Sit with your Team!

1

Team Project Kick-off

2

2

Team Project Overview

WolfCafe

© NC State Software Engineering Faculty

3

Project Deadlines

An overview of the project is listed below.

Team Project Part	Start Date	Team Deadline
Iteration 0	H Oct 15	W Oct 21 @ 11:45 pm
Iteration 1A	H Oct 22	W Oct 28 @ 11:45 pm
Iteration 1B	H Oct 29	W Nov 4 @ 11:45 pm
Team Member Peer Evaluation 1		M Nov 9 @ 11:45 pm
Iteration 2A	H Nov 5	W Nov 11 @ 11:45 pm
Iteration 2B	H Nov 12	W Nov 18 @ 11:45 pm
Final Project	H Nov 19	S Nov 21 @ 11:45 pm
Team Demo		H Nov 19 during lab
Reflection	S Nov 21 @ 11:45 pm	M Nov 23 @ 11:45 pm
Team Member Peer Evaluation 2		M Nov 23 @ 11:45 pm

4

Grading

Grade Item	Points
Technical Deliverables	380
Technical Process	130
Project Management	120
Team Collaboration	150
Peer Review (Inspections)	100
Total	**880**

Technical Deliverables: 150 points of the technical deliverables category is for the technical updates to the system. These are required elements.
You may earn extra credit by completing additional tasks beyond the required technical deliverables. Extra credit is maxed out at 90 additional points (~9% extra credit on the project). Extra credit must be planned and approved by your lab instructors.

5

Done Criteria (Functional Enhancements)

All implementation and tests should be PRed into main branch for done criteria evaluation.

- Appropriate requirement(s) describing the element modeled as a use case, including what is logged. The base system's AuditLogger records the user, their role, and the action; see technical-documents/design/security.md.
- Design describing change to the system.
- Working demo of functionality that meets requirements and management approval.
  - Backend demo through API calls
  - Frontend demo through browser
- Frontend should consider accessibility in regards to color/shapes, tab order, and tool tips to support screen readers.
- Passing unit and integration tests that cover at least 70% of lines and 50% of branches in the backend. ./mvnw verify checks these minimums and reports any shortfall as a build warning, and the GitHub Actions build writes the coverage on each run's summary page. A shortfall does not turn the build red, but the minimums must be met for the work to count as done.
- Passing system tests, as outlined in the system test plan, run by hand with actual results recorded.
- Code is commented.

Other deliverables will have other done criteria.

6

AI Expectations

Deliverable	AI use	Rule
Implementation code and automated tests	**Allowed**, under your team's AI policy	Someone other than the person who prompted reviews it before it merges. Everyone on the team must be able to explain any code that ships.
System test plan	**Allowed** to propose test cases	Your team checks every test against the requirements. **Actual results must come from running the test yourself.**
Requirements, design documents (class diagrams, sequence diagrams, wireframes, flow maps, design decisions), privacy policy, users' guide, developers' guide, human flourishing discussion	**Review only**	Your team writes these. AI may point out gaps, ambiguities, or places where a document no longer matches the implementation; your team decides what to change and makes the edit.
Inspection reports, bug reports, CoffeeMaker Investigation findings	**Exploration only**	AI may help you find something worth looking at. The findings and analysis are written by you.
Team goals, ground rules, Team AI Policy, project management plan and estimates	**Not allowed**	These record your team's decisions and judgment.
Weekly teaming reflections, peer evaluations, Team Project reflection	**Not allowed**	Your own words. Never give an AI tool your teammates' names or anything you are writing about them.
In-lab presentation	**Not allowed** for narration or the explanation of your work	If AI helped build what you are demonstrating, say so in the presentation.

The course-wide rules still apply: you verify anything AI produces, and "the AI wrote it that way" is not a defense for a defect.

