Tevin Pitts

Scan to report a  
safeguarding concern



or email the Designated  
Safeguarding Lead:  
Ian Wyles

[safeguarding@hyperiondev.com](mailto:safeguarding@hyperiondev.com)

# ***Stay Safe Series:***

Mastering Online Safety One Week or Step at a Time

---

While the digital world can be a wonderful place to make education and learning accessible to all, it is unfortunately also a space where harmful threats like online radicalisation, extremist propaganda, phishing scams, online blackmail and hackers can flourish.

As a component of this BootCamp the ***Stay Safe Series*** is designed to guide you through essential measures in order to protect yourself & your community from online dangers, whether they target your privacy, personal information or even attempt to manipulate your beliefs.

## Trustworthy Websites: How to Spot Secure Sites

---

- Look for the padlock.
- Check if there is a valid SSL/TLS certificate.
- **Look for a site seal.**
- **Check if the URL is legitimate.**
- Pop-up and Redirection ads are a red flag.



# Skills Bootcamp Progression Overview

## ✓ Criterion 1 - Initial Requirements

Specific achievements **within the first two weeks** of the program.

To meet this criterion, students need to, by no later than **01 December 2024**:

- **Guided Learning Hours (GLH):** Attend a **minimum of 7-8 GLH per week** (lectures, workshops, or mentor calls) for a total minimum of **15 GLH**.
- **Task Completion:** Successfully complete the **first 4 of the assigned tasks**.

## ✓ Criterion 2 - Mid-Course Progress

Progress through the successful completion of tasks **within the first half** of the program.

To meet this criterion, students should, by no later than **12 January 2025**:

- **Guided Learning Hours (GLH):** Complete at least **60 GLH**.
- **Task Completion :** Successfully complete the **first 13 of the assigned tasks**.



# Skills Bootcamp Progression Overview

## ✓ Criterion 3 – End-Course Progress

Showcasing students' progress nearing the completion of the course.

To meet this criterion, students should:

- **Guided Learning Hours (GLH):** Complete the **total minimum required GLH**, by the **support end date**.
- **Task Completion :** **Complete all mandatory tasks**, including any necessary resubmissions, by the end of the bootcamp, **09 March 2025**.

## ✓ Criterion 4 - Employability

Demonstrating progress to find employment.

To meet this criterion, students should:

- **Record an Interview Invite:** Students are required to record proof of invitation to an interview by **30 March 2025**.
  - **South Holland Students** are required to proof and interview by **17 March 2025**.
- **Record a Final Job Outcome :** Within 12 weeks post-graduation, students are required to record a job outcome.

**SKILLS  
FOR LIFE**

**SKILLS BOOTCAMPS**



Department  
for Education

# CoGrammar Git Workflow and Collaboration

# Learning Objectives & Outcomes

- Perform Git Collaboration and Best Practices
- Explain branching strategies (feature branches, master/main)
- Describe merging branches and resolving conflicts
- Explain and describe .gitignore
- Explain and perform to Pull Requests (PRs) and code reviews (via GitHub)
- Resolve merge conflicts effectively.
- Assess the impact of version control on collaboration.
- Collaborate on a shared project using remote repositories and platforms like GitHub.



# Git Workflow & Collaboration in Enterprise

## Google Android Development

- 2000+ developers
- 40+ million lines of code
- 40,000+ commits monthly
- Git-based version control essential

## Microsoft's Windows Development

- World's largest Git repo
- 4000+ developers
- 85+ million lines of code
- Git enables simultaneous development

## Why This Matters:

- Modern software development requires coordinating thousands of developers
- Version control is crucial for managing complex codebases and preventing conflicts
- Git skills are essential for working in any modern development team
- Understanding Git workflow is a fundamental skill for career growth in tech

