The Software Engineering of a Three-Tier Web-Based Student Information System (MyGJU)

Feras Al-Hawari, Anoud Alufeishat, Mai Alshawabkeh, Hala Barham, and Mohammad Habahbeh
Information Systems and Technology Center
German Jordanian University
Amman, Jordan

Abstract: this paper discusses how the software development team at the German Jordanian University (GJU) adopted the project management and software development processes in the ISO/IEC 29110 series to implement a complex Student Information System (SIS). Specifically, it identifies the key points to be taken into consideration in the analysis, design, implementation, testing, and deployment phases during the iterative and incremental SIS development process. The SIS is a distributed three-tier web-based application that enables registrars to perform various tasks such as: system setup, admission, registration, grades processing, graduation, and reporting. It was launched in the first 2015/2016 semester and enabled administration to maintain a comfortable learning environment, assess instructor performance, enhance teaching practices, and improve course content. The results of the system measurements and user survey assert that the SIS is feature rich, easy to use, fast, reliable, stable, highly available, and scalable.

Keywords: *student information system, registration, web-based applications, software engineering, ISO/IEC 29110, iterative and incremental software process, and waterfall process.*

1. INTRODUCTION

Recently, the GJU established a software development team to implement its custom SIS and Enterprise Resource Planning (ERP) applications in-house. The university took this step as its ERP software vendor was not able to deliver all the required applications albeit the deadline to complete the whole project was exceeded by several years. In addition, the delivered tools were still missing many basic features, cumbersome, and not integrated. Conversely, the in-house development efforts resulted in releasing several major applications like the SIS as well as the accounting and financial systems in a relatively short time (in about two years) while saving the university a lot of money. Furthermore, the new systems are feature-rich and customized to meet the university's academic policies and business processes.

Since the SIS is the most important application at any educational institution, it was one of the first systems to release and use at the GJU starting from the first 2015/2016 academic semester. The new SIS is a web-based application and it has three main views: student, staff, and administrator views. The student and staff views can be accessed via the Internet using the front-end of a web application that we refer to as MyGJU [1]. Whereas the administrator view is named MyGJU Admin and is only accessible through the university's Intranet for security reasons. The focus of this paper is on the software engineering of the MyGJU Admin application. Noting that the Accounting Information System (AIS) and the Financial Information System (FIS) were released along with the SIS, nevertheless they are not discussed in detail in this paper.

The MyGJU Admin application is a sophisticated SIS that allows registrars at the GJU to manage student information. It consists of several basic functional modules to support features such as: system setup (e.g., managing users, roles, countries, buildings, rooms, faculties, and departments), academic setup (e.g., managing courses, course sections, and study plans), admission, student records management (e.g., managing personal information, scholarships, schedules, grades, transcripts, major transfers, etc.), registration (i.e., adding and dropping courses), final exam scheduling, grades processing (i.e., entering grades, computing GPAs, and viewing transcripts), graduation, and reporting. It also addresses most of the issues that the previous SIS system could not adequately handle such as software ease of use, capability, performance, stability, reliability, and scalability. Furthermore, unlike the previous system, the MyGJU Admin is fully integrated with the rest of the university applications such as AIS, FIS, and Human Resources (still under development). Hence, guaranteeing the accuracy and consistency of the data being shared amongst them.

Several web-based student information systems were discussed in [2-19]. However, in this paper, we provide a comprehensive description for each of the software engineering life cycle phases that were followed to develop the MyGJU Admin application. Besides, we consider important aspects pertaining to a SIS such as usability, performance, authentication, and security. Another group of papers, such as [20-22], only discussed limited SIS features for specific technologies like computer-telephony

integration (CTI) and mobile phones. In [20], they used CTI and an interactive voice response system to allow students to register for courses. In this case the student starts by calling the system and, in turn, a computer responds back with a sequence of pre-recorded audio questions that the student needs to answer via the telephone's keypad in order to complete the registration process. Whereas in [21] and [22], they introduced mobile phone applications to enable students to add and drop courses.

While in [23-41], the researchers only focused on the design of certain aspects of a SIS rather than the development of the system as a whole as in this paper. For example, in [23-26] the focus was on university course advising. While in [27-30], [31], [32], [33], [34], [35], [36], and [37] they only discussed designing e-learning features, lecturer performance, assessment in the classroom, course planning, problem based learning, recording class attendance, enrolling students into examinations, and sending email alerts to students respectively. Moreover, the database analysis and design aspect of a SIS was presented in [38] and [39]. Furthermore, in [40] and [41], the speedy courses registration transaction function and the challenges in implementing such systems were considered respectively.

The rest of the paper is organized as follows: the adopted software engineering processes and established software development team are presented in section 2; the basic phases in the SIS development process that is analysis, design, development, testing, and deployment phases are discussed in sections 3, 4, 5, 6, and 7 respectively; validation results are considered in section 8; and a summary as well as future work are provided in section 9.

2. SOFTWARE ENGINEERING PROCESSES AND PROJECT TEAM

In this section, the project management and software development processes that are based on the systems engineering basic profile in the ISO/IEC 29110 series [42] and adopted to accomplish the MyGJU Admin project are discussed. Furthermore, the anatomy of the team that was capable of implementing such a complex application is presented.

2.1. Project Management Process

This process is adopted to deal with issues like project size, cost, schedule, progress, review, release, and closure. The following measures were done to achieve the aforementioned goals:

- **Project Plan**: A project plan was first prepared and presented to the Dean's council at GJU for discussion and then approval. Accordingly, it was decided to breakdown the project into three phases to be delivered after 12, 18, and 24 months from the start date of the project. The software modules to be delivered in each phase were also identified. Moreover, the resources and time needed to complete each software module were specified. Furthermore, the overall development, software, and hardware costs were estimated. It is worth mentioning that the approved plan also took into consideration the following factors: customer's (registration and admission department) priorities, requirements size, university budget, university calendar, development team size, developers' skills, and projected risks.
- **Project Progress**: A technical committee was formed to monitor and evaluate the project progress against the project plan based on a monthly progress report and feedback from the project manager. The project manager was responsible for publishing the monthly progress report, which shows the tasks breakdown for each software module in the project plan, tasks details, tasks assignments to developers, as well as the tasks anticipated and actual start and completion times. Furthermore, the project manager was responsible for presenting the project's progress status to the Dean's council once every 6 months.
- **Review Meetings**: Meetings were scheduled at least once a month between the development team and the customer to discuss the detailed requirements for each software module, change requests, functional specifications, input and output formats, data entry, test plans, and compliance with the requirements in the project plan. The meetings minutes were recorded and distributed to all participants for approval and feedback. All assignments were also tracked to guarantee their completion.
- **Version Control**: A version control strategy was specified to support the proposed three major deliverables and handle any intermediate software modifications. Accordingly, updates are usually deployed at off peak times (around midnight) with minimal down time (a few seconds). Moreover, all software modifications are documented and communicated to all parties upon every update.

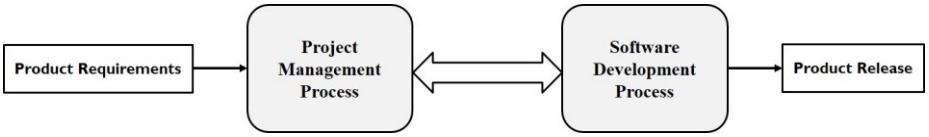

Figure 1. The adopted software project management and development processes

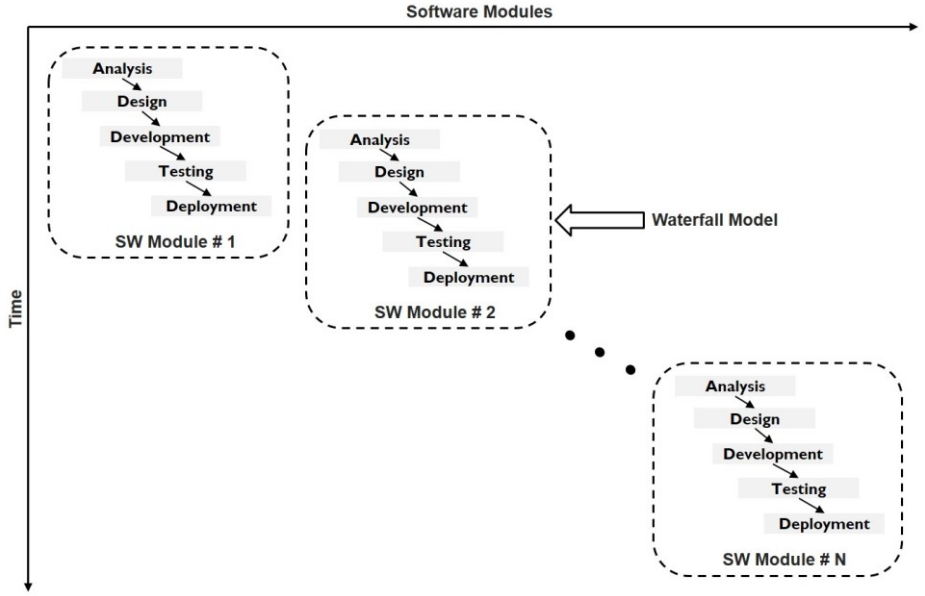

Figure 2: The iterative and incremental software development process

- **Risk Management**: Project risks (due to holidays, developer's incompetency, resignations, customer's hindrance, hardware delivery delay, etc.) were anticipated and identified on a regular basis. Accordingly, solutions (e.g., overtime, training, hiring, meetings), resources (equipment, money), and personal (perform a needed task, manage the risk, or clear equipment from customs) to overcome the side effects of any risk were promptly proposed, provided, and delegated.
- **Disposal Plan**: A plan was proposed, discussed, approved, scheduled, and executed to discontinue the old system and replace it with the newly developed system with minimal down time.

2.2. Software Development Process

The iterative and incremental software development process (methodology) shown in Figure 2 was used to implement the MyGJU Admin application. Accordingly, the application was developed through repeated waterfall [43] cycles (iterations) for the software modules (increments) that were gradually addressed as time progressed. This process makes the development of a complex project much easier as it is accomplished in smaller and manageable increments. In addition, it allows specifying more requirements and leads to a more stable application between increments. Hence, it is suitable to build a large and complex application, like the MyGJU Admin, which is delivered in several phases and comprised of several separate software modules such as buildings, rooms, courses, courses sections, study plans, grades, and reports.

The measures taken in each phase in the waterfall cycles (iterations) for some of the developed software modules will be discussed in the subsequent sections. It is worth mentioning that the development team held regular (on a weekly basis) meetings to review the documents or plans (e.g., functional specifications, design specifications, source code samples, algorithms, reusable modules, testing plans, and deployment plans) for each phase.

2.3. Team Anatomy

The project team consists of a project manager (also lead, architect, and developer), four developers, a tester, a database administrator, a system administrator, and a network engineer. The educational backgrounds, experiences, skills, and roles of the team members are summarized in Table 1.

Table 1. Team anatomy

Member	Degree	Experience	Roles	Skills
Project manager and lead	PhD	17 years in SW industry 5 years in academia	Manager, lead, architect, software development, and DB design	JSF, Java EE, Database
One developer	MS	10 years	Software development and DB design	JSF, Java EE, Database
Three developers	BS	3-5 years	Software development and DB design	JSF, Java EE, Database
Tester	BS	10 years	Software QA and testing	ERP senior user
DBA	BS	10 years	DBA	DBA
System Administrator	BS	10 years	Servers, storage, OS, VMs, application server, version control, backups, system security	Linux, Windows, MS Hyper-V, GIT, Backup Exec
Network Engineer	BS	7 years	Network VLANs and security	Cisco switches, ASA, and VLANs

Table 2. The academic and administrative user groups and some of their corresponding user roles.

