|  |  |
| --- | --- |
| F:\LICMA17\img\logoUL.jpg | F:\LICMA17\img\logoFS.jpg |

**University Management System**

OMO Team

Supervised by: Dr. Ali Ghorayeb

INFO401

Team

|  |  |
| --- | --- |
| C:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0229389.wmf  Obaida Ammar  id  71453824  obaidaammar99@gmail.com | C:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0229389.wmf  osama zammar  id  81636997  zammarosama@gmail.com |

C:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0229389.wmf

MOHAMMAD ABO ALFOUL

ID

TEL

EMAIL

Acknowledgement

We feel proud to present out ‘University Management Project’. It is our great pleasure. To express our gratitude towards our teachers Dr. Ali Ghorayeb and Dr. May Dehayni for guiding and providing us with the necessary information and skills that helped us prepare and organize our working methodology to complete the project in the most efficient and professional way. We would also like to thank all participants that contributed to the realization of our idea that turned into a project to carry on the university activities.

Table of contents

[**Table of contents** iii](#_Toc503515565)

[**Table of figures** v](#_Toc503515566)

[**Table of tables** vi](#_Toc503515567)

[Introduction 1](#_Toc503515568)

[Chapter 1. Preliminary Study 2](#_Toc503515569)

[1 Introduction 2](#_Toc503515570)

[2 Project objectives 2](#_Toc503515571)

[3 Similar applications 2](#_Toc503515572)

[4 Features 2](#_Toc503515573)

[5 Design 2](#_Toc503515574)

[Chapter 2. Database 3](#_Toc503515575)

[1 Introduction 3](#_Toc503515576)

[2 Implementation 3](#_Toc503515577)

[3 ER diagram 3](#_Toc503515578)

[4 Tables 3](#_Toc503515579)

[Chapter 3. Agile method 4](#_Toc503515580)

[1 Introduction 4](#_Toc503515581)

[2 Implementation 4](#_Toc503515582)

[3 Product backlog 4](#_Toc503515583)

[Chapter 4. UML 5](#_Toc503515584)

[1 Introduction 5](#_Toc503515585)

[2 Description of classes and interfaces 5](#_Toc503515586)

[3 Class diagram 5](#_Toc503515587)

[4 Use case diagram 5](#_Toc503515588)

[5 Sequence diagram 5](#_Toc503515589)

[Chapter 5. Design Patterns 6](#_Toc503515590)

[1 Introduction 6](#_Toc503515591)

[2 List of design patterns 6](#_Toc503515592)

[Chapter 6. Implementation 7](#_Toc503515593)

[1 Introduction 7](#_Toc503515594)

[2 Technique 7](#_Toc503515595)

[3 Constraints 7](#_Toc503515596)

[4 Security 7](#_Toc503515597)

[Conclusion 8](#_Toc503515598)

Table of figures

**No table of figures entries found.**

Table of tables

**No table of figures entries found.**

Summary

Within completing our Master’s Degree in Computer Science, we were interested to work on a project which mainly focus on providing management capabilities for users involved in a university. Throughout this project, we have worked as a team, preparing, organizing, and implementing our ideas to complete the project to the required standards and specifications.

Introduction

…

Chapter 1. Preliminary Study

# **Introduction**

The project consists of a desktop application developed to carry out and accomplish the activities done by a university. It automates the registration of student’s – instructor’s processes in university. It reduces the time it takes to register student’s courses and checking the courses prerequisites, grades and average calculations. This software is responsible for entering student’s data and keeping it from the day of submission till the student graduates, so it processes and maintains student’s data, major and marks. In addition, it maintains instructors’ data and their assignments to teach courses and permit them to assign grades to students in the courses they teach. An administrator, is responsible for managing all activities of students-instructors-courses.

# Project objectives

1. Manage the registration of new members in the university.
2. Manage courses that are offered by a certain faculty.
3. Manage professors’ registration to teach courses.
4. Manage students’ enrollment in courses and their grades.

# Similar applications

Lebanese University management system

# Features

1. Validate the registration of ‘Students’ and ‘Professors’ in the university.
2. Add courses to majors in order to be taught and enrolled in.
3. Assign ‘Professors’ to teach certain courses.
4. Enroll ‘Students’ in certain courses according to their major.
5. Assign grades to ‘Students’.
6. Generate transcript for ‘Students’.