7

AI Literacy Inventory

8

Problem Statement

9

Problem Statement - Required Elements

- Required Enhancements (Functional Elements)
  - Permissions - integrate CoffeeMaker into WolfCafe and add permissions by role
  - Create/Edit/Delete Staff and Customers
  - Customer Orders
  - View and Fulfill Orders
  - Customer Order Pickup
  - Tax Rate
- Users' Guide
- Developers' Guide
- Privacy Policy
- Human Flourishing

10

Problem Statement - Optional Elements

These elements are extra credit and the point value varies based on the complexity of the proposed/implemented solution

- Automated Frontend Testing
- Additional user role
- Anonymous orders
- Order history
- Refactoring
- Functionality
- Security audit
- Badges
- UI Enhancements
- GitHub Projects (for the whole team project - start this right away, should you select this option)

You need to let us know about which optional elements you plan to work on by the end of Iteration 1 at the latest!

There will be a Google form for this…

11

Project Overview

12

Iteration 0

Anything you don't finish in lecture today, you can work on in lab.

- Team Collaboration: Team Forming (in **lecture**)
  - Team Goals
  - Team Ground Rules
  - Team AI Policy
- Technical Process: Repository Setup (in lab)
- Peer Review: CoffeeMaker project selection (in lab)
- Technical Deliverables: Requirements (in **lecture**)
- Technical Deliverables: System Test Plan (on your own)
- Project Management: Project Management Plan (in lab)
- Project Management: Issues/Tasks (in lab)
- Project Management: Plan for Iteration 0 (in lab)

13

Iteration 1 & 2

Expectations: Each subteam is expected to deliver at least one major piece of functionality (meeting done criteria) each iteration!

Iteration #A – Design & System Test

- Technical Deliverables
  - Product Design
  - Product Development (not graded)
- Technical Process
- Project Management
  - Project Management Plan (w/ updates)
  - Check-in w/ Manager
- Team Collaboration
  - Technical Contribution to Iteration #A
  - Weekly Teaming Reflection
- Peer Review
  - Inspection – Iteration 1 – requirements and STP
  - Inspection – Iteration 2 – code from Iteration 1

Iteration #B – Implementation & Test

- Technical Deliverables
  - Product Development
- Technical Process
- Project Management
  - Project Management Plan (w/ updates)
  - Check-in w/ Manager
- Team Collaboration
  - Technical Contribution to Iteration #A
  - Weekly Teaming Reflection
  - Peer Evaluation
- Peer Review
  - Design Inspection
  - Iteration 1 – Peer Evaluation

14

Final Project & Reflection

Final Project

- Technical Deliverables
  - In-Lab Presentation & Demo (more next slide)
  - Deliver Product
  - Update Technical Documents
- Technical Processes
- Project Management
  - Update Project Management Plans
- Team Contribution
  - Technical Contribution
  - Peer Evaluation

Individual Reflection

- Team Forming
- Requirements
- Configuration management
- GitHub Actions
- Iterations
- Processes
- Communications
- Productivity
- Weekly Communication Reflections
- AI Use
- Feature-based Roles
- Leadership
- Enjoyment
- Advice
- Other

15

In-Lab Presentation & Demo

- You'll do a 8-10 minute presentation about your team project in lab!
- You'll be evaluated on
  - Introduction of your team and project
  - A description of your team's process
  - An overview of each team member's contributions
  - A demo of your team's functionality from the main branch
- Tell a story and showcase how you decided to interpret the problem statement.

16

Team Forming

17

Team Goals

- As a team, identify your goals for the project…
  - Do you want to get an A?
  - Do you want to pass?
  - Do you want to do the minimum necessary to get some grade?
  - Do you want to go above and beyond?
  - Do you want to make the base system better?
  - Do you want to learn a new technology?
- Record your team goals in your team wiki on a page called “Team Goals”

18

Team Ground Rules

- Create a set of team ground rules. Use your responses to the self-assessment to guide the discussion.
- Your Ground Rules must address:
  - Team Procedures
  - Work Quality
  - Team Participation
  - Accountability
  - Consequences for Non-Participation
- Record your Ground Rules in a wiki page called “Ground Rules”

19

Team AI Policy

- **Use for code and tests** — where does your team want AI involved? Generating test cases, scaffolding a component, explaining unfamiliar code, and writing production logic are very different decisions. Say which are in and which are out.
- **Review expectations** — how does AI-generated code get reviewed before it merges? The Team Project rules require that the person who prompted is not the only reviewer. What else does your team require?
- **Acknowledgment** — how does your team mark AI-assisted work so it is visible? Pick **one** mechanism — a note in the pull request description, a commit trailer, or a code comment — so it is consistent and someone can actually check it. This is how your team discloses AI use.
- **What “verified” means** — before AI-assisted code is merged, what has the author actually done? Run it, read it line by line, written a test that fails without it? Set the bar explicitly, because “it looked right” is the default otherwise.