User Role	User Group
Administrator	Administrative
Registrar	Administrative
Assistant Registrar	Administrative
President	Academic
Dean	Academic
Instructor	Academic
Teaching Assistant	Academic
Student	Academic

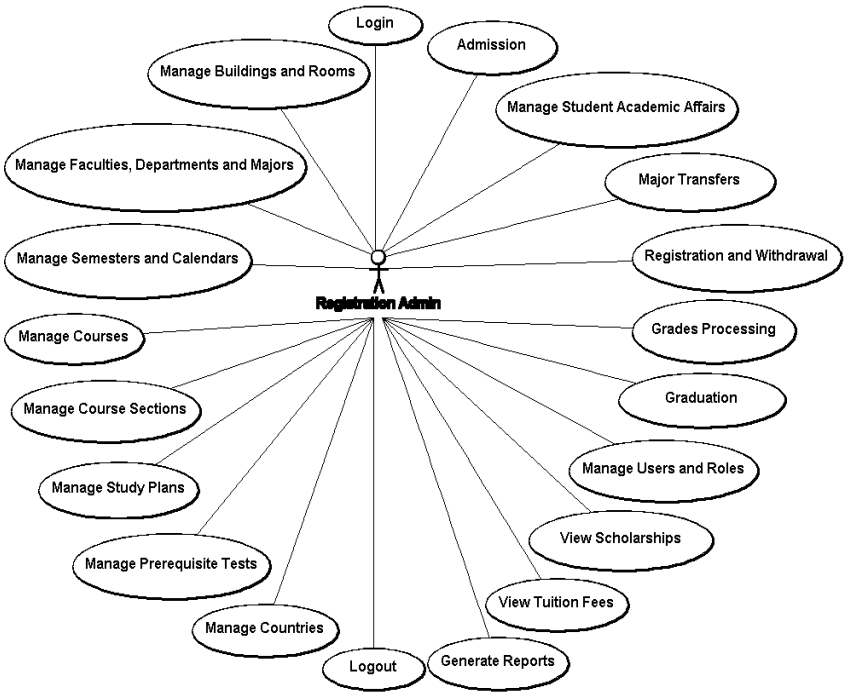

Figure 3. The use case diagram for the MyGJU Admin view

3. ANALYSIS PHASE

The requirements analysis phase involves specifying the workflows and features to be supported by the SIS and provided to its end users. The users of the MyGJU applications are subdivided into two groups: administrative and academic. Within each group, users might be assigned different roles (see Table 2) based on their needs and responsibilities. The administrative group contains administrators, registrars, and assistant registrars. While the academic group covers academic staff (e.g., deans, instructors, and teaching assistants) as well as students. Accordingly, the features of the MyGJU Admin tool target the administrative users, whereas the MyGJU system addresses the needs of the academic users. As the focus of this paper is on the MyGJU Admin application, we next consider its basic requirements and for completeness sake we also briefly discuss some of features of the MyGJU tool.

3.1. MyGJU Admin View

This application view is needed to automate and facilitate the daily tasks (e.g., setup, admission, course scheduling, registration, grades computation and viewing, and graduation) that are performed by the administrative staff of the admission and registration department at the GJU. The activities supported by the MyGJU Admin view are summarized in the use case diagram shown in Figure 3 and can be categorized into the following six groups: system setup, academic setup, students affairs, users affairs, financial affairs, and reports. The system setup group is discussed first and it supports the following features.

Managing buildings and rooms. The registrars need to add the university buildings in the system and then manage the rooms in each building (see Figure 4) in order to later schedule the course sections into their suitable rooms. This feature, for example, also allows a registrar to specify the names of the buildings, the number of floors in each building, the names of the rooms, the capacity of each room (see Figure 5), and the types of the available equipment in each room.

Managing faculties, departments and majors. Managing this information in the system is pivotal as each student is admitted to a faculty, department, and major. This capability, for example, enables the user to add a new faculty (e.g., School of Management and Logistic Sciences) to the existing list of faculties in the system, manage its corresponding departments (e.g., International Accounting and Management Sciences), and then manage the majors of each department.

Managing semesters and calendars. Based on this feature, a registrar can add a new academic semester and then specify its name as well as its start and end dates. In addition, he/she can define the periods of the various event types (e.g., registration, withdrawal, and grades submission periods) in the semester's calendar (see Figure 6). Each calendar event controls when the users may or may not perform its respective operations. For example, registrars and students can only add and drop course sections during the Registration or Add/Drop periods. Whereas instructors may only submit the class grades during the Grades Submission period.

Managing countries. The user can use this capability to add countries and nationalities in the system. This data is needed, for example, to determine the tuition fees for the students as the tuition fees for the international students are usually more than those for the Jordanian students.

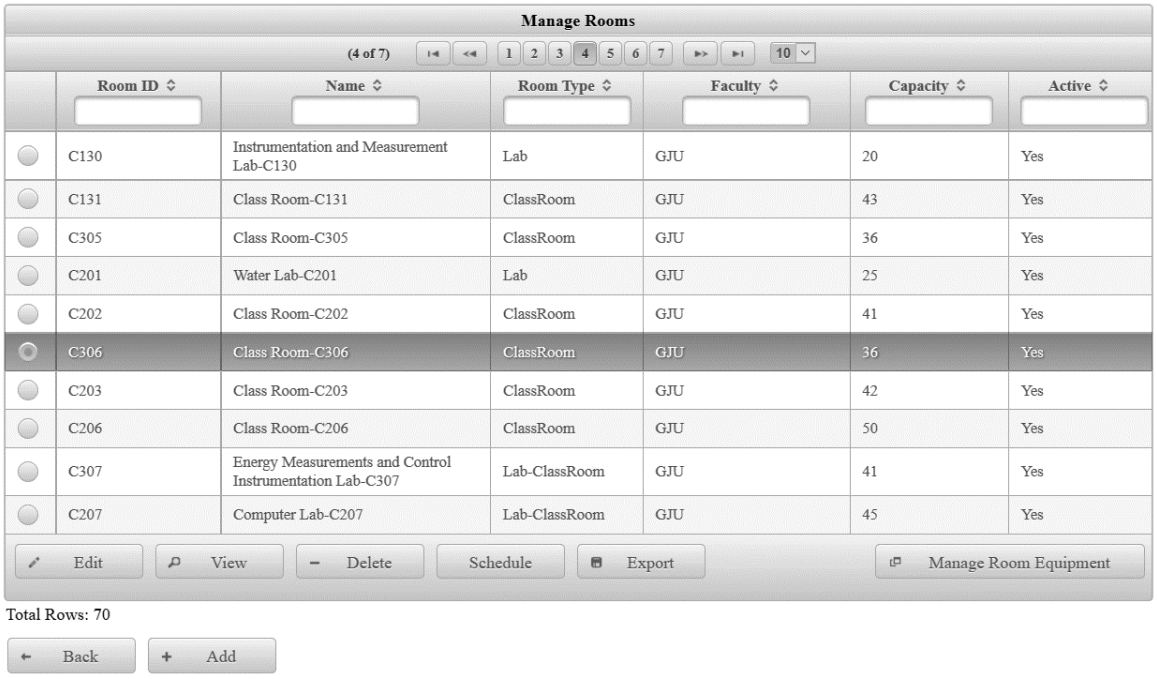

Figure 4. The manage rooms screen in the MyGJU Admin view

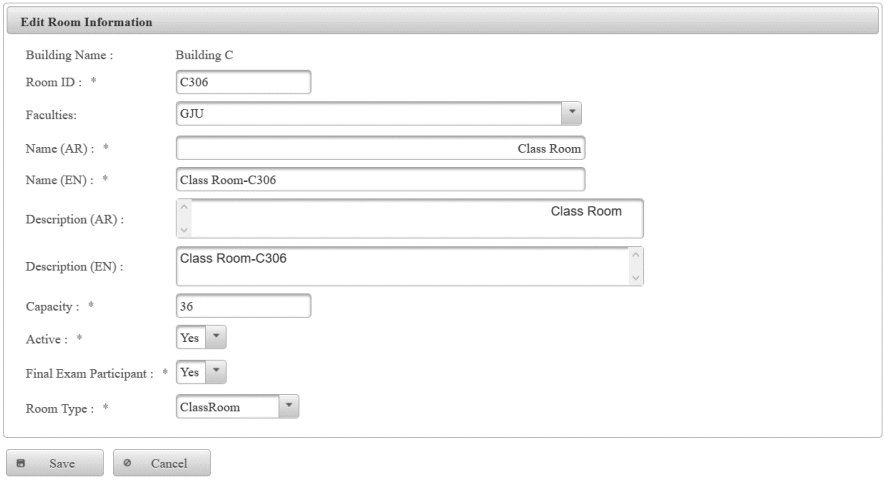

Figure 5. The edit room information screen in the MyGJU Admin view

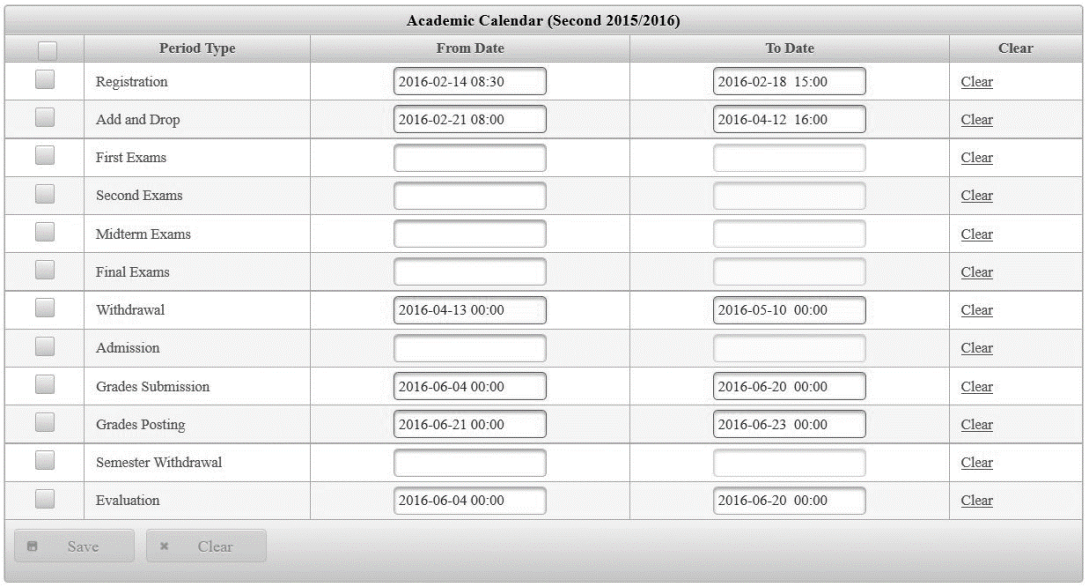

Figure 6. The academic calendar for the second 2015/2016 semester in the MyGJU Admin view

After completing the system setup tasks, the registrar needs to perform some of the academic setup activities that are discussed next.

Managing courses. This feature, for example, lets the registrar catalogue new courses, name a course, assign a code to a course, and edit the attributes of a course (e.g., credit hours, theoretical hours, practical hours, financial hours, type, and level). Besides, a course can be associated with a degree, faculty and department. Moreover, the user may define pre- and co-requisite courses as well as equivalent courses for any given course.

Managing course sections. The course sections scheduling task is considered as the basis for the registration process as the students will be enrolled only in the offered and available course sections. The registrars add, edit, and delete course sections based on the requirements that are provided to them by the various faculties at the GJU. The add course section feature in the MyGJU Admin (see Figure 7) is very flexible as it allows specifying the course section number, associating the course section with a course code, scheduling the course section on certain days and times, and assigning one or multiple instructors (e.g., a professor and a teaching assistant) to teach the course section. In addition, it allows allocating the course section to a room that accommodates the desired number of students (i.e., the entered course section capacity). Note that the MyGJU Admin automatically and on-the-fly populates the available rooms list with the rooms that fit the entered capacity and that are free on the desired days and times. Furthermore, it detects and reports any conflicts in the instructor's schedule in advance.

Managing prerequisite tests and exempted language courses. The prerequisite tests are usually taken by the newly accepted students in order to evaluate, for example, their language skills. Based on the test scores, it will be decided whether or not to exempt a student from some of the language courses. Both of those capabilities are available in the MyGJU Admin tool.

