CSC 1202: SOFTWARE DEVELOPMENT PROJECT

Lecturer: Dr. Peter Khisa Wakholi

Biography

Dr. Peter Wakholi (Ph.D.), is a lecturer in the School of Computing and Informatics at Makerere University and C.E.O and Founder of OMNI-Tech Ltd. He holds a PHD in Information Sciences with Research in Business processes in mobile systems from University of Bergen, Norway. An M.Sc. Information Systems from London Southbank University, United Kingdom. A BSc. Computer Science (1st Class Hons) from Makerere University, Uganda and a Higher National Diploma in Electrical Engineering from Kyambogo University, Uganda.

Email: pwakholi@cit.ac.ug

Lecture Hours

- Monday 8:00 – 10:00 AM Bsc Csc Day Venue: LLT 6B
- Thursday 8:00 —10:00 Bsc Csc Day Venue: LLT 3B
- Thursday 19:00 – 21:00 Bsc Csc EVE Venue: LLT 2A
- Saturday 8:00 Am – 10:00 AM Bsc Csc EVE Venue: LLT 2A

Course Description

This course aims to give students experience in developing complete software applications and systems. The focus will be on practicing programming and program documentation combining all the skills and knowledge acquired in other courses.

This unit will allow students to apply the knowledge and practice the skills acquired in the prerequisite and qualifying units in designing and building a substantial software development system in diverse application domains. Working in groups, students will need to carry out the full range of activities, including requirements capture, analysis, design, coding, testing, and documentation. Students will use the **XP methodology** and make use of professional tools for the management of their projects.

Course Objectives

The aims of the course unit are to:

a) practically give students skills in integrating different programming concepts into a single application.
b) introduce the student to hands-on aspects of software development.
c) nurture the student's ability to independently read different literature sources to identify what they can use to develop an application.

Learning outcomes

By the end of the course, the student should b e able to:

a) Apply the skills he/she has acquired to integrate different programming concepts into a single application.
b) Apply the skills he/she has acquired to software development
c) Solve real-life problems and challenges
d) Produce written reports of student's work in a style that conforms to their discipline and is appropriate for clients to check
e) Give an oral presentation of the student's work and its rationale for the student's discipline
f) Work with a small group of students to deliver substantial software development results
g) Follow a systematic process of work with appropriate mechanisms for dealing with risks, such as unforeseen factors and delays
h) Use previously unfamiliar tools and integrate with previously unfamiliar software/systems, learning their use from examples and documentation
i) Analyze needs, develop requirements, and demonstrate that the delivered results meet those needs
j) Apply discipline expertise in software development to solve problems and meet identified needs.

Detailed Course Content

The main content in the course is translation of a real-life problem into a working computer program. For each year the course lecturer will provide a theme where students can be challenged to develop a software application that a real-life problem. In addition, challenging practical challenges will be sourced from the industry and organizations that students can work on.

Study Materials

The study materials will include examples and case studies of real-world challenges; software development tools and hardware equipment; open sources repositories on Github. Gitlab etc.

Mode of Delivery

Learning will be largely by self - study/research. Students will be given programming Assignments that will be submitted after a specific period. The skills which that will be expected in the assignment will be indicated to the student. The course lecturer of staff will meet students at the issue of each of the assignment and address any outstanding issues as well as expectations. A minimum of three quizzes related to Django web framework and Python will be given. Where necessary a guest speaker or project owner from the industry will be invited to provide additional context on the project or the state-of-the-art industry tools and methods.

Mode of Assessment

Two assignments (each taking two weeks) and three assignments (each taking 3 weeks) will be given. The first two assignments will constitute the Quizzes (30%) while the last three assignments will constitute the project (40%) with oral examination - individual (30%).

Reading List

Students can read any literature like books and online tutorials that can help in addressing the problem in the project at hand. Students can also review available resources on GitHub and other common software repositories.

