UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

COS 301 Software Engineering

Mini-Project Demo 2 Instructions
2025

---

1. Introduction

This document provides the Demo 2 instructions for the COS 301 Mini-Project 2025. In general, demos are assessment opportunities that we use to track progress and milestones. Though demos are mostly technically focused, a certain soft skill is required to convey your demo well. This document provides an outline of all the technical requirements as well as some suggestions for what makes a good demo.

The demo is on **Friday, 25 April 2025. The demo will be IN PERSON at IT 4-66.** Project Managers, please make sure to book one demo slot for your team when the demo slots open on Hyperperform.

The general theme of the demo is for us to see the **final, completed implementation** of your **reference MPDB project** that you have **engineered**. This demo is very much a product demo and needs to focus on showing a **COMPLETE WORKING PRODUCT**. Each team will get **20 minutes** to demo their Mini-Project.

2. Prerequisites

You have received the Mini-Project 2025 Playbook. This document serves as the Mini-Project proposal received from the client (the lecturers of this module). It is not a formal requirement specification but rather a proposal for a project, along with client wishes. This document derives the components and deliverables of the Mini-Project.

You also have to be registered on Hyperperform. This is a requirement.

3. Plagiarism

The Mini-Project shares a lot of overlap with existing tutorials and courses that are publicly available. We would like to remind the students here that they are allowed to use references as inspiration but cannot clone or use those references as if they were their own. Students have to do their own novel work and cannot claim ownership of work that is not theirs. Please be warned that if your project resembles one of these existing courses or tutorials, we will treat it as a plagiarism case.

4. Demo Components

During the **20 minutes** demo, we want your team to present to us:

1. Project Introduction
2. Live Demo:
    a. Show us all your working features
    b. Show us your WOW-factor
    c. Show us your “delivery” component (Docker/Installation/Distr).
    d. Show us your CI/CD.
    e. Show us your testing (unit and integration/e2e)
3. Admin Elements:
    a. Project Plan
    b. Role Breakdown + Contributions
    c. What has changed in the documentation (Agile)?
4. Q&A (please leave 5 minutes for this component)

Some general notes are given below.

Project Introduction

Introduce us to your team’s project by giving us:

- Your team name/number
- Background: Provide a brief background on your specific implementation of MPDB
- Scope: Clearly define the scope of your project by specifying which features you will be implementing and any limitations or exclusions.

Keep this very short and concise.

Live Demo

This is the technical part of the demo. Some general notes:

- You HAVE to demo a LIVE version of your platform.
- The demo has to be LIVE and not recorded or static mock-ups.
- You need to prepare for this from a technical point of view. Think about sharing screens, VPN and Wifi issues, etc.

Working Features

For this component, we need to SEE the connection between what you PLANNED to do vs. what was done in the end. Show us a WORKING implementation **of all of the specified requirements**. Importantly, these HAVE to be **finalised** features. For demo 2, the final code has to be on the **main/master** branch. This is the only demo-able state of the project that we will be looking at.

Wow-Factor

Make sure to highlight and show us your wow factor. Tell us why you picked the wow-factor and what makes it so “WOW”.

CI/CD

Finally, you must present your WORKING CI/CD. This includes:

- A Git repository on GitHub where you version-control your code.
- Tell us which Git branching strategy you are using
- What is your review process
- Show us your CI/CD pipeline in action. This means:
    - It should run actions for:
        - **Linting**
        - **Building**
        - **Testing**

Testing

Furthering on from the technical aspects. We require you to show us the following:

- The tests that you have identified that need to be run against your set of identified requirements
- There has to be:
    - Finalised **unit tests.**
    - Finalised **integration tests/end-to-end tests.**
- These tests **HAVE TO BE IMPLEMENTED AND WORKING.**

Admin:

Do not spend a lot of time on this, just take us through
1. That you have all the elements required and;
2. What has changed since demo 1.

Project Plan

Share your plan for the project with us. The deliverables here are:

- A **project board on GitHub** with tickets/cards clearly shows a project plan/roadmap.
- Tell us what **Software Engineering Methodology** you are following and why.
- What is your team composition, i.e. roles allocated to members?

