CAPSTONE PROJECT FINAL REPORT

Submitted to
Computer Science Department
College of Computing Sciences and Engineering
Kuwait University

Advisor

Dr. Maha Alabuljalil

Group Members

Asmaa Sultan 2151117051

Amna Almutairi 2131110038

DATE OF SUBMISSION

September 9, 2020

Scheduley

For Fast Track Course Scheduling

Scheduly

Acknowledgements

The team would like to thank our advisor for her valuable vision that helped bring forth the motivation behind the project. And her efforts to support us throughout the software development. And to extend thanks to the committee for their valuable time to answer our questions and provide valuable feedback and guidance. We also thank our friends and families for their support and encouragement.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION AND BACKGROUND

The vital presence of universities in our modern life dictates that we must make their experience as smooth as can be for our students and future workforce. From that belief, we picked up the initiative to create a student course advisory website specific to the department of Computer Science at Kuwait University.

The project includes developing a Django web-app (Vincent, 2018) named Scheduly. Scheduly is meant to deal with Computer Science students, instructors, and staff. What it does is provide those parties with a smart way to plan next semester course schedule. That is, to offer students a suggested plan so they can make informed decisions. And to digitally communicate information between the faculty and the student such as list of courses students wish to take.

The team followed the software engineering agile development methodology Extreme Programming (Sommerville, 2015) that guided us throughout the many phases of this project. From planning, analysis, design, implementation, and testing, to deployment. The first phases of the project included deciding the functionalities the system should offer as well as the restrictions of our system as well as prototyping. Following that, the design and implementation of the system was made to bring the functionalities to real use. To finish with deployment and testing of the system of all angles such as usability tests, component tests, integrations tests and performance tests.

In this document, the development of this app is detailed throughout the many phases we had.

1.1 BACKGROUND

The Computer Science department students at Kuwait University used traditional methods to plan for courses. While most students needed to approach the student advisor in their early years, the student advisor reported this was not the case. The student advisor also reported this problem was very common and by the time the students approached them it was too late to have an efficient plan. This led to issues like student having delayed progress, bad GPAs, and in some severe cases end up being forced out from university due to regulation regarding both delayed progress and bad GPA. The team decided to formulate an automated solution for this problem as extending the usage of the system to fit faculty and course schedulers. The features of this system will help both the faculty and student make informed decisions by using this system without wasting time, paper, and energy.

2. USER AND SYSTEM REQUIREMENTS

This section will include details about the project's different features and how the team applies requirement engineering to find them.

2.1 REQUIREMENTS ELICITATION APPROACH

Requirement engineering mechanisms are meant to help understand and analyze the requirements to be provided by the system for the users of the system. It is also to negotiate and validate those requirements such that the team can specify them unambiguously. Specifically, the steps included are as follows:

- **Inception**: asking questions to establish a basic understanding of the problem to offer solutions.
    - Interviews with stakeholders such as students showed the team that most students chose courses based only on availability.
    - Interviews with stakeholders such as instructors showed the team that instructors choices were based on course preferences.
    - Interview with schedule supervisor showed the team that the supervisor had to consider the instructors time preferences before assigning them to teach a course.
    - Interview with the student advisor showed the team that students often do not check with the advisor therefore were lost in their academic progression.
- **Elicitation**: to bring forth requirements that are most important.
    - From previous steps the team concluded it was vital to inform the students about courses eligibility and help them plan the future course load well.
    - From previous steps the team wanted to automate this solution to reduce the time spent between communicating between each student, instructor, and the schedule supervisor.
- **Elaboration**: to create different user scenarios to better describe data, functional, and behavioral requirements.

    - Such as students of new admission, students that made progress with the Computer Science department.
- **Negotiation**: to decide which requirements are mandatory and which might be prone to later modifications.
    - Negotiation with stakeholders about what they liked the system to take in consideration and what is realistic. Such as the times of the courses were not an important factor such as the course itself for students. But were a factor in supervisor's course scheduling decisions.
- **Specification**: a prototype is made to design web interface requirements and how each request is carried out.
- **Validation**: to answer a set of questions such as: Are the requirement consistent with the overall objective of the system? Are the requirements bounded and unambiguous?
    - Such as the number of students demanding a certain course could be smaller than the actual number of students not using the system.
    - Deciding the number of factors that is needed for the optimal solution; should the GPA play a major role? The many scenarios proved it should not. As student interviews showed many students rarely repeat a course unless they had a failing grade in it.

Sources of requirements gathering included but not limited to the following:

- **Brain Storming**: team members and their supervisor gather frequently to propose different student scenarios and discuss possible solutions to be embedded gradually to the system prototype. This helped specify how each user interaction should be enabled as an input and what is the expected output.
- **Focus Group**: students of classes between 2012-2016 were asked to answer the questions while also providing any comment they believe is necessary. Many students had enough knowledge and experience to provide, due especially to their delayed academic progress which is what our system targets. Each student

offered a different opinion and during the elicitation phase, the team embedded what is solvable into the system.

- **Interviews**: other stakeholders such as instructors, the general advisor, and other decision-makers in the Computer Science department were interviewed to get different perspectives, understand what the optimal solution is, and what the system is required to offer.

2.2 SYSTEM'S FUNCTIONAL REQUIREMENTS

The requirements found with the above methods are listed by the specialty of user

2.2.1 STUDENTS

- Students must be added by the admin.
- Student could contact the admin or staff in person or through email.
- Students login with their university email and assigned password set by the admin.
- Students can change their login information after their first login.
- Students have a saved progress containing all compulsory Computer Science courses.
- Students can view and modify the progress.
- Student can view the plan that is based on the saved progress.
- Students have a saved wish list containing elective Computer Science courses.
- Students can modify their wish list.
- Students plan is formulated based on student progress and student's year of study.
- Student plan is divided into semesters.
- Student plan consists of Computer Science courses and their respective dependencies.
- Student plan includes the number of credits the student needs to take that is outside of Computer Science department.

2.2.2 INSTRUCTORS

- Instructors must be added by the admin.
- Instructors could contact the admin or staff in person or through email.
- Instructors login with their university email and assigned password.
- Instructors can change their login information after their first login.
- Instructors have a saved wish list containing all Computer Science courses.
- Instructors can modify their wish list.
- Instructors have saved time preferences for each wish list selection.
- Instructors can modify time preferences.
- Instructors can view statistics regarding students with courses on their wish list.

2.2.3 SUPERVISOR

- Supervisor must be added and assigned by the admin.
- Supervisor could contact the admin or staff in person or through email.
- Supervisor login with their university email and assigned password
- Supervisor can change their login information.
- Supervisor has a saved wish list containing all Computer Science courses.
- Supervisor can modify their wish list.
- Supervisor can view statistics regarding courses offered by instructors.
- Supervisor can view statistics regarding instructors' time preferences.

2.2.4 ADMIN

- Admin can add both instructors and students to the system.
- Admin can delete both instructors and students from the system.
- Admin can add Computer Science courses to the system.
- Admin can edit Computer Science courses in the system.
- Admin can notify students and instructors to modify their wish list.
- Admin sets allowed wish list times.

2.3 SYSTEM'S NON-FUNCTIONAL REQUIREMENTS

This section describes the non-functional the system has such as:

- Security: the systems handle of private information is secured such as hashing passwords that even the system admin cannot view.
- Reliability: the system can handle various requests and is accessible at any time, proved with tests on the system done by the team.
- Usability: the system's design such as inputs and interface were devolved with various type of users in mind from new to returning.
- Scalability: the system used amazon web services to hold more requests with load balancers and servers.

3. SYSTEM ARCHITECTURE

The system's external architecture is as seen in Figure 1

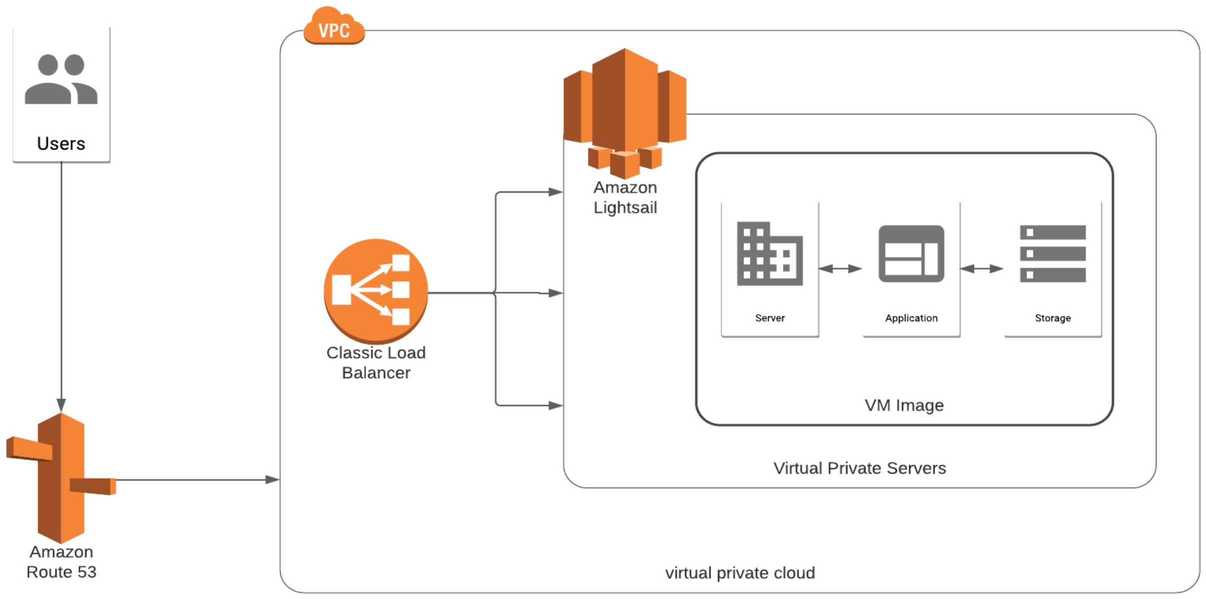


FIGURE 1: HIGH LEVEL SYSTEM ARCHITECTURE

The team used the cloud computing service Lightsail (Lightsail Documentation, 2020) which is provided by the Amazon Web Services (AWS Documentation, 2020). Which is a Virtual Private Cloud services platform. This services platform provided the team with a bundle of existing AWS products. Specifically:

- Amazon Route 53: the highly available and scalable cloud Domain Name System web services which effectively route end users to the infrastructures running in AWS. And allowed the team to register the domain name (http://www.thescheduly.com). As well as the DNS service to route our domain name to the fixed and public static IP of our infrastructures. And finally, health checking where the service sends automated requests over the internet to our infrastructure to verify it is reachable, available, and functional.
- Load Balancer: the service to distribute incoming web traffic among multiple instances helped the team increase availability and fault tolerance of the application such that if an instance is corrupted users are directed to other instances.
- Lightsail instances: or the virtual private server, is the cloud virtual machines with preconfigured Linux enviorment. Which had preinstalled Bitnami Django stack from the Bitnami software library. Which provided the framework Django, Python, and the Web Server Apache. All needed for running the web-app inside the instances.

The client-server software architecture or the architecture within the virtual image of Lightsail instance is as seen from Figure 2. The web-app or the system handles user requests through the web browser. The web-browser requests from an Apache HTTP server through the user's browser and directs them to a worker process running the Django application. Database used was SQLite, the files that built the app's internal architecture written in Python.

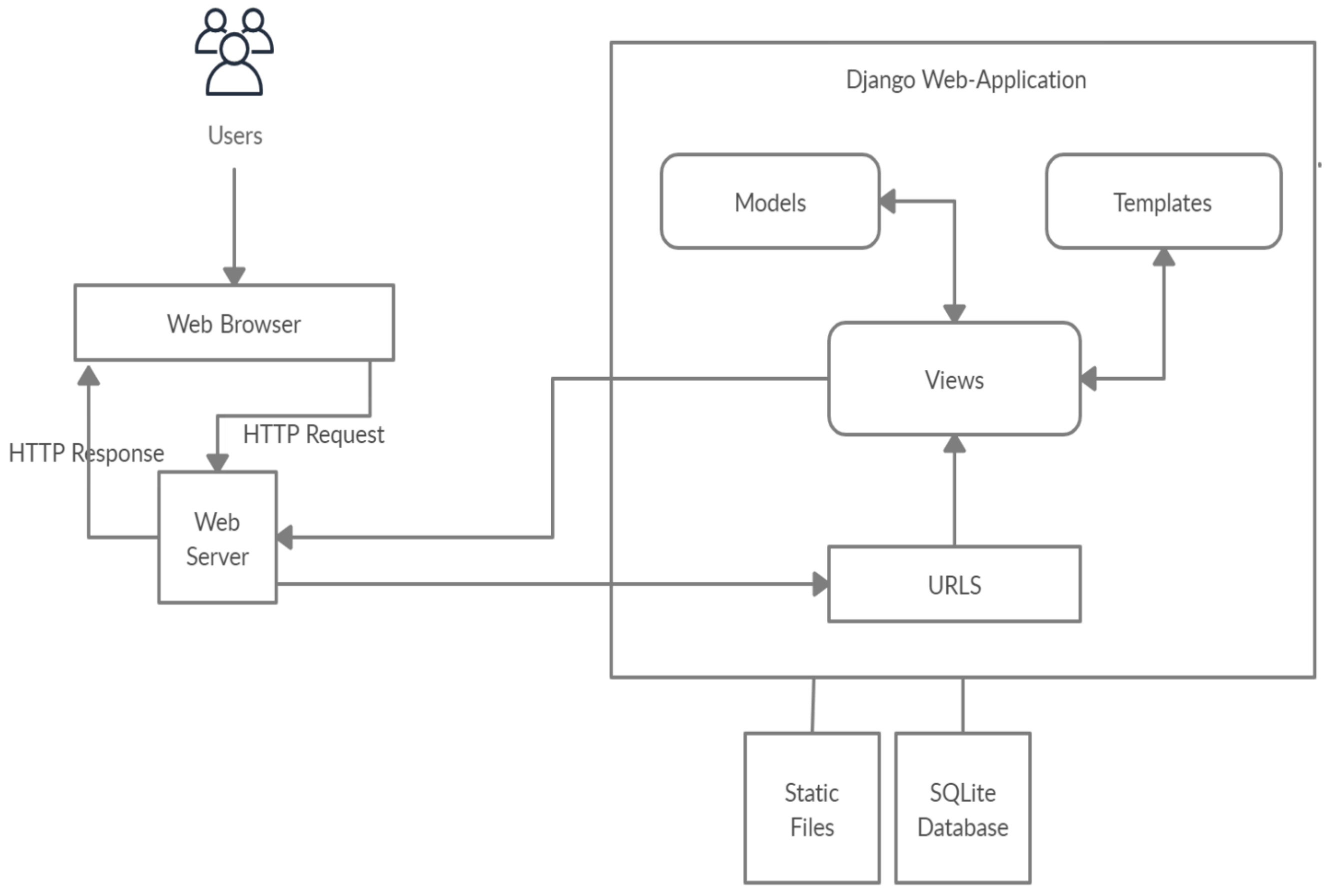


FIGURE 2: SYSTEM CLIENT-SERVER ARCHITECTURE

And finally, Figure 3 shows the component that classifies our group of classes for interchangeability and modularity of codes.

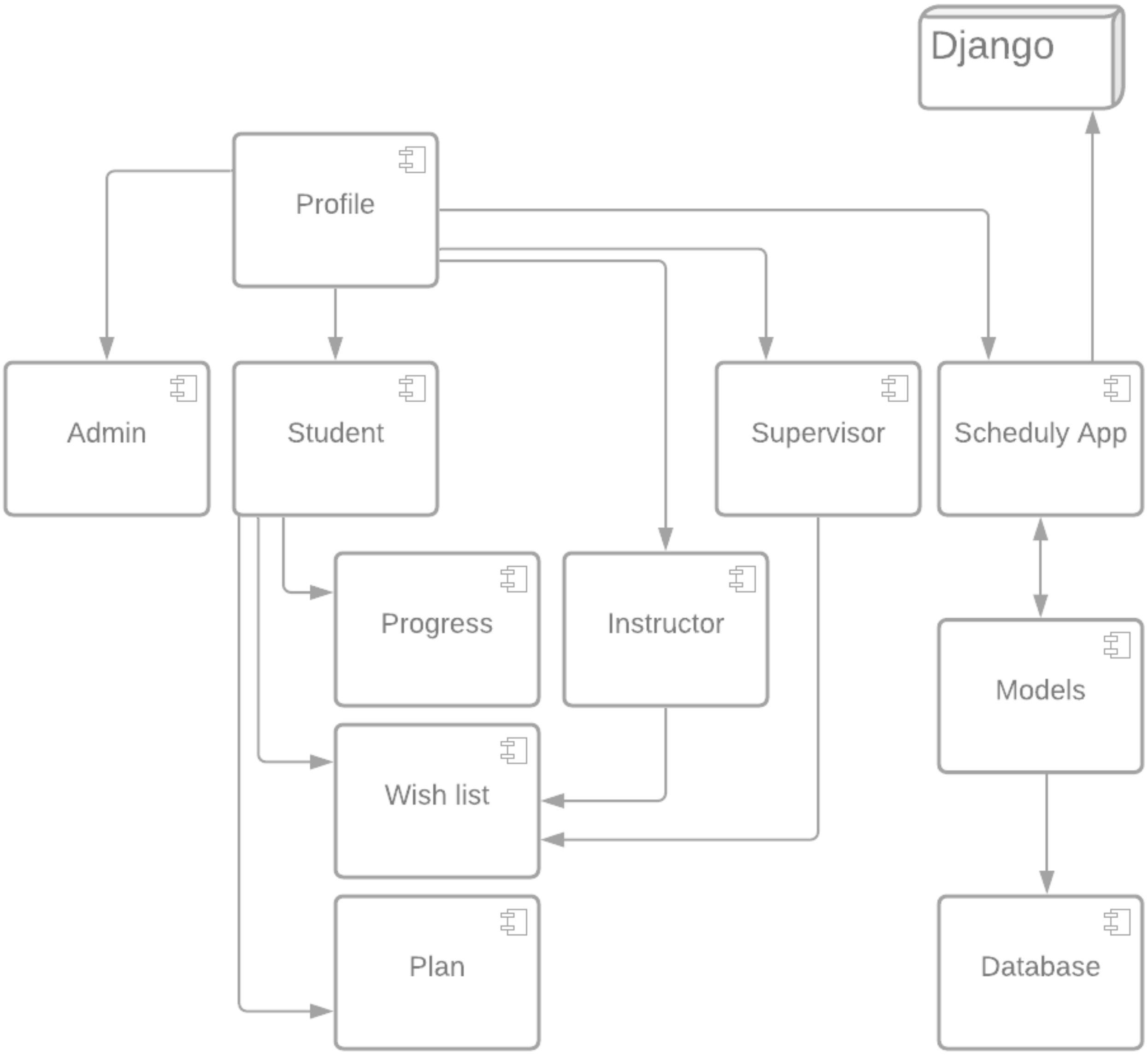


FIGURE 3: SYSTEM HIGH-LEVEL SOFTWARE ARCHITECURE

4. SYSTEM DESIGN ARTIFACTS

This section discusses the system design and to highlight the details and the component of the system. The internal architecture heavily relied on the Django architecture of Models-Templates-Views. With models, we manipulated most data relative to our users so we could process the many requests of our system such as students and their information, instructors and their information and courses and their information, the plan, the academic progress, the wish lists, and the relationship between them. Based on the entities defined with models the relationship between them and the web-app is reduced to HTTP requests and Django's framework.

The system used a web-app to interact between a student and their progress, plan, wish list. The instructor to interact with a wish list and statistics. The supervisor which is the special case of instructor who can view other instructor choices, to do so on the website along with the same instructor jobs. And finally, the admin to manage the users, courses, and sending emails to notify students when to add their choices to the system.

4.1 DESIGN DECISIONS

For a student, the login page is the first step of interacting with our system. The login needed from the student is their email and password. Following that, the student is presented the following actions: 1) Edit their profile information 2) view or update their academic progress 3) view their suggested plan 4) view or edit their elective courses wish list. The information the student can edit is restricted to the email and password since other information such as ID number or year cannot be decided by the student. The student's academic progress is left with the student as a form that contains all Computer Science department compulsory courses and math courses. That is because there are many courses outside this department that do not have dependencies that could affect the student's long-term progress such as the math and compulsory courses. Computer Science elective courses were kept off the student's progress as well for the same reason. For the student's plan, the student is presented with a table that suggests the courses from the next semester of the student, so their future academic progress is optimal. For the student's wish list, the student is presented with the page that displays all Computer Science department elective courses and then they add it to their own wish list.