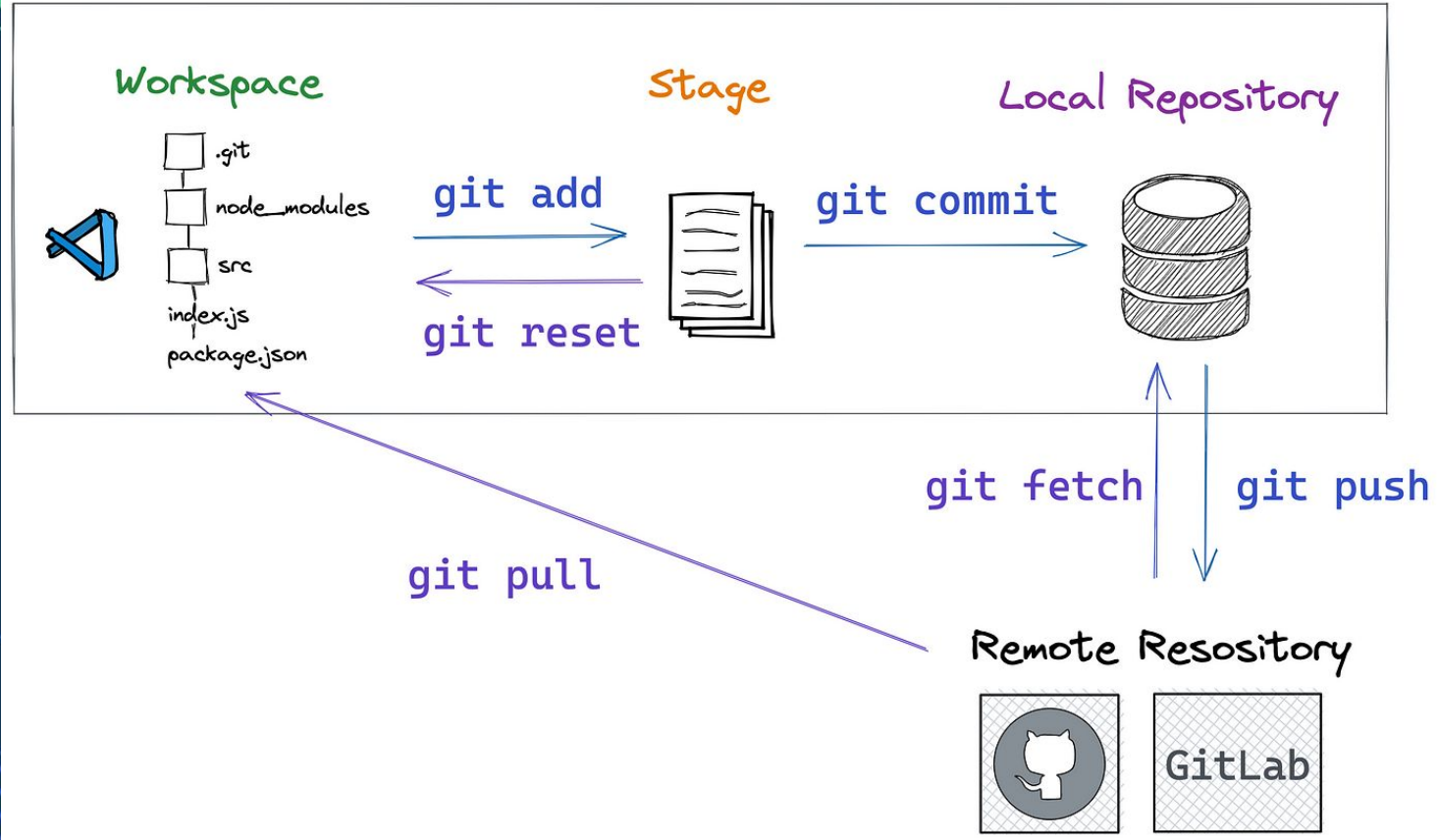


# Introduction



# Recap

Local



# Branching Strategies







# What Are Branches in Git?

- **What is a branch?**
  - A branch is like a separate workspace in your project where you can work without affecting the main code.
- **Purpose of the Main/Master Branch:**
  - The *main branch* (or master) is the stable, production-ready version of your project.
  - Changes merged here should always work as intended.
- **Why Protect the Main Branch?**
  - Prevent accidental changes by requiring reviews before merging.
  - Keep the project stable for releases or deployments.

