When The Ring Bells

Requirements Specification and Analysis

1.0

10.11.2019

Ersin ÇEBİ
Zafer KALYONCU
Serdar ŞAHİN
Çağatay DEMİRCAN

Prepared for
SE301 Software Engineering

IŞIK UNIVERSITY
COMPUTER
SCIENCE AND
ENGINEERING

Table of Contents

<When The Ring Bells>

REQUIREMENTS ANALYSIS DOCUMENT

1. Introduction

The purpose of this system is to design a attendance application for use in schools and in similar places. Designed system is user-friendly system. The biggest reason for making this system is that the existing polling applications used create many difficulties for students, teachers and other users. Our biggest aim is to minimize these problems in the application we designed and to use the application we designed in schools and other used areas.

1.1. Purpose of the System

This system was designed with the aim of providing easy, credible for teachers who want to take attendance, users can display their classes, and participation percentages, which class their missed and detailed view of classes. It makes the whole process an easy affair without too much hassle as follow a few easy steps.

1.2. Scope of the System

The system serves schools for student teacher and student affairs relations.These three users have some different and some same privileges and features. The system can serve many schools, not a school.

Teachers using the system can easily create student attendance.

At the same time, students can control their participation in classes as a percentage.

1.3. Objectives and Success Criteria of the Project

1- Paper saving
2- Prevent loss of time
3- To allow teachers to take attendance in a simple and comfortable way
4- Ensuring that everyone can see the detail of courses

1.4. Definitions, Acronyms, and Abbreviations

RAD: Requirements Analysis Document

1.5. Overview

When The Ring Bells is an attendance tracker system. Teachers and instructors can track student progress and manage classes. Students can see their own progress and manage their classes at start of semester. Users have a

lot of functions in general but if Student wants to take class, it must be approved by Student Affairs. In When The Ring Bells, students can see their attendance percentages, lectures that they missed or joined. Also teachers can their students attendance percentages. While they taking attendance, system will allow to choose between options about student condition. For example student might be joined, missed or excused. This will help the solve problem of tracking attendance and offers more organized tracking.

2. Current System

Currently, many schools and organizations are taking attendance with the old system or paper. These systems are very difficult for users. For example, minus systems present many difficulties both on the management side and on the student side. The difficulties are usually due to the fact that the interface is poorly designed or program they use does not work efficiently. In addition, many teachers use paper in the classroom because the because there is no system. Of course, there are other side effects, such as wasting paper. As a result, the systems used in many schools are not liked or found to be sufficient by their users.

3. Proposed System

Our online attendance tracking app allows users to see the their class details, system focused on improving teacher and student relations. The admin in the system can accept or reject the request that sent from student affair. Finally, the system allows students to enroll or drop their classes as given time. The online attendance application prevents the waste of unnecessary paper and the time lost in the virtual environment.

3.1. Overview

The system provides many possibilities for the user who is a member of the system. The system is not registered through the web application. Therefore, users, roles and courses are given by the customer to the founders of the system and added to the database.

Students submit requests to register for opened courses.Of course they can do it until the last registration day. if the student enrolls in the course, he / she can see his / her participation in the courses as a percentage.

The teacher sends a request to the student affairs in order to be able to add himself / herself to the course. The teacher can create attendance for the lesson time and see past attendance.

The student affairs may approve or reject the requests of teachers and students

<When The Ring Bells>

3.2. Functional Requirements

In this system ‘when the ring bells’ users who authorized can take attendance, can add and remove classes and class students. Also authorized users can display students percentages of attendance and students can display their attendance percentages per classes.

Admin user; can define user authorization. By that each user that is students or teachers will be registered by admin.

Student affairs; is also can assign users but those users can only be students. Also can open classes. Also student affairs can assign students and teachers into that classes.

Teachers; mainly can take attendance on a class. Also teachers can assign students into classes.

Students; can display their classes.Also students can see their attendance percentages. System shows which date students missed or attended for student.Students can see their classes and their class teacher’s information.