For an instructor, the login process as well as editing profile is similar to that of the student. But the profile of the instructor included their respective un-preferred times of teaching which they can edit anytime. The instructor also has a wish-list page that shows both compulsory and elective Computer Science courses so the instructor can view or edit it. And a statistics page that shows the courses and statistics regarding number of students who have each course added to the students' wish-lists.

For the supervisor, the same as the instructor with an extra page of viewing other instructors statistics such as their un-preferred times of teaching, the courses and the number of instructors who have the course added to their wish-list.

For an admin, the login process is similar to the previous users. The admin has pages to manage the system such as adding the users or deleting them. The admin also can add courses or edit them. And a page that allows the admin to change the wish list allowed time which also automatically sends out emails for the students and instructors to inform them of this change. The admin is also tasked with assigning the supervisor role to an instructor or removing it.

For web-app interface is designed simply to provide formality to the academic purpose it served. Also, with ease of use in mind considered, as all the actions and pages are easily accessed from a navbar.

4.2 ENTITY-RELATIONSHIP MODEL AND DIAGRAM

For our data and database, the entity-relationship model was used to implement the information needed for and from the users as single model or entity and the relations between them as one to one relationship, one-to-many relationship, and many-to-many relationship, or another entity in some cases.

The following Entity Relationship Diagram shows how the data was manipulated by the web-app. The entities Student and Instructor were linked with the build-in Django user profile. While the instructor types of normal instructor, supervisor, or admin, were merged in the model Instructor. The course entity has three relationships vital to the algorithms such as 1) Dependency, which is a many-to-many relationship to implement prerequisites of the course as another entity. 2) Progres-Course, which is the many-to-many relationship to prove the student has passed a course or not, while having the attribute credit that helped our algorithm suggest the credits the student needs to take outside the Computer Science compulsory courses. 3) WishList-Course, which is the many-to-many relationship to link each course and the wish-list it was added to by either instructor or student, The wish-list entity has the role of linking the owner of that wish-list and the owner either student or instructor. While the subject entity is used for algorithms as another table to get results from without overlapping with the courses entity.

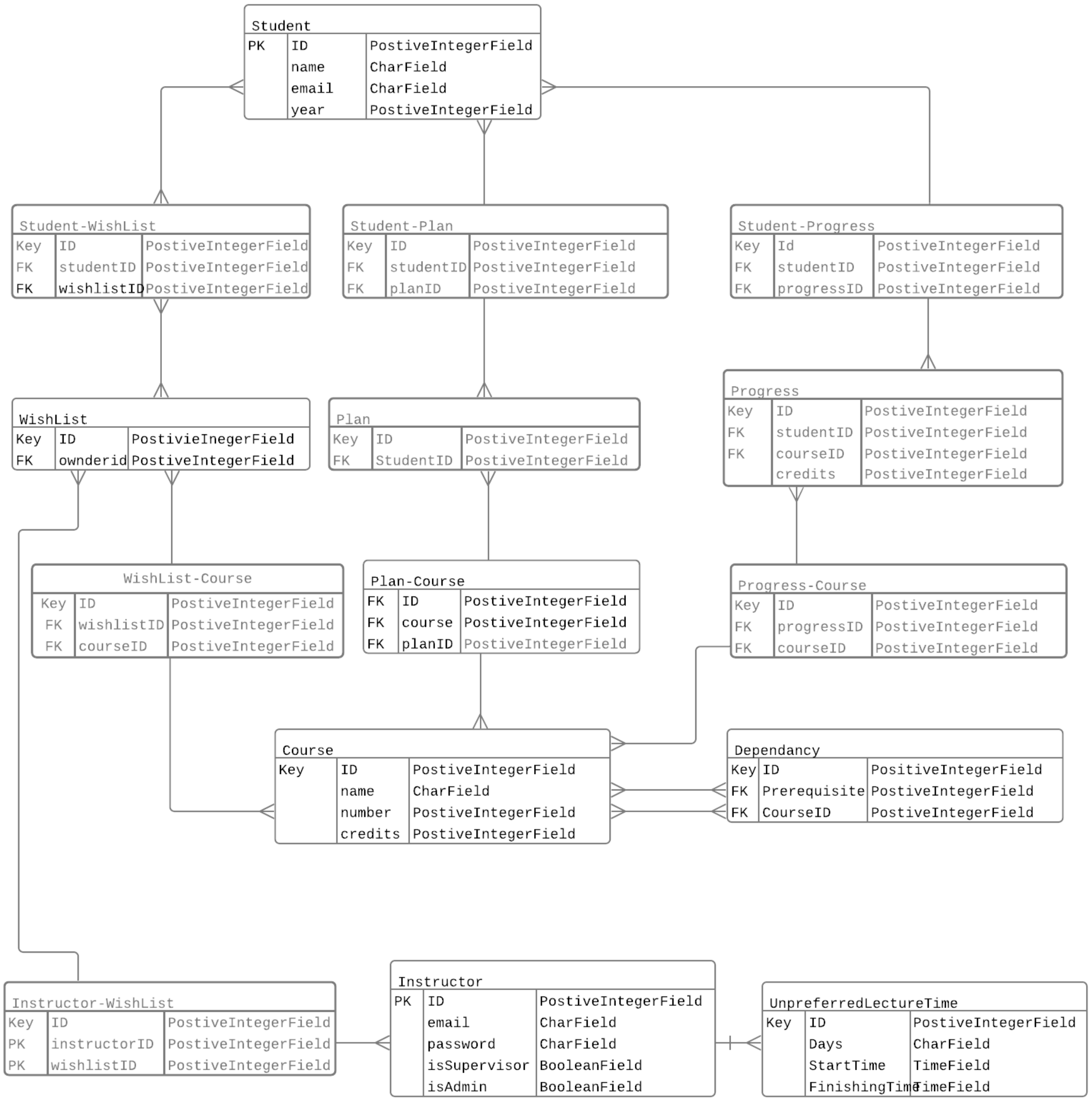

Student
PK ID PostiveIntegerField
name CharField
email CharField
year PostiveIntegerField
Student-WishList
Key ID PostiveIntegerField
FK studentID PostiveIntegerField
FK wishlistID PostiveIntegerField
Student-Plan
Key ID PostiveIntegerField
FK studentID PostiveIntegerField
FK planID PostiveIntegerField
Student-Progress
Key Id PostiveIntegerField
FK studentID PostiveIntegerField
FK progressID PostiveIntegerField
WishList
Key ID PostivieInegerField
FK ownderid PostiveIntegerField
Plan
Key ID PostiveIntegerField
FK StudentID PostiveIntegerField
Progress
Key ID PostiveIntegerField
FK studentID PostiveIntegerField
FK courseID PostiveIntegerField
credits PostiveIntegerField
WishList-Course
Key ID PostiveIntegerField
FK wishlistID PostiveIntegerField
FK courseID PostiveIntegerField
Plan-Course
FK ID PostiveIntegerField
FK course PostiveIntegerField
FK planID PostiveIntegerField
Progress-Course
Key ID PostiveIntegerField
FK progressID PostiveIntegerField
FK courseID PostiveIntegerField
Course
Key ID PostiveIntegerField
name CharField
number PostiveIntegerField
credits PostiveIntegerField
Dependancy
Key ID PositiveIntegerField
FK Prerequisite PostiveIntegerField
FK CourseID PostiveIntegerField
Instructor-WishList
Key ID PostiveIntegerField
PK instructorID PostiveIntegerField
PK wishlistID PostiveIntegerField
Instructor
PK ID PostiveIntegerField
email CharField
password CharField
isSupervisor BooleanField
isAdmin BooleanField
UnpreferredLectureTime
Key ID PostiveIntegerField
Days CharField
StartTime TimeField
FinishingTime TimeField

FIGURE 4: ER DIAGRAM

4.3 CLASS DIAGRAM

The following conceptual class diagram shows the blueprint for the objects that the system uses.

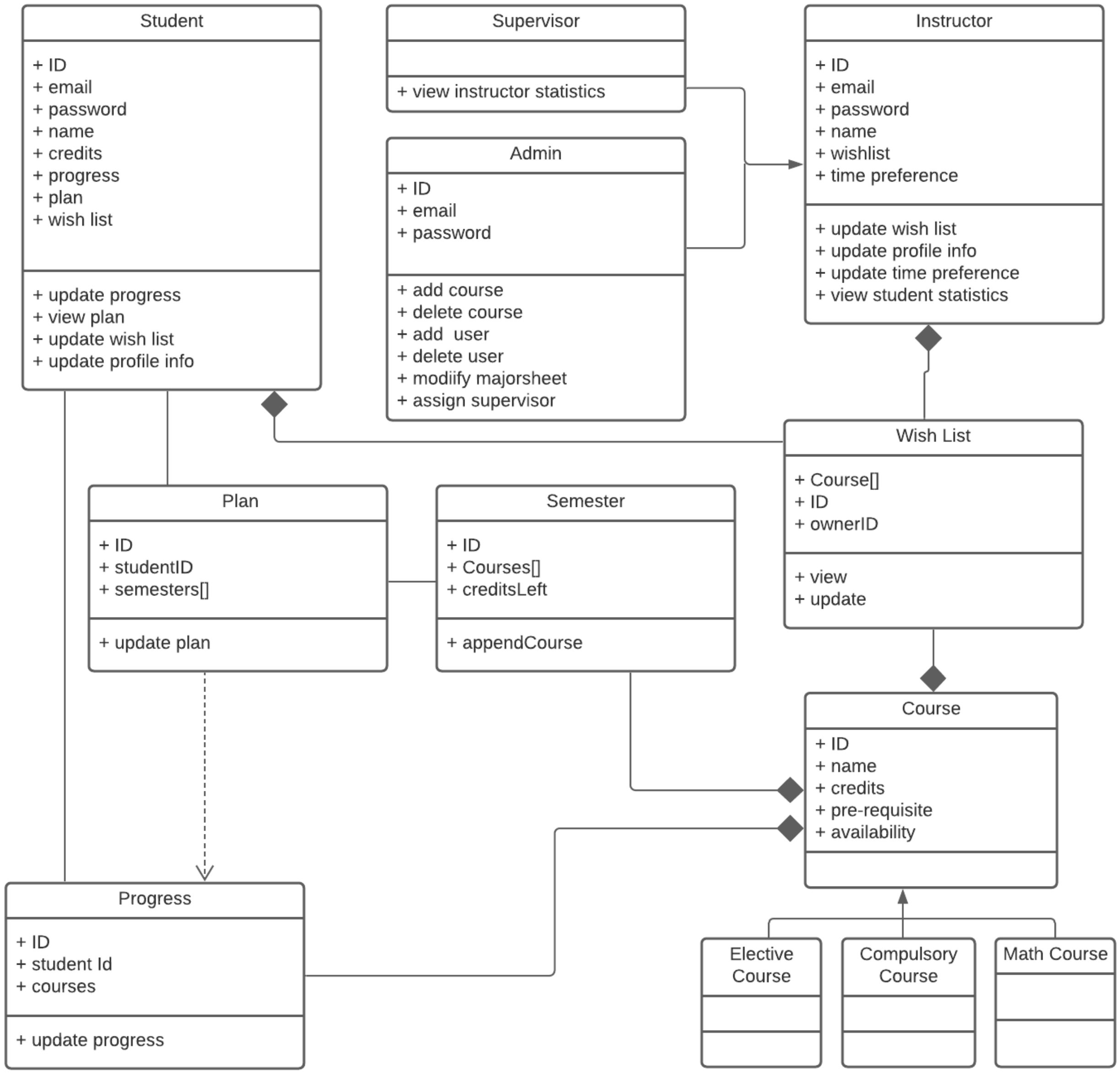


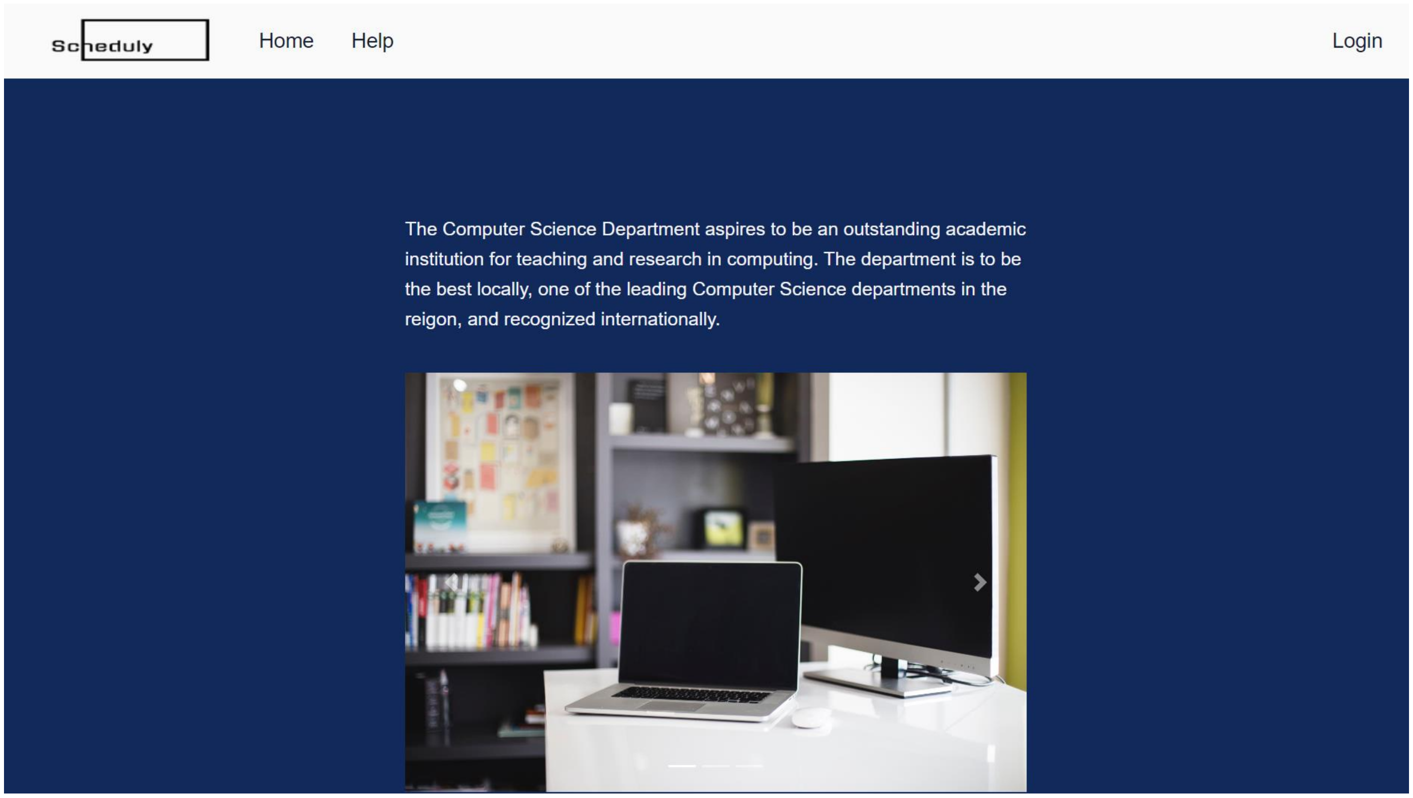
FIGURE 5: CLASS DIAGRAM

For the different type of users, we have four types of classes, the student, the instructor, and the two extensions of instructor: the supervisor and the admin. All mentioned users have a wish list that is composed of courses. While student has classes plan and progress with the plan having dependency with the progress. The course has three different type of courses that inherit from it: Compulsory Course for Compulsory courses in the Computer Science department. Elective Course for electives from the same department, and finally compulsory math courses from the math department.

4.4 USER INTERFACES

In this section we show pages the team developed for the web-app interfaces. Design decisions of the interface varied such as simplicity fit for the academic usage and to include ease of access. The interface design followed the eight golden rules of interface design (Shneiderman, 2010) including:

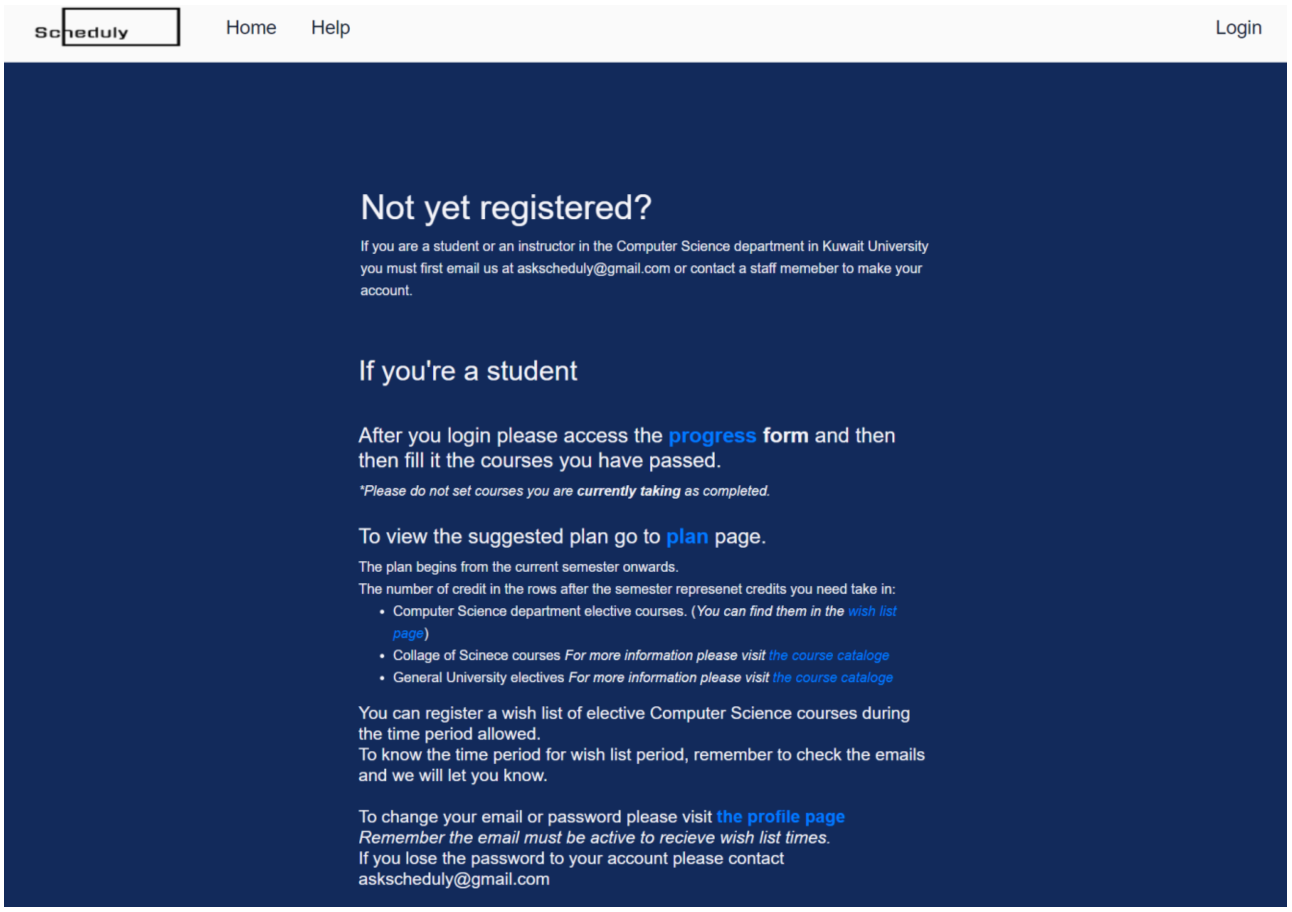
- Consistency: the design is consistent through every page in the web-app
- Seek universal usability: the diverse users from new to returning can easily find their way around the web-app.
- Offer informative feedback: all frequent and minor actions in the system have an appropriate response.
- Design dialogs that yield closure: for sequences of actions a notification of completion is offered.
- Permit easy reversal of actions: errors can be undone to allow all users to have efficient use of the system and encourage exploration of unfamiliar options.
- Keep users in control: the interface responds to user actions and stay consistent with familiar behavior.
- Reducing short-memory load: each functionality is given its own page and can be reached from the navbar so it can be easily accessed



The Computer Science Department aspires to be an outstanding academic institution for teaching and research in computing. The department is to be the best locally, one of the leading Computer Science departments in the reigon, and recognized internationally.