Documentation:

The deliverables for Documentation are:

- **A README.md** file, which is a home page for your project. Please put links here to all relevant resources like:
    - Project Information
        - Short description of your project
        - This description should also appear in the About field in your GitHub repository
        - The description should follow the following format: ”TeamName - ProjectName - Project Description.”
    - Demo Links (To videos, documentation, etc)
        - A link to your recorded demo video is named “TeamName-Demo2.”
        - A link to the Requirement Specification (SRS)
        - A link to your design specification.
        - A link to your GitHub Project Board - Project management tool
    - Team members and roles
        - Short profile description of each member of your team.
        - The individual professional profile of each team member, including a link to LinkedIn, is visible on your GitHub.
    - Your GitHub repository should contain relevant information such as the following:
        - Git structure (**mono repo**)
        - Git organisation and management
        - Branching strategy
        - Code Quality badges (only the applicable ones):
            - Code Coverage — Coveralls, Codecov, SonarCloud
            - Build status - GitHub Actions
            - Requirements - shields.io
            - Issue tracking - GitHub Issues

- **A requirement specification** (in whatever format of your liking, such as LaTeX or Google Docs). This requirement specification is a formal document capturing the requirements of the client. Usually, it also contains some background information about the project. The wishes from the client were given in the Mini-Project 2025 Playbook (see link above). You need to formally list the **technical and non-technical requirements** in a requirement specification. This should contain not only the direct wishes of the client but also any other requirements that are missing in the details. The requirement specification also captures use cases (sometimes in the form of a use-case diagram).
- **A design specification** (in whatever format you like, such as LaTeX or Google Docs) is a formal document that captures your visual and architectural design elements. These include wireframes, mockups, and **screenshots of views/features/pages** that clearly illustrate what is to be implemented. Further, the design specification highlights your **architectural specifications** through an **architecture diagram**.

5. Rubric

The scoring weight for each component is given in the rubric below:

Project Overview (2 min)	5
Live Demo - Features (8 mins)	30
Live Demo - Wow-Factors (2 mins)	15
Live Demo - CI/CD + Testing (2 mins)	15
Live Demo - Delivery (2 mins)	15
Admin - Project Plan, Roles, Documentation (2 mins)	10
Overall Demo Presentation	10
TOTAL	**100**

* Leave time for Q&A at the end.

6. Further Tips

- Present the demo using PowerPoint, Google Slides or Canva (https://www.canva.com/).
- Speak clearly.
- Be prepared and organise your demo according to the instructions and rubric.
- Ensure that your README includes links to any artefact or deliverable we might ask for. These include your repository, documentation, project board, and proof of testing.
- Stick to your allowed time frame. No extra time is allowed.
- Ensure your diagrams correlate with how you learned to design them from COS 301 and COS 214 (consider draw.io for diagrams, as a visual paradigm can be a headache).
- Groups from 2022 for reference:
  - https://github.com/COS301-SE-2022/Pure-LoRa-Tracking
  - https://github.com/COS301-SE-2022/The-Au-Pair
  - https://github.com/COS301-SE-2022/Japanese-Writing-Evaluator
  - https://github.com/COS301-SE-2022/CryptoHub
  - https://github.com/COS301-SE-2022/ReverseHand
  - https://github.com/COS301-SE-2022/Map-out-game-reserves-using-aerial-photographs
- Groups from 2023 for reference:
  - https://github.com/COS301-SE-2023/AI-Photo-Editor
  - https://github.com/COS301-SE-2023/Blue-Skies

    - https://github.com/COS301-SE-2023/Domain-Pulse-A-Sentiment-Analysis-Platform
    - https://github.com/COS301-SE-2023/Avalanche
    - https://github.com/COS301-SE-2023/WriteToPdf
- Groups from 2024 for reference:
    - https://github.com/COS301-SE-2024/GameOnConnect
    - https://github.com/COS301-SE-2024/Web-Exploration-Engine
    - https://github.com/COS301-SE-2024/Dispute-Resolution-Engine
    - https://github.com/COS301-SE-2024/MoodMix
    - https://github.com/COS301-SE-2024/Crop-Prediction-System