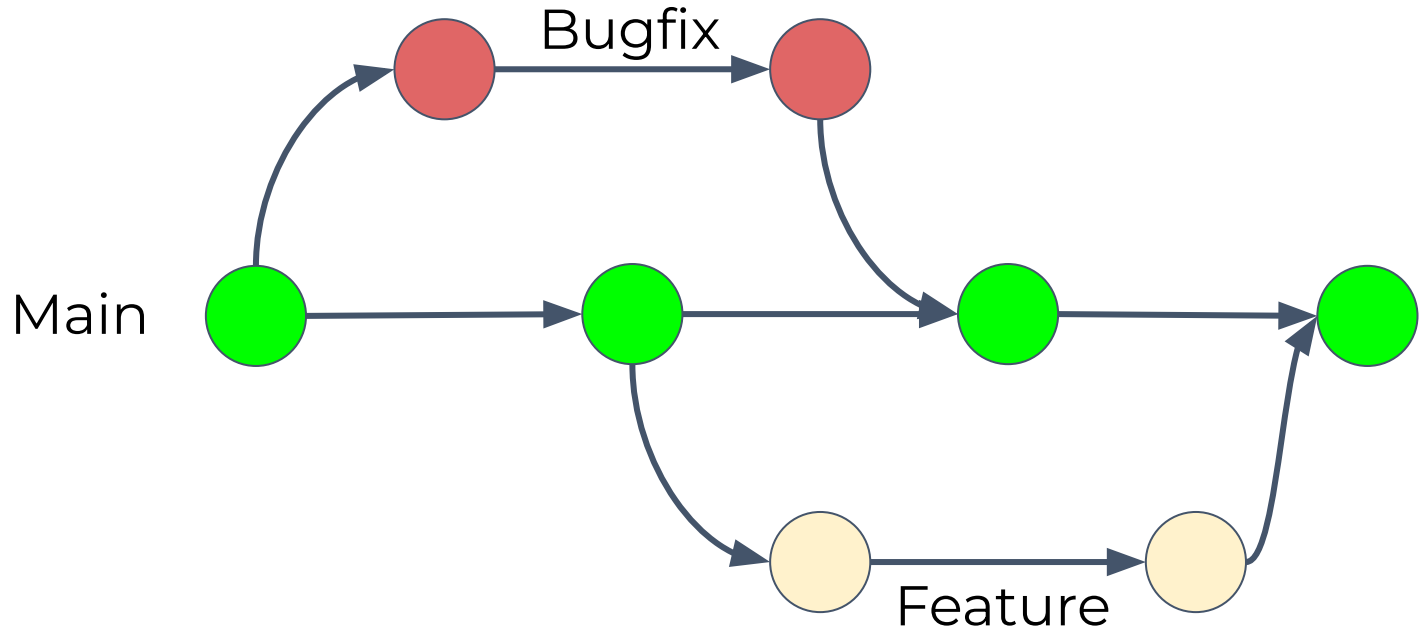
# Feature Branches: Work Without Worry

- **What is a Feature Branch?**
  - A branch dedicated to developing a specific feature or fix.
  - Allows independent work without affecting others.
- **Why Use Feature Branches?**
  - Keep the main branch stable.
  - Organize your work better.
- **Branch Lifecycle:**
  - Create → Work → Test → Merge → Delete

# Feature Branches: Work Without Worry

- **Good Naming Practices:**
  - Be descriptive but concise.
  - Use formats like:
    - `feature/new-login-page`
    - `bugfix/fix-login-issue`
    - `hotfix/urgent-deploy-fix`
- **Why Naming Matters:**
  - Clear names help the team understand the branch's purpose.
  - Avoids confusion in collaborative projects.
- **Tip:**
  - Use lowercase and dashes (-) for readability.
- Example list of well-named branches vs. poorly named branches:
  -  `feature/add-user-auth`
  -  `1234` or `newbranch`.

# Branching Strategies: A Visual Guide



# Collaboration Fundamentals



# Understanding Remote and Local Repositories

- **What is a Remote Repository?**
  - A version of your repository hosted online (e.g., GitHub).
  - Accessible by your entire team, enabling collaboration.
- **Local vs Remote:**
  - *Local*: Stored on your computer.
  - *Remote*: Shared on a platform for teamwork.
- **Why Use GitHub? (Or GitLab, Bitbucket).**
  - Easy sharing, and collaboration tools, collaborate with others, back up your code, track changes over time.