FIGURE 6: HOMEPAGE FOR VISITORS

Figure 5 shows the homepage from a new user or a logged-out user's point of view.



Not yet registered?

If you are a student or an instructor in the Computer Science department in Kuwait University you must first email us at askscheduly@gmail.com or contact a staff memeber to make your account.

If you're a student

After you login please access the progress **form** and then then fill it the courses you have passed.

Please do not set courses you are **currently taking as completed.*

To view the suggested plan go to plan page.

The plan begins from the current semester onwards.
The number of credit in the rows after the semester represenet credits you need take in:

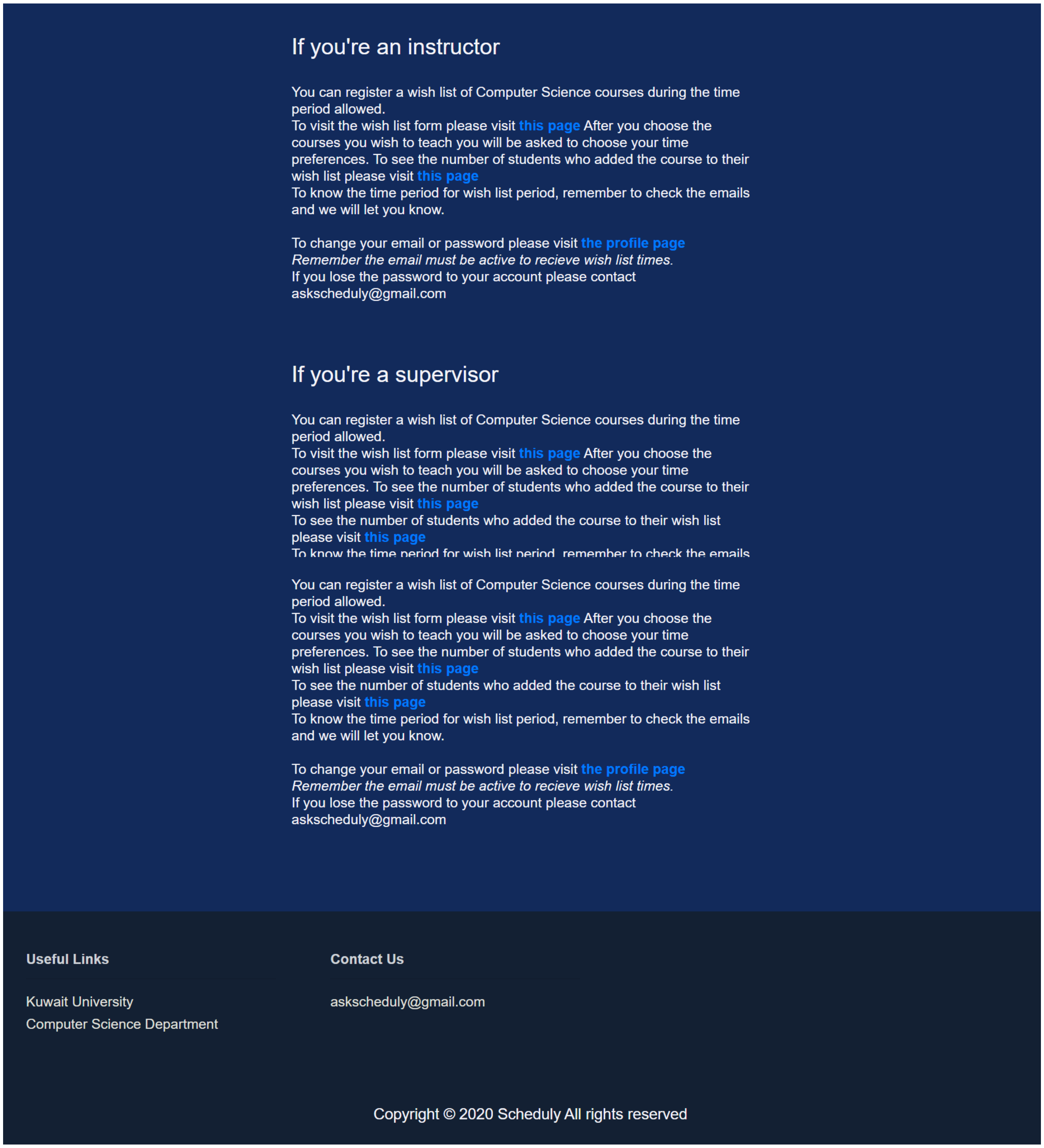
- Computer Science department elective courses. (*You can find them in the wish list page*)
- Collage of Scinece courses *For more information please visit the course cataloge*
- General University electives *For more information please visit the course cataloge*

You can register a wish list of elective Computer Science courses during the time period allowed.
To know the time period for wish list period, remember to check the emails and we will let you know.

To change your email or password please visit **the profile page**
Remember the email must be active to recieve wish list times.
If you lose the password to your account please contact askscheduly@gmail.com

FIGURE 7: HELP PAGE

Visitors and students can access a help page from the homepage or any other menu bar that directs them to system features and offer guidance regarding how to use and what to expect when accessing the features.


If you're an instructor

You can register a wish list of Computer Science courses during the time period allowed.
To visit the wish list form please visit this page After you choose the courses you wish to teach you will be asked to choose your time preferences. To see the number of students who added the course to their wish list please visit this page
To know the time period for wish list period, remember to check the emails and we will let you know.

To change your email or password please visit the profile page
Remember the email must be active to recieve wish list times.
If you lose the password to your account please contact askscheduly@gmail.com

If you're a supervisor

You can register a wish list of Computer Science courses during the time period allowed.
To visit the wish list form please visit this page After you choose the courses you wish to teach you will be asked to choose your time preferences. To see the number of students who added the course to their wish list please visit this page
To see the number of students who added the course to their wish list please visit this page
To know the time period for wish list period, remember to check the emails

You can register a wish list of Computer Science courses during the time period allowed.
To visit the wish list form please visit this page After you choose the courses you wish to teach you will be asked to choose your time preferences. To see the number of students who added the course to their wish list please visit this page
To see the number of students who added the course to their wish list please visit this page
To know the time period for wish list period, remember to check the emails and we will let you know.

To change your email or password please visit the profile page
Remember the email must be active to recieve wish list times.
If you lose the password to your account please contact askscheduly@gmail.com

Useful Links

Kuwait University
Computer Science Department

Contact Us

askscheduly@gmail.com

Copyright © 2020 Scheduly All rights reserved

FIGURE 8: HELP PAGE

Instructors and supervisor can access a help page from the homepage or any other menu bar that directs them to system features and offer guidance regarding how to use and what to expect when accessing the features.

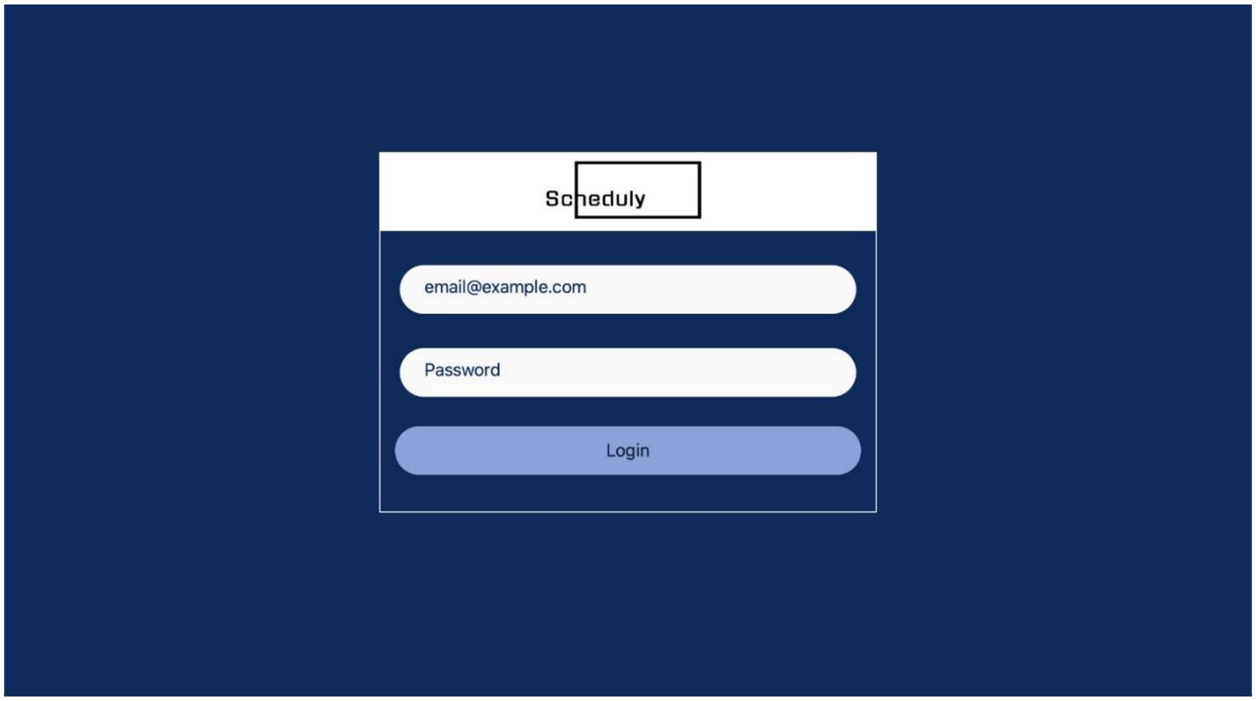


FIGURE 9: LOGIN PAGE

The login information needed to access the system is the email (university or personal email) and password that users get from admin or staff.

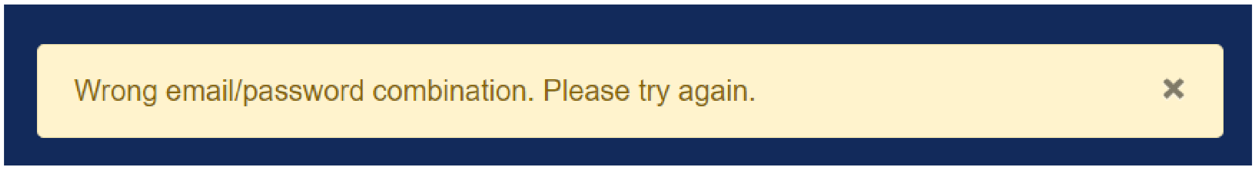

Wrong email/password combination. Please try again.

FIGURE 10: INFORMATIVE FEEDBACK #1

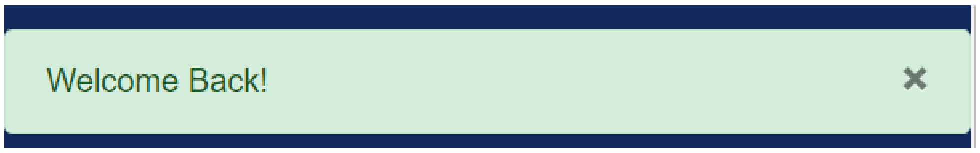

Welcome Back!

FIGURE 11: INFORMATIVE FEEDBACK #2

Login errors are handled and communicated to the users of the system as well as system status and action closures.

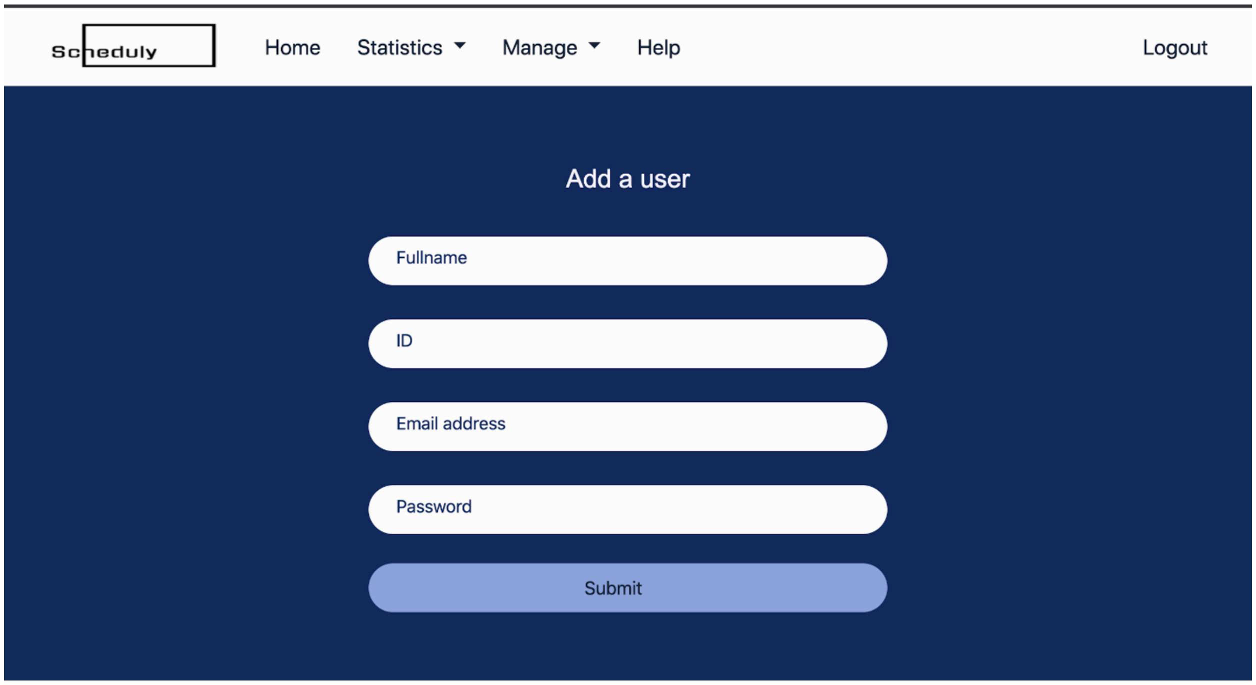
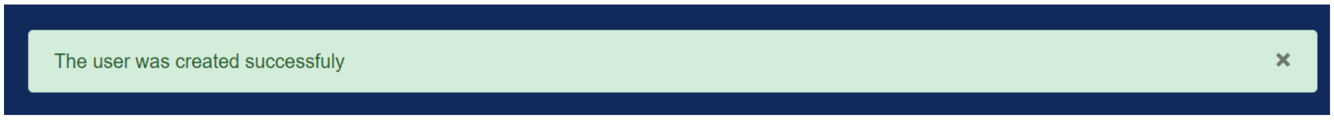


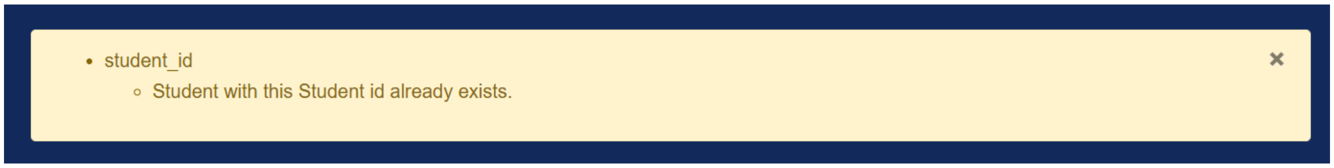
FIGURE 12: ADMIN PAGE TO ADD A USER

For the admin to add a user the following information is needed: their name, their unique university ID, and their unique email address. Following that, the admin can assign a password that the users can change in their respective interfaces.



The user was created successfuly

FIGURE 13: INFORMATIVE FEEDBACK #4



- student_id
  - Student with this Student id already exists.

FIGURE 14: INFORMATIVE FEEDBACK #4

Usage errors are handled and communicated to the admin of the system as well as system status and action closures.

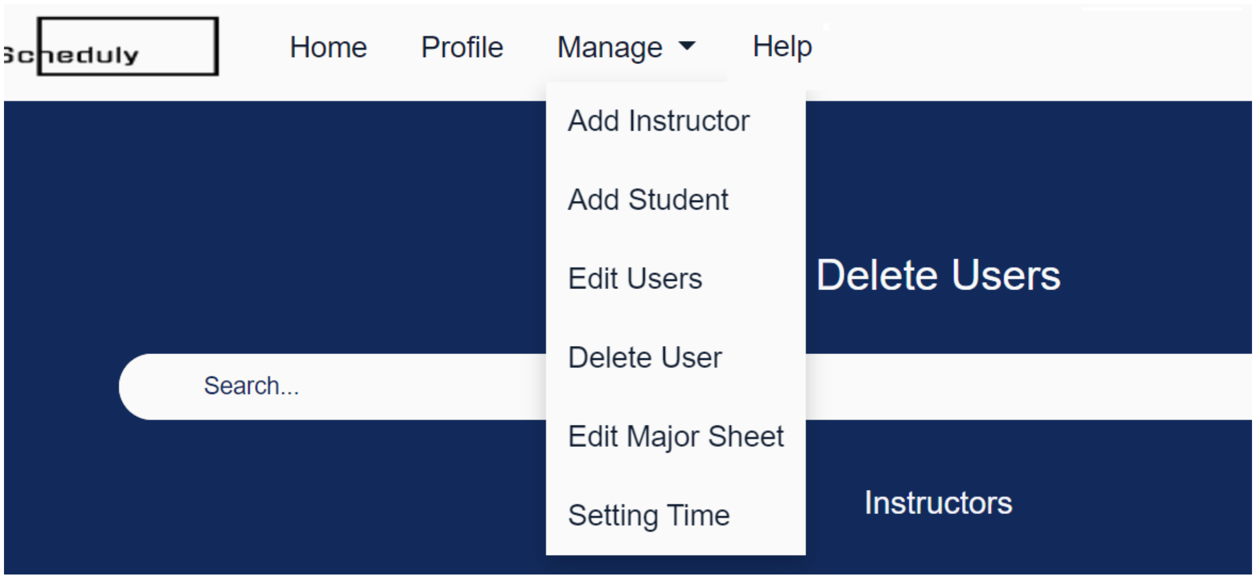
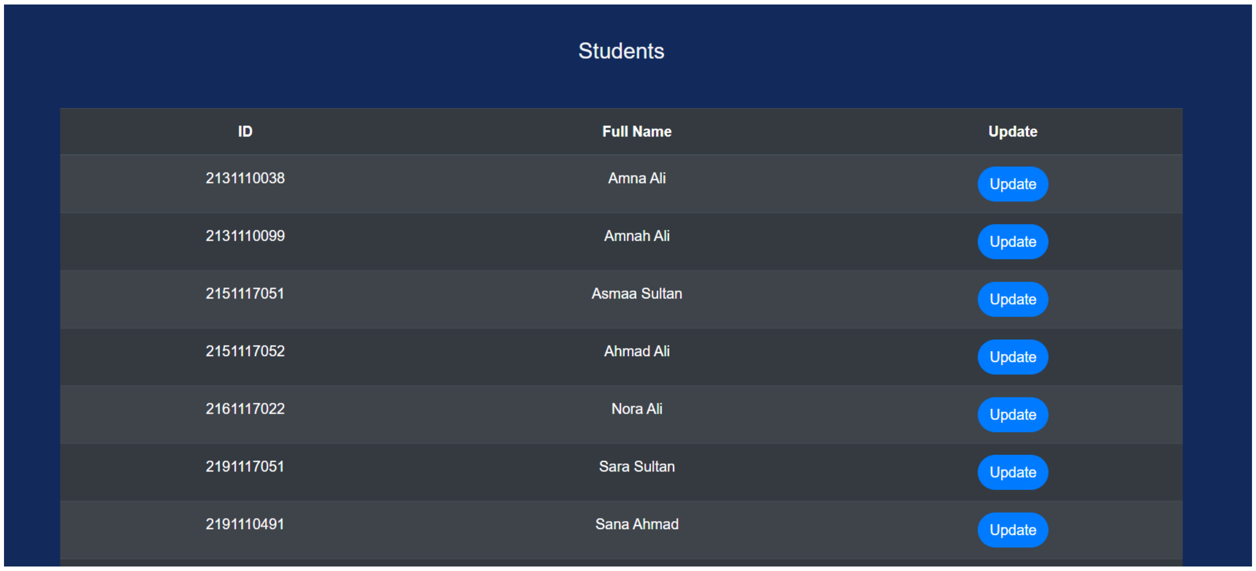


FIGURE 15: ADMIN MANEGEMENT MENU

The menu with the pages that the admin uses. The editing users, deleting users, and editing major sheet pages all include a search options due to the large number of options there is.