There are other items to consider in the project writeup.

20

Feature-based Roles

Role-Based Development

© NC State Software Engineering Faculty

Role-Based Development

- Roles provide a mechanism for **owning and identifying who should complete what work tasks**
- User Role Division
  - Each developer implements the use case from the perspective of a user (e.g., Service provider, barista, machine fixer)
- Specialization Roles
  - Each developer owns an area of development (e.g., team leader, development leader, QA leader, planning leader)
- Feature-Based Roles
  - Each developer owns an aspect of developing a use case (e.g., backend, frontend – data entry, frontend – data reporting & display)

© NC State Software Engineering Faculty

Feature-Based Roles

Divides functionality into three key areas and specializations

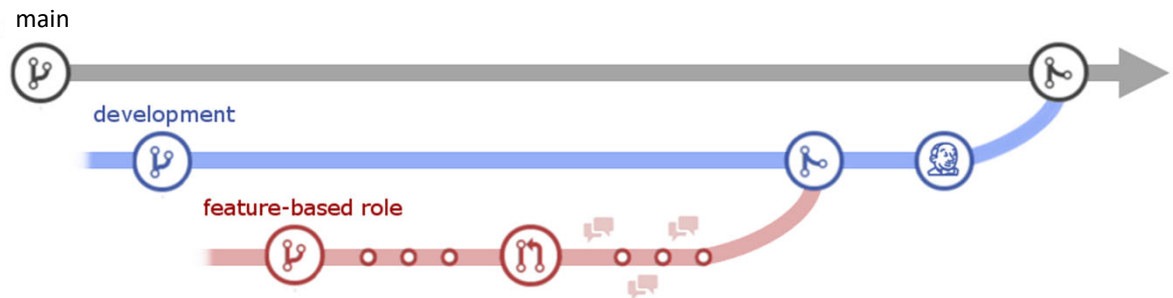
- **Backend**
  - Database/Hibernate + Java backend + API implementation
- **Frontend – Data Entry**
  - Form elements for the functionality
- **Frontend – Data Reporting & Display**
  - Organized reports and display for users