Managing study plans. The student needs to be linked to a specific study plan when admitted to a faculty. The study plan states the graduation requirements that the student needs to satisfy. It consists of several sections. Each section includes several courses (see Figure 8). The student must complete the total required credit hours of each study plan section to eventually fulfill the study plan's total required credit hours and hence be eligible to graduate. For example, based on the study plan shown in Figure 8, the students admitted to the Spatial Planning Master program need to finish 18, 9, 9, and 8 credit hours from the Compulsory Requirements, Graduation Projects, Free electives, and Remedial Courses study plan sections respectively to satisfy the study plan's total credit hours of 44. Furthermore, note that the course sections that a student may enroll in depend on whether or not their corresponding courses are included in the student's study plan. The MyGJU admin supports an easy to use wizard to enter the study plan basic information (e.g., name, effective year, degree, faculty, department, major, and total credit hours), define the study plan sections, assign courses to each study plan section, and define requisite courses for some of the courses in the study plan. Furthermore, it enables registrars to link students to their respective study plans.

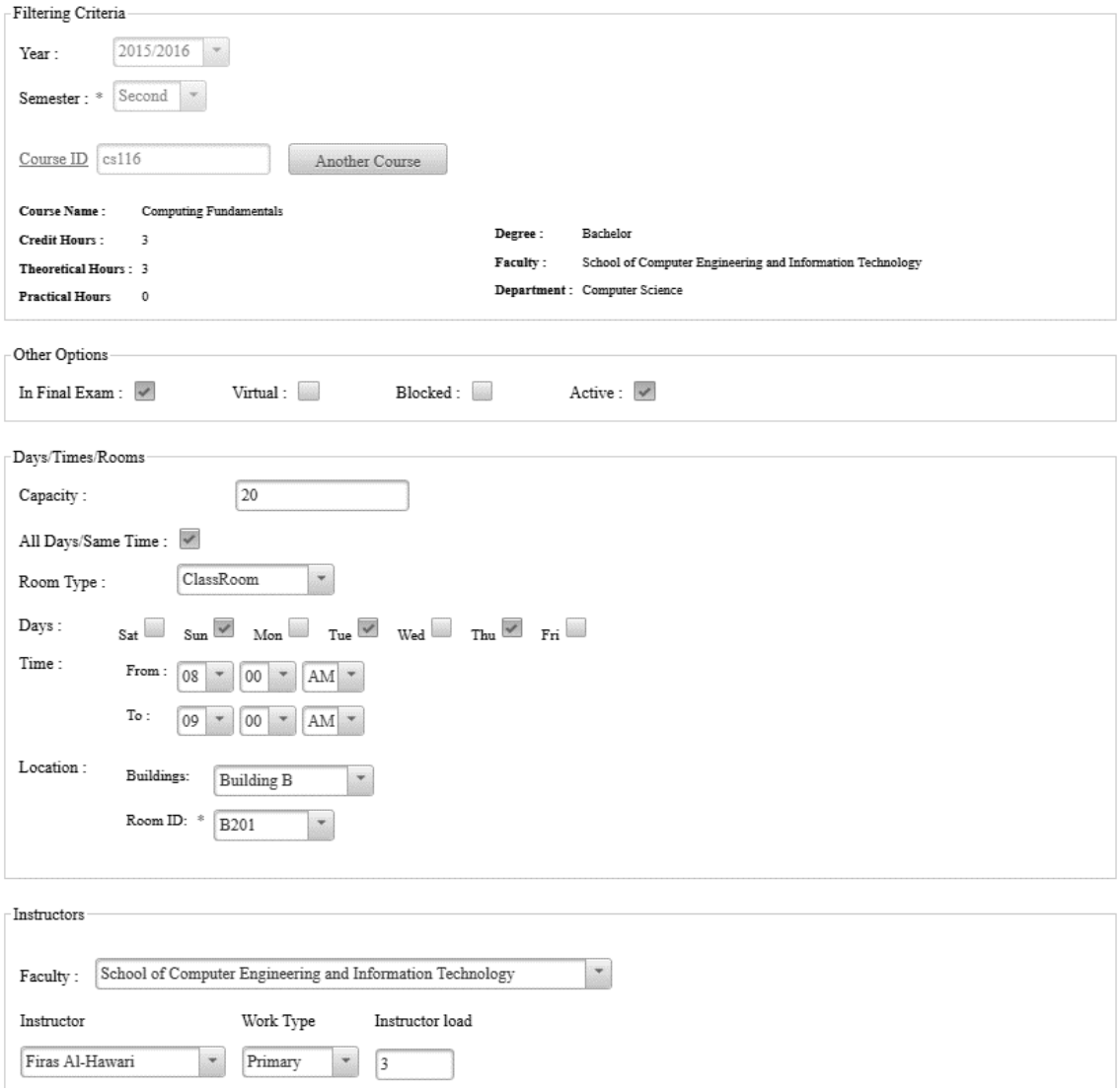

Figure 7. The add course section screen in the MyGJU Admin view

Study Plan Information

Study Plan:	Spatial Planning Master 2012/2013	**Study Plan Credit Hours:**	44
Degree:	Master	**Faculty:**	School of Architecture and Built Environment
Department:	Spatial Planning Master	**Major:**	Spatial Planning Master
Active From :	First Semester 2012/2013		

Study Plan Sections

Compulsory Requirements

Course ID	Course Name	Pre-requisites	Co-requisites	Pre Tests	Credit Hours	Theoretical Hours	Practical Hours
SP710	Planning Studio I				3	3	0
SP740	Planning Theories and Strategies				3	3	0
SP770	Sustainable Planning I				3	3	0
SP780	Spatial Socio-Economic Development Planning				3	3	0
SP711	Planning Studio II				3	3	0
SABE721	Research and Presentation Skills				2	2	0
SABE722	Technical Writing Skills				1	1	0

Section Total Credit Hours: 18
Section Required Credit Hours: 18

Graduation Projects

Course ID	Course Name	Pre-requisites	Co-requisites	Pre Tests	Credit Hours	Theoretical Hours	Practical Hours
SP799 A	Master Thesis / Spatial Planning				9	9	0
SP799 B	Master Thesis / Spatial Planning				0	0	0
SP799 C	Master Thesis / Spatial Planning				3	3	3
SP799 D	Master Thesis / Spatial Planning				6	6	6

Section Total Credit Hours: 18
Section Required Credit Hours: 9

Free Electives

Course ID	Course Name	Pre-requisites	Co-requisites	Pre Tests	Credit Hours	Theoretical Hours	Practical Hours
SP751	Appropriate Technology				3	3	0
SP783	Tourism Planning				3	3	0
SP701	Special Topics in Urban Design and Urban Regeneration				3	3	0

Section Total Credit Hours: 54
Section Required Credit Hours: 9

Remedial Courses

Course ID	Course Name	Pre-requisites	Co-requisites	Pre Tests	Credit Hours	Theoretical Hours	Practical Hours
ARC541	Urban Design				5	1	8
ARC346	Urban Studies				3	2	2

Section Total Credit Hours: 8
Section Required Credit Hours: 8

Figure 8. The study plan screen in the MyGJU Admin view

Acceptance

Application Id: 9426
Degree: Bachelor
Program: * Regular Program
Enrollment Year: 2015 / 2016
Enrollment Semester: Second

Department Selected

	Priority	Department Name
◉	First Choice	Computer Engineering
○	Second Choice	Pharmaceutical & Chemical Engineering
○	Third Choice	Communication Engineering

Major : * Computer Engineering
Study Plan: Computer Engineering 2014
Acceptance : * Competition
GJU Email: gju_test
Previously Dismissed: ☐

← Back Accept

Figure 9. The acceptance screen in the MyGJU Admin view

Upon finishing the system and academic setup activities, the registrar is required to conduct some of the following student affairs related tasks.

Managing admission applications. The prospective students may apply to one of the GJU degree programs by submitting an online admission application. The registrars accept or reject those applications based on the decision of the admission committee. An admitted student should be assigned a program, faculty, department, major, study plan, username, as well as an automatically generated student identification number (see Figure 9).

Managing student details. The registrar needs to manage the following student data: personal and academic information, schedules, grades, transcript, semester status, hold status, academic notes, transfer credit, substitute courses, prerequisite tests grades, and exempted courses.

Major transfers. The MyGJU Admin enables a registrar to transfer a student to another major. In that case, the system automatically transfers or exempts courses from the current student transcript based on whether or not the courses are in the study plan of the new major.

Registration and withdrawals. The registrar should be able, when needed, to add, drop, and withdraw courses on behalf of the students.

Processing Grades. The MyGJU Admin supports the following features related to grades: posting the submitted grades to the transcripts of the students, calculating the GPA for each student, viewing the transcripts, and editing the grades in a transcript.

Graduation. The MyGJU Admin enables registrars to find the expected to graduate students, graduate the students that completed the graduation requirements specified in their corresponding study plans, and generate the certificates for the graduating students.

In regard to the user affairs group, the MyGJU Admin supports the following activities: managing the users' (e.g., administrators, registrars, and instructors) accounts, managing the users' roles, and viewing the academic advisors.

As far as the financial affairs related tasks, the system allows the registrars to view tuition fees, view scholarships definitions, and link students to their scholarships. Noting that the fees and scholarships are defined by the accountants in the AIS, which is not discussed in this paper.

The MyGJU Admin also enables the registrars to generate various custom reports that aid in academic planning, monitoring, evaluation, and decision making. In addition, it uses a state-of-the-art graph coloring based integer linear programming (ILP) approach to generate a conflict free (e.g., a student cannot have two exams at the same time, a student should not have more than two exams in the same day) final exams timetable. Furthermore, it allows registrars to manage room bookings.

3.2. MyGJU Student View

This view targets the primary end users (i.e., the students) and its details are not considered in this paper. Using this application view, a student may view his/her academic information, study plan, schedule, grades, transcript, tuition fees, financial statement of account, as well as all the offered course sections in a given semester. In addition, students can add and drop courses during the registration period and they must evaluate their instructors at the end of each semester. The evaluation scores are considered by the university administration as one of the important factors that can be used to measure the instructors' performance and to review the educational process.

3.3. MyGJU Instructor View

The details of this view are not discussed in this paper, but it basically allows an instructor to view his/her personal information, schedules, and evaluations. Also, an instructor may manage the grades of his/her students, and may view all the offered course sections in a semester. Whereas the instructors who are given extra academic roles (e.g., advisors, chairs, or deans) may also view the information (e.g., study plans, transcripts, and financial statements) of their respective students in order to monitor their progress and to provide them with proper academic guidance. Furthermore, a dean or chair may also

perform extra administrative tasks such as increasing the capacity of a course section, approving or rejecting the submitted grades, assigning students to advisors, and generating various academic reports.

4. DESIGN PHASE

Designing an enterprise application is a daunting task because it requires specifying many UI screens and usability flows, identifying hundreds of modules and determining the relationships amongst them, and then connecting everything together in order to realize the desired application features. Therefore, a divide and conquer design strategy is very suitable in this case. Accordingly, the design problem is divided into several manageable parts, designing them independently, and merging the partial designs into a solution for the whole problem. In the rest of this section, we first present the adopted three-tier application architecture, and then we demonstrate how the divide and conquer strategy is applied to design each tier.

4.1. Three-Tier Web Application Architecture

The MyGJU Admin is an enterprise web application that is developed using the Java EE APIs [44]. Specifically, it is a distributed client-server application that is based on a three-tier architecture. Hence, its functionality is separated into three major tiers: client, web, and data (see Figure 10). Each tier has its own responsibilities and is distributed to a different physical location in a computer network. The three application tiers are discussed next.

Client Tier: This is the front-end (i.e., graphical user interface) of the web application. Specifically, in this case, it is the HTML pages (i.e., screens) that the user interacts with to login and use the supported application features. The HTML pages are displayed in the web browser that runs on the user's device (e.g., laptop, desktop, or smartphone). The web browser uses the HTTPS protocol to securely communicate with the web tier over a computer network.

Web Tier: In this case, it is the MyGJU Admin application that is deployed on the Java EE application server, which is hosted on the machines in the data center. The application is comprised of JavaServer Faces (JSF) pages [45], managed beans, and Java classes. The JSF pages contain JSF, HTML, and PrimeFaces [46] elements. The state of each JSF page is stored in its corresponding managed bean. The JSF framework acts as the controller in the Model View Controller (MVC) design pattern and hence it connects the View (JSF pages) to the Model (managed beans). The managed beans implement the application logic and uses Data Access Objects (DAOs) to query or update the data in the data tier. A DAO uses the Java Database Connectivity (JDBC) API [47] to interact with the database. The web server translates each JSF page to an HTML document and sends it back to the browser for display. Furthermore, the JSF framework supports other services such as data conversion, validation and error handling, as well as internationalization (supporting multiple languages) and Ajax.