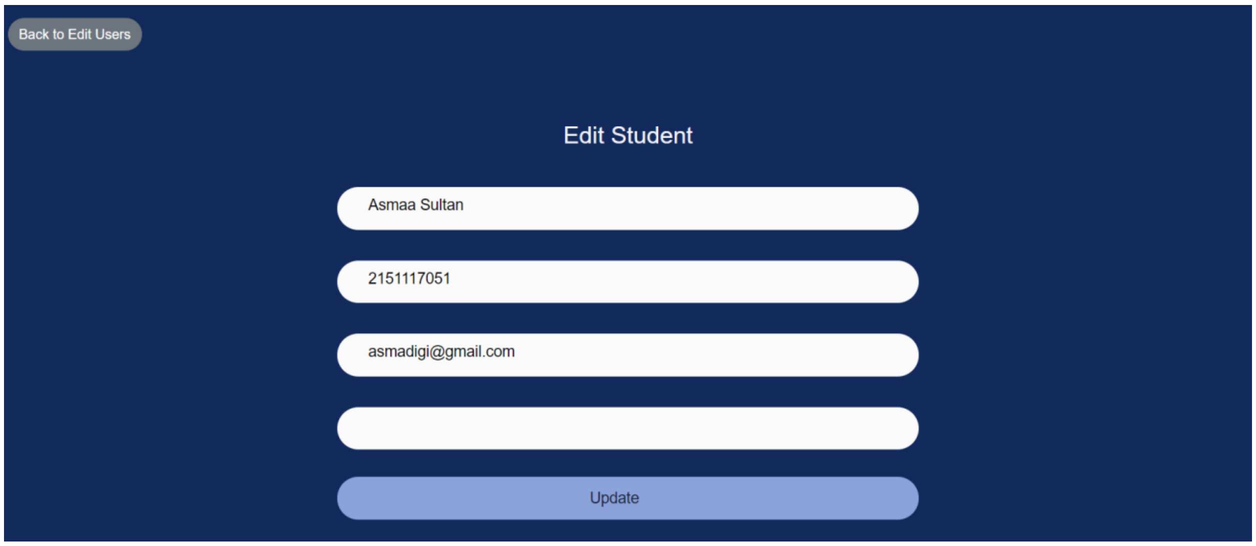


Students

ID	Full Name	Update
2131110038	Amna Ali	Update
2131110099	Amnah Ali	Update
2151117051	Asmaa Sultan	Update
2151117052	Ahmad Ali	Update
2161117022	Nora Ali	Update
2191117051	Sara Sultan	Update
2191110491	Sana Ahmad	Update

FIGURE 16: ADMIN PAGE TO EDIT REGISTERED USERS

The admin can view the list of registered users (both instructors and students) where the admin can find the selected user profile to be edited.

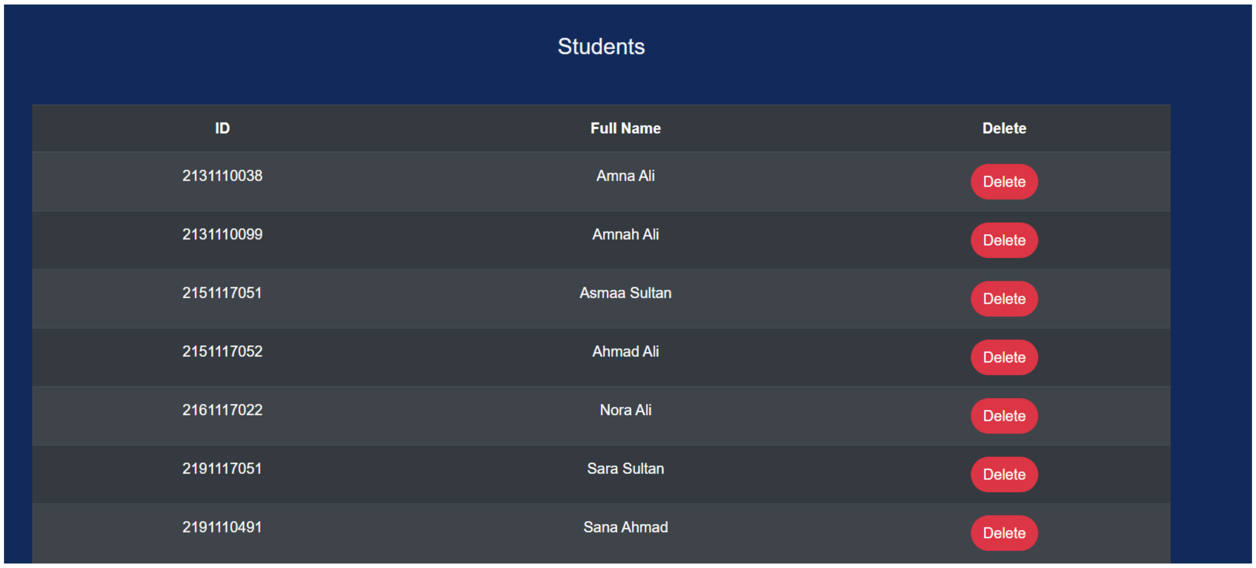


Edit Student

Asmaa Sultan
2151117051
asmadigi@gmail.com

FIGURE 17: PAGE TO EDIT A STUDENT

The page where the admin can edit a student profile has the option reverse the selected action.

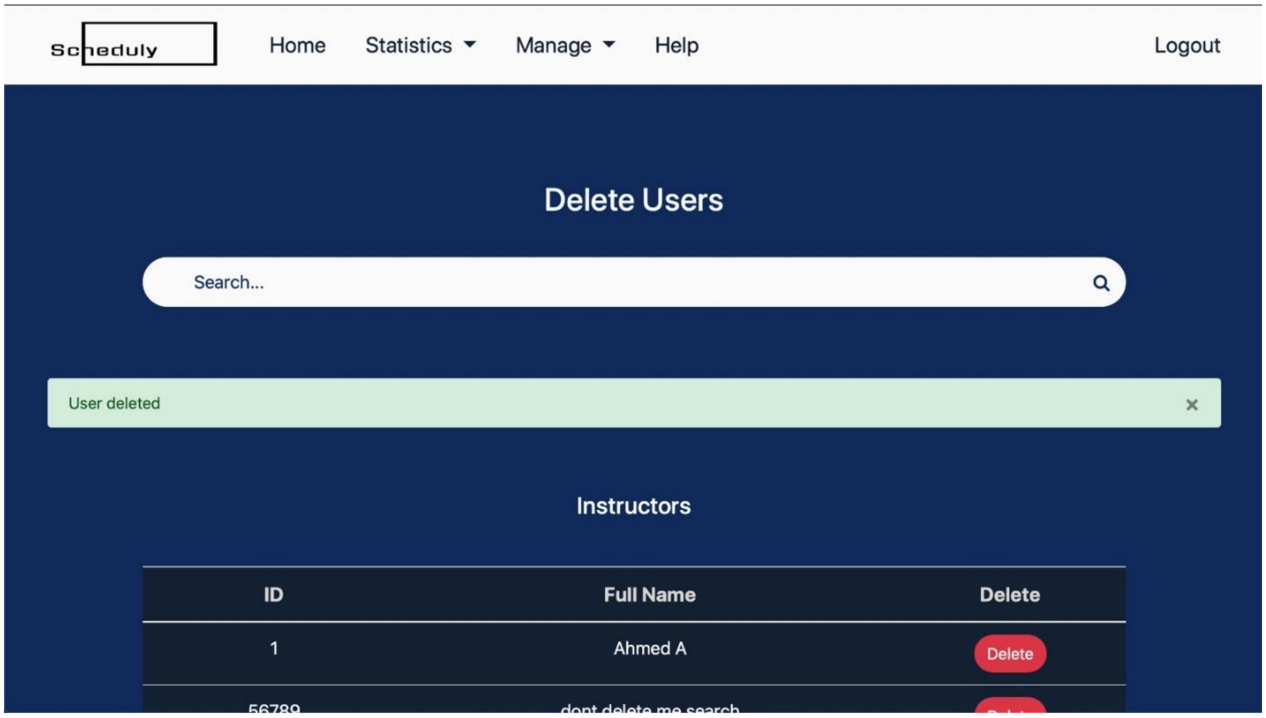


Students

ID	Full Name	Delete
2131110038	Amna Ali	Delete
2131110099	Amnah Ali	Delete
2151117051	Asmaa Sultan	Delete
2151117052	Ahmad Ali	Delete
2161117022	Nora Ali	Delete
2191117051	Sara Sultan	Delete
2191110491	Sana Ahmad	Delete

FIGURE 18: PAGE TO VIEW LIST OF USERS TO DELETE

The admin can view the list of registered users (both instructors and students) where the admin can find the selected user profile to be deleted.



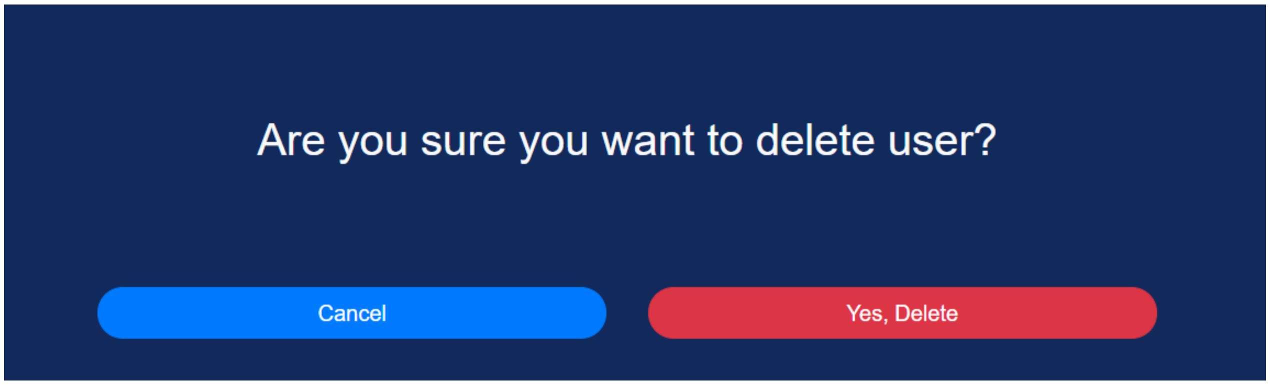
Delete Users

User deleted

Instructors

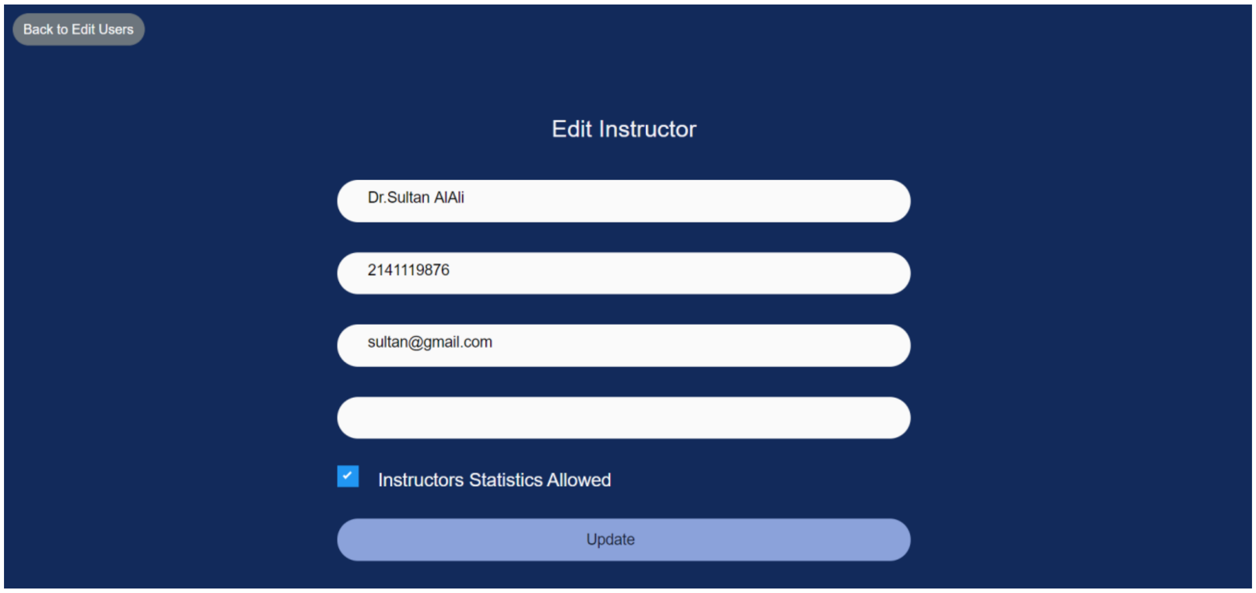
ID	Full Name	Delete
1	Ahmed A	Delete
56789	dont delete me search	

FIGURE 19: INFORMATIVE FEEDBACK #5



Are you sure you want to delete user?

FIGURE 20: REVERSAL OF ACTIONS #2



Dr.Sultan AlAli

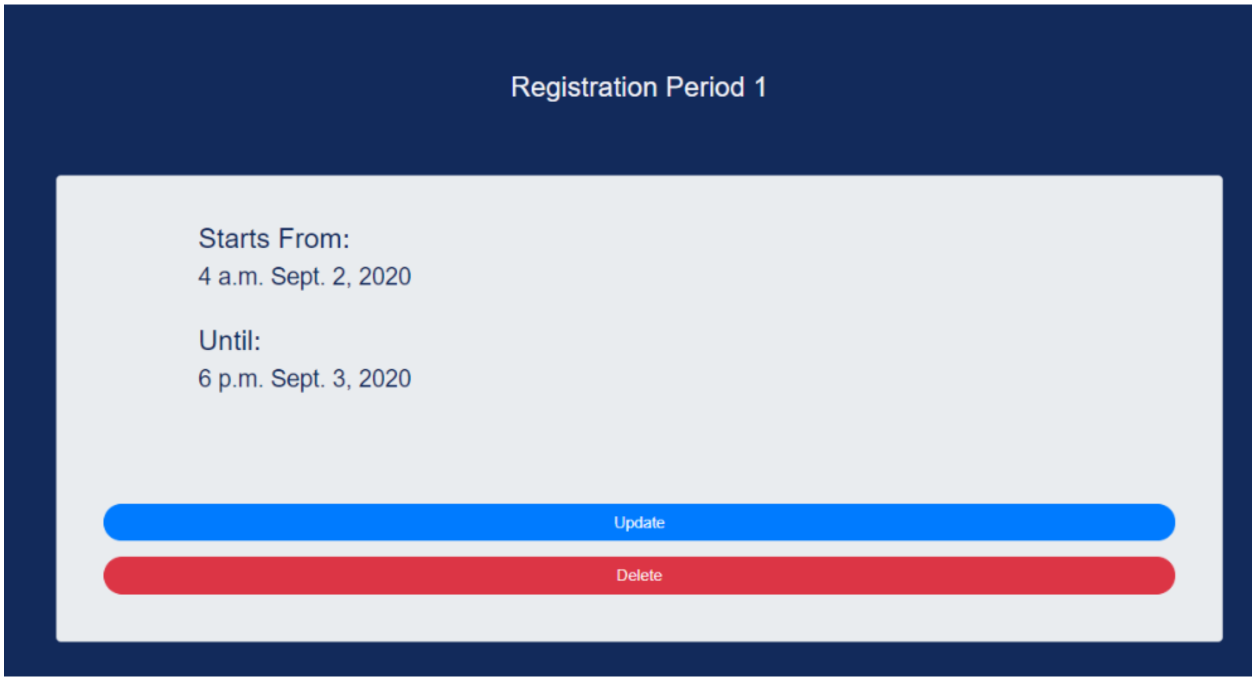
2141119876

sultan@gmail.com

✓ Instructors Statistics Allowed

FIGURE 21: ADDING INSTRUCTOR ADMIN PAGE

Editing or adding an instructor includes the option for the admin to assign the supervisor.

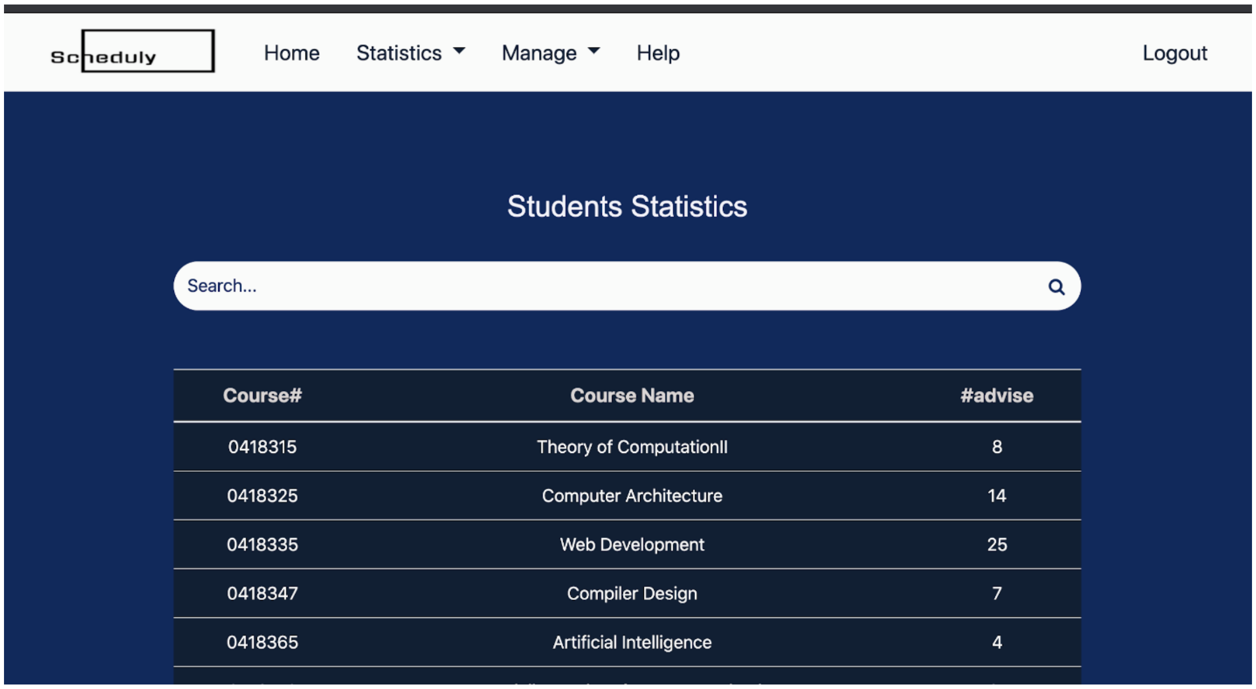


Starts From:
4 a.m. Sept. 2, 2020

Until:
6 p.m. Sept. 3, 2020

FIGURE 22: SETTING WISH-LIST TIME PAGE FOR ADMIN

After updating the wish-list registration time, the system automatically sends emails to notify both students and instructors.