REST API – collaboration between roles to identify what is needed

© NC State Software Engineering Faculty

23

Feature-Based Roles & Git

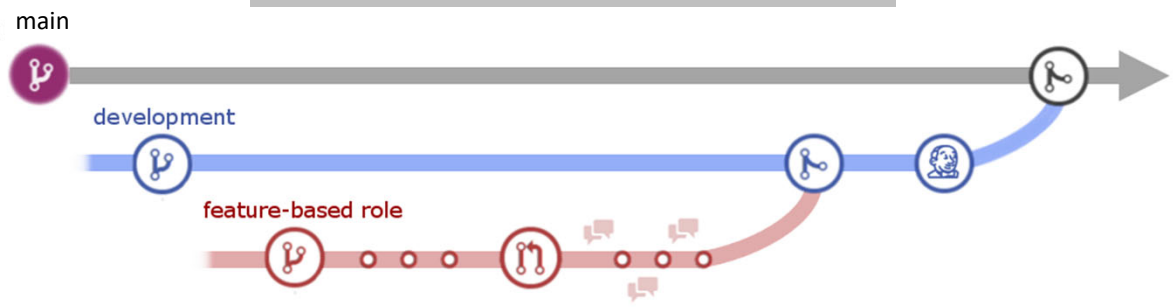


© NC State Software Engineering Faculty

24

Feature-Based Roles & Git

Start with base project in main branch

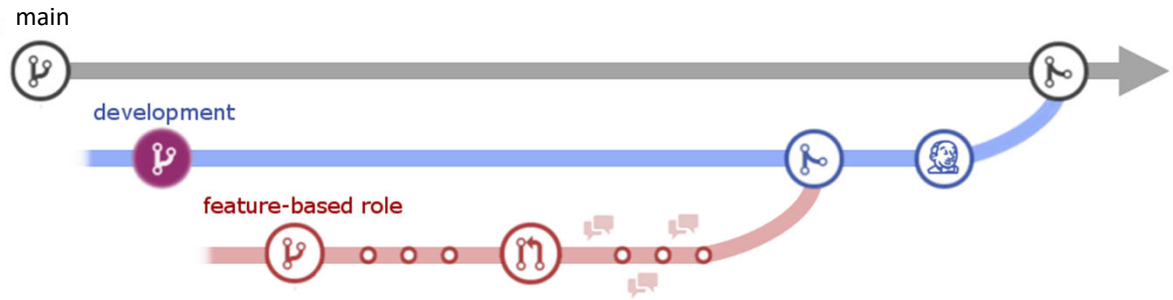


© NC State Software Engineering Faculty

25

Feature-Based Roles & Git

Make a development branch

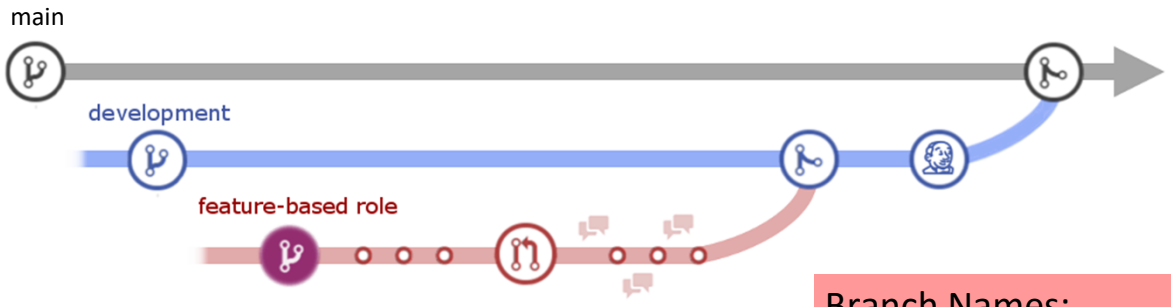


© NC State Software Engineering Faculty

26

Feature-Based Roles & Git

And a branch for each feature-based role



Branch Names:

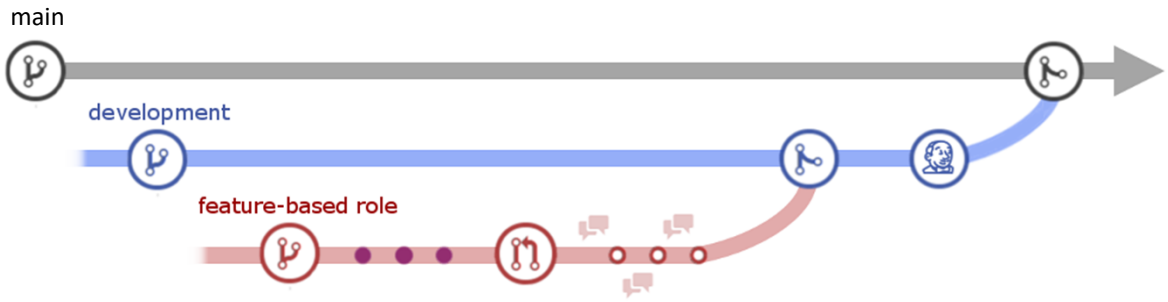
- `backend`
- `frontend-entry`
- `frontend-report`

© NC State Software Engineering Faculty

27

Feature-Based Roles & Git

Develop on feature-based role branches.

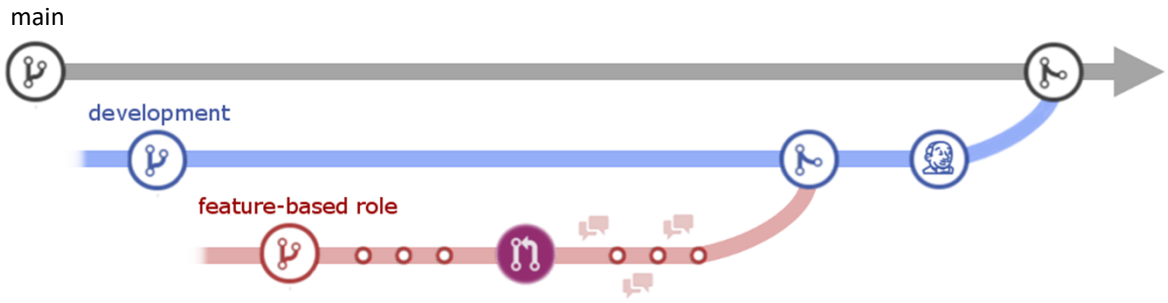


© NC State Software Engineering Faculty

28

Feature-Based Roles & Git

Pull request feature-based role branch to development.

