Cornell WebDev Club

Workshop #4: Final Project & Deployment

April 15, 2025

Attendance

Agenda:

1.	• Project Structuring and Best Practices
2.	• GitHub Collab
3.	• Deploying your Project
4.	• Homework

Goal:

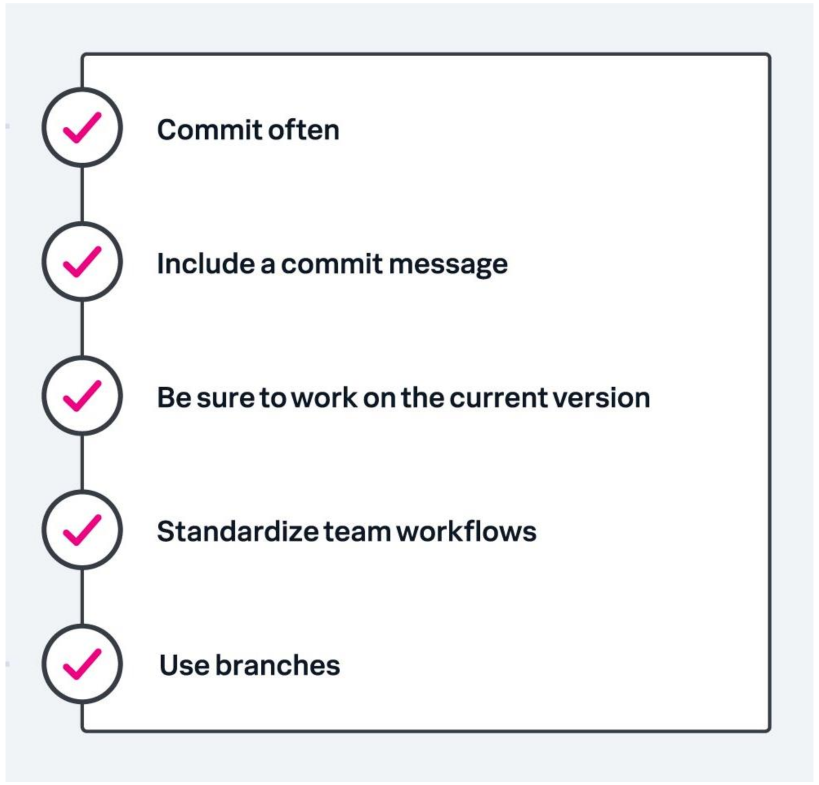
Participants will finish the workshop with a fully integrated web project, showcasing a static website enhanced with interactivity and connected to a basic backend service.

1. Project Structuring and Best Practices

1. Project Structuring and Best Practices

1. Organizing and Managing Code

- **Frontend & Backend Separation –** keep a clear distinction between frontend (UI) and backend (databases)
- **Modular Architecture –** break down code into different components, services, and modules
- **Folder Structure –** clearly label and maintain logical structure of folders
  - ex) /src, /components, /services, /routes
- **Scalability & Maintainability –** efficiently handle increasing workloads by using strategies like vertical and horizontal scaling
  - Vertical Scalability = scaling up and increases system's capability by upgrading existing hardware (storage, RAM)
  - Horizontal scaling = scaling out and adding more machines or instances to distribute the load