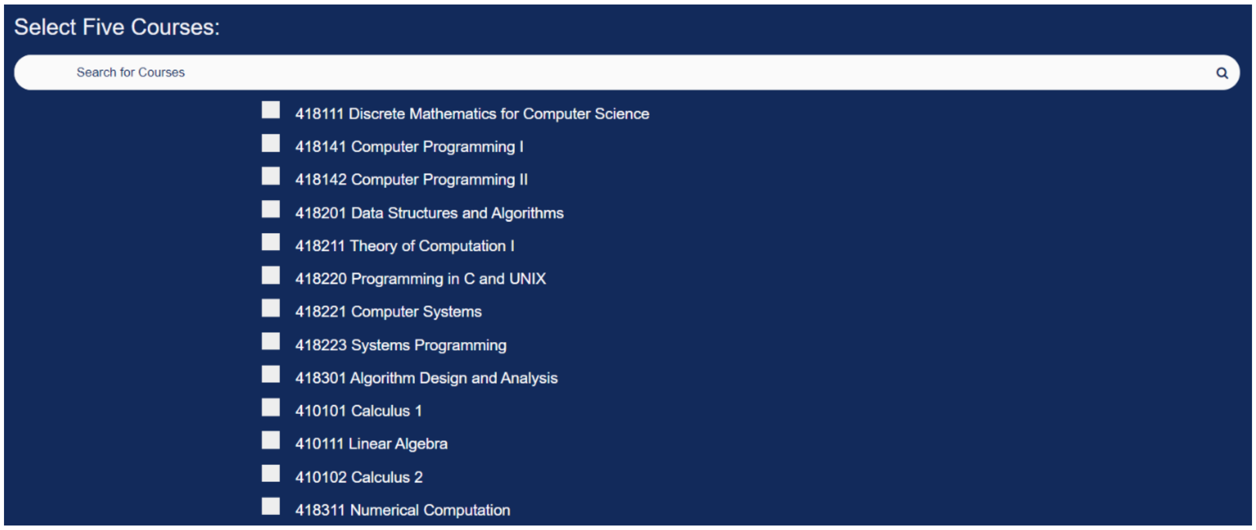


Students Statistics

Course#	Course Name	#advise
0418315	Theory of ComputationII	8
0418325	Computer Architecture	14
0418335	Web Development	25
0418347	Compiler Design	7
0418365	Artificial Intelligence	4

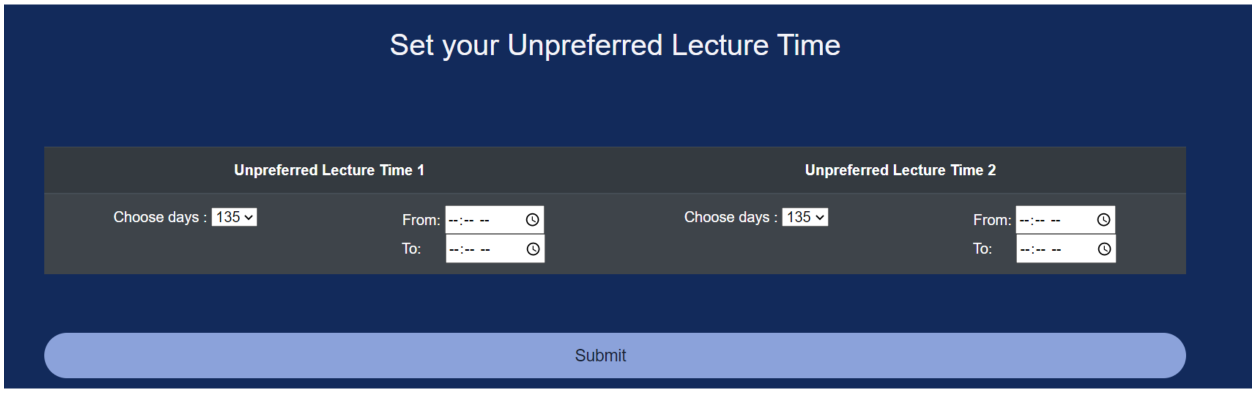
FIGURE 23: INSTRUCTOR PAGE TO VIEW STATISTICS FOR ELECTIVE CS COURSES

This informational page is to communicate to the instructors the courses that are in demand by students.


Select Five Courses:

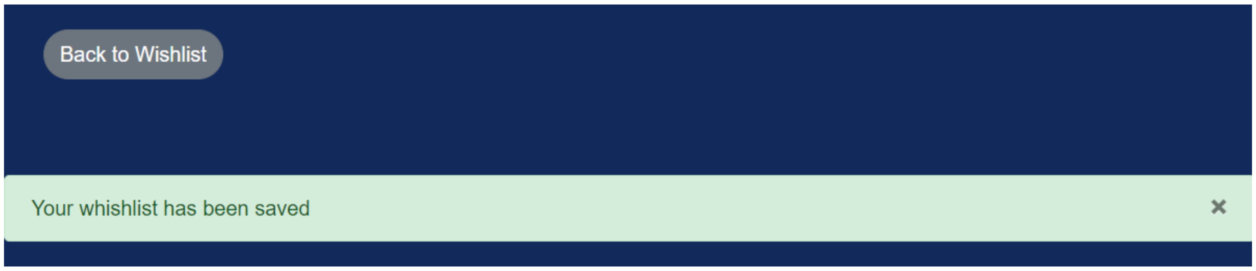
- ☐ 418111 Discrete Mathematics for Computer Science
- ☐ 418141 Computer Programming I
- ☐ 418142 Computer Programming II
- ☐ 418201 Data Structures and Algorithms
- ☐ 418211 Theory of Computation I
- ☐ 418220 Programming in C and UNIX
- ☐ 418221 Computer Systems
- ☐ 418223 Systems Programming
- ☐ 418301 Algorithm Design and Analysis
- ☐ 410101 Calculus 1
- ☐ 410111 Linear Algebra
- ☐ 410102 Calculus 2
- ☐ 418311 Numerical Computation

FIGURE 24: INSTRUCTOR PAGE TO VIEW OR UPDATE WISH LIST


Set your Unpreferred Lecture Time

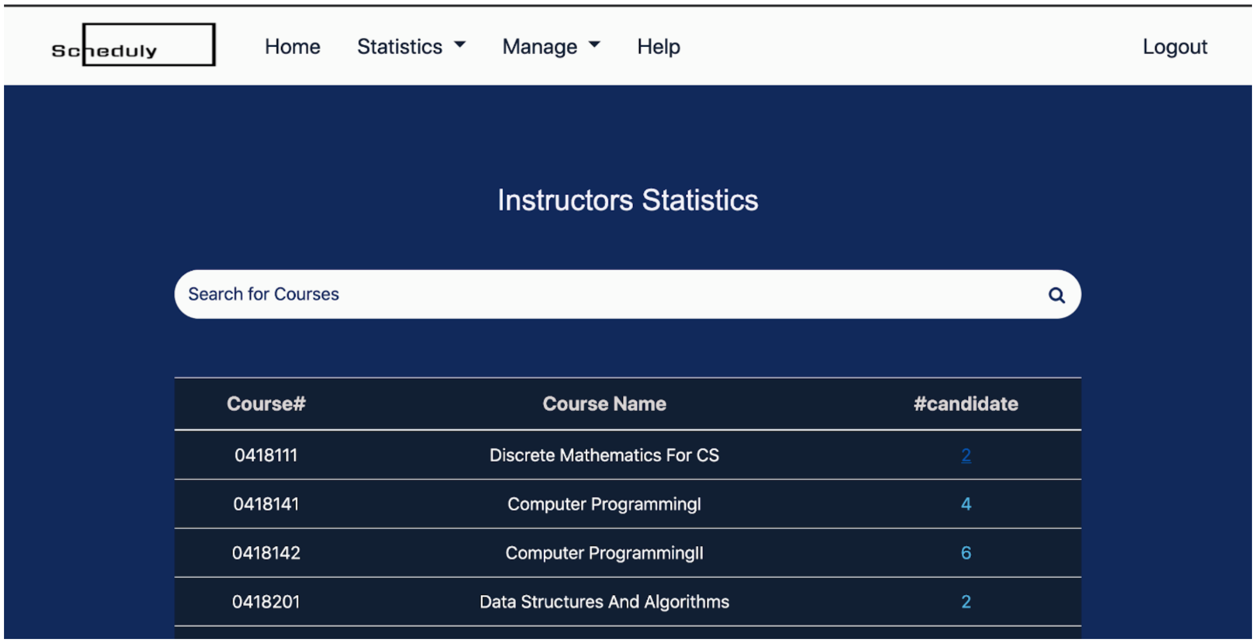
Unpreferred Lecture Time 1		Unpreferred Lecture Time 2	
Choose days : 135	From: --:-- -- To: --:-- --	Choose days : 135	From: --:-- -- To: --:-- --

FIGURE 25: FURTHER INTRUCTOR WISH LIST CHOICE



Your whishlist has been saved

FIGURE 26: INFORMATIVE FEEDBACK #6

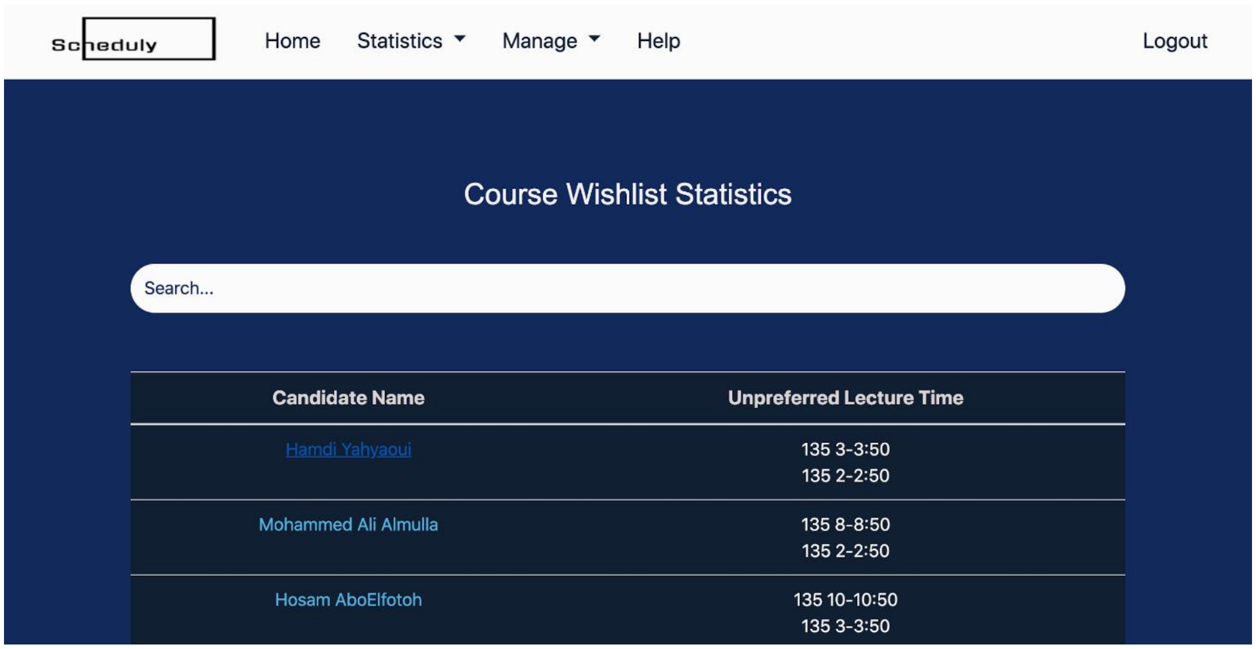


Instructors Statistics

Course#	Course Name	#candidate
0418111	Discrete Mathematics For CS	2
0418141	Computer ProgrammingI	4
0418142	Computer ProgrammingII	6
0418201	Data Structures And Algorithms	2

FIGURE 27: SUPERVISOR PAGE TO VIEW STATISTICS FOR COMULSORY COURSES

This informational page is to communicate to the schedule supervisor the number of instructors offering to teach a certain course.



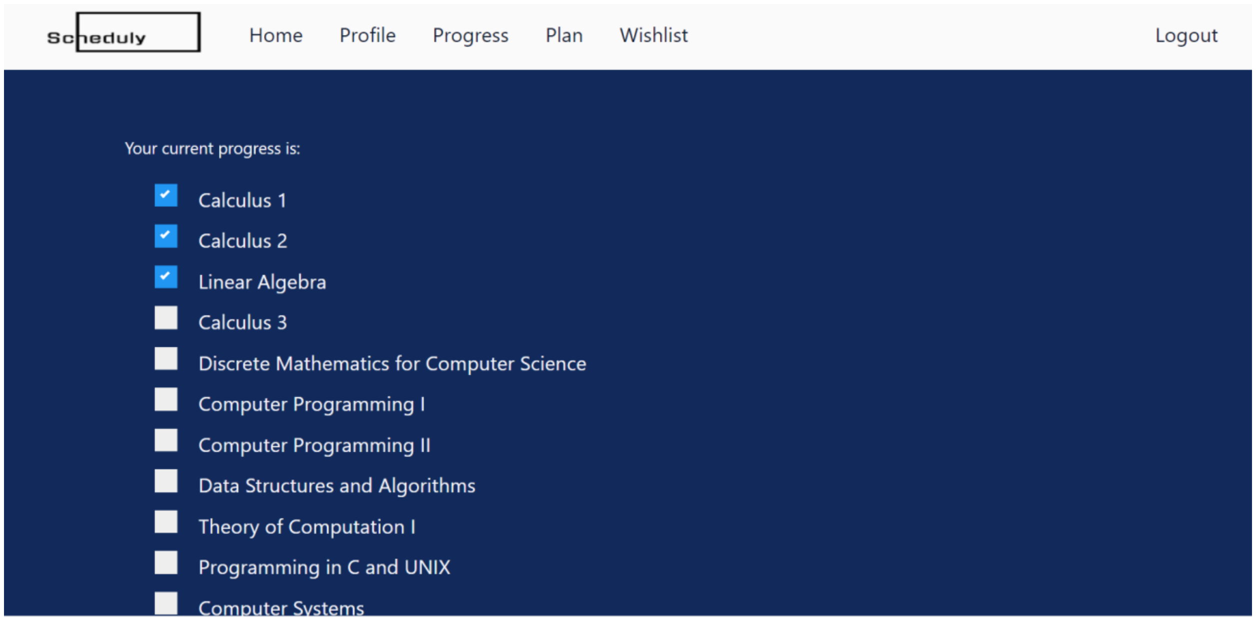
Course Wishlist Statistics

Search...

Candidate Name	Unpreferred Lecture Time
Hamdi Yahyaoui	135 3-3:50 135 2-2:50
Mohammed Ali Almulla	135 8-8:50 135 2-2:50
Hosam AboElfotoh	135 10-10:50 135 3-3:50

FIGURE 28: SUPERVISOR PAGE TO VIEW STATISTICS FOR INSTRUCTOR'S TIME PREFERENCES

This informational page is to communicate to the schedule supervisor the time preferences of instructors offering to teach a certain course.

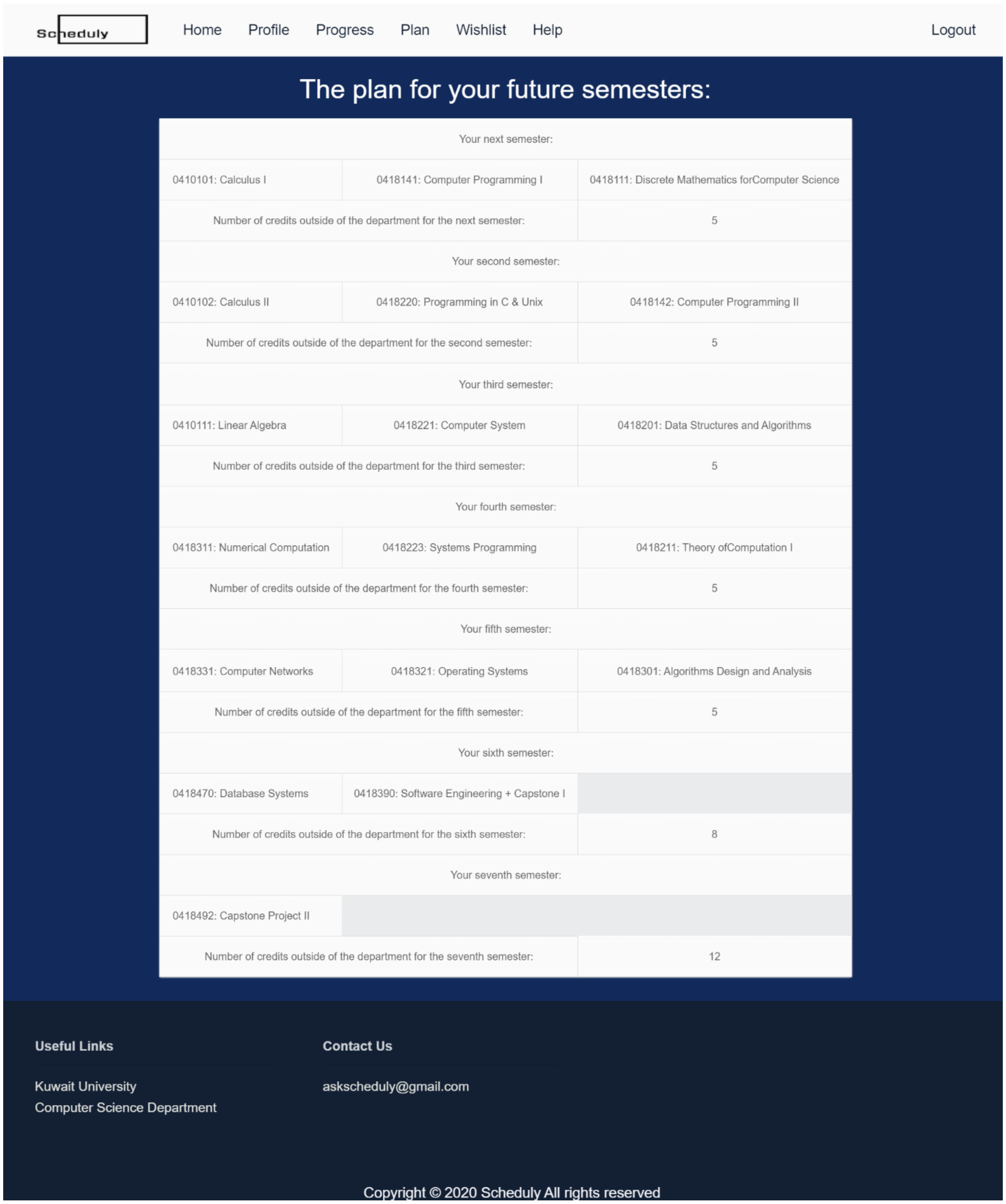


Your current progress is:

- ✓ Calculus 1
- ✓ Calculus 2
- ✓ Linear Algebra
- ☐ Calculus 3
- ☐ Discrete Mathematics for Computer Science
- ☐ Computer Programming I
- ☐ Computer Programming II
- ☐ Data Structures and Algorithms
- ☐ Theory of Computation I
- ☐ Programming in C and UNIX
- ☐ Computer Systems

FIGURE 29: STUDENT PAGE FOR UPDATING OR VIEWING MAJOR SHEET PROGRESS

This page is to offer the student the option to update or view their progress.



The plan for your future semesters:

Your next semester:		
0410101: Calculus I	0418141: Computer Programming I	0418111: Discrete Mathematics forComputer Science
Number of credits outside of the department for the next semester:		5
Your second semester:		
0410102: Calculus II	0418220: Programming in C & Unix	0418142: Computer Programming II
Number of credits outside of the department for the second semester:		5
Your third semester:		
0410111: Linear Algebra	0418221: Computer System	0418201: Data Structures and Algorithms
Number of credits outside of the department for the third semester:		5
Your fourth semester:		
0418311: Numerical Computation	0418223: Systems Programming	0418211: Theory ofComputation I
Number of credits outside of the department for the fourth semester:		5
Your fifth semester:		
0418331: Computer Networks	0418321: Operating Systems	0418301: Algorithms Design and Analysis
Number of credits outside of the department for the fifth semester:		5
Your sixth semester:		
0418470: Database Systems	0418390: Software Engineering + Capstone I	
Number of credits outside of the department for the sixth semester:		8
Your seventh semester:		
0418492: Capstone Project II		
Number of credits outside of the department for the seventh semester:		12

Useful Links

Kuwait University
Computer Science Department

Contact Us

askscheduly@gmail.com

Copyright © 2020 Scheduly All rights reserved

FIGURE 30: STUDENT PLAN EXAMPLE CASE

4.5 ALGORITHMS

To implement features of the system such as formulating the plan, validating the progress, and retrieving statistics. The team developed algorithms using python, which allowed the data to be treated as objects and could be easily iterated through and manipulated.

All statistics are handled by many-to-many relations with python inbuilt count function

```
If the post request is of a course to be added to a student's or instructor's wish list
        Add a new relation
If the get request is to view statistics
        Count number of relations
```

Python handles the relations as objects which makes it easier for information to be counter or filtered.

The course class had the following methods that returns a value to assures dependencies:

```
function is_available():
        return if all is true
                        from Progress filter (course, student id)
                        for course in self.dependancy.all()
function is_available_after():
         return true if all is true
            course in courses_completed
            for course in self.number.prerequisite.all()
```

The progress object existence proves the student finished the course. These methods are called often in the progress algorithm and in the plan algorithm.

For the optimal plan

```
    To fetch list of courses completed = [progress.course for progress in
Progress.objects.filter(studentID=this.user)
    Fetch the number of credits of the student
    If the student is a freshmen
        Set the limit of the courses in a semester to 3
    Else
        Set the limit of the courses in a semester to 2
    Couses_left = list of courses – list of courses completed
    while courses_left:
         courses_available = [
            course for course in courses_left
            if course.is_available_after(courses_completed) ]
         if not courses_available:
            courses_completed.extend(semester)
            semesters.append(semester)
         while courses_available:
```

```
        course = courses_available.pop()
        courses_left.remove(course)
        if number of courses in this semester is less than limit of courses in a semester:
            add new course to the semester
            credits for the semester -= number of the credits of added course
    append to plan the semester
```

5. MODIFICATIONS OF THE ORIGINAL PLAN

During the development of this project the team and the entire world faced an unforeseeable event. The pandemic that changed a lot also changed the working plan on this project. During that, as team members tried to develop their own vision of the website, each member found a more efficient way to implement the requirements.