# What is a Pull Request?

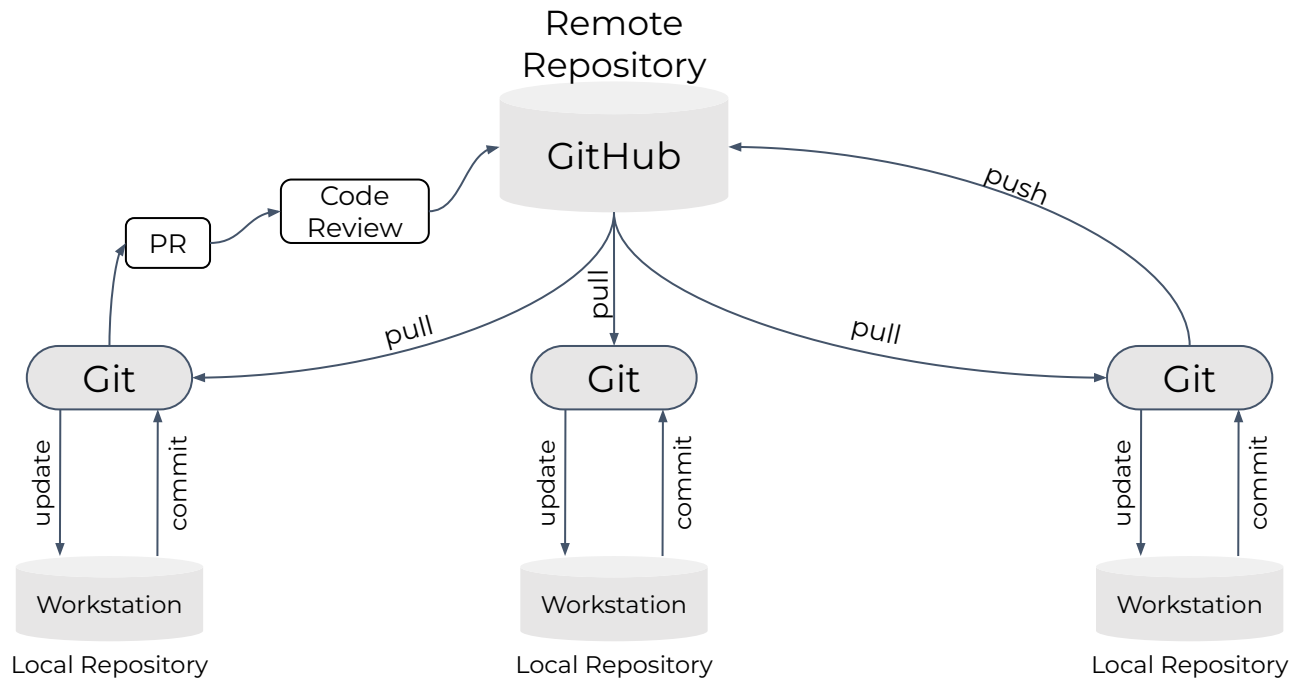
- **Purpose of PRs:**
  - A formal request to review and merge your work into a shared branch.
  - Encourages collaboration and ensures quality control.
- **When to Create a PR?**
  - After completing a feature or fix in your branch.
- **PR Workflow:**
  1. Submit a PR on GitHub.
  2. Request reviews from team members.
  3. Address feedback and make changes if needed.
  4. Merge your branch when approved.

# Code Reviews: Building Better Software Together

- **Why Code Reviews Matter:**
  - Catch bugs or issues early.
  - Ensure adherence to team coding standards.
  - Encourage knowledge sharing among team members.
- **Writing Effective PR Descriptions:**
  - Be clear and concise.
  - Describe *what* you did and *why* you did it.
  - Mention any specific areas needing attention during the review.
- **Best Practices for PRs:**
  - Small, focused changes (don't lump multiple features).
  - Respond to feedback promptly.