3.3. Nonfunctional Requirements

Usability;The user can do whatever he wants with a minimum of clicks. For example, a teacher spends a lot of time creating polling on existing systems. At the same time, the student spends a lot of time searching for continuity information.

Interface: In this project we used bootstrap to be nice looking and efficient.

Supportability: Operator must be able register new teachers and students without modification to the existing system. Also this system must be able to be maintained and adapted.

Packacing: There will be no constraints delivery. When The Ring Bells is a website.

Reliability: The system must be running 100% of the time when teacher taking attendance.

Implementation: The system will be implemented on Angular CLI it is a typescript framework. In addition, firebase will be used as the database. User Interface should be responsive, bootstrap will be use so it will be smart phone friendly.

Performance: The system must allow least 100 parallel users.

Scenario name: Listing Assigned Classes
Participant actor instances: Ersin**:**Teacher
Flow of events: 1. Ersin opens the application page. 2. Ersin sees the login screen with only two parameters on those are email and password. 3. Ersin enters his email and password, clicks on login button. 4. Ersin redirects to the main page. 5. In the main page Ersin sees the classes that assigned to him on that semester. 6. Ersin sees the logout button, and clicks it. 7. Ersin redirects to login screen.

3.4. System Models

Scenario name: Listing Assigned Classes Students
Participant actor instances: Ersin**:**Teacher
Flow of events: 1. Ersin opens the application page. 2. Ersin sees the login screen with only two parameters on those are email and password. 3. Ersin enters his email and password, clicks on login button. 4. Ersin redirects to the main page. 5. In the main page Ersin sees the classes that assigned to him on that semester. 6. Ersin sees ‘Detail’ button each on every item of class list. 7. Ersin click the ‘Detail’ button of desired class. 8. Ersin redirects to classes page. 9. Ersin sees ‘student list’ in the classes page. Also displays the percentages of each student in the class. 10. Ersin sees the logout button, and clicks it. 11. Ersin redirects to login screen.

Scenario name: Opening Attendance
Participant actor instances: Ersin**:**Teacher
1. Ersin opens the application page. 2. Ersin sees the login screen with only two parameters on those are email and password. 3. Ersin enters his email and password, clicks on login button. 4. Ersin redirects to the main page. 5. In the main page Ersin sees the classes that assigned to him on that semester. 6. Ersin sees ‘Detail’ button each on every item of class list. 7. Ersin click the ‘Detail’ button of desired class. 8. Ersin redirects to classes page. 9. Ersin sees the ‘open attendance’ button and clicks it. 10. Ersin redirect to the related page. 11. In the page Ersin defines the name of attendance and the date of attandance. Clicks ‘Save’ button. 12. Ersin sees the logout button, and clicks it. 13. Ersin redirects to login screen.

<When The Ring Bells>

Scenario name: Taking Attendance
Participant actor instances: Ersin**:**Teacher
1. Ersin opens the application page. 2. Ersin sees the login screen with only two parameters on those are email and password. 3. Ersin enters his email and password, clicks on login button. 4. Ersin redirects to the main page. 5. In the main page Ersin sees the classes that assigned to him on that semester. 6. Ersin sees ‘Detail’ button each on every item of class list. 7. Ersin click the ‘Detail’ button of desired class. 8. Ersin redirects to classes page. 9. In the class page for displaying opened attendance, Ersin clicks the ‘opened attendance’ button and redirects to taking attendance page. 10. In the page Ersin sees three options for every student on the list these are present, absent, excused. 11. Ersin selects the related option for every student. 12. Clicks ‘save’ button for saving the attendance. 13. Ersin sees the logout button, and clicks it. 14. Ersin redirects to login screen.

Scenario name: Sending Request to Student Affair
Participant actor instances: Ersin**:**Teacher
1. Ersin opens the application page. 2. Ersin sees the login screen with only two parameters on those are email and password. 3. Ersin enters his email and password, clicks on login button. 4. Ersin redirects to the main page. 5. Ersin sees the open courses list and clicks on it. 6. Ersin redirect to the classes list, and sees the search bar. 7. Ersin searches the desired course of him. 8. Ersin sees ‘send request’ button each on every course. 9. Ersin clicks and sends request for the desired course. 10. Ersin sees the logout button, and clicks it. 11. Ersin redirects to login screen.

