UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

COMPUTER SCIENCE

COS 301 Software Engineering

Capstone Project: Demo 2 Instruction

S. Baror, V. Pieterse, T Mautsa & C. Trivilla

1 Introduction

The Capstone Project Demo 2 is **Friday, 24 May 2019**. The Demo's Bookings will be on Google Calendar, with the Google Calendar link provided on the COS 301 portal at the Link section. **The booking will be available from 09h00 on Wednesday, 15th May and will close at 23h30 on Friday 17th May 2019**. Demo 2 is focused on Architectural design and the deployment of your system. Ensure to dedicate 5-10 minutes reflection on your collaboration, then complete your iPeer before the deadline. There will be no extension after the deadline.

2 Live Demo

During Demo 2 and subsequent Demos, **You MUST Demo from your Git Master Branch.** Demos displayed outside your Git Master branch will loss you marks. During the live demo, you should do the following:

- Each team member briefly tell us what they did since Demo 1
- have a slide show that contains the following.
  - That briefly describes the architectural designs of your system and the reasons for your architectural decisions — 1 slide.
  - Give three to five core quality requirements of your system. How do you intend to or are you already addressing these quality requirements — 1 slide
- You should show the following live :
  - Implementation of at least three use cases different from the use cases you showed at Demo 1. Ensure that one of the use cases implemented is a core functionality of your project.
  - **Automated tests** both unit and integration testing of the implemented use cases
  - Any other cool stuff you have implemented

3 Deliverables

You should maintain the deliverable for your project. All the appropriate deliverable, as well as your use of the tools specified in Section 5, will be evaluated with every demo. The appropriate artefacts should at all times reflect the detail and current state of the project under construction. You should strive to have a working prototype of the implementation available in your master branch at all times.

3.1 COS 301 SE-Git Organisation

From Demo 2, the COS 301 Git Organisation is your landing page. The Link to the COS 301 SE Git Organisation is also available on the link section of the COS 301 web page. Your team's COS 301 SE Git Organisation should have the following:

1. Your readme should contain a short description of your project.
2. A PDF of your Architectural Designs documentation
3. A PDF user manual
4. A PDF of your coding standard/quality
5. Link to your Project management - scrum board tool of your choice (Waffle, Zenhub, etc.)
6. Individual profiles of the team members.

3.2 Working prototype

You should implement your system in such a way that you always have a working prototype of the system in your git master branch. The features that are not implemented yet should be mocked. For demo 2 we expect at least three new uses cases to be implemented and tested. The working prototype must be available on the master branch of your git repo.

3.3 Architectural Requirements and design documentation

This document should be kept in sync with your implemented project. The instructions for Demo 2 is given in the next section.

3.4 Coding standards document

The coding standards document should describe your conventions and styles to ensure a uniform style, clarity, flexibility, reliability and efficiency of your code. Also, document the file structure of your repository. See Chapter 18 in the textbook.

3.5 Testing policy document

You should have automated tests. Use a tool such as Travis CI (or any other appropriate tool) to manage and automate testing and deployment of your system. The testing policy document should describe the procedure you are following for testing. Point to your git repository of test cases and test reports. See Chapter 18 in the textbook as well as the documentation of your chosen testing tool.

3.6 User manual

The user manual should start by having a brief description of the project in layman's terms (avoid technical terms). Include a deployment *picture* i.e. something like your UML deployment diagram that should be in your requirements and design document, yet with pretty pictures of the devices you use. Write the document using the guidelines given in the UserManual.pdf document you can find in the Instructions folder on the CS Web page. The detail description of use cases should be only for the use cases that are already implemented (no imaginary or 'we may have' use cases).

4 Requirements and design specifications

This document is a growing document. You should correct errors in the previous version and add more content for every demo.

The main focus of your documentation for the Capstone Demo 2 is the Architectural Design of your project. You should use the mini project Requirements and design specification document as a guideline.

You should add the following to your Demo 1 documentation without removing the existing sections:

- Deployment model
- Architectural designs and requirements
- Constraints
- Technology decisions

Pay close attention to improve/correct the Quality requirements - Quantify the core quality requirements of your system.

4.1 Introduction

Explain the vision and objectives. State the business need for the application and summarise the scope of the project as it concerns the architectural designs

4.2 Deployment model

Show the deployment model using UML deployment diagram syntax.

4.3 Architectural requirements

The requirements and design documentation should include the architectural design structure of your system. See Chapters 6 in the textbook for reference. You should be able to determine the architectural, structural design of your system. For example, is your system based on N-tier, client-server, MVC, Service-oriented architecture (SOA), Microservices or any other architectural style. Justify your choice(s).

4.4 Quality requirements

Specify and quantify each of the quality requirements relevant to the system. Examples of quality requirements include performance, reliability, scale-ability, security maintainability, usability. *Quantify the quality requirements of your system. E.g., Availability: Test system is expected to be at least 99.5% up time. Scalability: Test system is designed to handle 50millions users request per sec. For security: Role-based access control is used.* Each of these quality requirements needs to be either quantified or at least be specified in a testable way.

5 Assessment rubric

Item	**Marks**
Live Demo	
At least 3 working use cases (10 marks each)	30
Architectural decisions and justifications	10
Unit and integration testing (5 marks each)	10
Documentation	
Corrections to document of demo 1 where mistakes were made	5
Architectural design & justifications of your design decisions in your documentation	20
Quality requirements and justifications	10
Deployment diagram	10
Quality of reflection in iPeer (individual)	5
Total	100

Note that individuals who are identified as social loafers or diligent isolates will be penalized by down-scaling of marks of team deliverables.

6 Project Client

You should have regular discussions/meeting with your project owner (client). Seek their approval of all artefacts that we require for COS 301. Give your client access to your git.

Your client is welcome to attend your demo, but it is not required. Ideally, you should arrange to see your client for an additional half hour or more before or after your demo. The demo itself may be rushed and can not serve as an opportunity for you to ask your client some questions.

If your client needs access to campus – for the demo or any other meeting, please provide the following detail at least 36 hours before the time.

- Client Name
- Client Email
- Client Phone number

- Date
- Time
- Venue
- Vehicle description
- Vehicle registration number.