Data Tier: The application data is stored in database tables and is managed in the data tier. The tables are managed by a database management system (DBMS) that is hosted on a database server. The web application interacts with the DBMS to query or update the data.

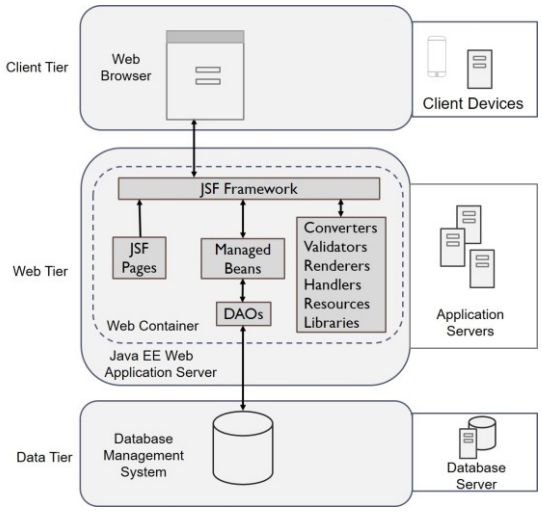

Figure 10. The three-tier architecture of the MyGJU Admin

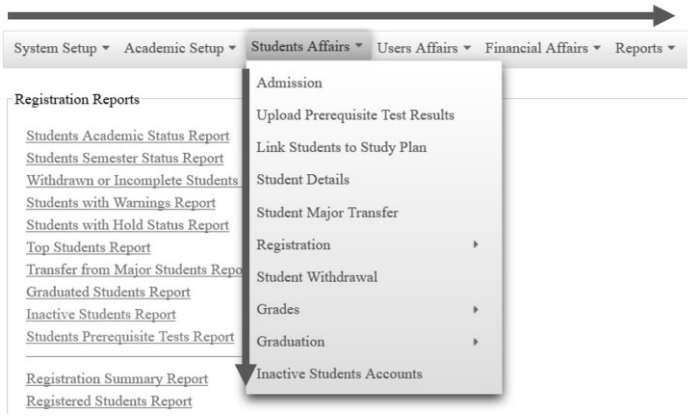

Figure 11. The main menu bar in the MyGJU Admin view

4.2. Client Tier Design

The client tier design maps to designing the user interface (UI) of the web application. The main goal of the UI design in this context is to allow the user to perform all the functional activities in the use case diagrams that were introduced in the requirement analysis phase. Accomplishing that goal, usually requires tens of sub menus and hundreds of screens. Hence, special care should be taken to simplify that design process and to make sure the resulting UI is effective, efficient, and easy to use.

The complex UI design can be simplified by breaking it into high and low level designs. The menu design is considered as the high-level structural design, while the design of the screens that are associated with each menu item is considered as the low-level detailed design. Therefore, the menu organization plays a key role in simplifying the UI design (by subdividing the problem into several manageable parts i.e., the menu items and their corresponding screens) and enhancing the user experience (by allowing the user to quickly find the menu items to perform the desired tasks).

For example, the menu bar for the MyGJU Admin tool (see Figure 11) was designed to allow easy access to the aforementioned six functional groups of tasks and to reflect the order by which those tasks may be performed by the registrars. Based on that, the menus for the preparatory system and academic setup groups were placed in the beginning of the menu bar. Then came the Students Affairs, Users Affairs and Financial Affairs menus, while the Reports menu was placed at the end of the menu bar according to their usage order as discussed earlier. The same applies to the order of the items in each menu. For example, a registrar may first admit some students, then check their academic details, next add or drop courses for a student, later post the grades of all students, and finally graduate some students. Accordingly, the Admission, Student Details, Registration, Grades, and Graduation menu items under the Students Affairs menu were ordered as shown in Figure 11.

An outcome of the menu design is the mapping of each group of related tasks to a corresponding menu item. Consequently, the screens related to each menu item can be considered as an independent group of screens that can be easily identified and designed. For example, the screens to manage buildings and rooms, manage the calendar events, manage course sections, manage admission applications, and enter grades are accessed from different menu items and hence can be easily designed independently of each other.

Furthermore, to enhance the user experience, each screen was designed to be clear, legible, and concise as illustrated in Figures 2-7. In addition, each screen can be accessed from one entry point (e.g., menu item, link, or button), and the number of steps and screens needed to accomplish a certain task were reduced as much as possible. Moreover, all similar screens were designed in a consistent manner. For example, all the manage screens contain a table of selectable rows (see Figure 4). The footer of the table contains the buttons that apply to a selected row (e.g., Edit, View, and Delete). While, the Add and Back buttons are usually located at the end of the screen as they are independent from the data in the table. Maintaining a consistent UI also required having consistent labels, messages, style (e.g., font, color, size, and background), icons, UI elements, element placement (e.g., buttons and menu placement), and behavior (e.g., what to expect in case of an error).

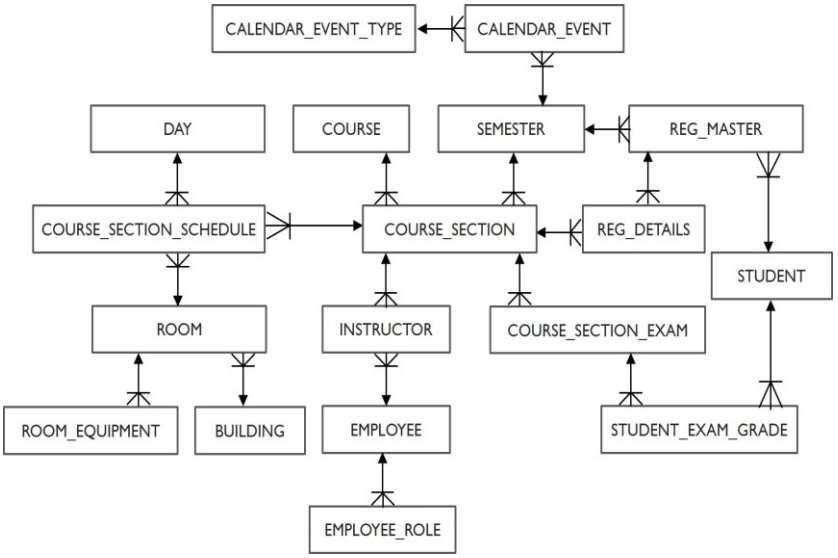

Figure 12. ER diagram for the rooms, courses sections, registration and entering grades features

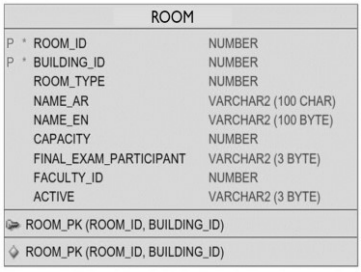

Figure 13. The database table for a room.

4.3. Data Tier Design

The design of the data tier is basically the database design. The database for the MyGJU Admin system is huge and is comprised of about 250 tables and views. Again, the complex database design process was simplified by splitting the tables into independent groups, designing each group of tables independently from its counterparts, and then establishing the relationships amongst the tables in the different groups to complete the whole database design. Logically, the table group identification is based on the UI organization. Therefore, the related screens that are associated with a menu item to accomplish a specific functional task will usually be associated with a group of related database tables.

For example, the Entity-Relationship (ER) diagram shown in Figure 12 captures the relationships amongst the database tables for some of the features in the MyGJU Admin view such as the manage buildings and rooms, manage course sections, registration, and entering grades. In an ER diagram an entity (i.e., table) is represented via a box, while any two related entities are connected via a line that has different shapes at its ends to represent the cardinality of the relationship (e.g., one-to-many, many-to-many, and one-to-one). Moreover, each entity in the ER diagram may be annotated with its corresponding attributes. For example, the attributes of the room entity (e.g., identifier, building, type, name, capacity, and faculty) may be specified as shown in Figure 13.

Based on the example ER model, a building may contain many rooms (i.e., one-to-many relation), but a room belongs to one building (i.e., one-to-one relation). In addition, a course may have many course sections, but each course section is associated with one course. A course section can have multiple instructors (e.g., professor and teaching assistant) and schedules (e.g., different start and end times on different days). Moreover, a student may have one registration schedule (i.e., one record in the REG_MASTER table) per semester, while a registration schedule can be associated with the course sections that the student is enrolled in (note that each association between a student and a course section is recorded in the REG_DETAILS table and is linked to the corresponding record in the REG_MASTER table). Furthermore, an instructor may conduct several exams in a course section (e.g., first, second, and final), but he/she can enter only one grade for each student per exam.

Besides, the ER diagram illustrates how the different groups of tables were associated with each other in a step to complete the whole database design. For example, the registration tables were associated with the manage course sections tables via the crow foot linking the REG_DETAILS and COURSE_SECTION entities. While, the crow foot between the COURSE_SECTION_SCHEDULE and ROOM entities associated the manage course sections tables with the manage buildings and rooms tables.

4.4. Web Tier Design

The design of the web tier depends on the design of the client and data tiers, hence it is discussed last in this section even though the web tier is located in the middle of the application's architecture stack. In this tier, each front-end screen is usually associated with a managed bean class, while each back-end table is mapped to a model class. During the design phase, the structure of those classes can be described using class diagrams, whereas a snapshot of the instances of those classes at a certain time may be captured using object diagrams. While an activity diagram, for example, may be used to describe the needed application interaction and logic to accomplish a certain task or a set of related tasks. The application interaction shows how the user interacts with the application to achieve a specific goal. While the application logic describes how the application processes the client's requests, communicates with the database (if needed), and then sends the responses back to the client to deliver the desired service.

This design methodology is used to describe the various task flows in the MyGJU applications. Typically, one or several task flows might be associated with a group of related screens that support some of the offered features (e.g., manage course sections and registration). For example, we next discuss how the registration flow in the MyGJU Admin application was designed using this methodology (note that the registration flow from the MyGJU student view is different and will be presented in another paper).

The three screens that are used in the registration flow in the MyGJU Admin are shown in Figure 14, whereas the activity diagram for that flow is shown in Figure 15. For simplicity of presentation, note that the activity diagram only spanned the client and web tiers, but it did not cover the data tier despite the fact that the web tier usually communicates with it to query or update the data. Moreover, a registrar (client) may navigate to (i.e., reach the starting point of the activity diagram) the Student Registration screen (top screen in Figure 14) from the Registration sub menu shown in Figure 11.