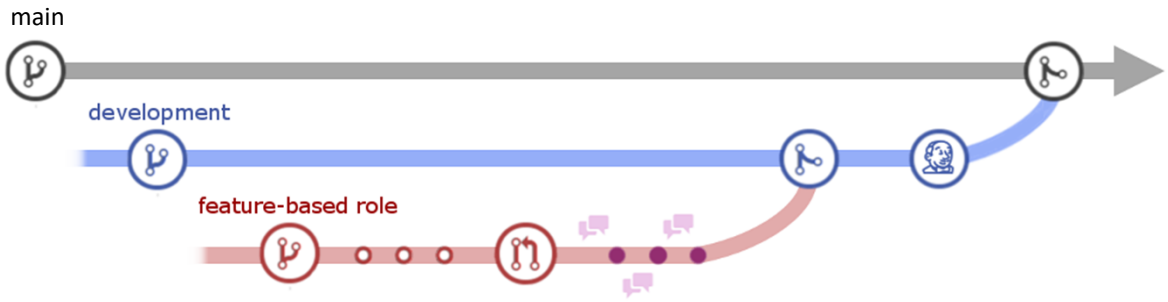


© NC State Software Engineering Faculty

29

Feature-Based Roles & Git

Review & Discuss!

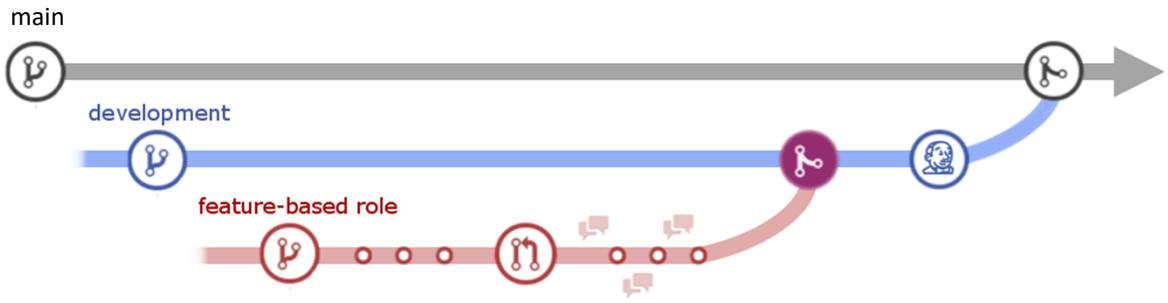


© NC State Software Engineering Faculty

30

Feature-Based Roles & Git

Merge feature-based role branch into development.

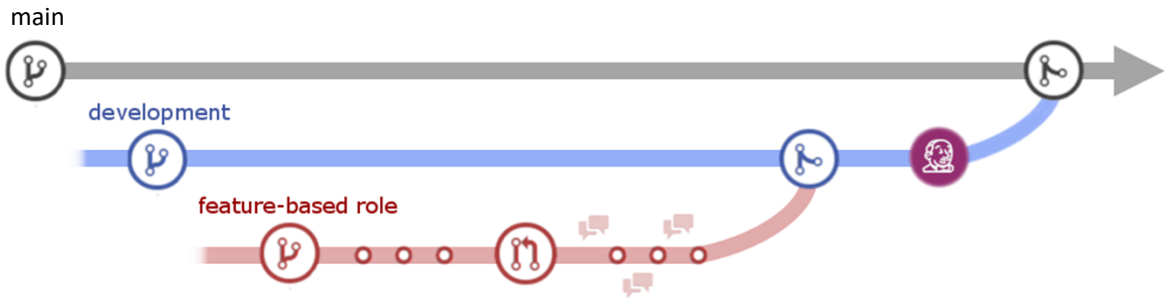


© NC State Software Engineering Faculty

31

Feature-Based Roles & Git

Building the development branch checks integration.