Scenario name: Edit student information
Participant actor instances: Ali: Student affairs
Flow of events: 1. Ali enter the website on his browser. 2. He sees the login screen. 3. The website asks for ali's registered user name and password. 4. He sees the page of student affairs. 5. Ali, click the button of students he can see all students in school and he can change information of students. 6. He press save button after making changes.

Scenario name: Requests section
Participant actor instances: Akif: Student affairs
Flow of events: 1. Akif enter the website on his browser. 2. He sees the login screen. 3. The website asks for akif's registered user name and password. 4. He sees the page of student affairs. 5. Arif, clicks the notification icon on the top right corner. and the request page is displayed. At this page may approve or reject requests.He press save button after making changes. 6. He press save button after making changes.

Scenario name: Opening a course
Participant actor instances: Cem: Student affairs
Flow of events: 1. Cem enter the website on his browser. 2. He sees the login screen. 3. The website asks for cem's registered user name and password. 4. He sees the page of student affairs. 5. He presses the course button on the right. 6. He presses the plus button on this page and comes up with the lessons he can add. Clicks one of these lessons. 7. He press save button after adding lesson.

Scenario name: Adding students and teachers to course
Participant actor instances: Kemal: Student affairs
Flow of events: 1. Kemal enter the website on his browser. 2. He sees the login screen. 3. The website asks for Kemal's registered user name and password. 4. He sees the page of student affairs. 5. He presses the course button on the right. 6. Kemal selects any course from the page that comes in front of him. 7. He click the add button at the bottom left of the course page. 8. He see a list of instructors. he can add students and teachers from this list. 9. He press save button after adding teacher and student to course.

Scenario name: Show Profile
Participant actor instances: Zafer Student
Flow of events: 1. Zafer enter the whentheringbells. 2. Zafer enters the application by typing the e-mail address and password. 3. Zafer sees the profile button at the top of the page. 4. Zafer clicks the profile button to go to the profile page. 5. Zafer sees his profile

Scenario name: Student and teacher deletion scenario
Participant actor instances: Kemal: Student affairs
Flow of events: 1. Serdar enter the website on his browser. 2. He sees the login screen. 3. The website asks for Serdar's registered user name and password. 4. He sees the page of student affairs. 5. He presses the course button on the right. 6. Kemal selects any course from the page that comes in front of him. 7. He click the add button at the bottom left of the course page. 8. He see a list of students. He, will be able to delete the students and teachers on the list. 9. He press save button after adding teacher and student to course.
Scenario name: Viewing the courses the student has taken.
Participant actor instances: Zafer Student

<When The Ring Bells>

Flow of events:

1. Zafer enter the whentheringbells.
2. Zafer enters the application by typing the e-mail address and password.
3. Zafer sees his courses in the middle of the page.

Scenario name: Adding courses(request)

Participant actor instances: Zafer Student

Flow of events:

1. Zafer enter the whentheringbells.
2. Zafer enters the application by typing the e-mail address and password.
3. Zafer sees the courses button at the next to the profile button.
4. Zafer clicks the courses button to go to the courses page.
5. Zafer sees all the courses opened in the school.
6. Zafer clicks the send request button next to the course.

Scenario name: Percentage attendance

Participant actor instances: Zafer Student

Flow of events:

1. Zafer enter the whentheringbells.
2. Zafer enters the application by typing the e-mail address and password.
3. Zafer sees his courses in the middle of the page.