Weekly schedule

WK	Topic	Learning Activity	Resources
01	**Introduction to Software Project** • Course objectives • Study materials • Delivery modes • Course Project • Team formation • Learning materials	Presentation by the course lecturer	• Course Outline
02	• Assignment brief • Xtreme Programming methodology	Presentation by the course lecturer	• Course outline • Lecture slides
03	Working with GIT, IDEs	Guest Lecture	Git Tutorial (w3schools.com)
04	Django Framework overview- hands-on	Guest Lecture	https://realpython.com/ https://www.w3schools.com/django/index.php
05	Javascript fundamentals	Guest Lecture	https://www.w3schools.com/js/

06	Django Quiz Deliverable 2 presentation		
07	Backend programming and REST API	Guest Lecture	https://www.tutorialspoint.com/restful/index.htm
08	Working With JSON Data in Python		https://realpython.com/courses/working-json-data-python/
09	Javascript Quiz Deliverable 2 presentation		https://www.w3schools.com/js/
09	Writing Unit tests	Guest Lecture	https://docs.python.org/3/library/unittest.html
10	Writing functionality tests	Course lecturer	
11	• Progress with requirements (user stories); • Implements & test requirements • **Submit deliverable 2: Working prototype 1**	Project	
12	• Final project presentation & demo (in-tutorial) • Concluding remarks and feedback	Project	

Project Assignment

To-Do App

A to-do app is a software application that lets you make a list of tasks that you need to complete. You can make daily or weekly lists of tasks in a to-do app. A user sets up an account before they can create their tasks. The app lets the user keep track of their chores. Once you complete a task, mark it "completed" and update your to-do list. The user should be able to specify the expected date and time of each task and when (in minutes) they should be reminded. A reminder should be sent using an email to the user notifying them of the task. The email should contain a link to the task, which opens when clicked. Tasks that are incomplete but whose time has passed should be marked as skipped. The user can delete tasks. All tasks should not be stored for more than seven days.

Requirements

You must use a Django framework to develop the back-end system. You must build a minimal web application using tools like JavaScript, HTML, and CSS for the front end. You may provide functional requirements, testing, productivity scripting, and deployment. Also, use Extreme Programming (XP), an agile software development framework that aims to produce higher quality software and higher quality of life for the team.

You will be expected to host your app on an online server by using Django/Flask framework.

The examination is based on time (the amount of time estimated for this assignment) and skill (good code and good use/understanding of the tools used).

Deliverable 1: Date Due: 9th March 2023

1. Forms groups of 4 students.
2. Create a Project in Gitlab
3. Register each of the members in the group with an account. Add the account username to your registration.
4. Provide an account that will be used to monitor your work. Submitting an account that does not work is tantamount to failure.
5. Create a functional and technological specification on the repo in readme.md.
6. Submit a report to MUELE

What to submit

- A PDF report submit to MUELE
- List of group members, registration numbers, and GIT account **(5 Marks)**
- Identify two user stories and provide a summary of your understanding of the assignment in terms of user stories **(8 Marks)**
- For each user story identified, provide **(20 Marks)**
    - List of implementation tasks identified and written in card format.
    - Provide a plan that shows the list of stories and the order in which they will be implemented. Your plan shall be used as the basis for evaluating deliverables 2 and 3.

Deliverable 2: 26th April 2023

- For the current iteration based on the first user story, provide **(10 Marks)**
    - Your designs (UI, database, process, etc., as appropriate)
    - Provide your test suite.

- Screenshots of your current prototype **(10 marks)**
- A working prototype hosted on an online server **(30 Marks)**
- Links to your code in GIT explain the blocks related to the user stories implemented **(30 marks to be assigned to individuals based on GIT activity)**
- Challenges faced and how you overcame them **(10 marks)**

Deliverable 3: 31st May 2023

1. Individual oral code and Project screencast: Your code will be in Gitlab. Every student will individually present the code and system parts. You should show the code and explain how it works. Present how you've used the frameworks in your code and show any interesting things.
2. Documentation: We are coders, and we read code, commits, and branches as part of the examination. Marks will be awarded to only code committed to the main branch. Label it "Main" in GIT for easy identification. Make sure you commit/frequently push with commit comments that explain what you have done. This is very good practice and the best documentation.

Examination Criteria:

The projects are graded according to the following criteria:

1. Maturity in development demonstrated by the students **(10 Marks)**
2. Proper use of XP techniques and correct use of Django frameworks **(10 marks)**
3. Completion of a mid-size project **(50 marks for individual contribution)**
4. Creativity in your implementation **(10 marks)**
5. Code quality, refactored code - easy to read code (code is the documentation) **(10 Marks)**
6. The timely delivery of the project **(10 marks)**
7. Individual reflection on the exercise, What you have learned, what you have done, what could have been done better, and how you hope to improve or get better **(10 Marks)**

*** Plagiarism is an offense. See university policy - https://policies.mak.ac.ug/sites/default/files/policies/Makerere-Academic-Integrity-Policy.pdf

Plagiarized work will not be marked.