# Pull Requests and Code Reviews



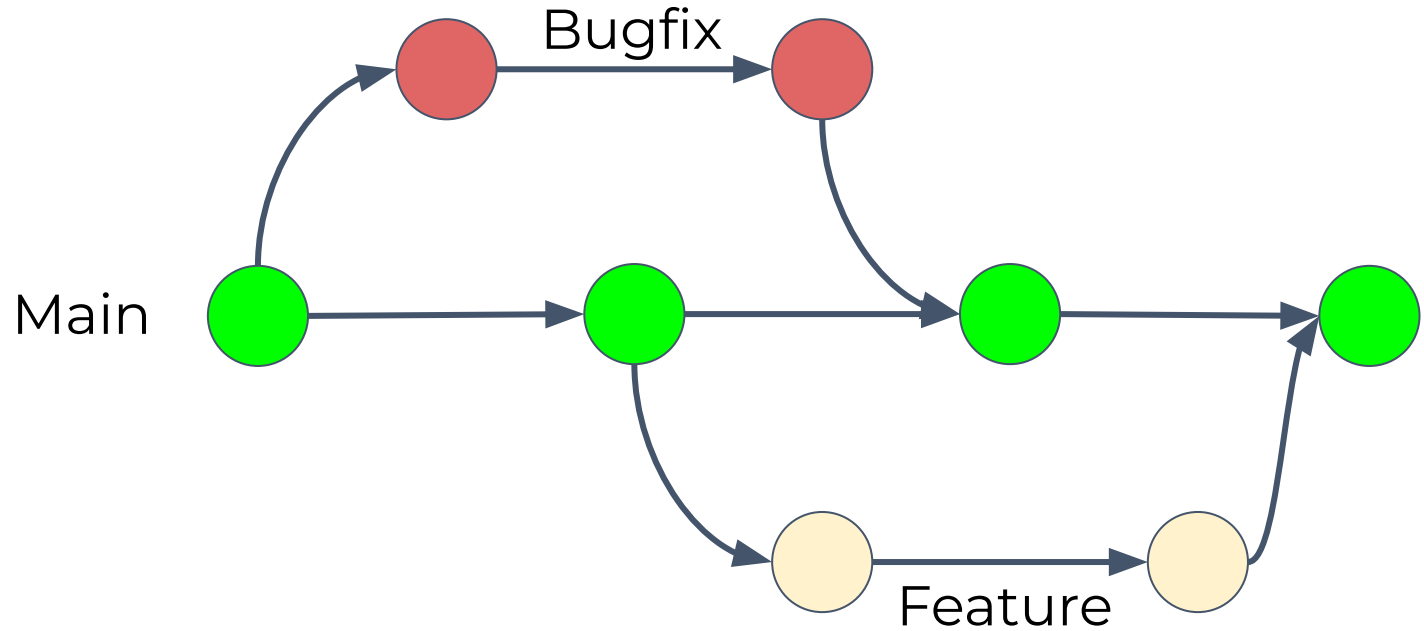
# Merge Operations & Conflict Resolution



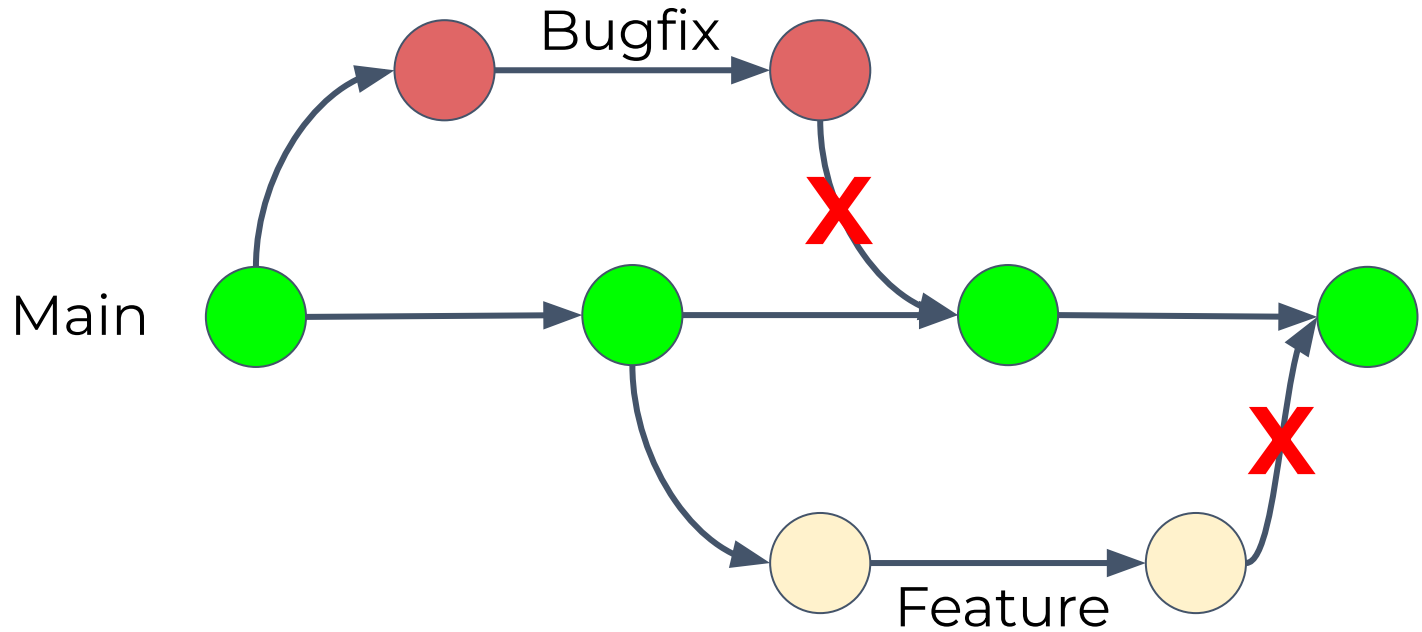
# What is a Merge and Why Do We Need It?