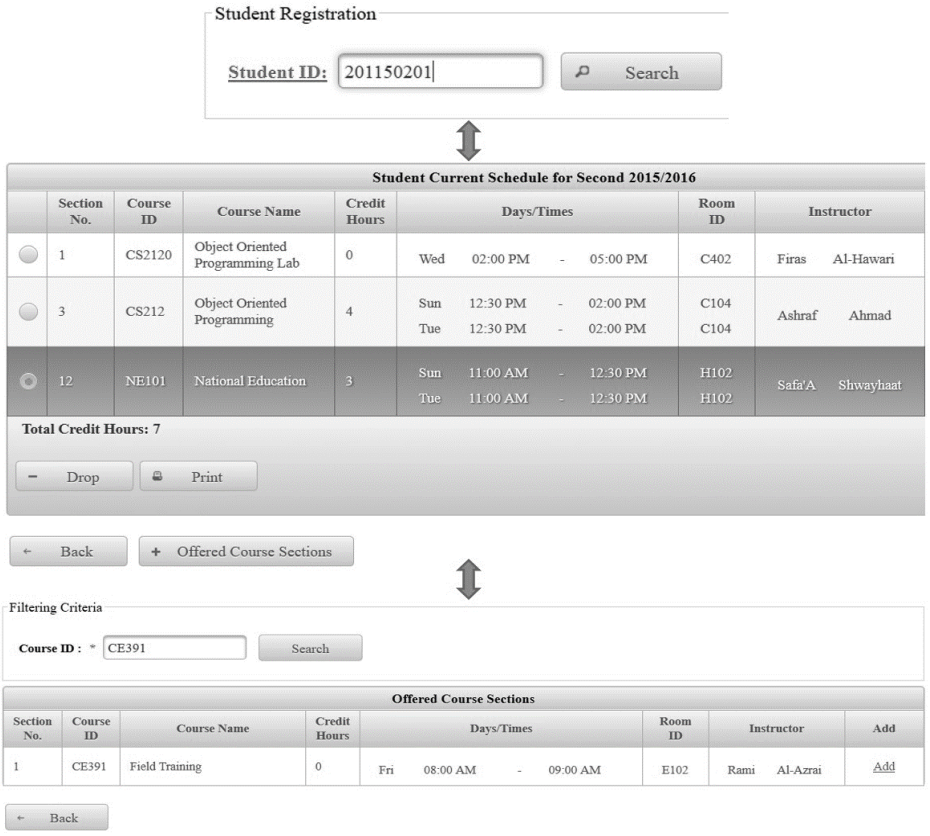

Figure 14. A snapshot of all the screens used in the registration flow in the MyGJU Admin application

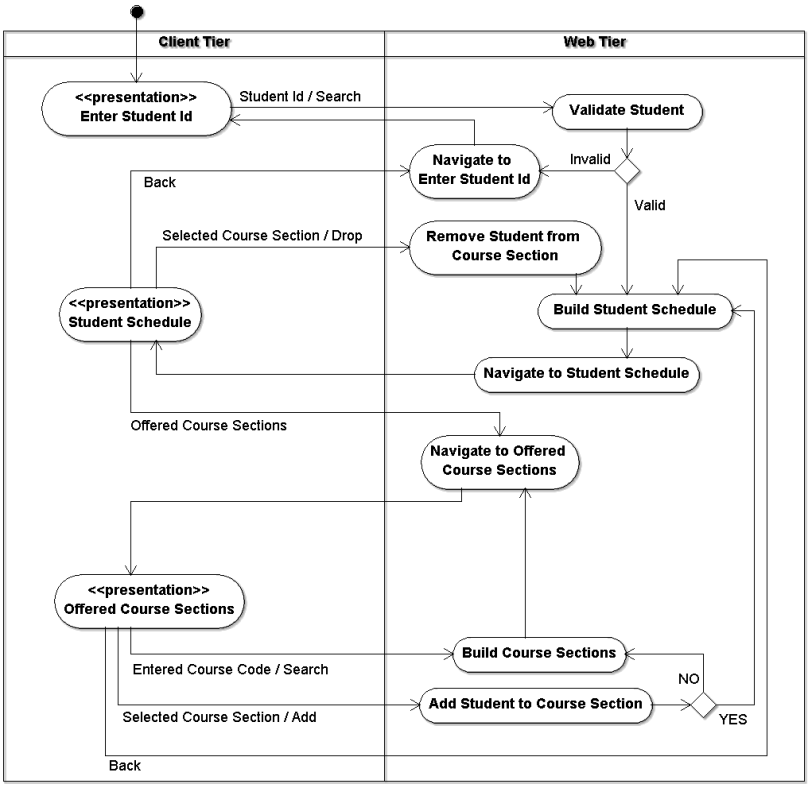

Figure 15. The activity diagram for the registration flow in the MyGJU Admin application

Based on the shown activity diagram, when a registrar enters a student Id and clicks the Search button in the Student Registration screen, the client's browser sends the entered student Id within a search request to the web tier for processing. If the student Id is valid (e.g., Id is numeric, Id exists, the student account is active, the student is not dismissed from university, and the student did not graduate yet), the web tier (specifically the responsible managed bean object) retrieves the student's schedule (i.e., the list of course sections that the student is enrolled in) from the database and sends it back to the registrar's browser for display. Otherwise, it displays an error message on the top of the Student Registration screen explaining what went wrong. When the Student Current Schedule screen (the middle screen in Figure 14) is displayed, the registrar may perform the following actions: select and then drop a course section from the student's schedule (by selecting a row and then clicking the Drop button), print the student's current schedule (by clicking on the Print button), go back to the previous page (by clicking on the Back button), or navigate to the Offered Course Sections screen (by clicking on the Offered Course Sections button). After a registrar navigates to the Offered Course Sections screen (bottom screen in Figure 14), he/she can enter a course code (e.g., CE391) and then click the Search button to send a search request to the web tier for processing. The managed bean object in the web tier, in turn, retrieves all the course sections (if any) for the received course code from the database and sends them back to the browser for display. Now, the registrar may try to enroll the student in one of the offered course sections by clicking on the Add link of the desired course section. In response to that, the browser sends an add course section request to the web tier to enroll the student in the selected course section, if possible. The student will be enrolled in the selected course section only when all the following conditions are satisfied: the course section's capacity is not exceeded, the student's account or scholarship (if any) balance is enough to cover the cost of the course, and the total credit hours in the student's schedule did not exceed the maximum allowed credit hours. Accordingly, if the course section is added successfully to the student's schedule, the web tier rebuilds the new student schedule and sends it back to the browser for display. Otherwise, it displays an error message on the top of the Offered Course Sections screen informing the registrar of what happened so he/she may try to enroll the student in another course section, if desired.

It is worth mentioning that when the web application uses JSF pages, the web server performs several pre-processing steps ahead of sending a reply back to the browser. For example, to send the student's schedule back to the browser, the JSF framework on the web server generates the HTML document that encapsulates the student's schedule table by rendering (encoding) each JSF element in the corresponding JSF page to an HTML element. The web server, in turn, packages the generated HTML document in an HTTP response and sends it back to the browser at the client side for display.

5. DEVELOPMENT PHASE

Implementing the features associated with one or several screens in a three-tier web application usually requires developing the following basic components:

JSF Pages: A JSF page is considered as the view layer in the web tier. In our case, it may contain a combination of HTML, JSF, and PrimeFaces elements (tags) to represent the UI components (e.g., form, table, calendar, button, text field, and checkbox) in a certain page (screen) in the system.

Managed Beans: A managed bean is a special Java class that is managed by the web container. In other words, the web container instantiates, destroys, and manipulates the managed bean object. Each JSF page in the application is usually associated with a managed bean to store its state and to handle its business logic. Therefore, a managed bean has attributes to store the values of the UI components that comprise its corresponding JSF page. Besides, it implements several methods to support the following functions: get and set the values of the UI components, validate the UI components' data, and handle the events fired by the UI components. Based on that, a managed bean represents the model layer in the web tier, while the web container is considered as the controller layer in the MVC design pattern.

Entities: The definition of each database table is mapped to a corresponding entity class in order to enable a Java program to access the table's data. Hence, an instance of a certain entity class would map to a row in its corresponding table, while attributes in that instance would map to individual cells in its row. This mapping process is called Object Relational Mapping (ORM) and it can be automated using tools like Hibernate [48].

Data Access Objects (DAOs): A DAO is a Java object or API that is usually used by a managed bean (in the web tier) to interact with the data tier i.e., the database. For example, a DAO would contain methods to query, add, delete, or update a row in a database table. It uses the JDBC API to access the database and return the retrieved row data as entity objects back to the calling managed bean for processing.

Next, for example, we show the implementation details for the basic application components that pertain to the manage rooms feature. Then, we discuss the implementation choices that may affect the performance of the application.

5.1. Basic Components Implementation

The partial JSF page for the manage rooms screen is shown in Figure 16. It shows the dataTable element (a PrimeFaces component) and some of its attributes starting at line 2. In addition, the column element for the roomId column and the commandButton element for the View button are shown starting at line 10 and line 15 respectively. Whereas, the unseen elements for the rest of the table columns and buttons are developed in a similar manner as their shown counterparts. The table rows are stored in, and accessed from, the roomsList attribute in the managed roomsBean instance (at line 3). While, the selected table row (i.e., the selected room) is saved in the selectedRoom attribute in the roomsBean object (at line 5). Furthermore, table pagination is enabled and a default number of rows per page is set to ten at line 6 and line 7 respectively. Moreover, the viewSelectedRoom method would be invoked on its roomsBean object to handle the click View button action event as shown at line 17.

The partial Java code for the managed RoomsBean class that is associated with the manage rooms page is shown in Figure 17. The @Named annotation at line 1 is used so that the RoomsBean instance would be treated by the web container as a CDI (Context Dependency Injection) managed bean. While the @ViewScoped annotation at line 2 means that the managed bean has a view scope (lifetime). The web container instantiates a view scoped bean when the user navigates to its corresponding JSF page

(i.e., the manage rooms page in this case) and keeps it alive as long as the user is interacting with that same JSF view. Once the user navigates to a different page, the web container discards the managed bean to free up the server's memory. Consequently, using view scoped beans rather than request scoped beans also improves the application's performance because the container would not rebuild them on every post back to the same view.

In addition, the code at lines 4 and 5 (in Figure 17) injects the ScratchBean instance into a RoomsBean object. A ScratchBean is a session scoped bean that lives and holds its data as long as its established session. It is developed to allow a view scoped bean to pass data to another bean before it goes out of scope. For example, the user may select a building in the manage buildings screen (not shown here) and then navigate to the manage rooms page to view the selected building's rooms. Since the ManageBuildings bean (not shown here) is also view scoped, it first saves the selected buildingId (which is needed by the bean of the next page) in the scratchBean instance so it will not be discarded upon navigating to the manage rooms page. Consequently, after constructing the RoomsBean object, the init method (at line 11) would be invoked to get the saved and needed buildingId from the scratchBean object (at line 12), instantiate a RoomsDao object (at line 13), and pass the buildingId to the buildRooms method to retrieve the rooms of the previously selected building from the database (at line 14). Similarly, for example, a RoomsBean instance would save the selected roomId in the scratchBean instance (at line 19) ahead of being trashed by the web container upon navigating to the view_room page (at line 20), which needs that data to retrieve (from the database) and then view the information of the selected room.

The partial Java code for the Room class is shown in Figure 18. It corresponds to the ROOM table shown in Figure 13. While the partial Java code for the RoomsDao is shown in Figure 19. The RoomsDao provides methods to query and manipulate (i.e., insert, delete, and update) the data in the ROOM table. In this example, we only show the buildRooms method that starts at line 2 (which is used in the init method of the RoomsBean class) to illustrate how the DAO interacts with the database via the JDBC API to build an ArrayList of Room instances (i.e., room rows) for the selected building. At line 4, the parent (i.e., GeneralDao class) getConnection method is invoked to get a connection to the database. Then at lines 7-9, the SQL query to obtain the building's rooms is constructed. Next at lines 10-12, a PreparedStatement object is used to execute the SQL query and obtain the ResultSet object.. Then at lines 14-19, the attributes of each Room instance are populated with the values of the columns of each corresponding table row. At line 20, a Room instance gets added to the rooms list. Finally at lines 22-24, the database resources are released.

```
<h:form id= "manage_rooms_form">
    <p:dataTable id= "manage_rooms_tbl"
                 value="#{roomsBean.roomsList}"
                 var="room“
                 selection="#{roomsBean.selectedRoom}“
                 paginator="true"
                 rows="10"
                 <!-- other attributes here !--> >

        <p:column headerText="#{msgs.room_id}">
            <h:outputText value="#{room.roomId}" />
        </p:column>

        <!-- other columns here !-->

        <p:commandButton id="view_button"
                         value="#{msgs.view}"
                         action="#{roomsBean.viewSelectedRoom()}"
                         <!-- other attributes here !--> />

        <!-- other buttons here !-->
    </p:dataTable>

    <!-- other components here !-->
</h:form>
```

Figure 16. A partial snapshot of the manage rooms JSF page

```
@Named
@ViewScoped
public class RoomsBean implements Serializable {
    @Inject
    private ScratchBean scratchBean;

    private ArrayList<Room> roomsList;
    private Room selectedRoom;
    private RoomsDao roomsDao;
    // other attributes here ...

    @PostConstruct
    public void init() {
        int buildingId = scratchBean.getSelectedBuildingId();
        roomsDao = new RoomsDao();
        roomsList = roomsDao.buildRooms(buildingId);
    }

    // Getters and setters here

    // The view room button handler
    public String viewSelectedRoom() {
        scratchBean.setSelectedRoomId(selectedRoom.getRoomId());
        return "view_room.xhtml"; // navigate to the view_room page
    }

    // other methods here ...
}
```