- The plan no longer includes courses outside of the Computer Science department except for Math Courses. Reasons for that:
    - Non-department courses have many choices and different circumstances.
    - Some math courses are a prerequisite of some of the department's courses.
- The progress includes only math and Computer Science courses.

Another major modification of the original plan was the duration of the project as it was postponed.

6. IMPLEMENTATION FRAMEWORK AND DETAILS

The services offered by the system are targeted for web browsers on desktop and mobile platforms.

- Django Framework: the python-operated framework was used to develop and construct the major component critical to the system
- Bootstrap: the open-source CSS framework helped build responsive and mobile-friendly front-end interactions with users.
- Selenium Framework: the tool designed for testing web applications helped run python-operated tests on this web-app.

6.1 COMPONENT AND CODE REUSE

Components of the system were based on built-in Django library functions such as:

- The Django model layer which is the abstraction for structuring and manipulating the data of the web application.
- The Django view layer to encapsulate logic of request and response.
- The Django template layer for rendering information into HTML files.
- Django built-in template tags and filters to adjust variables on each individual page.
- Django build-in authentication system that handles both authentication and authorization.
- Django built-in QuerySet to retrieve data the database.
- Django Database SQLite to store the data of the system.
- Django built-in library Forms was a component heavily used to create and manipulate form data.
- Django build in library FormSet for dynamic forms.
- Django built-in admin library was expanded to fit the admin intended of this system.
- Python OOP programming to manipulate and access data as objects.
- Python itertools module for iterating through objects.

7. TESTING

The testing phase in this project included a set of tests varying from quality to reliability. This section hopes to guide throughout the process the team followed.

7.1 TESTING PLAN

The team had chosen to work with the python-operated Django framework since it was an environment supported by deployment and tests. As well as its support of running component tests with easy commands. Each unit tests made sure invalid inputs for each component were not something the system cannot handle. Following that the integration tests made sure mentioned components worked well together and correctly. Stress and Performance tests automated HTTP requests to the see how the system remained durable. And finally, reliability tests meant to show how the code results remained consistent.

7.2 UNIT TEST CASES

The components of the system which are the smallest parts of that were tested with unit test cases across the different Django files, specifically: the python units that handled views, models, forms, and redirecting HTTP and URL requests.

#	Test Target	Test Name	Test Result
1	Form	test_add_date_time_interval_form	Pass
2	Form	test_invalid_add_date_time_form_types_time	Pass
3	Form	test_invalid_add_date_time_form_types_date	Pass
4	Form	test_invalid_add_date_time_form_types_time	Pass
5	Form	test_user_update_form	Pass
6	Form	test_invalid_user_update_form	Pass
7	Form	test_user_login_form	Pass
8	Form	test_invalid_user_login_form	Pass
9	Form	Instructor profile change email	Pass
10	Form	test_invalid_user_login_form_no_username	Pass

11	Form	test_invalid_user_login_form_empty_fields	Pass
12	Form	test_invalid_add_course_form_empty_fields	Pass
13	Form	test_invalid_add_course_form_empty_name	Pass
14	Form	test_invalid_add_course_form_empty_credits	Pass
15	Form	test_invalid_add_course_form_empty_number	Pass
16	Form	test_invalid_add_course_form_invalid_number	Pass
17	Form	test_invalid_add_course_form_invalid_credits	Pass
18	Form	test_invalid_add_course_form_negative_credits	Pass
19	Form	test_valid_add_course_form	Pass
20	Form	test_invalid_add_elective_course_form_empty_fields	Pass
21	Form	test_invalid_add_elective_course_form_empty_name	Pass
22	Form	test_invalid_add_elective_course_form_empty_credits	Pass
23	Form	test_invalid_add_elective_course_form_empty_number	Pass
24	Form	test_invalid_add_elective_course_form_invalid_number	Pass
25	Form	test_invalid_add_elective_course_form_invalid_credits	Pass
26	Form	test_invalid_add_elective_course_form_negative_credits	Pass
27	Form	test_valid_add_elective_course_form	Pass
28	Form	test_valid_add_elective_course_form_with_total_num_of_students	Pass
29	Form	test_invalid_add_elective_course_form_with_total_num_of_students	Pass
30	Form	test_invalid_add_elective_course_form_with_total_num_of_students_negative	Pass
31	Model	test_create_Instructor	Pass
32	Model	test_create_Student	Pass

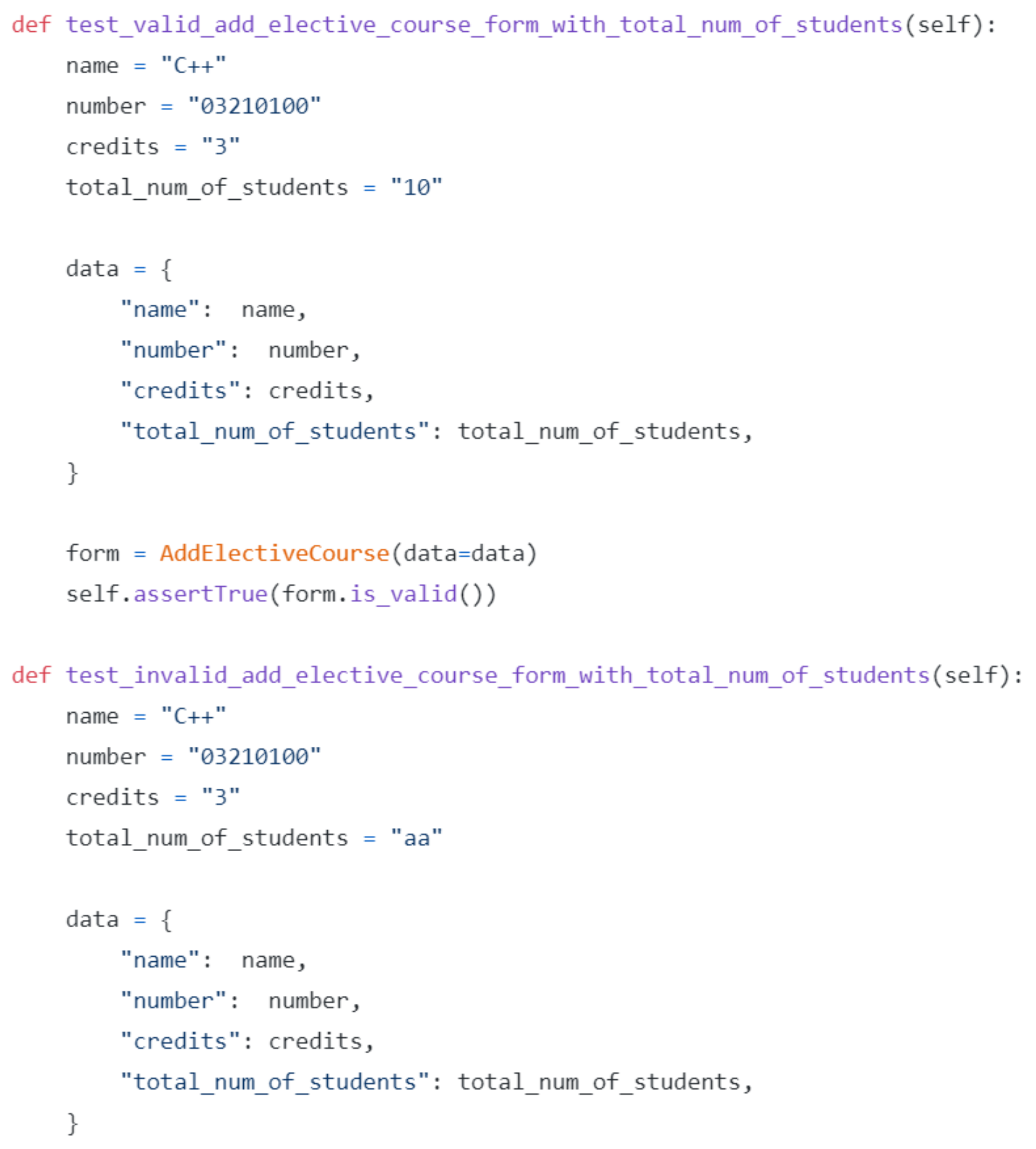
33	Model	test_create_DateTimeInterval	Pass
34	Model	test_create_Profile	Pass
35	Model	test_create_Course	Pass
36	Model	test_create_dependant_Course	Pass
37	Model	test_create_ElectiveCourse	Pass
38	Model	test_create_Progress	Pass
39	Model	test_create_Wishlist	Pass
40	Model	test_create_Semester	Pass
41	Model	test_create_UnpreferredLectureTime	Pass
42	URL	test_user_login_url_is_resolved	Pass
43	URL	test_home_url_is_resolved	Pass
44	URL	test_user_profile_url_is_resolved	Pass
45	URL	test_no_access_url_is_resolved	Pass
46	URL	test_add_instructor_url_is_resolved	Pass
47	URL	test_add_student_url_is_resolved	Pass
48	URL	test_edit_major_sheet_url_is_resolved	Pass
49	URL	test_add_course_url_is_resolved	Pass
50	URL	test_add_elective_course_url_is_resolved	Pass
51	URL	test_edit_course_url_is_resolved	Pass
52	URL	test_delete_course_url_is_resolved	Pass
53	URL	test_delete_course_confirmation_url_is_resolved	Pass
54	URL	test_list_to_edit_url_is_resolved	Pass
55	URL	test_edit_instructor_url_is_resolved	Pass
56	URL	test_edit_student_url_is_resolved	Pass
57	URL	test_delete_user_url_is_resolved	Pass

58	URL	test_delete_user_confirmation_url_is_resolved	Pass
59	URL	test_delete_users_url_is_resolved	Pass
60	URL	test_setting_time_url_is_resolved	Pass
61	URL	test_list_time_url_is_resolved	Pass
62	URL	test_update_time_url_is_resolved	Pass
63	URL	test_delete_time_url_is_resolved	Pass
64	URL	test_delete_time_confirmation_url_is_resolved	Pass
65	URL	test_instructor_wishlist_url_is_resolved	Pass
66	URL	test_instructor_wishlist_settime_url_is_resolved	Pass
67	URL	test_instructors_statistics_url_is_resolved	Pass
68	URL	test_candidate_info_url_is_resolved	Pass
69	URL	test_reset_instructors_statistics_url_is_resolved	Pass
70	URL	test_reset_instructors_statistics_confirmation_url_is_res olved	Pass
71	URL	test_progress_url_is_resolved	Pass
72	URL	test_plan_url_is_resolved	Pass
73	URL	test_student_wishlist_url_is_resolved	Pass
74	URL	test_students_statistics_url_is_resolved	Pass
75	URL	test_reset_students_statistics_url_is_resolved	Pass
76	URL	test_reset_students_statistics_confirmation_url_is_resolv ed	Pass
77	View	test_user_login_GET	Pass
78	View	test_user_logout_GET	Pass
79	View	test_home_GET	Pass
80	View	test_list_to_delete_GET	Pass

81	View	test_list_time_GET	Pass
82	View	test_edit_major_sheet_GET	Pass
83	View	test_add_course_GET	Pass
84	View	test_add_student_view	Pass

Table 1: Unit tests and results

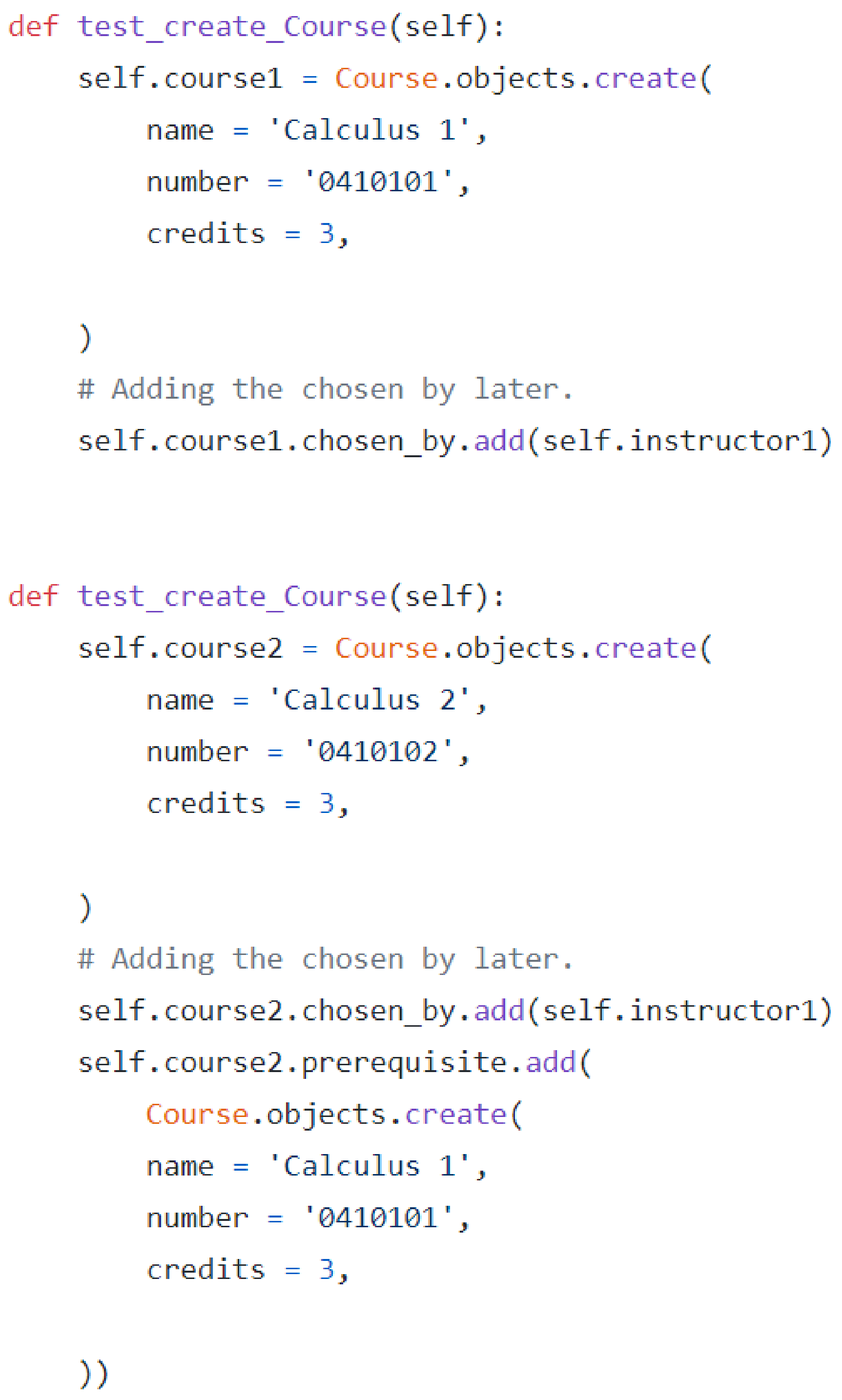
7.2.1 FORM TESTS SAMPLES



```
def test_valid_add_elective_course_form_with_total_num_of_students(self):
    name = "C++"
    number = "03210100"
    credits = "3"
    total_num_of_students = "10"

    data = {
        "name":  name,
        "number":  number,
        "credits": credits,
        "total_num_of_students": total_num_of_students,
    }

    form = AddElectiveCourse(data=data)
    self.assertTrue(form.is_valid())

def test_invalid_add_elective_course_form_with_total_num_of_students(self):
    name = "C++"
    number = "03210100"
    credits = "3"
    total_num_of_students = "aa"

    data = {
        "name":  name,
        "number":  number,
        "credits": credits,
        "total_num_of_students": total_num_of_students,
    }
```

FIGURE 31: FORM TEST SAMPLE

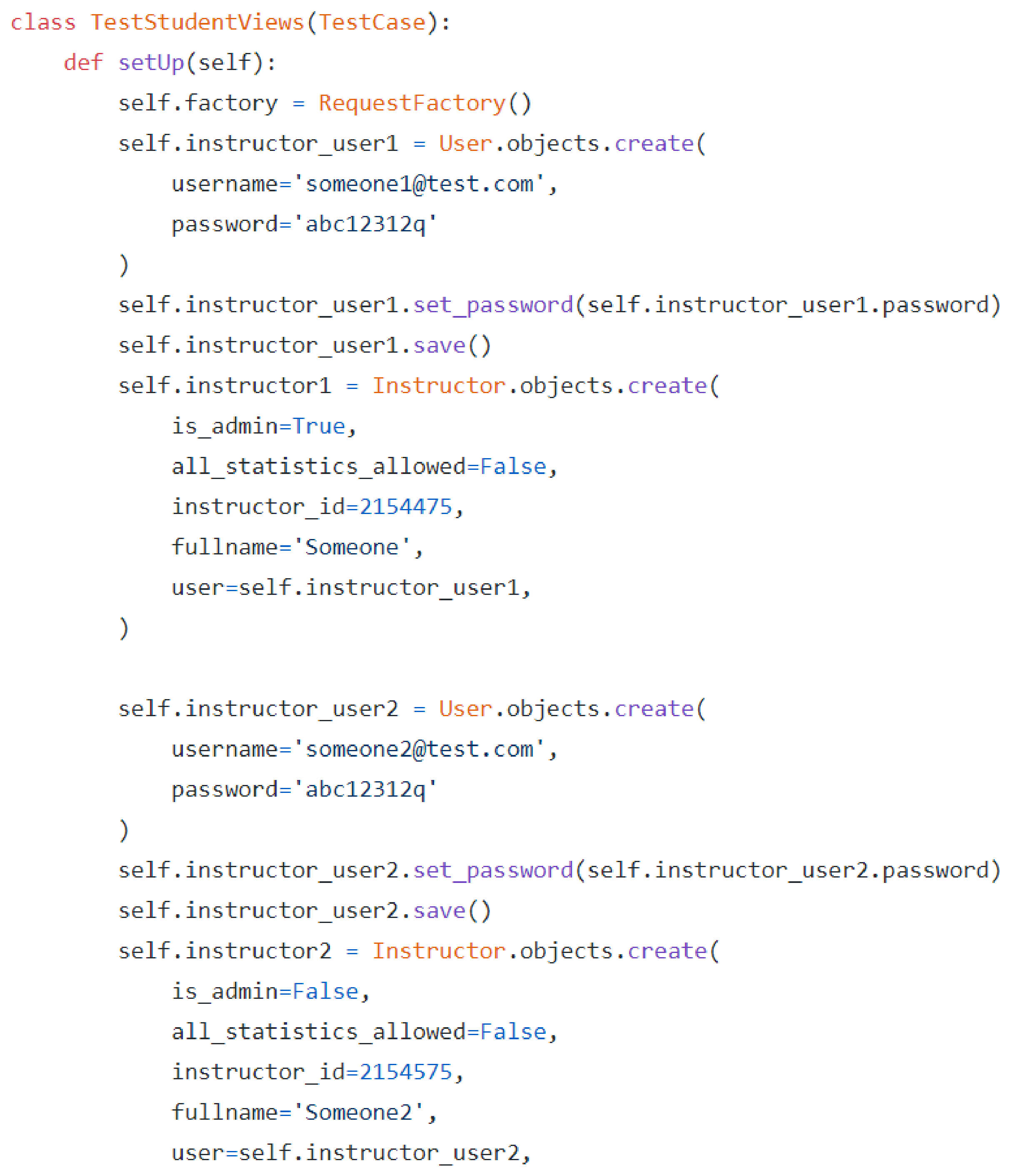
7.2.2 MODEL TESTS SAMPLES



```
def test_create_Course(self):
    self.course1 = Course.objects.create(
        name = 'Calculus 1',
        number = '0410101',
        credits = 3,

    )
    # Adding the chosen by later.
    self.course1.chosen_by.add(self.instructor1)


def test_create_Course(self):
    self.course2 = Course.objects.create(
        name = 'Calculus 2',
        number = '0410102',
        credits = 3,

    )
    # Adding the chosen by later.
    self.course2.chosen_by.add(self.instructor1)
    self.course2.prerequisite.add(
        Course.objects.create(
        name = 'Calculus 1',
        number = '0410101',
        credits = 3,

    ))
```