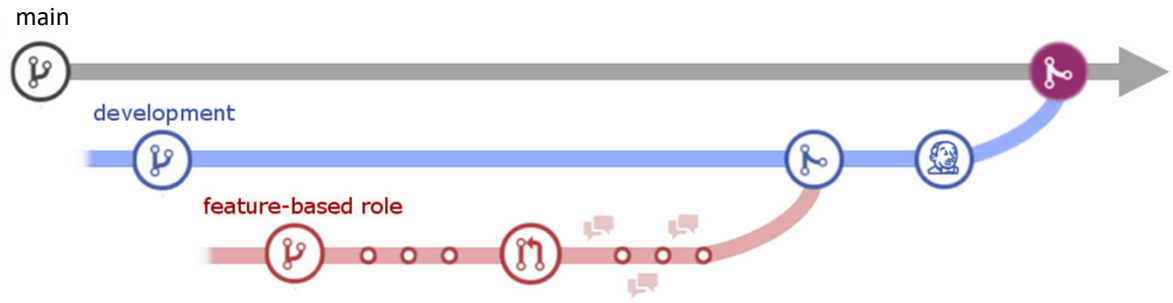


© NC State Software Engineering Faculty

32

Feature-Based Roles & Git

Merge development into main via pull request to deliver functionality!



© NC State Software Engineering Faculty

33

Feature-based Roles

Feature-based roles provide a mechanism for splitting up the work to implement your requirements. Your team will support *two feature-based sub teams* of 2-3 people each. Each feature-based sub-team will implement at least one feature each iteration.

Each use case can typically be broken into several main features. These may include backend business logic, backend API, frontend data entry, frontend display, and frontend reporting. Each team member must own one or more of the features. These roles can change and shift as necessary over the course of the project.

In keeping with good git practices, remember that each unit of work (which could be a feature) needs to be in its own topic branch, for example, `unityID/uc27-frontend-data-entry` .

Identify the topic branches you foresee and which team member will own each branch. **Document the branches and their owners in the wiki.**

Note: feature-based roles provide a way to break up the workload between teammates; however, good communication is required to ensure that each feature comes together into a cohesive working whole.

34

Requirements Exploration

Stakeholders and Personas

© NC State Software Engineering Faculty

Requirements Engineering (1)

- Systematic way of developing requirements through an iterative process
- Results in a specification of the system that stakeholders understand
  - natural language (e.g., traditional/plan-driven requirements, use cases, or user stories)
  - easy to understand pictures (UML Use Case Diagrams)
- A stakeholder is a key representative of the groups with a vested interest in your system and direct or indirect influence on its requirements.

© NC State Software Engineering Faculty

Requirements Engineering (2)

- **Analysis**: studying user needs to generate a system definition that users understand (often done in conjunction with Product Managers)
  - Fact-finding
  - Communication
  - Fact-validation
  - Moving from user requirements to product requirements
- **Modeling**: translating requirements that the user understands into a form that software engineers understand
  - Representation
  - Organization

© NC State Software Engineering Faculty

37

Activity 1: Identify Stakeholders

Given the WolfCafe Problem Statement…

- Who are my **stakeholders**?
- As a team, **list as many stakeholders as possible**!
- Are there any stakeholders who might abuse the system?

38

Activity 2: Narrow Stakeholders

Given your exploration of the stakeholders…

- Narrow down to 4-5 key stakeholders
- One should be an adversarial stakeholder… someone who wants to misuse or abuse the system to gain more than their peers

40

Activity 3: Create Personas

Given the stakeholders, let's create personas!

- Name and personal characteristics
- Job
- Location
- Interests
- Background and Skills
- Motivations and Attitudes
  - What are their motivations for using new technologies?
  - What is their computer self-efficacy?
  - What is their attitude towards risk?
- Attitude to Technology
  - What is their information processing style?
  - How do they learn about new technologies? Process or tinkering?
- Inspiration: https://gendermag.org/custom_persona.php

41

Requirements Exploration

- Document your team's personas on your Wiki!
- Use your personas to explore the requirements.

- Use the personas to tell a story in your presentation!

42