✓ Commit often

✓ Include a commit message

✓ Be sure to work on the current version

✓ Standardize team workflows

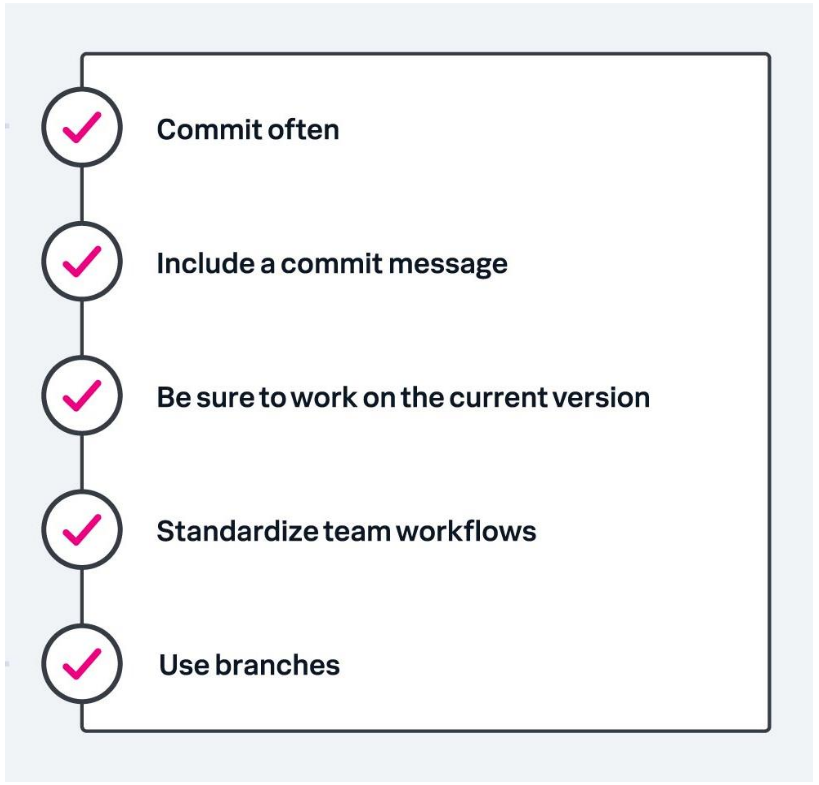
✓ Use branches

1. Project Structuring and Best Practices

1. Organizing and Managing Code
2. **Clean Code & Best Practices**
   - **Consistent Naming Conventions –** use meaningful, readable variable and function names
     - Use CamelCase to name your files and functions
     - ex) NavBar.js
   - **Code Readability –** keep functions small, use comments where necessary, and follow DRY (Don't Repeat Yourself)

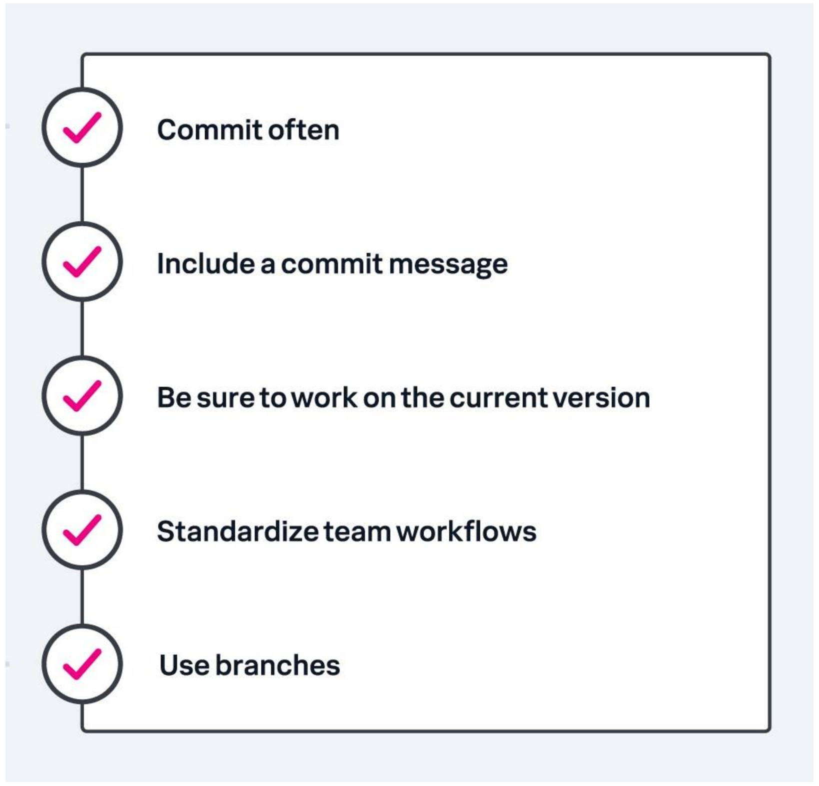
```
print(data.get("name", "Name not found"))
```

   - **Error Handling –** implement proper logging and exception handling
     - Include print statements to ensure code is working
   - **Performance Optimization –** minimize redundant operations and optimize API calls


✓ Commit often
✓ Include a commit message
✓ Be sure to work on the current version
✓ Standardize team workflows
✓ Use branches

1. Project Structuring and Best Practices

1. Organizing and Managing Code
2. Clean Code & Best Practices
3. Version Control (Git Basics)
   - **Branching Strategy –** follow, main, dev, and feature branches workflow
   - **Commit Best Practices –** write clear, specific commit messages, commit often, and avoid large commits
   - **Code Reviews & Collaboration –** use pull requests BEFORE anything and conduct peer reviews
   - **CI/CD Integration –** automate testing and deployment for smoother releases
     - CI = Continuous Integration
     - CD = Continuous Deployment
       - Both automate code building, testing, and deployment, ensuring faster, reliable, and consistent software releases with minimal manual intervention


Commit often
Include a commit message
Be sure to work on the current version
Standardize team workflows
Use branches

2. GitHub Collab

2. GitHub Collab

GitHub Collab = workflow where multiple developers contribute to a shared repository using branches, commits, pull request, and code reviews to manage and merge changes efficiently