Figure 17. A partial snapshot of the RoomsBean class

```
public class Room implements Serializable {
    private int roomId;
    private int buildingId;
    private int roomType;
    private String nameAR;
    private String nameEN;
    private int capacity;
    private boolean finalExamParticipant;
    private int facultyId;
    private boolean active;

    // getters and setters here ...
}
```

Figure 18. A partial snapshot of the Room class

```
public class RoomsDao extends GeneralDao {
    public ArrayList<Room> buildRooms(int buildingId) throws Exception {
        try {
            Connection conn = getConnection();
            ArrayList<Room> rooms = new ArrayList<>();

            String sql = "SELECT * FROM ROOM "
                       + " WHERE BUILDING_ID=? "
                       + " ORDER BY ROOM_ID ASC";

            PreparedStatement ps = conn.prepareStatement(sql);
            ps.setInt(1, buildingId);
            ResultSet rs = ps.executeQuery();

            while (rs.next()) {
                Room room = new Room();
                room.setBuildingId(rs.getInt("BUILDING_ID"));
                room.setRoomID(rs.getInt("ROOM_ID"));
                room.setNameAR(rs.getString("NAME_AR"));
                room.setNameEN(rs.getString("NAME_EN"));

                // populate the rest of the room attributes

                rooms.add(room);
            }

            rs.close();
            ps.close();
            conn.close();

            return rooms;
        } catch (Exception e) {
            throw new Exception(e.getMessage());
        }
    }

    // other methods here ...
}
```

Figure 19. A partial snapshot of the RoomsDao class

5.2. Application Performance Considerations

It is worth mentioning in this context that table pagination in combination with lazy loading should be used to view large data tables in order to improve the performance of the application. In that case, the managed bean will retrieve only the number of rows of the selected table page (which is usually a small number e.g., it is 10 rows per page in Figure 4) as opposed to querying all the rows in the table, which in some cases can be in the range of tens of thousands of rows. Consequently, that makes the application more efficient as it reduces the demand on the database and limits the amounts of data transmitted over the network and then cached in the web tier memory

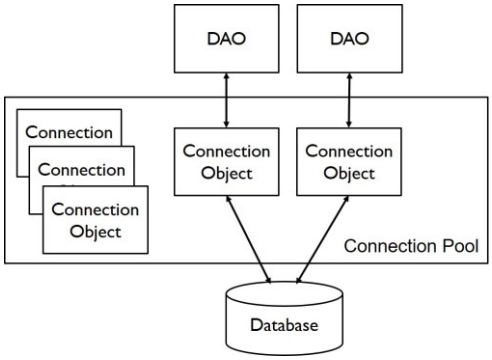

Figure 20. Interactions between the DAOs and the Connection Pool

```
1.  public class GeneralDao implements Serializable {
2.    private DataSource ds;

3.    public Connection getConnection() throws Exception {
4.      Connection connection = null;

5.       ds = (DataSource) new InitialContext().lookup("jdbc/mygju");
6.       connection = ds.getConnection();

7.       return connection;
8.    }

9.    // other methods here ...
10. }
```

Figure 21. A partial snapshot of the GeneralDao class

In addition, it is obvious that the choice of the managed beans scopes may affect the performance of a distributed web-application with thousands of users. Therefore, the view scoped beans should be used for all the Ajax-enabled pages that need to save the page state across several post backs to the same view, while a limited number of session scoped beans should be used to store a small size of scratch data and logged-in user information (e.g., username, full name, preferences, and role) for the lifetime of the session. Accordingly, the memory consumption is optimized as the data is kept in memory only as long as it is needed. Furthermore, the CPU usage is reduced as the unnecessary and possibly time consuming data reloads are avoided.

The web application cannot open a connection to the database for every page request as that can be time consuming. Moreover, the application cannot keep open a large number of connections to the database as they consume the resources of the database server and the limit that the database usually puts on the maximum number of concurrent connections can be reached. Such problems are solved by setting up a JDBC connection pool in the Java EE web application server. Consequently, a DAO can obtain an already opened connection object from the available connection objects in the connection pool (see Figure 20). When the connection object is no longer needed by the DAO and the close method is invoked on it (as shown at line 24 in Figure 19), the open connection object will be returned to the pool, but not closed, so it can be reused by another DAO. Accordingly, the connection pool minimizes the time to get a database connection object. The parent GeneralDao class that is extended by all the application's DAOs implements the getConnection method that starts at line 3 in Figure 21. At line 5, a data source object is obtained from the JDBC connection pool. Then at line 6, the getConnection method of the data source object is called to obtain a connection object from the connection pool.

6. TESTING PHASE

Testing such a complex system can be simplified by performing the tests at different levels during and after the software implementation phase. The three standard levels of software testing that we followed are discussed next.

6.1. Unit Testing

At this testing level, each software component (i.e., unit) was tested in isolation from other components to decide whether each unit operates as specified. A software unit can be a method in a class, a class as whole (e.g., managed bean, entity, or DAO), and a JSF page (i.e., a screen) and its underlying code. All components were tested to determine whether they work correctly in all scenarios (functionality), complete in a reasonable time (performance), handle invalid input gracefully (robustness), and support both English and Arabic (internationalization). Whereas, each UI page was also tested to check if it is easy to use (usability) and behaves the same in all browsers (compatibility). The unit tests were performed by each respective developer while implementing the corresponding components. Hence, they allow identifying and fixing software faults at an early stage in the development cycle, which leads to the following benefits: strengthens the software foundation, minimizes the time to integrate the various components as they are already tested and functional ahead

of that stage, and reduces maintenance costs as it is more expensive to fix a fault in the final phases of the software engineering cycle as opposed to repairing it in the initial phases.

6.2. Integration Testing

In this case, the integrated and related software components (e.g., manage buildings and rooms screens, manage course sections screens, registration screens, etc.) were each tested as a group to verify if they function correctly and perform as expected when they interact with each other. Progressively the verified modules that will be used in a sequence of a certain workflow were combined and tested to determine how they act in a larger group. For example, a workflow test may start by adding a new room in the system (in the manage rooms module), then assigning a course section to that room (in the manage course sections module), and finally adding students to that course section (in the registration module). Such tests were conducted initially by the development team and later by the quality assurance (QA) engineers either manually or automatically using tools like Selenium [49].

6.3. System and Acceptance Testing

This level deals with testing the MyGJU Admin application as a whole. We performed three types of tests at the system level: load, alpha, and beta. The load test is a non-functional test that was conducted ahead of releasing the system to gain confidence in the ability of the application to scale up and service many concurrent users in a fast and reliable manner under normal and heavy load conditions. We ran several automated load tests in which we increased the number of concurrent users as follows: 50, 100, 250, 500, and 1000. We also varied the think times, inter-arrival times, and workflows to mimic various realistic scenarios. Whereas, the alpha and beta tests were conducted by technical employees and end users respectively to find software defects and to make sure that the system is easy to use and meets the functional requirements ahead of its release. The developers, QA engineers and the IT engineers in the computer center participated in the alpha testing. Subsequently, the registrars as well as the instructors and students of the school of architecture and built environment at the GJU were chosen to use and test the beta version of the application during the summer 2015/2016 semester. Such tests were very effective as the majority of the features and workflows in the system were used and exercised in realistic scenarios by the various testers. The feedback of the testers as well as the results of the various tests were all immediately incorporated and addressed in the system. Accordingly, the application was successfully launched in the first 2015/2016 semester with a lot of positive feedback received from the end users regarding its capabilities, ease of use, performance, and reliability.

7. DEPLOYMENT PHASE

The system requirements, application security, as well as user authentication and authorization are important topics that we addressed in the system deployment phase and hence are discussed in the following subsections.

7.1. System Requirements

In order to efficiently run the MyGJU Admin application, several hardware and software requirements must be satisfied. First, the web application and database have to run on powerful servers with enough cores (in our case its 24 cores each) and memory size (32 GB each). Besides that, the servers should be connected to network switches via high bandwidth links (10 Gbps links). Whereas, several web application server and DBMS settings must also be fine-tuned. The settings of the following application server resources need to be considered: JDBC connection pool, network listeners, thread pool, and Java virtual machine (JVM). In addition, DBMS settings such as number of connections, number of cursors, and size of buffer cache should be optimized. The system requirements are usually determined based on the observed resource utilization when running a load test for that purpose and while the actual application is deployed.

7.2. Application Security

The following security schemes are applied to protect the MyGJU application servers from any unauthorized access:

- The secure Linux operating system is installed on all servers.
- All unnecessary services on all machines are turned off.
- The operating systems and applications are always patched with the latest hotfixes to close any

vulnerabilities that may be exploited by hackers.

- Anti-virus and anti-spyware software is installed on all computers to protect them from such threats.
- Several security policy rules are defined on the firewall (see Figure 22) to: restrict access from the Internet to the servers on the Intranet (e.g., the MyGJU Admin and database servers), permit only specific users (e.g., system and database administrators) on the Intranet (i.e., from the inside) to access the MyGJU Admin and database servers, permit all users to access the MyGJU server in the demilitarized zone (DMZ) only via the server's public IP address and the HTTPS service port.
- A strong password policy is enforced on all users and administrators to avoid compromising any of their accounts.

Moreover, the application DAOs interact with the database via the JDBC PreparedStatement API in order to prevent SQL injections. In addition, the https protocol is enabled in the application server to secure the communication channels between the clients and the application.

7.3. User Authentication and Authorization

The setup of the user accounts is a very important aspect to consider before releasing the system. In that regard, we implemented a Single Sign-On (SSO) authentication process that allows a user to login to the MyGJU tools as well as other key applications (e.g., E-Mail, E-Learning) using the same credentials (i.e., using one username and password). That saves users from having to memorize many passwords and reduces helpdesk cost by minimizing the requests for password resets. The Microsoft Active Directory (AD) [50] is used at the GJU as a directory service to store and verify the login credentials of all staff members and students. For security purposes, the information of the staff members and students are placed in different organizational units respectively within the GJU domain in the AD. Besides that, the Lightweight Directory Access Protocol (LDAP) [51] is used by the MyGJU Admin tool via the Java Naming and Directory Interface (JNDI) [52] to communicate with the AD for user authentication.

A partial code snippet for the authentication code is shown in Figure 23. At lines 1, 2, and 3-4 the search base, filter, and controls are initialized respectively. At line 5 the env Hashtable object is instantiated and then populated with the username, password, and ldapURL (contains the ldap server IP address and port) objects at lines 6, 7, and 8 respectively. Then, the ctx DirContext object is instantiated at line 9. Consequently, the search method is invoked on the ctx object at line 10 in order to find the mailNickname attribute as shown at lines 11-13. If the mailNickname attribute is not null as checked at line 14, then the user is considered authenticated. Otherwise, the user login fails.

After authenticating a user in the AD, the user information that is stored in the MyGJU Admin database is used by the system for authorization purposes. For example, a registrar will be denied access to the MyGJU Admin if his/her account does not exist or is inactive.

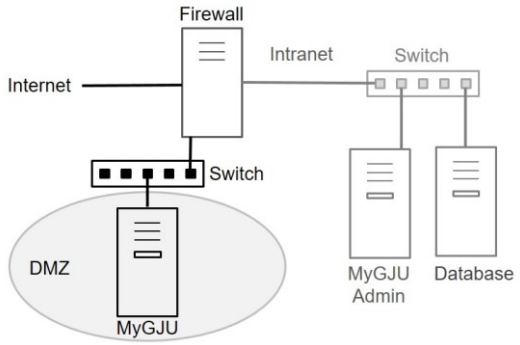

Figure 22. The firewall and network setup for the MyGJU applications

```
1.  String searchBase = "OU=Registrars,DC=gju,DC=edu,DC=jo";
2.  String searchFilter = "(&(objectClass=person)(mail=" + username + "))";
3.  SearchControls searchCtrls = new SearchControls();
4.  searchCtrls.setSearchScope(SearchControls.SUBTREE_SCOPE);

5.   Hashtable<String, String> env = new Hashtable<String, String>();
6.   env.put(Context.SECURITY_PRINCIPAL, username);
7.   env.put(Context.SECURITY_CREDENTIALS, password);
8.   env.put(Context.PROVIDER_URL, ldapURL);
9.   DirContext ctx = new InitialDirContext(env);

10.  NamingEnumeration ne = ctx.search(searchBase, searchFilter, searchCtrls);
11.  SearchResult sr = (SearchResult) ne.next();
12.  Attributes attributes = sr.getAttributes();
13.  Attribute attribute = attributes.get("mailNickname");

14.  if (attribute != null) {
15.     authenticated = true
16.  }
```

Figure 23. A partial code snippet for the MyGJU authentication code

8. VALIDATION AND RESULTS

The results of the conducted system measurements and user survey are discussed in the following subsections to illustrate that the system is feature rich, fast, reliable, stable, available, scalable, and easy to use.