- **What is a Merge?**
  - Combining changes from one branch into another.
  - Ensures all features and fixes come together.
- **When to Merge:**
  - After completing a feature or fix in a feature branch.
  - Before a release to ensure all work is integrated.

# What Are Branches in Git?



# Handling Merge Conflicts



# Handling Merge Conflicts

- **Why Do Merge Conflicts Happen?**
  - Two branches modify the same line in a file.
  - Changes are made to the same file in ways Git cannot automatically combine.
- **How to Resolve Conflicts:**
  1. Identify the conflict (Git will mark files with conflicts).
  2. Edit the file to keep the desired changes.
  3. Mark the conflict as resolved (`git add`).
  4. Complete the merge (`git commit`).
- **Tools to Simplify Conflict Resolution:**
  - VS Code Git integration.
  - GitHub's conflict resolution editor.

# Best Practices & Tools





# Managing What Git Tracks with .gitignore

- **What is .gitignore?**
  - A file to tell Git which files or directories to ignore (not track).
  - Keeps sensitive or irrelevant files out of the repository.
- **Common Use Cases:**
  - Ignoring files like logs, temporary files, and environment variables.
  - Excluding OS or editor-specific files (e.g., `.DS_Store`, `*.swp`).
- **How to Use .gitignore:**
  - Add a `.gitignore` file at the root of your project.
  - Use patterns to specify ignored files (e.g., `*.log`, `/node_modules/`).
- **Best Practice:**
  - Always add `.gitignore` when initializing a project to avoid tracking unnecessary files.



# Guidelines for Effective Code Reviews

- **What is Code Review?**
  - A systematic examination of code by peers to improve quality and ensure adherence to team standards.
- **Review Guidelines:**
  - Focus on the code, not the person.
  - Check for functionality, readability, and adherence to standards.
  - Ensure the code is well-tested.
- **Providing Constructive Feedback:**
  - Be specific: "Consider renaming this variable to make it clearer."
  - Be polite: "What if we refactor this function for better readability?"
  - Avoid negative or personal comments.

# Using GitHub's Review Features Effectively

- **GitHub Review Features:**
  - Leave inline comments on specific lines of code.
  - Approve or request changes on pull requests.
  - Use suggestions for quick fixes.
- **Best Practices for Reviewers:**
  - Understand the feature's purpose before reviewing.
  - Test locally if necessary.
  - Avoid nitpicking minor issues unless they impact functionality.
- **Best Practices for Submitters:**
  - Write clear commit messages.
  - Use meaningful PR descriptions (what/why/how).
  - Address feedback promptly and update your PR.

## Poll

What is a common reason merge conflicts occur in Git?

1. Two developers created new files with the same name in different branches.
2. The same line in a file was modified by different branches.
3. Git cannot track changes made to binary files.

## Poll

You've accidentally committed sensitive API keys to your repository. What's the most effective long-term solution?

1. Delete the commit and force push to remove the history
2. Add the file to .gitignore, rotate the API keys, and store them in environment variables
3. Simply remove the keys from the current version and commit the change

# Lesson Conclusion and Recap

Recap the key concepts and techniques covered during the lesson.

- **Branching Strategies:** Key strategies like feature branches and main branch usage help organise work and streamline collaboration.
- **Merging and Conflict Resolution:** Merging branches and handling merge conflicts ensure smooth integration of changes from different contributors.
- **Using .gitignore:** The .gitignore file helps manage which files to track or ignore, keeping the repository clean and focused.
- **Pull Requests (PRs) and Code Reviews:** PRs and code reviews support collaborative development, allowing team members to review, discuss, and improve code before merging.
- **Remote Repositories and GitHub Collaboration:** Leveraging platforms like GitHub enhances teamwork, making it easier to share, collaborate, and track project progress.

# Resources

## Resources

- **Software:**

- [Git - Downloading Package](#)
- [Download GitHub Desktop](#)

- **Additional Resources**

- [Hello World - GitHub Docs](#)
- [Get started with GitHub documentation](#)

- **Books:**

- [Pro Git book](#)

# Questions and Answers





# Thank you for attending



Department  
for Education

CoGrammar