4. Zafer sees his **P**ercentage attendance under the course.

Scenario name: drop course
Participant actor instances: Zafer Student
Flow of events: 1. Zafer enter the whentheringbells. 2. Zafer enters the application by typing the e-mail address and password. 3. Zafer sees his courses in the middle of the page. 4. Zafer click the course which his want to drop 5. Zafer sees the course and click drop button

Scenario name missed lessons
Participant actor instances: Zafer Student
Flow of events: 1. Zafer enter the whentheringbells. 2. Zafer enters the application by typing the e-mail address and password. 3. Zafer sees his courses in the middle of the page. 4. Zafer click the course which his want to see missed lesson 5. Zafer sees the missed lessons

Scenario name **adminApprovesChanges**
Participation Actor instances **john:Admin, alice:StudentAffairs**
Flow of Events 1. **John enters the website.** 2. **John sees login screen.** 3. **John clicks log in button.**

4. **John logs in.**
5. **John clicks requests button.**
6. **John selects change requests.**
7. **John sees requests.**
8. **John selects the change.**
9. **John sees information about the change and message from Alice.**
10. **John clicks approve button.**
11. **Website sends a notification to Alice.**

Scenario name	**adminApprovesRemoveStudent**
Participation Actor instances	**john:Admin, alice:StudentAffairs**

Flow of Events

1. **John enters the website**
2. **John sees login screen.**
3. **John clicks log in button.**
4. **John logs in.**
5. **John clicks requests button.**
6. **John selects approve requests.**
7. **John selects students.**
8. **John sees the requests.**
9. **John selects the request.**
10. **John sees information about request that gives information about student and class.**
11. **John approves request.**

12. **John clicks manage class button.**
13. **John selects the class.**
14. **John clicks delete a student.**
15. **John selecst the student.**
16. **Website sends a notification to Alice.**

Scenario name	**adminApprovesDeleteStudent**
Participation Actor instances	**john:Admin, alice:StudentAffairs**

Flow of Events

1. **John enters the website**
2. **John sees login screen.**
3. **John clicks log in button.**
4. **John logs in.**
5. **John clicks requests button.**
6. **John selects delete requests.**
7. **John sees the delete requests.**
8. **John selects the request.**
9. **John sees information about request that gives information about student and message from Alice.**
10. **John clicks approve button.**
11. **John click delete user button.**
12. **John selects the student.**
13. **Website sends a notification to John.**

Scenario name **adminApprovesClassOfTeacher**
Participation Actor instances **john:Admin, alice:StudentAffairs**
Flow of Events 1. **John enters the website** 2. **John sees login screen.** 3. **John clicks log in button.** 4. **John logs in.** 5. **John clicks requests button.** 6. **John selects approve requests.** 7. **John selects teachers.** 8. **John sees the requests.** 9. **John selects the request.** 10. **John sees information about request that gives information about teacher and class.** 11. **John clicks approve button.** 12. **John clicks manage class button.** 13. **John selects the class.** 14. **John clicks add a teacher.** 15. **John enters the teacher's information.** 16. **Website sends a notification to Alice.**

Scenario name **adminApprovesClassOfStudent**
Participation Actor instances **john:Admin, alice:StudentAffairs**
Flow of Events 1. **John enters the website** 2. **John sees login screen.** 3. **John clicks log in button.** 4. **John logs in.** 5. **John clicks requests button.** 6. **John selects approve requests.** 7. **John selects students.** 8. **John sees the requests.** 9. **John selects the request.** 10. **John sees information about request that gives information about student and class.** 11. **John approves request.** 12. **John clicks manage class button.** 13. **John selects the class.** 14. **John clicks add a student.** 15. **John enters the student's information.** 16. **Website sends a notification to Alice.**

<When The Ring Bells>

Use Cases

Use case name: Listing Assigned Classes
Participant actor instances: İnitiated by Teacher
Flow of Events: 1. Opening the application page. 2. Application responded by showing login screen. 3. Application asks teachers email and password. 4. Teacher enters the related info's to the text-boxes. 5. Clicks to ‘login’ button to submitting. 6. System checks for the related data from database. 7. System redirecting teacher to main page and loads the classes data in to main page.
Entry Condition: User opens the application page. Clicks the login button on login page. User has not to be registered.
Exceptional Cases: **Invalid Character:** If the App detects invalid characters into the text-boxes or not filled any text-box, it warns the User about the entering invalid characters and wants to retry from the User by displaying alert.