FIGURE 32: MODELS TEST SAMPLE

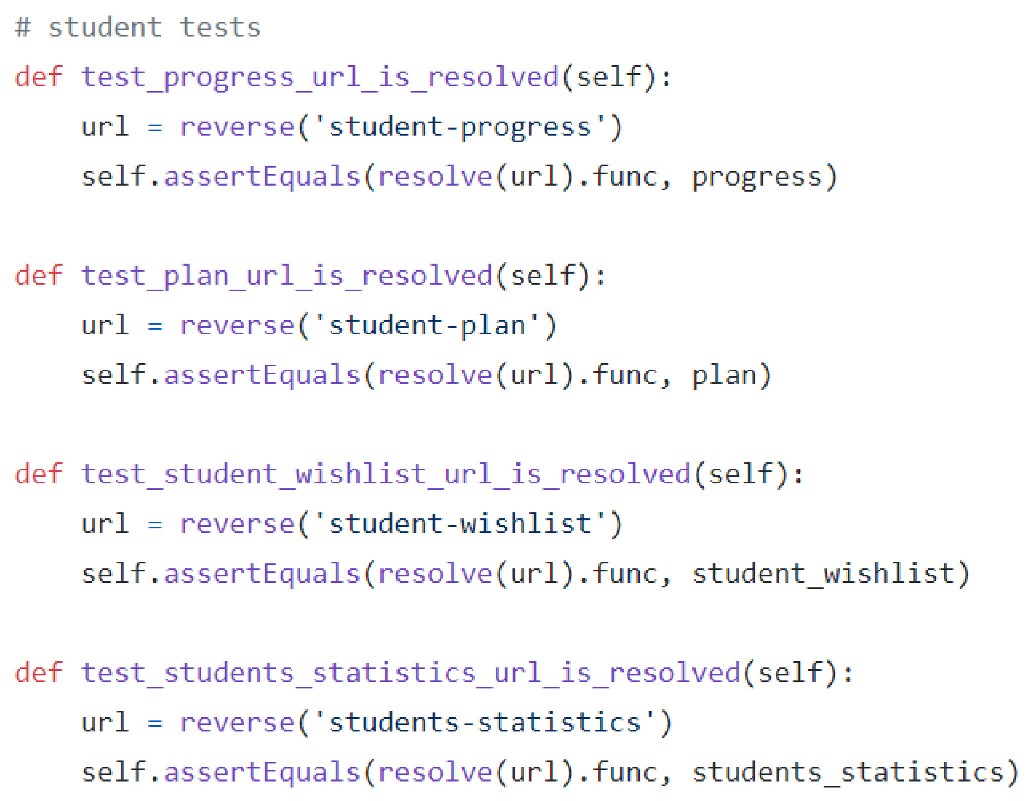
7.2.3 VIEW TESTS



```
class TestStudentViews(TestCase):
    def setUp(self):
        self.factory = RequestFactory()
        self.instructor_user1 = User.objects.create(
            username='someone1@test.com',
            password='abc12312q'
        )
        self.instructor_user1.set_password(self.instructor_user1.password)
        self.instructor_user1.save()
        self.instructor1 = Instructor.objects.create(
            is_admin=True,
            all_statistics_allowed=False,
            instructor_id=2154475,
            fullname='Someone',
            user=self.instructor_user1,
        )

        self.instructor_user2 = User.objects.create(
            username='someone2@test.com',
            password='abc12312q'
        )
        self.instructor_user2.set_password(self.instructor_user2.password)
        self.instructor_user2.save()
        self.instructor2 = Instructor.objects.create(
            is_admin=False,
            all_statistics_allowed=False,
            instructor_id=2154575,
            fullname='Someone2',
            user=self.instructor_user2,
```

FIGURE 33VIEW TEST SAMPLE

7.2.4 URL TESTS



```python
# student tests
def test_progress_url_is_resolved(self):
    url = reverse('student-progress')
    self.assertEquals(resolve(url).func, progress)

def test_plan_url_is_resolved(self):
    url = reverse('student-plan')
    self.assertEquals(resolve(url).func, plan)

def test_student_wishlist_url_is_resolved(self):
    url = reverse('student-wishlist')
    self.assertEquals(resolve(url).func, student_wishlist)

def test_students_statistics_url_is_resolved(self):
    url = reverse('students-statistics')
    self.assertEquals(resolve(url).func, students_statistics)
```

FIGURE 34: URL TEST SAMPLE

7.3 INTEGRATION TESTS

Integration tests were conducted to see if components were interacting well with one another or not. For this, the Chrome webdriver was used with the python-supported Selenium framework to run automated web-interactions and find the many errors in the system.

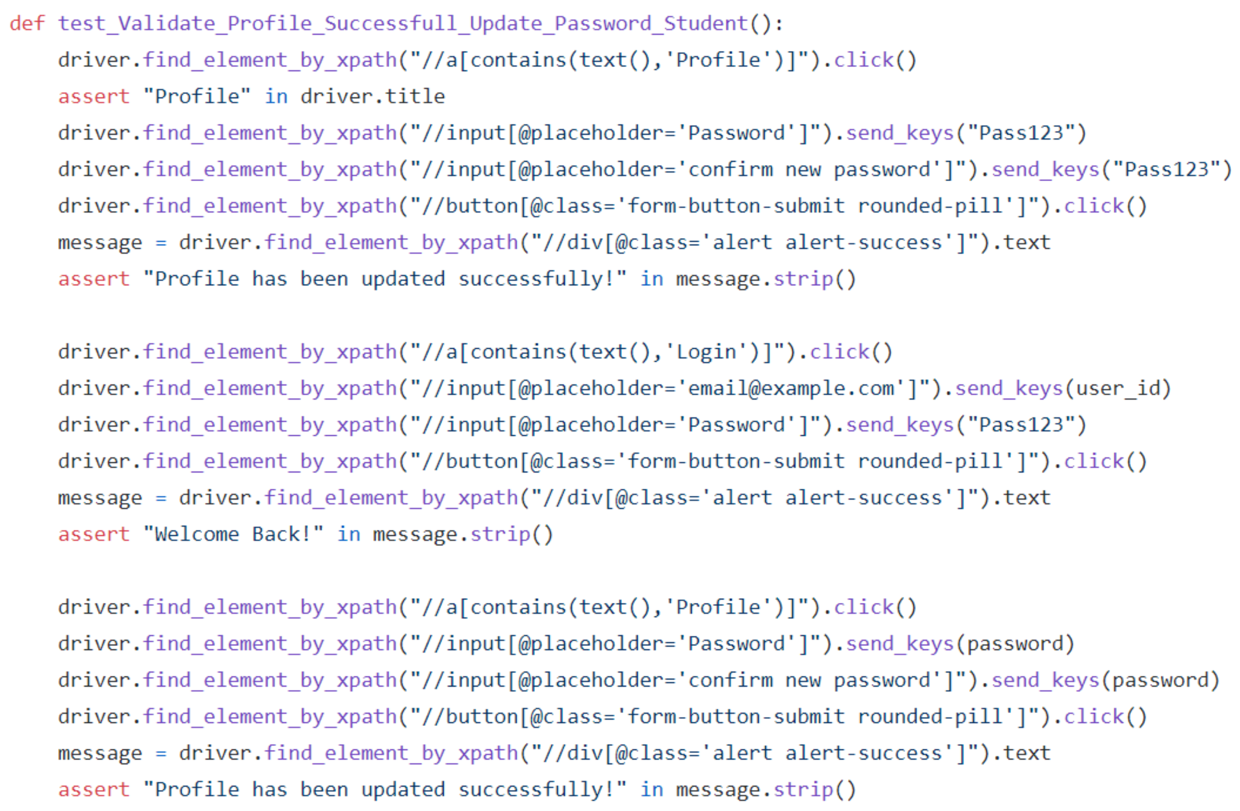
Test #	**Test Name**	**Test Result**
1	test_Validate_Login_Supervisor	Pass
2	test_Validate_HomePage_Supervisor	Pass
3	test_Validate_Profile_Successfull_Update_Password_Supervisor	Pass
4	test_Validate_Error_Message_Profile_Update_Password_Supervisor	Pass
5	test_Validate__Profile_Update_Password_Supervisor_Same_Pass	Pass
6	test_Validate_Help_Page_Supervisor	Pass
7	test_Add_Wishlist_Supervisor	Pass
8	test_Validate_Login_Student	Pass
9	test_Validate_HomePage_Student	Pass
10	test_Validate_Profile_Successfull_Update_Password_Student	Pass
11	test_Validate_Error_Message_Profile_Update_Password_Student	Pass
12	test_Validate__Profile_Update_Password_Student_Same_Pass	Pass
13	test_Validate_Help_Page_Student	Pass
14	test_Validate_Progress_Student	Pass

15	test_Validate_Plan_Student	Pass
16	test_Add_Wishlist_Student	Pass
17	test_Validate_Login_Instructor	Pass
18	test_Validate_HomePage_Instructor	Pass
19	test_Validate_Profile_Successfull_Update_Password_Instructor	Pass
20	test_Validate_Error_Message_Profile_Update_Password_Instructor	Pass
21	test_Validate__Profile_Update_Password_Instructor_Same_Pass	Pass
22	test_Validate_Help_Page_Instructor	Pass
23	test_Students_Statistics_Instructor	Pass
24	test_Add_Wishlist_Instructor	Pass
25	test_Validate_LoginAdmin	Pass
26	test_Validate_HomePage_Admin	Pass
27	test_Validate_Profile_Successfull_Update_Password_Admin	Pass
28	test_Validate_Error_Message_Profile_Update_Password_Admin	Pass
29	test_Validate__Profile_Update_Password_Admin_Same_Pass	Pass
30	test_Validate_Help_Page_Admin	Pass
31	test_Students_Statistics_Admin	Pass
32	test_Validate_Add_Instructor_Admin	Pass
33	test_Validate_Update_Instructor_Admin	Pass
34	test_Validate_Delete_Instructor_Admin	Pass

35	test_Validate_Add_User_Admin	Pass
36	test_Validate_Update_User_Admin	Pass
37	test_Validate_Delete_User_Admin	Pass
38	test_Validate_Add_Course_Admin	Pass
39	test_Validate_Update_Course_Admin	Pass
40	test_Validate_Delete_Course_Admin	Pass
41	test_Validate_Update_Setting_Time_Admin	Pass
42	test_Validate_Delete_Setting_Time_Admin	Pass
43	test_Validate_Add_Setting_Time_Admin	Pass
44	test_Add_Wishlist_Admin	Pass
45	test_Instructor_Statistics_Admin	Pass

Table 2: Integration tests and results

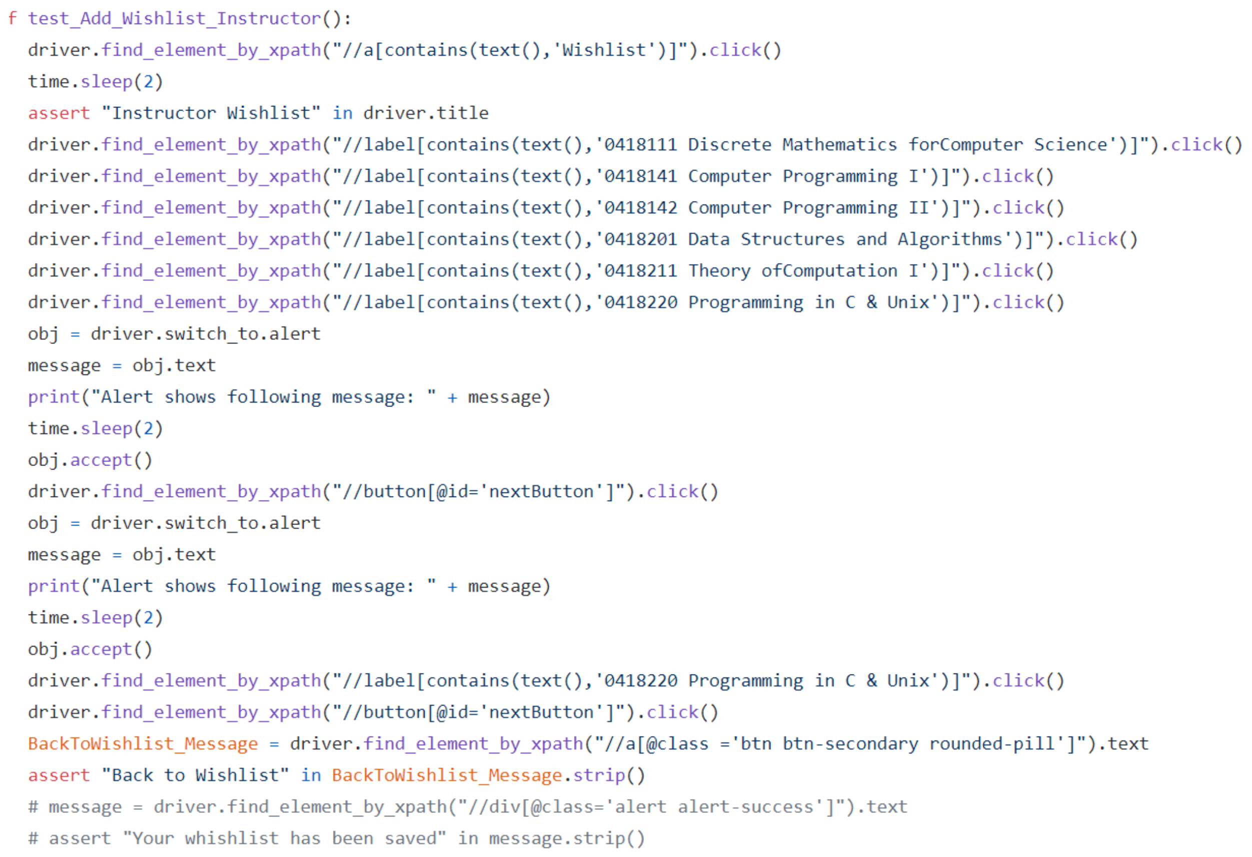
7.3.1 STUDENT INTEGRATION TEST SAMPLE



```
def test_Validate_Profile_Successfull_Update_Password_Student():
    driver.find_element_by_xpath("//a[contains(text(),'Profile')]").click()
    assert "Profile" in driver.title
    driver.find_element_by_xpath("//input[@placeholder='Password']").send_keys("Pass123")
    driver.find_element_by_xpath("//input[@placeholder='confirm new password']").send_keys("Pass123")
    driver.find_element_by_xpath("//button[@class='form-button-submit rounded-pill']").click()
    message = driver.find_element_by_xpath("//div[@class='alert alert-success']").text
    assert "Profile has been updated successfully!" in message.strip()

    driver.find_element_by_xpath("//a[contains(text(),'Login')]").click()
    driver.find_element_by_xpath("//input[@placeholder='email@example.com']").send_keys(user_id)
    driver.find_element_by_xpath("//input[@placeholder='Password']").send_keys("Pass123")
    driver.find_element_by_xpath("//button[@class='form-button-submit rounded-pill']").click()
    message = driver.find_element_by_xpath("//div[@class='alert alert-success']").text
    assert "Welcome Back!" in message.strip()

    driver.find_element_by_xpath("//a[contains(text(),'Profile')]").click()
    driver.find_element_by_xpath("//input[@placeholder='Password']").send_keys(password)
    driver.find_element_by_xpath("//input[@placeholder='confirm new password']").send_keys(password)
    driver.find_element_by_xpath("//button[@class='form-button-submit rounded-pill']").click()
    message = driver.find_element_by_xpath("//div[@class='alert alert-success']").text
    assert "Profile has been updated successfully!" in message.strip()
```

FIGURE 35: STUDENT INTEGRATION TEST SAMPLE

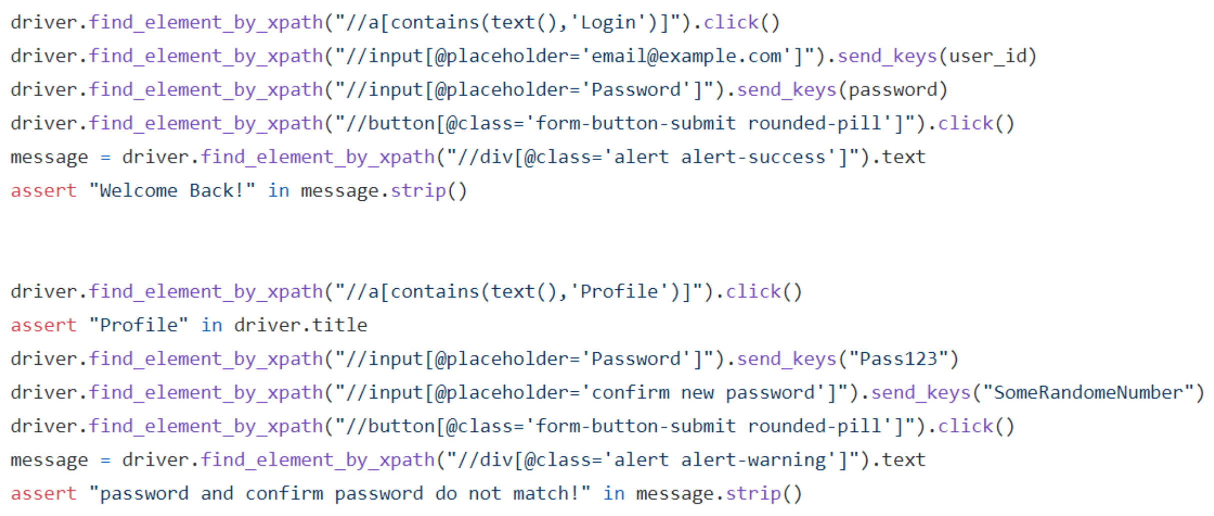
7.3.2 INSTRUCTOR INTEGRATION TEST SAMPLE



```
f test_Add_Wishlist_Instructor():
  driver.find_element_by_xpath("//a[contains(text(),'Wishlist')]").click()
  time.sleep(2)
  assert "Instructor Wishlist" in driver.title
  driver.find_element_by_xpath("//label[contains(text(),'0418111 Discrete Mathematics forComputer Science')]").click()
  driver.find_element_by_xpath("//label[contains(text(),'0418141 Computer Programming I')]").click()
  driver.find_element_by_xpath("//label[contains(text(),'0418142 Computer Programming II')]").click()
  driver.find_element_by_xpath("//label[contains(text(),'0418201 Data Structures and Algorithms')]").click()
  driver.find_element_by_xpath("//label[contains(text(),'0418211 Theory ofComputation I')]").click()
  driver.find_element_by_xpath("//label[contains(text(),'0418220 Programming in C & Unix')]").click()
  obj = driver.switch_to.alert
  message = obj.text
  print("Alert shows following message: " + message)
  time.sleep(2)
  obj.accept()
  driver.find_element_by_xpath("//button[@id='nextButton']").click()
  obj = driver.switch_to.alert
  message = obj.text
  print("Alert shows following message: " + message)
  time.sleep(2)
  obj.accept()
  driver.find_element_by_xpath("//label[contains(text(),'0418220 Programming in C & Unix')]").click()
  driver.find_element_by_xpath("//button[@id='nextButton']").click()
  BackToWishlist_Message = driver.find_element_by_xpath("//a[@class ='btn btn-secondary rounded-pill']").text
  assert "Back to Wishlist" in BackToWishlist_Message.strip()
  # message = driver.find_element_by_xpath("//div[@class='alert alert-success']").text
  # assert "Your whishlist has been saved" in message.strip()
```

FIGURE 36: INSTRUCTOR INTEGRATION TEST SAMPLE

7.3.3 SUPERVISOR INTEGRATION TEST SAMPLES



```
driver.find_element_by_xpath("//a[contains(text(),'Login')]").click()
driver.find_element_by_xpath("//input[@placeholder='email@example.com']").send_keys(user_id)
driver.find_element_by_xpath("//input[@placeholder='Password']").send_keys(password)
driver.find_element_by_xpath("//button[@class='form-button-submit rounded-pill']").click()
message = driver.find_element_by_xpath("//div[@class='alert alert-success']").text
assert "Welcome Back!" in message.strip()


driver.find_element_by_xpath("//a[contains(text(),'Profile')]").click()
assert "Profile" in driver.title
driver.find_element_by_xpath("//input[@placeholder='Password']").send_keys("Pass123")
driver.find_element_by_xpath("//input[@placeholder='confirm new password']").send_keys("SomeRandomeNumber")
driver.find_element_by_xpath("//button[@class='form-button-submit rounded-pill']").click()
message = driver.find_element_by_xpath("//div[@class='alert alert-warning']").text
assert "password and confirm password do not match!" in message.strip()
```