# Design

1. Students and Instructors registration in the university

Save users’ info in database after admin validate their registration

Admins Validate the registration

Of Students and Professors

Students and Professors register in the

University system

System

Database

1. Students checking their grades in registered courses.

Students request to view their

grades

System requests specific student grades in registered courses

Database

System

Display student grades

Retrieve requested student grades in registered courses

1. Professors assign grades for students in the courses they teach.

Instructor request registered students in the courses they teach

System request students enrolled in a specific course the professor teach

Database

System

Retrieve students enrolled in a specific course taught by the professor

Professor assign grades to students

System validates the new assigned students’ grades

1. Administrators Add courses.

Administrator add courses to specific major

System adds new course to database records

Database

System

1. Administrators Edit, Delete courses.

Administrator request a specific course stored in the university database

System requests a specific course

System

Database

Retrieve requested course

System validates the administrator request to Edit or Delete a course

Administrator Edit or Delete the requested course

1. Administrator Edit Student or Professors info.

Administrator request a specific student or professor stored in the university records

System requests a specific student or professor

System

Database

Retrieve requested student or professor

Administrator Edit or Delete the requested Student or Professor

System validates the administrator request to Edit or Delete a Student or Professor

1. Administrator assign Professors to teach courses.

Administrator request to assign professor to teach a course

System requests available courses

System

System retrieves available courses

Database

Administrator register a professor in specified courses

System validates the assignment of professor in specified courses

1. Administrator enroll Students in courses.

Administrator request to enroll a student in courses

System requests courses of Student major

Database

System

System retrieves courses of Student major

Administrator enroll student in specified courses

System validates the enrollment of student in specified courses

1. Administrator assign grades for students.

Administrator request student grades

System requests student grades

System

Database

System retrieves student grades

Administrator assign student grades

System validates the student grades

1. Administrator manage History of previously deleted components of university.

Administrator request to manage university history

System requests specified history (previously deleted component)

System

Database

System retrieves specified history component

System validates the retrieval of previously deleted component back to system

Administrator request to retrieve previously deleted component from history

1. Administrator request transcript for a student.

Administrator request a transcript for a specific student

System requests the transcript of specified student

Database

System

System retrieves student transcript

Chapter 2. Database

# Introduction

The Database of this project was used in order to permanently store the information of users and the information needed by the application to run properly. It is the heart of this project where all the relations between the different users and the information related to these relations is found in the Database.

# Implementation

The Database was implemented using the PhpMyAdmin Database administration service and the MYSQL language. The servers used were XAMPP and EasyPHP depending on the developer’s preference.

# ER diagram

# 

# Tables

The tables can be found in detail inside the ER Diagram.

Chapter 3. Agile method

# Introduction

Agile is an iterative way of managing projects and developing software that makes it easier for teams to deliver value to their customers more quickly and effectively. The methodology followed in this project supports Agile method in which the team focused on delivering value faster and with fewer complications by systematically managing the project and developing the software in an iterative fashion by delivering small but consumable increments, and by continuously evaluating the requirements, plans, and results the team was always able to respond to change in a timely manner.

# Implementation

The agile methodology used is ‘Scrum’ which is framework used to establish a hypothesis, test it, reflect on the experience, and make adjustments. The working procedures of the product development is split into 2-4 weeks iteration.

The steps followed involve:

1. **Definition of the project:** The team along with (users-customers), definition of project’s goals, objectives and requirements.
2. **Creation of a backlog:** which is a prioritized list of tasks that need to be carried out. It is realized through the collaboration of team, users, product owner.
3. **Sprint Planning:** the sprint is planned by the selection of tasks with the highest priority from the backlog and by the determination of how much work the team can be committed to complete in the upcoming sprint.
4. **Execution of the sprint:** at this stage the team focus on completing the planned tasks for this sprint with a daily scrum meeting to check the progress and address any issues.
5. **Sprint review:** at the end of the sprint, the team provide a demo of new features to present what it accomplished during the sprint.
6. **Sprint retrospective:** the team retrospect on the sprint, discussing what went well, what didn’t and what can be improved for the next sprint.
7. **Repeat:** This process is repeated for each sprint until the project is completed. The product is incrementally developed and delivered to the customer is small chunks.
8. **Continuously improve:** Agile methodologies focus on continuous improvement. The team reflects on its progress and makes adjustments as necessary to improve processes, tools, and communication for the next sprint