Use case name: Listing Students in The Assigned Class
Participant actor instances: İnitiated by Teacher
Flow of Events: 1. Opening the application page. 2. Application responded by showing login screen. 3. Application asks teachers email and password. 4. Teacher enters the related info's to the text-boxes. 5. Clicks to ‘login’ button to submitting. 6. System checks for the related data from database. 7. System redirecting teacher to main page and loads the classes data in to main page. 8. Teacher clicks the ‘Details’ button of required class and system takes the ID of the class, redirects the detail page and load the related data of class. 9. For finding classes students system uses ID for related tables for filtering students and lists.
Entry Condition: If required class has no enrolled student NULL data will return.
Exceptional Cases: **User Session:** If the proces cant be done in the required time, session will be expired and user will be redirected to the login page.

Use case name: Oppening Attandance
Participant actor instances: İnitiated by Teacher
Flow of Events: 1. Opening the application page. 2. Application responded by showing login screen. 3. Application asks teachers email and password. 4. Teacher enters the related info's to the text-boxes. 5. Clicks to ‘login’ button to submitting. 6. System checks for the related data from database. 7. System redirecting teacher to main page and loads the classes data in to main page. 8. Teacher clicks the ‘Details’ button of required class and system takes the ID of the class, redirects the detail page and load the related data of class. 9. Teacher clicks ‘open attendance’ button and system redirects the teacher into attendance creating page and the fields that needs to be filled.
Quality Requirements: If teacher leaves required fields empty, system displays a warning message, like "This area cannot be empty.”
Exceptional Cases: **User Session:** If the process cant be done in the required time, session will be expired and user will be redirected to the login page.

<When The Ring Bells>

Use case name: Taking Attandance
Participant actor instances: İnitiated by Teacher
Flow of Events: 1. Opening the application page. 2. Application responded by showing login screen. 3. Application asks teachers email and password. 4. Teacher enters the related info's to the text-boxes. 5. Clicks to ‘login’ button to submitting. 6. System checks for the related data from database. 7. System redirecting teacher to main page and loads the classes data in to main page. 8. Teacher clicks the ‘Details’ button of required class and system takes the ID of the class, redirects the detail page and load the related data of class. 9. Teacher clicks ‘opened attendance’ button and system redirects the teacher into attendance list page and loads the student list with it. 10. Teacher selects options(present, absent, excused) for each student and clicks save. 11. System takes the submitted data, saves it into attendance table, each tuple with related student.
Quality Requirements: If teacher leaves required fields empty, system displays a warning message, like "This area cannot be empty.”
Exceptional Cases: **User Session:** If the proces cant be done in the required time, session will be expired and user will be redirected to the login page.

Use case name: Sending Request to Student Affair
Participant actor instances: İnitiated by Teacher
Flow of Events: 1. Opening the application page. 2. Application responded by showing login screen. 3. Application asks teachers email and password. 4. Teacher enters the related info's to the text-boxes. 5. Clicks to ‘login’ button to submitting. 6. System checks for the related data from database. 7. System redirecting teacher to main page and loads the classes data in to main page. 8. Teacher clicks the ‘open courses’ button, system redirects to the page and loads the related data. 9. Teacher clicks the ‘send reques’ for desired course. 10. System saves the request into ‘requests’ table.
Exceptional Cases: **User Session:** If the proces cant be done in the required time, session will be expired and user will be redirected to the login page.

Use case name: Edit student information
Participant actors: Student affairs
Flow of events: 1. User open the website. User see login page. 2. The user enters email and password to the login fields. 3. If the email and password match the email and password stored in the database, they are authenticated. 4. The system shows the user the categories of student affairs. 5. User clicks the list button where all students in the school appear. 6. System displays all users in the database. 7. User changes the list and presses the save button. 8. The system saves the new list to the database.
Entry Condition: User opens the website. Login with username and password. User have to login as student affairs and user clicks list button.
Exit Condition: After the user has made the change, press the save button and it should be saved to the database correctly.
Quality Requirements:
Exceptional Cases: 1. If the user enters the user name or password incorrectly, system display error message "please check your username or password". 2. If a database error occurs, it indicates a system error, such as “An error has occurred, please try again”.