FIGURE 37: SUPERVISOR INTEGRATION TEST SAMPLE

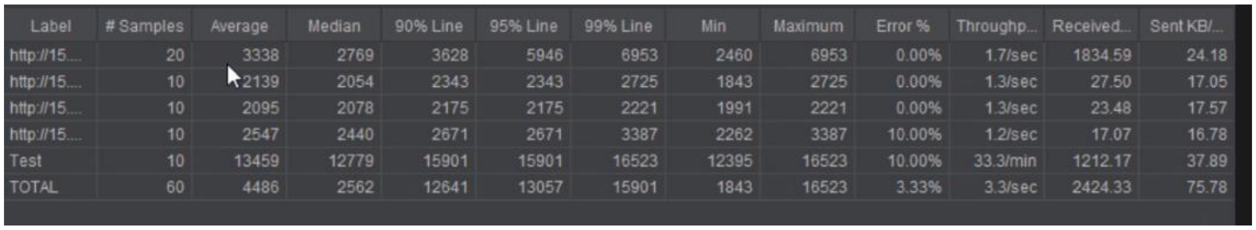
7.3.4 ADMIN INTEGRATION TEST SAMPLES

```
def test_Validate_Update_Instructor_Admin():
    driver.get("http://15.236.43.203:8000/manage/edit/users/")
    heading = driver.find_element_by_xpath("//h4[@class='title pt-4']").text
    assert "Instructors" in heading.strip()
    driver.find_element_by_xpath("//td[contains(text(),'New_Instructor')]//following-sibling::*/a").click()

    driver.find_element_by_xpath("//input[@name='fullname']").clear()
    driver.find_element_by_xpath("//input[@name='fullname']").send_keys("Updated_New_Instructor")
    driver.find_element_by_xpath("//input[@name='instructor_id']").clear()
    driver.find_element_by_xpath("//input[@name='instructor_id']").send_keys(87181)
    driver.find_element_by_xpath("//input[@name='username']").clear()
    driver.find_element_by_xpath("//input[@name='username']").send_keys("Updated_New_Instructor_87181@gmail.com")
    driver.find_element_by_xpath("//input[@name='password']").clear()
    driver.find_element_by_xpath("//input[@name='password']").send_keys("NewPass123")
    driver.find_element_by_xpath("//span[@class='checkmark']").click()
    driver.find_element_by_xpath("//button[@class='form-button-submit pr-5 btn-lg rounded-pill']").click()
    message = driver.find_element_by_xpath("//div[@class='alert alert-success']").text
    assert "The user was updated successfuly" in message.strip()
```

FIGURE 38: ADMIN INTEGRATION TEST SAMPLE

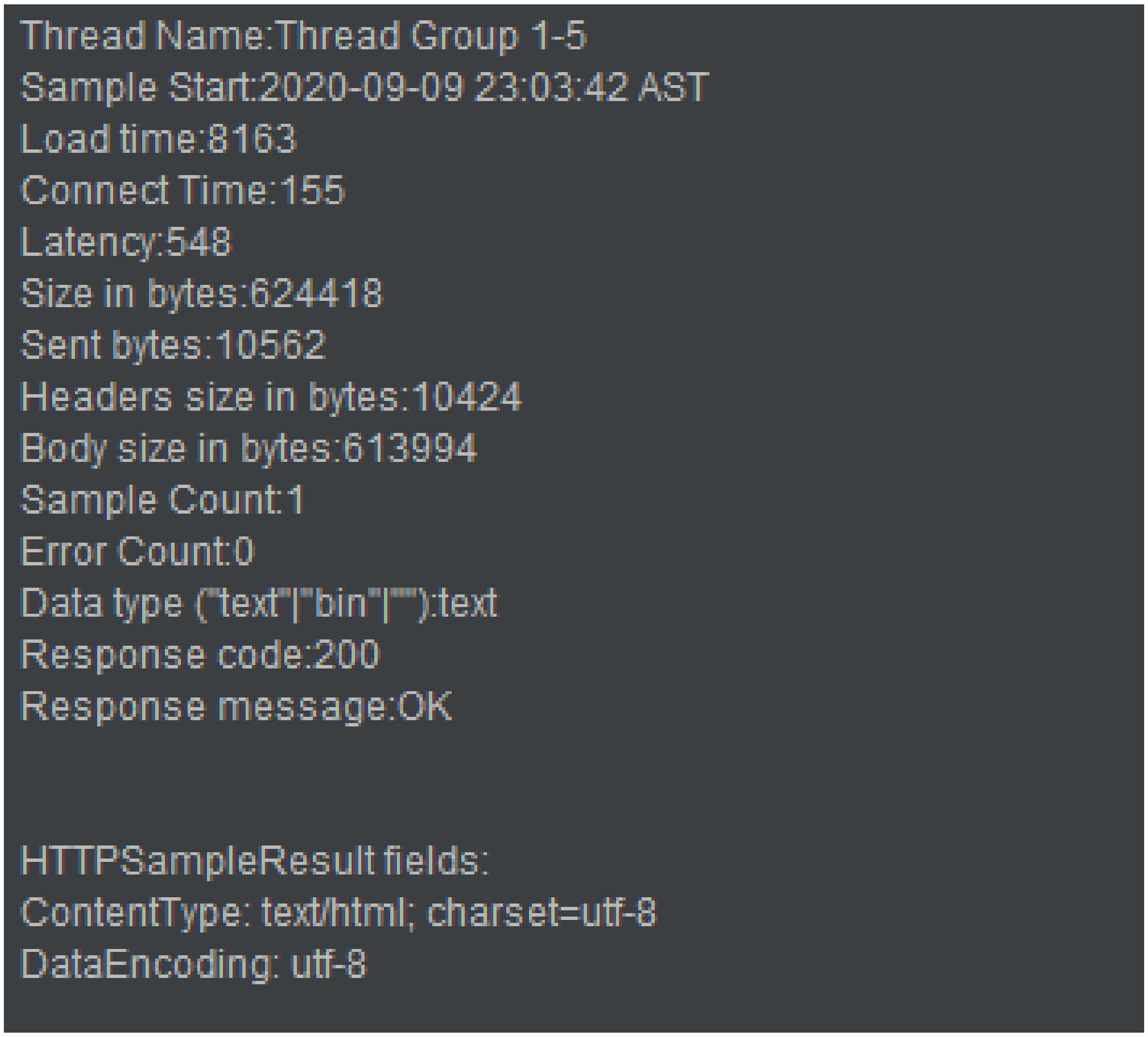
7.4 STRESS AND PERFORMANCE TESTS

Following deployment, the team tested the stress and performance of the system with JMeter. The test handled sending tests to the system's server to studty the performance of the system. To concurrently run valid user requests through HTTP Requests and check if they fail or not. The system easily handled around



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughp...	Received...	Sent KB/...
http://15....	20	3338	2769	3628	5946	6953	2460	6953	0.00%	1.7/sec	1834.59	24.18
http://15....	10	2139	2054	2343	2343	2725	1843	2725	0.00%	1.3/sec	27.50	17.05
http://15....	10	2095	2078	2175	2175	2221	1991	2221	0.00%	1.3/sec	23.48	17.57
http://15....	10	2547	2440	2671	2671	3387	2262	3387	10.00%	1.2/sec	17.07	16.78
Test	10	13459	12779	15901	15901	16523	12395	16523	10.00%	33.3/min	1212.17	37.89
TOTAL	60	4486	2562	12641	13057	15901	1843	16523	3.33%	3.3/sec	2424.33	75.78

FIGURE 39: JMETER RESULTS



```
Thread Name:Thread Group 1-5
Sample Start:2020-09-09 23:03:42 AST
Load time:8163
Connect Time:155
Latency:548
Size in bytes:624418
Sent bytes:10562
Headers size in bytes:10424
Body size in bytes:613994
Sample Count:1
Error Count:0
Data type ("text"|"bin"|""):text
Response code:200
Response message:OK


HTTPSampleResult fields:
ContentType: text/html; charset=utf-8
DataEncoding: utf-8
```

FIGURE 40: SAMPLE HTTP REQUEST RESPONE

Thanks to the scalability abilities of AWS the errors were lowers than thought. Increasing the number of users did not crash the system but some requests did fail. Figure 41 shows the error percentage as the number of tests increase.

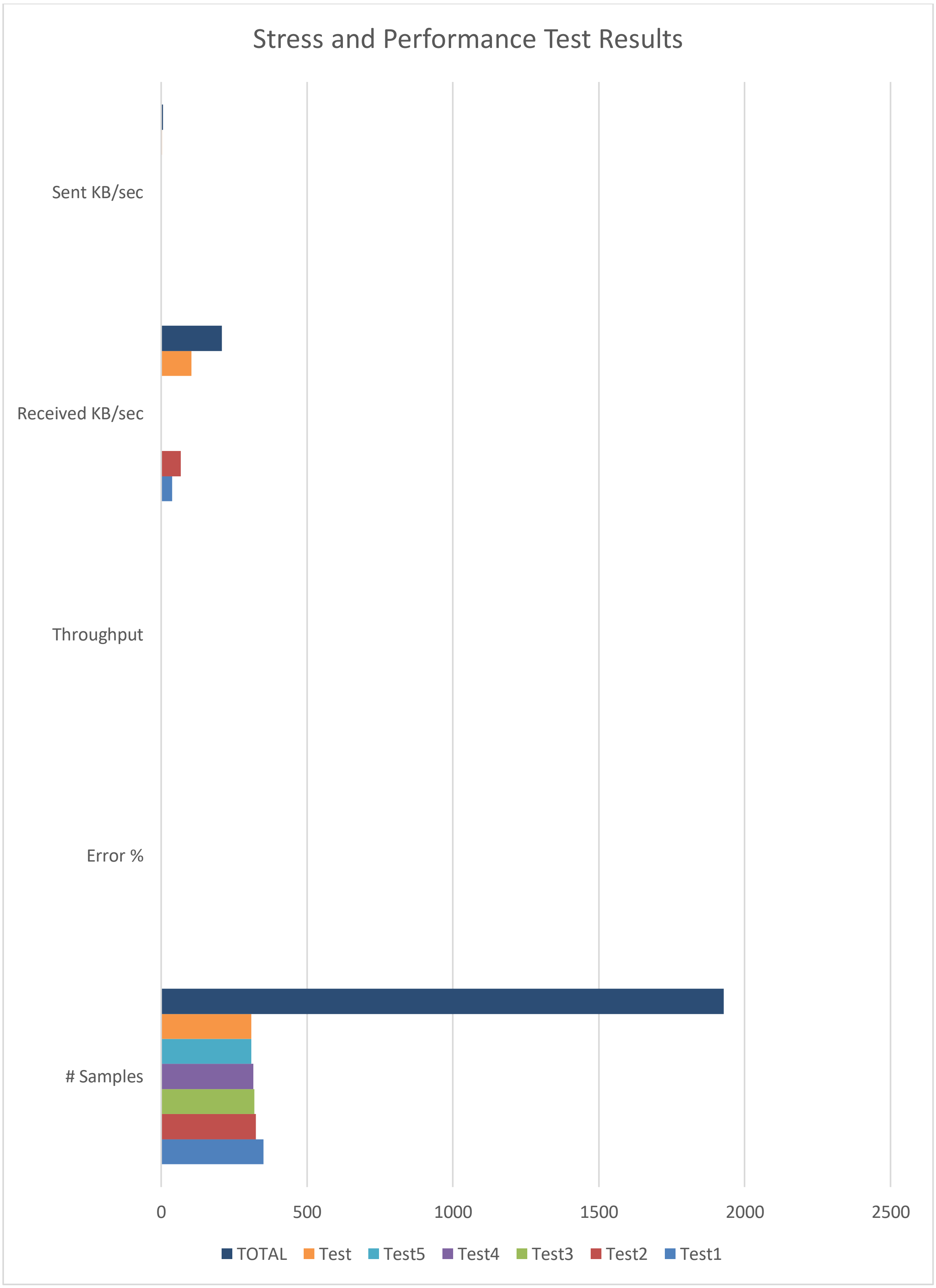
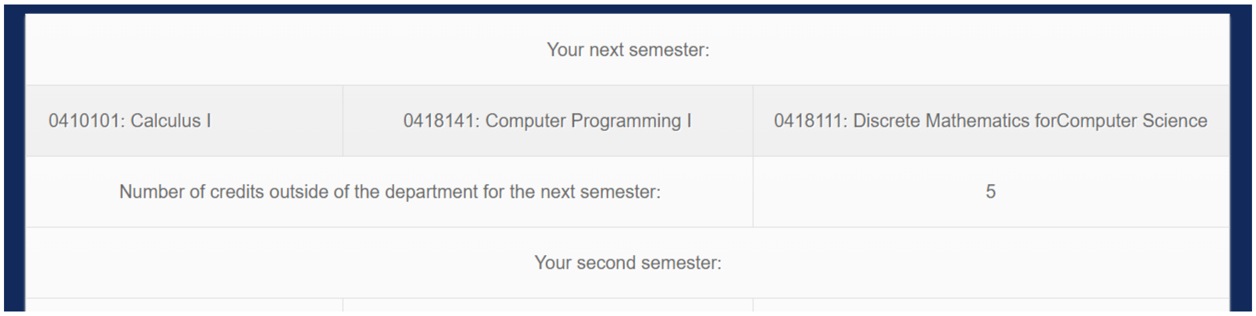


FIGURE 41: STRESS AND PERFORMANCE TEST

7.5 RELIABILITY TESTS

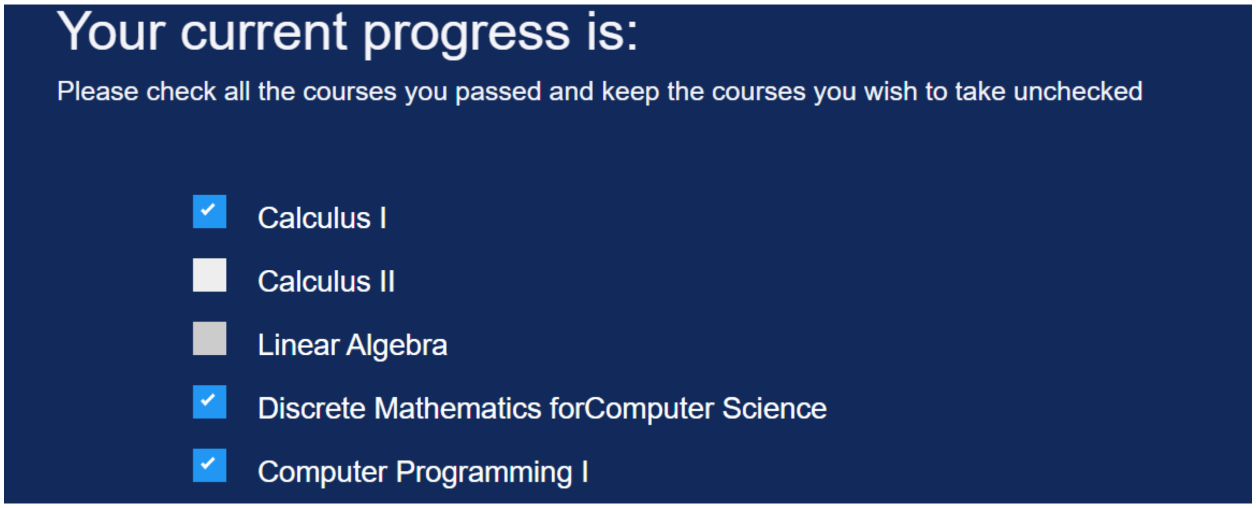
For reliability tests the team decided to test the student plan test to make sure the results remained consistent throughout a student's progress.



Your next semester:		
0410101: Calculus I	0418141: Computer Programming I	0418111: Discrete Mathematics forComputer Science
Number of credits outside of the department for the next semester:		5
Your second semester:		

FIGURE 42: PLAN FOR THE NEXT SEMESTER

After setting the student to have followed the previous plan:



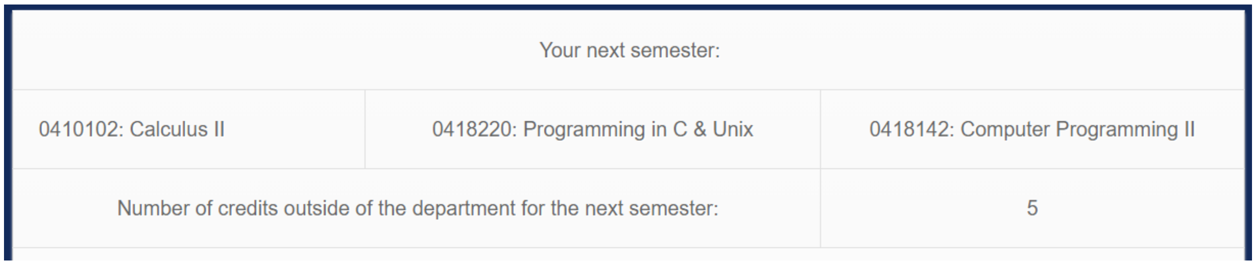
Your current progress is:

Please check all the courses you passed and keep the courses you wish to take unchecked

- ✓ Calculus I
- ☐ Calculus II
- ☐ Linear Algebra
- ✓ Discrete Mathematics forComputer Science
- ✓ Computer Programming I

FIGURE 43: FOLLOWING THE PLAN

The new plan now suggests the same semester that guarantees best progress.



Your next semester:		
0410102: Calculus II	0418220: Programming in C & Unix	0418142: Computer Programming II
Number of credits outside of the department for the next semester:		5

FIGURE 44: PLAN FOR NEXT SEMESTER CHANGED

8. TOOLS AND COMPONENT REUSE

The tools used to help develop the website by documentation the progress, to code, or to communicate between team members included:

- Atom text editor Atom: the open-source text editor was used to write the python codes that built the system and create the local web server for testing and deployment.
- Lucid Charts: to create diagrams to serve the purpose of visualization and understanding the components of the system, activities, and user behavior.
- Visual Paradigm: to create diagrams to serve the purpose of visualization and understanding the components of the system, activities, and user behavior.
- Creately: to create diagrams to serve the purpose of visualization and understanding the components of the system, activities, and user behavior.
- Git and GitHub: to host code without fear of losing it.
- Microsoft Word: to document progress of the project.
- Microsoft Teams and Zoom: to run meetings between team members.
- Apache JMeter: a load testing tool for analyzing and measuring performance of services.

9. CONCLUSIONS AND LESSONS LEARNED

In conclusion, Scheduly is the system this team developed with software engineering methodologies. It strived to solve a problem in today's department of Computer Science in Kuwait University, which we mentioned in the introduction. The web-app developed with the powerful Python programming language. Throughout the development of this web-app the team faced many challenges of all natures, technical or social. From the first phases of analysis and requirements gathering, the team learned how to prioritize the many variables in this scheduling environment of a university. Studying the many cases and interviewing the different type of users showed that for a problem to be solved many outlooks need to be studied and viewed. The designing phases showed the team how to find quicker solutions to bigger problems such as relying on solving smaller problems such as the scheduling of the entire semester is parted to helping scheduling for the students first and then for the instructors to finally reach the supervisor intended to set the schedule and having him be fully informed. During the implementation phases the team learned how powerful new technologies are and how important it is to learn them and make use of them the best way possible. And how during future projects we need to factor the time and effort into learning said technologies would affect the work done, so it should be the case that we make time to learn about new upgrades and evolution in technologies in the field of Computer Science. Other lessons learned is to expect risks of all types and how important it is to be proactive when it comes to those risks and try to be avoid them all together because mitigating them would delay the work, or how to find ways to adapt to unmanageable risks to keep projects going.

The team is honored to have serviced the same department of study that taught them the many skills used in this project.

BIBLIOGRAPHY

AWS Documentation. (2020). Retrieved from Amazon: https://docs.aws.amazon.com/index.html?nc2=h_ql_doc_do

Lightsail Documentation. (2020). Retrieved from https://lightsail.aws.amazon.com/ls/docs/en_us/overview

Makai, M. (2012). Retrieved from https://www.fullstackpython.com/

Norvig, S. R. (2009). *Artificial Intelligence: A Modern Approach.*

Shneiderman, B. (2010). *Designing the User Interface.*

Sommerville, I. (2015). Software Engineering. Retrieved from https://www.geeksforgeeks.org/software-engineering-extreme-programming-xp/

Vincent, W. S. (2018). *Django for APIs: Build web APIs with Python and Django.*