8.1. Measurement Results

The system provided several capabilities to permit the registrars to complete all the needed basic tasks as summarized in Table 3. Based on that, it enabled the registrars to accept 1488 new students, complete the major transfers for 154 students, open 3973 new course sections, withdraw courses for 831 students, and graduate 563 students. In addition, it allowed both registrars and students to complete 81842 add and drop course section operations. Moreover, it enabled instructors to enter 42268 grades.

The mean and standard deviation (stddev) of the elapsed times it took registrars to perform some common tasks in the fall 2016/2017 semester are reported in Table 4. Accordingly, it took a registrar about 52 seconds (on average) to open a new course section using the screen in Figure 7. Hence, it only took registrars about 17 hours (on average) to open all of the course sections (i.e., 1164 course sections) for that semester, which indicates that registrars accurately achieved this basic task with ease and in a short time. Whereas, registrars were able to compute the GPA for any student and view any study plan in less than 2 seconds and 1 second (on average) respectively (i.e., almost at the click of a button). While, they enrolled students in course sections using the flow discussed in subsection 4.4 in about 16.4 seconds (on average) per operation (i.e., an add course section operation). It is worth mentioning that in these cases the elapsed time may include think time, entry time, communication time, and execution time. Furthermore, the mean and stddev values were computed based on the measured elapsed times of all the operations to perform each respective task of interest. For example, the mean and stddev values in the Add Course Section column in Table 4 were computed based on the elapsed times (shown in Figure 24) to complete each of the 564 add course section operations that were performed by registrars on behalf of students during the registration period for the latest fall semester.

The total number of user sessions per day since the launch of the system in September 2015 till the end of November 2016 are shown in Figure 25. This data illustrates that the system was available every day (i.e., 443 days) during that period. Accordingly, and given the fact that the system was only brought down for scheduled updates that took around 30 seconds per month during that period, the system achieved 0.999989 availability (i.e., it is highly available). Moreover, the system was able to handle a minimum workload of 100 sessions on April 30, 2016 and a maximum workload of 13692 sessions on September 30, 2015 (i.e., during the registration period for the fall 2015/2016 semester). Hence, it proved to be scalable as it withstood a workload that scaled up to 136.92 times the minimum recorded workload without any degradation in performance. Whereas, the recorded number of new user sessions per minute during the peak day (i.e., September 30, 2015) are shown in Figure 26 and demonstrate that the system also supported a maximum of 55 new sessions per minute at the beginning of the registration period (which started at 8am and ended at 3pm) during that day. As far as the seven obvious peaks in Figure 25, they are related to the registration and grades submission periods in the fall, spring, and summer 2015/2016 semesters as well as the registration period in the fall 2016/2017 semester.

Table 3. The number of performed basic tasks since the launch of MyGJU till the end of November 2016

	Admission	Major Transfers	Study Plans	Course Sections	Registration	Withdrawals	Grades	Graduation
#	1488	154	30	3973	81842	831	42268	563

Table 4. The mean time and standard deviation to perform some of the basic tasks using the MyGJU Admin

	New Course Section		Compute Student GPA		View Study Plan		Add Course Section	
	Mean	Stddev	Mean	Stddev	Mean	Stddev	Mean	Stddev
Time (seconds)	51.988	13.754	1.848	1.099	0.791	0.385	16.422	18.930

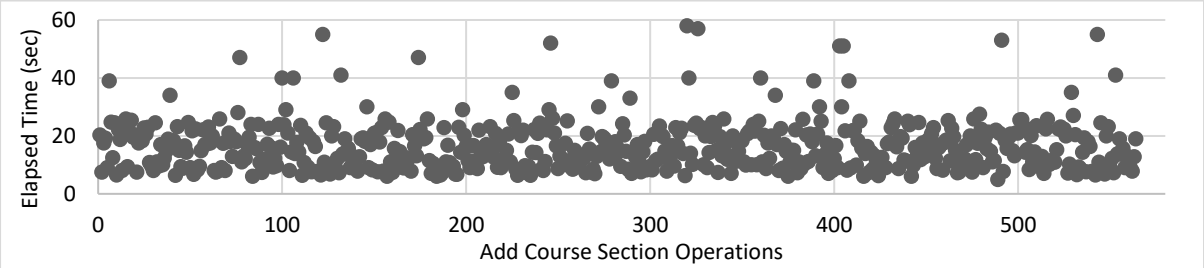

Figure 24. Elapsed times to add course sections by the registrars in the registration period for the First 2016/2017 semester

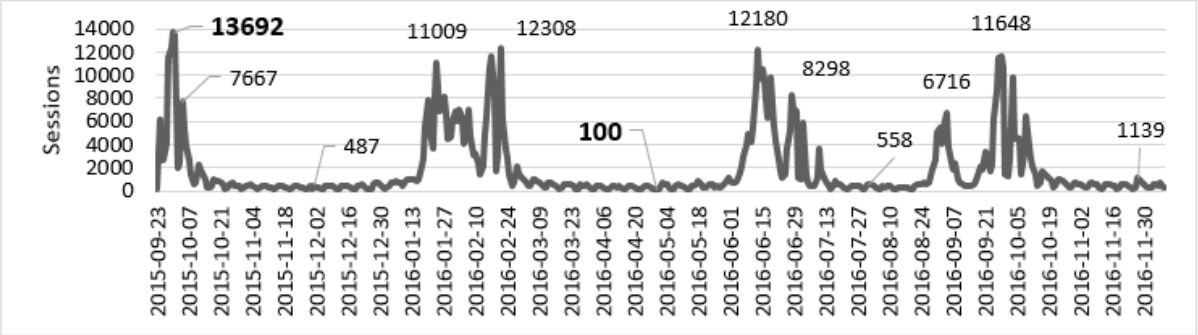

Figure 25. Total number of user sessions per day since the launch of MyGJU till the end of November 2016

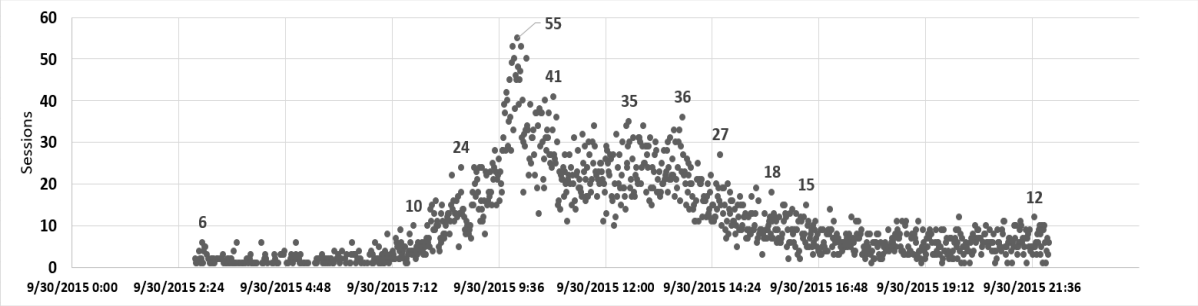

Figure 26. Total number of new user sessions per minute on September 30, 2015 (i.e., peak day in Figure 24)

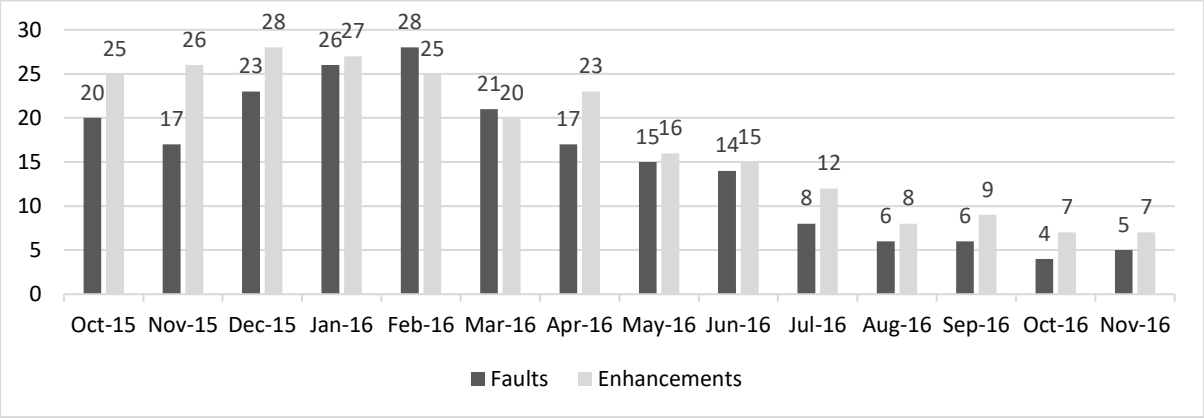

Figure 27. Number of faults and enhancements addressed by the development team monthly till November 2016

The total number of faults and enhancements that were addressed by the development team every month since the release of phase one of the system at the end of September 2015 till the end of November 2016 are shown in Figure 27. It is worth noting that phase two and three of the project were completed within that time period, as according to the development plan that was mentioned in subsection 2.1 they were supposed to be delivered six and twelve months after phase one (i.e., end of March and September 2016 respectively). Based on that, the higher number of updates during the first six months (i.e., October 2015 to March 2016) can be attributed to the incremental delivery of phase

two and the initial feedback from the users. While as the new development winded down and most of the encountered faults were fixed, the number of desired updates significantly decreased in the last five months indicating the high level of stability and reliability that the software has reached at this point.

8.2. MyGJU User Survey and Results

A MyGJU application user survey that targeted both students and registrars (i.e., application administrators) was conducted over a three weeks period. The questionnaires shown in Table 6 (for students) and Table 7 (for registrars) were about the MyGJU experience (question 1), performance (question 2), reliability (question 3), stability (question 4), availability (question 5), and ease of use (questions 6-8). The answers to each question were based on a five points Likert scale [53] as shown in Table 5. The students and registrars survey results found in Table 6 and Table 7 respectively show that 1409 students and 14 registrars participated in each survey. For each question, the number of responses to each answer point, total number of responses to the question, and overall question score are shown. Moreover, the total responses to all questions points and questions as well as the overall survey score are provided in the last row in each table.