Use case name: Requests section
Participant actors: Student affairs
Flow of events: 1. User open the website. User see login page. 2. The user enters email and password to the login fields. 3. If the email and password match the email and password stored in the database, they are authenticated. 4. The system shows the user the categories of student affairs. 5. User, click the notification icon on the top right. 6. System displays all request in the database. 7. At this page user may approve or reject requests. 8. User presses the save button. 9. The system saves the new list to the database.
Entry Condition: User opens the website. Login with username and password. User have to login as student affairs and see the notification icon in the top right.
Exit Condition: After the user has made the confirmation, press the save button and it should be saved to the database correctly.
Quality Requirements:
Exceptional Cases: 1. If the user enters the user name or password incorrectly, system display error message "please check your username or password". 2. If a database error occurs, it indicates a system error, such as “An error has occurred, please try again”.

Use case name: Opening a course

Participant actors: Student affairs

Flow of events:

1. User open the website. User see login page.
2. The user enters email and password to the login fields.
3. If the email and password match the email and password stored in the database, they are authenticated.
4. The system shows the user the categories of student affairs.
5. User, clicks the course button on the right .
6. System displays all courses in the database.
7. User, if wants to add a new course, he presses the plus button to add new courses. Type the name of the course and press the save button.
8. The system saves the new list to the database.

Entry Condition: User opens the website. Login with username and password. User have to login as student affairs and clicks the course button on the right.

Exit Condition: After the user has made the confirmation, press the save button and it should be saved to the database correctly.

Quality Requirements:

Exceptional Cases:

1. If the user enters the user name or password incorrectly, system display error message "please check your username or password".
2. If a database error occurs, it indicates a system error, such as “An error has occurred, please try again”.
3. If the course you are trying to add has already been added. system error such as"This course has already been added".

Use case name: Adding students and teachers to course

Participant actors: Student affairs

Flow of events:

1. User open the website. User see login page.
2. The user enters email and password to the login fields.
3. If the email and password match the email and password stored in the database, they are authenticated.
4. The system shows the user the categories of student affairs.
5. User, clicks the course button on the right .
6. System displays all courses in the database.
7. User, clicks one of the courses in front of him.
8. System draws teachers and students from the database if any.
9. User, presses edit button.
10. System, pulls the entire list of students and teachers from the database.
11. User, presses the save button after adding the desired user.
12. The system saves the new list to the database.

Entry Condition: User opens the website. Login with username and password. User have to login as student affairs and clicks the course button on the right.

Exit Condition: After the user has made the confirmation, press the save button and it should be saved to the database correctly.

Quality Requirements:

Exceptional Cases:

1. If the user enters the user name or password incorrectly, system display error message "please check your username or password".
2. If a database error occurs, it indicates a system error, such as “An error has occurred, please try again”.
3. If the teacher you are trying to add has already been added. system error such as"This teacher has already been added".

4. If the student you are trying to add has already been added. system error such as"This student has already been added".

Use case name: Student and teacher deletion scenario

Participant actors: Student affairs

Flow of events:

1. User open the website. User see login page.
2. The user enters email and password to the login fields.
3. If the email and password match the email and password stored in the database, they are authenticated.
4. The system shows the user the categories of student affairs.
5. User, clicks the course button on the right .
6. System displays all courses in the database.
7. User, clicks one of the courses in front of him.
8. System draws teachers and students from the database if any.
9. User, presses edit button.
10. System, pulls the entire list of students and teachers from the database.
11. User, presses the save button after deleting the desired user.
12. The system saves the new list to the database.

Entry Condition: User opens the website. Login with username and password. User have to login as student affairs and clicks the course button on the right.

Exit Condition: After the user has made the confirmation, press the save button and it should be saved to the database correctly.