Steps to GitHub Collab

1. Clone the Repository

```
git clone <repository-url>
cd <repository-name>
```

2. Create a New Branch

```
git checkout -b feature-branch
```

3. Make Changes & Commit

```
git push origin feature-branch
```

4. Push to GitHub

```
git add .
git commit -m "Your meaningful commit message"
```

5. Create a Pull Request

Best Practices

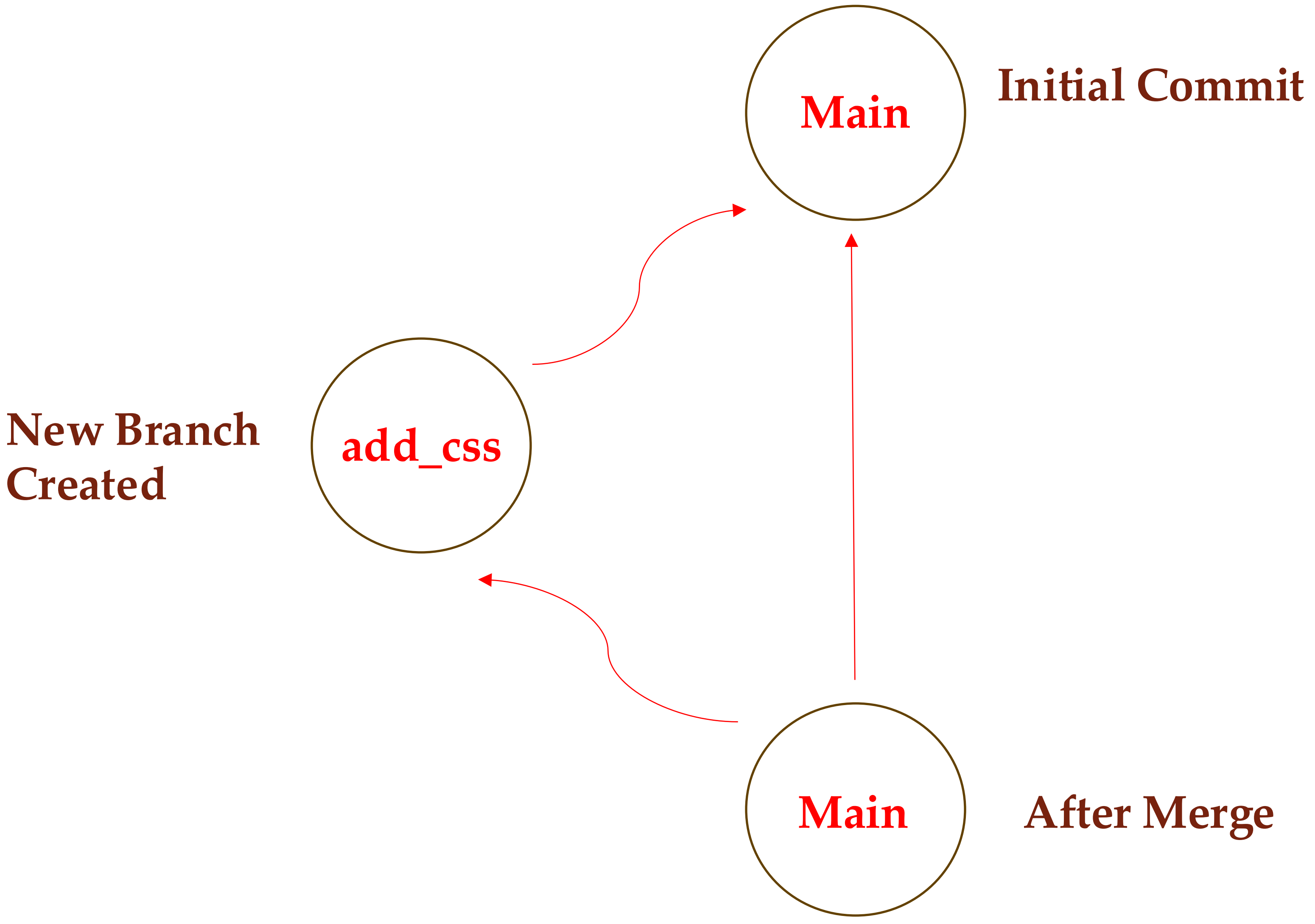
- Use meaningful commit messages
- Keep branches small & specific
- Pull latest changes before working:

```
git pull origin main
```

- Resolve merge conflicts carefully

Demo: GitHub Collaboration Practice

3. Git and GitHub Basics



3. Deploying your Project

3. Deploying your Project

Frontend Deployment

- GitHub Pages – great for hosting static sites directly from a repository. Free and easy to set up

- Netlify & Vercel – ideal for modern web apps, offering automatic deployment from GitHub, custom domains, and serverless functions

Backend Deployment

- Heroku – simple, beginner-friendly cloud platform with easy setup for Node.js, Python, and other frameworks

- Vercel & Netlify – serverless functions allow you to run backend logic without managing servers

- AWS/DigitalOcean – advanced options for complete control, useful for handling larger-scale applications

Scalability Considerations

- Auto-Scaling – cloud platforms like AWS, Heroku, and Vercel automatically adjust resources based on demand

- Load Balancing – distributes traffics across multiple servers to prevent overload

- Database Scaling – using read replicas, caching, and optimized queries to handle large data loads efficiently

Demo: Deploying a Project

4. Homework

4. Homework

Homework: Deploying Final Website

Instructions:

- Complete frontend to your bio webpages
- Complete backend to your bio webpages
- Connect backend to frontend
- **Publish your project and share the live URL**

Make Sure to:

- Go through the steps of organizing and managing code
- Consider Best Practices for clean code
- Manage your errors