Table 5. The question answers and the answer scores based on a five points Likert scale

Answer	**Strongly Agree**	**Agree**	**Neutral**	**Disagree**	**Strongly Disagree**
Answer Score	5	4	3	2	1

Table 6. The students survey results

#	Question	Answers					Total Answers	Score
		5	4	3	2	1		
1	MyGJU is much better than the previous system	638	484	190	19	35	1366	4.22
2	MyGJU is fast	344	678	228	70	46	1366	3.88
3	I faced very few errors while using MyGJU	338	669	227	56	70	1360	3.84
4	MyGJU is stable	357	716	216	41	36	1366	3.96
5	I can access MyGJU almost any time	492	638	144	51	41	1366	3.99
6	MyGJU is very easy to use	518	753	74	28	36	1409	4.20
7	I can accomplish tasks using MyGJU with very few steps (clicks)	303	744	207	76	35	1365	3.88
8	The user interface (tabs, menus, screens) is modern, consistent, and user friendly	312	768	175	65	45	1365	3.90
All Questions		3302	5450	1461	406	344	10963	3.9997

Table 7. The registrars survey results

#	Question	Answers					Total Answers	Score
		5	4	3	2	1		
1	MyGJU is much better than the previous system	7	6	1	0	0	14	4.42
2	MyGJU is fast	8	3	2	1	0	14	4.29
3	I faced very few errors while using MyGJU	6	4	3	1	0	14	4.07
4	MyGJU is stable	7	4	3	0	0	14	4.29
5	I can access MyGJU almost any time	9	3	2	0	0	14	4.50
6	MyGJU is very easy to use	8	3	2	1	0	14	4.29
7	Adding a course section is simple and quick	6	4	3	0	1	14	4.00
8	The registration flow is simple and informative	6	4	2	1	1	14	3.92
All Questions		57	31	18	4	2	112	4.22

Table 8. The percentages of all the students and registrars survey answers

	Strongly Agree	**Agree**	**Neutral**	**Disagree**	**Strongly Disagree**
Students	30.12%	49.71%	13.33%	3.70%	3.14%
Registrars	50.89%	27.68%	16.07%	3.57%	1.79%

Based on the results in Table 6 and Table 7, it is clear that the majority of students (1122 out of 1366 responses i.e., 81% of the responses) and registrars (13 out of 14 responses i.e., 93% of the responses) strongly agreed or agreed that MyGJU is much better than previous commercial system. While, the majority of the students responses strongly agreed or agreed that MyGJU is fast (1022 out of 1366 responses i.e., 75% of the responses), reliable (1007 out of 1360 responses i.e., 74% of the responses), stable (1073 out of 1366 responses i.e., 79% of the responses), available (1130 out of 1366 responses i.e., 83% of the responses), and easy to use (3398 out of 4139 responses i.e., 82% of the responses). Whereas, the majority of the registrars responses strongly agreed or agreed that MyGJU is fast (11 out of 14 responses i.e., 79% of the responses), reliable (10 out of 14 responses i.e., 72% of the responses), stable (11 out of 14 responses i.e., 79% of the responses), available (12 out of 14 responses i.e., 86% of the responses), and easy to use (31 out of 42 responses i.e., 74% of the responses). Finally, the overall percentages shown in Table 8 asserted that the majority of students (79.83%) and registrars (78.57%) either agreed or strongly agreed that MyGJU is collectively better than the previous system, fast, reliable, stable, available, and easy to use.

9. SUMMARY AND FUTURE WORK

The project management and software development processes that are based on the engineering basic profile in the ISO/IEC 29110 series and used to implement a complex student information administration system named MyGJU Admin were presented in this paper. A detailed discussion of the various phases in the adopted iterative and incremental software development process was also provided. Moreover, the reported system measurements and user survey results asserted that the introduced system is easy to use, feature rich, fast, reliable, stable, available, and scalable.

The system also provided information and features that helped administration maintain a comfortable learning environment, meet student demands, assess instructor performance, enhance teaching practices, and improve course content as discussed below:

- The number of enrolled and graduated students is used by administration to estimate the maximum number of prospective students such that a low student-to-faculty ratio is maintained in order to keep the classes small and allow the students to receive more one-on-one attention.
- The system allows giving some instructors an advisory role and then assigning students to each advisor. Accordingly, advisors can access the information (e.g., study plans, schedules, grades, holds, and financial statements) of their respective students in order to monitor their academic progress and provide them with needed academic guidance to guarantee their success.
- The historical registration data is utilized by the different faculties to determine the number and capacity of the course sections to be offered in each semester in order to meet the students' demands.
- The students can evaluate their instructors and courses every semester. Based on that, the university faculties can assess the performance of the faculty members, enhance the teaching practices, and improve the courses content.

As far as future work, we are planning to introduce several new features and tools related to the SIS at the GJU such as: a simple message service (SMS) framework to send important notifications (e.g., exam dates, schedule changes, and announcements) to the users, student attendance management, course materials (e.g., syllabus, lecture notes, home works, and exams) management, and MyGJU mobile application.

REFERENCES

1. MyGJU. German Jordanian University. Available from https://mygju.gju.edu.jo [last accessed November, 2016].
2. Bharamagoudar SR, Geeta RB, Totad SG. Web based student information management system. International Journal of Advanced Research in Computer and Communication Engineering 2013; 2(6).
3. Kumar BA. Thin client web-based campus information Systems for Fiji national university. arXiv preprint arXiv:1102.0583 2011.
4. Peng Y, Liu N, Li Y, Shao Z. Design and implementation of the online course registration system at Tsinghua University. IEEE International Conference on Systems and Informatics (ICSI) May 19, 2012; 1179-1182.

5. Yang D, Zhu S, Zhou S, Wang J, Cai A. The design and implement of Web MIS of students based on servlet + JDBC. First International Workshop on Education Technology and Computer Science March 7, 2009; 1022-1025.
6. Harris CS, Herring T. Web-enabled systems for student access. California State University 1999; 74(2):2-.
7. Luck M, Joy M. A secure on-line submission system. Software-Practice and Experience 1999; 29(8):721-40.
8. Qingshan Y, Xianli Z, Mingying Z. Design and implementation of college student management information system based on .Net three-layer structure. IEEE International Conference on Intelligent Computing and Integrated Systems October 22, 2010.
9. Gast H, Haug A, Loos R, Simonis V, Weiss R. CIS: A web-based course information system.
10. Athineos S, Karolidis D, Prentakis P, Samarakou M. A web based registration system for higher educational institutions in Greece: the case of energy technology department-TEI of Athens.
11. Steenkamp AL, Basal A. Building an integrated student information system in a K-12 school system. Proceedings of Information Systems Education Conference (ISECON) 2009; v26.
12. Hashim NM, Mohamed SN. Development of student information system. International Journal of Science and Research (IJSR) 2013; 2:256-60.
13. Liu Z, Wang H, Zan H. Design and implementation of student information management system. IEEE International Symposium on Intelligence, Information Processing, and Trusted Computing October 28, 2010; 607-610.
14. Liu Y, Gao F, Liu Y. Design and implementation of student registration system for universities. IEEE 2nd International Conference on Consumer Electronics, Communications, and Networks (CECNet) April 21, 2012; 1760-1763.
15. Alshareef A, Alkilany A, Alweshah M, Abu Bakar A. Toward a student information system for Sebha university, Libya. IEEE Fifth International Conference on Innovative Computing Technology (INTECH) May 20, 2015; 34-39.
16. Kannan PT, Bansal SK. Unimate: a student information system. IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI) August 22, 2013; 1251-1256.
17. Tang YF, Zhang YS. Design and implementation of college student information management system based on web Services. IEEE International Symposium on IT in Medicine and Education (ITIME'09) August 14, 2009; 1:1044-1048.
18. Mei-shan J, Chang-li Q, Jing L. The design of student information management system based on B/S architecture. IEEE 2nd International Conference on Consumer Electronics, Communications, and Networks (CECNet) 2012.
19. Norasiah MA, Norhayati A. Intelligent student information system. IEEE 4th National Conference on Telecommunication Technology (NCTT) January 14, 2003; 212-215.
20. Fahmy M. Automated student's courses registration using computer-telephony integration. International Arab Journal on Information Technology 2007; 4(4):353-8.
21. Adebisi AA, Oluwatobi AN, Adeola AO. Design and implementation of a mobile students' course registration platform. International Journal of Advanced Technology and Engineering Exploration (IJATEE); 2:25-30.
22. Ali O, Mohideen MK, Salleh N. MyKICT: Design and development of mobile application for course pre-registration. IEEE 2nd International Conference on Information Science and Security (ICISS) December 14, 2015; 1-4.
23. Roushan T, Chaki D, Hasdak O, Chowdhury MS, Rasel AA, Rahman MA, Arif H. University course advising: overcoming the challenges using decision support system. IEEE 16th International Conference on Computer and Information Technology (ICCIT) March 8, 2014; 13-18.
24. Laghari MS. Automated course advising system. International Journal of Machine Learning and Computing 2014; 4(1):47.
25. Kadam KS, Chandure OV. A review paper on student information supervision system.
26. Mihali R, Sobh T, Vamoser D. SKED: A course scheduling and advising software. Computer Applications in Engineering Education. 2004 Jan 1;12(1):1-9.
27. Dogan B, Dikbıyık E. OPCOMITS: Developing an adaptive and intelligent web based educational system based on concept map model. Computer Applications in Engineering Education. 2016 May 1:24(5):676-691.
28. Popović O, Marković DS, Popović R. mTester—Mobile learning system. Computer Applications in Engineering Education. 2016 Feb 1:24(3):412-420.
29. Zafar A, Albidewi I. Evaluation study of eLGuide: A framework for adaptive e-learning. Computer Applications in Engineering Education. 2015 Jul 1;23(4):542-55.
30. Rai A, Yadav A, Yadav D, Prasad R. A conceptual framework for E-learning. IEEE International Conference in MOOC Innovation and Technology in Education (MITE) December 20, 2013; 209-213.
31. Rashid TA, Ahmad HA. Lecturer performance system using neural network with Particle Swarm Optimization. Computer Applications in Engineering Education. 2016 Apr 1:24(4):629-638.
32. Rodriguez-Sanchez MC, Torrado-Carvajal Á, Vaquero J, Borromeo S, Hernandez-Tamames JA. A new open-source technological system for real-time assessment in the classroom. Computer Applications in Engineering Education. 2015 May 1;23(3):412-21.
33. Cuevas PL, Muñoz-Merino PJ, Fernandez-Panadero C, Kloos CD. CourseEditor: A course planning tool compatible with IMS-LD. Computer Applications in Engineering Education. 2013 Sep 1;21(3):421-31.
34. Muñoz-Merino PJ, Delgado Kloos C. A software player for providing hints in problem-based learning according to a new specification. Computer Applications in Engineering Education. 2009 Sep 1;17(3):272-84.
35. Li S, Zheng M, Fan H. Recording class attendance in a student registration system.
36. Ala'a M. Online registration system. International Journal of Computer Science and Security (IJCSS) 2010; 4(3):331.
37. Mabunda BT, Dehinbo JO. Enhancing online university class management system with instant Email feedback alert. In Proceedings of the World Congress on Engineering and Computer Science 2012; 1.

38. Adamov A, Mehdiyev S, Seyidzade E. Good practice of data modeling and database design for UMIS. Course registration system implementation. IEEE 8th International Conference on Application of Information and Communication Technologies (AICT) October 15, 2010; 1-4.
39. Mozgaleva PI, Zamyatina OM, Gulyaeva KV. Database design of information system for students' project activity management. IEEE International Conference on Interactive Collaborative Learning (ICL) December 3, 2014; 886-890.
40. Moertini VS, Yuliaty T, Rumono W. Academic IS for higher education institutions: the design of speedy courses registration transaction function. IEEE International Conference on Electrical Engineering and Informatics (ICEEI) July 17, 2011; 1-5.
41. Mukerjee S. Student information systems–implementation challenges and the road ahead. Journal of Higher Education Policy and Management 2012; 34(1):51-60.
42. Laporte CY, O'Connor RV, Paucar LH. The implementation of ISO/IEC 29110 software engineering standards and guides in very small entities. In International Conference on Evaluation of Novel Approaches to Software Engineering. Springer International Publishing. 2015 Apr 29:162-179.
43. Royce, WW. Managing the development of large software systems. Proceedings of IEEE WESCON. 1970 Aug 1:26(8): 382-338.
44. Juneau, J. Introducing Java EE 7: A look at what's new. Apress 2013.
45. Juneau, J. JavaServer Faces: Introduction by example. Apress 2014.
46. Caliskan, M, Varaksin, O. PrimeFaces cookbook. Packt Publishing 2015.
47. White, S. JDBC API tutorial and reference: universal data access for the Java 2 platform. Addison-Wesley Longman Publishing Co., Inc. 1999.
48. Hibernate ORM. Available from http://hibernate.org/orm/ [last accessed May 28, 2016].
49. SeleniumHQ. Available from http://docs.seleniumhq.org [last accessed May 28, 2016].
50. Iseminger, D. Active directory services for Microsoft windows 2000. Microsoft Press 1999.
51. Zeilenga, K. Lightweight directory access protocol (LDAP): Technical specification road map. 2006.
52. Lee, R, Seligman, S. The JNDI API tutorial and reference: building directory-enabled Java applications. Addison-Wesley Longman Publishing Co., Inc. 2000.
53. Likert, R. A technique for the measurement of attitudes. Archives of psychology. 1932.