# Product backlog

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **PRODUCT BACKLOG** | | | | | | |
| **Team:** | Obaida Ammar | Osama Zammar | Mohammad Abo Alfoul |  |  |  |
| **Date:** | 5/1/2023 |  |  |  |  |  |
| **Priority** | **Story Point Estimate** | **Sprint Number** | **Feature** | **Story** | **Story Type** | **Done (Yes/No)** |
| 1 | 1 | 1 | Student | I can register to the system if I am a new member. | Story | Yes |
| 2 | 1 | 1 | Professor | I can register to the system if I am a new member. | Story | Yes |
| 3 | 3 | 1 | Administrator | I can accept or unaccept new registered members (Student or Professor). | Story | Yes |
| 4 | 2 | 2 | Administrator | I can add courses to majors offered by the university. | Story | Yes |
| 5 | 1 | 2 | Administrator | I can delete course from university records. | Story | Yes |
| 6 | 3 | 3 | Administrator | I can add professors to teach available courses. | Story | Yes |
| 7 | 1 | 3 | Administrator | I can remove professor from a course that he/she teaches. | Story | Yes |
| 8 | 3 | 3 | Administrator | I can enroll students in courses of their major only if they have passed prerequisite courses. | Story | Yes |
| 9 | 1 | 3 | Administrator | I can drop a student from a course he/she has enrolled in. | Story | Yes |
| 10 | 2 | 4 | Professor | I can assign grades for students enrolled in courses that I teach. | Story | Yes |
| 11 | 2 | 4 | Professor | I can edit the grades for students enrolled in courses that I teach (only if I didn’t submit the grades yet). | Story | Yes |
| 12 | 1 | 4 | Professor | I can submit the final grades that I have assigned to students so that I can’t edit them again. | Story | Yes |
| 13 | 2 | 5 | Administrator | I can edit the grades that have been submitted by professors. | Story | Yes |
| 14 | 4 | 5 | Administrator | I can generate a student transcript. | Story | Yes |
| 15 | 3 | 5 | Student | I can view my grades in courses I am enrolled in. | Story | Yes |
| 16 | 4 | 5 | Student | I can generate a transcript of all grades and courses that I have completed so far. | Story | Yes |
| 17 | 3 | 6 | Administrator | I can edit the information of a course. | Story | Yes |
| 18 | 3 | 6 | Administrator | I can edit the information of a student. | Story | Yes |
| 19 | 1 | 6 | Administrator | I can delete a student from university records | Story | Yes |
| 20 | 3 | 6 | Administrator | I can edit the information of a professor. | Story | Yes |
| 21 | 1 | 6 | Administrator | I can delete a professor from university records. | Story | Yes |
| 22 | 3 | 7 | Administrator | I can retrieve previously deleted university members (Student or Professor). | Story | Yes |
| 23 | 3 | 7 | Administrator | I can retrieve previously deleted courses. | Story | Yes |
| 24 | 4 | 7 | Administrator | I can retrieve previously deleted relation (Professor teach course). | Story | Yes |
| 25 | 4 | 7 | Administrator | I can retrieve previously deleted relation (Student – enrolled courses – grades) | Story | Yes |

Chapter 4. UML

# Introduction

# Description of classes and interfaces

# Class diagram

# Use case diagram

# Sequence diagram

Chapter 5. Design Patterns

# Introduction

Design patterns are typical solutions to common problems in software design. Each design pattern used is like a customized blueprint to solve a design problem in the project and accelerate the development process. It also provides development paradigms which helps save time without having to reinvent patterns every time a problem arises. In addition, standardization related to design pattern is also very useful to facilitate code readability.

# List of design patterns

1. **Architectural Patterns:**

MVC: used to separate the application into 3 main groups (model, view, controller)

1. **Structural Patterns:**

DAO: used to separate low level data accessing API or operations from high level business services.

Flyweight: used to reduce the number of objects created and to decrease memory footprint and increase performance.

1. **Creational Patterns:**

Singleton: restrict the instantiation of a class and ensures that only one instance of the class exist in Java Virtual Machine.

Abstract Factory: it provides a way to create families of related objects without imposing their concrete classes, by encapsulating a group of individual factories that have a common theme without specifying their concrete classes.

Chapter 6. Implementation

# Introduction

# Technique

# Constraints

# Security

Conclusion