Quality Requirements:

Exceptional Cases:

1. If the user enters the user name or password incorrectly, system display error message "please check your username or password".
2. If a database error occurs, it indicates a system error, such as “An error has occurred, please try again”.
3. If there is no teacher you are trying to delete. System error like "This teacher doesn't already exist".

4. If there is no student you are trying to delete. System error like "This student doesn't already exist".

Use case name: Student add course(request)
Participant actors: Student
Flow of events: 9. Student open the web application and see login page 10. Student enter his/her email and password click the login button 11. Student click courses button on top side 12. System show courses on firebase database . 13. Student send request for any course 14. System check this request date 15. System sending request which student’s request, to student affairs
Entry Condition: The Student is loggen into WhenTheRingBells
Exit Condition: if Student successfully spend request or if the time for adding courses has passed..
Quality Requirements: - If Student sending request which sended request same course, system displays a warning message, like " request already sent.”

Use case name: ListStudentsCourse
Participant actors: Student
Flow of events: 1. Student open the web application and see login page 2. Student enter his/her email and password click the login button. 3. App responds by showing the main page. 4. App take course on database and shows students his/her courses on the middle of page . 5. The student sees the courses she/he has taken
Entry Condition: The Student is loggen into WhenTheRingBells
Exit Condition: Student successfully inspects his/her courses according to database datas.

Use case name: Percentage attendance
Participant actors: Student
Flow of events: 1. Student open the web application and see login page 2. Student enter his/her email and password click the login button. 3. App responds by showing the main page and student’s courses. 4. The system shows the student the student's attendance rate with the attendance of the teacher.

<When The Ring Bells>

Entry Condition: The Student is loggen into WhenTheRingBells
Exit Condition: Student successfully inspects his/her Percentage attendance according to database datas.

Use case name: Show profile
Participant actors: Student
Flow of events: 1. Student open the web application and see login page 2. Student enter his/her email and password click the login button 3. Student click profile button on top side 4. System show student information on database . 5. Student send request for any course 6. System check this request date 7. System sending request which student's request, to student affairs
Entry Condition: The Student is loggen into WhenTheRingBells
Exit Condition: Student successfully inspects his/her profile according to database datas.

Use case name: Drop Course
Participant actors: Student
Flow of events: 1. Student open the web application and see login page 2. Student enter his/her email and password click the login button. 3. App responds by showing the main page. 4. App take course on database and shows students his/her courses on the middle of page . 5. The student sees the courses she/he has taken 6. Student click course which he/she want to drop 7. Student sees course information and click the top side drop course 8. System delete student's course on database
Entry Condition: The Student is loggen into WhenTheRingBells
Exit Condition: Student successfully drop course or

the time for drop courses has passed..

Use case name: MissedLesson
Participant actors: Student
Flow of events: 1. Student open the web application and see login page 2. Student enter his/her email and password click the login button. 3. App responds by showing the main page. 4. App take course on database and shows students his/her courses on the middle of page . 5. The student sees the courses she/he has taken 6. Student click course which he/she want to see missed lesson 7. System show course informations which system take database. 8. Student sees her/his missed lessons
Entry Condition: The Student is loggen into WhenTheRingBells
Exit Condition: Student successfully inspects his/her profile missed lesson to database datas.

Use case name: Admin Approves Change
Participant actors: Initiated by Admin
Flow of events: 1. Admin sees the request. 2. Admin clicks the request button. 3. Admin chooses the request. 4. Admin sees detailed information about the change request. 5. Admin approves the change. 6. Student Affairs is notified when Admin approves the change.
Entry condition: Admin is logged into When The Ring Bells. Admin clicks request page button.
Exit conditions: When the system sends notification to Student Affairs, **OR** Student affairs receives an explanation indicating why Admin did not approved the change request.
Quality requirements: Student Affairs gets respond within 1 day. Admin sees detailed infomation about changes and when is the dead line of request.

Use case name: Admin Approves Remove Student
Participant actors: Initiated by Admin
Flow of events: 1. Admin sees the request. 2. Admin clicks the request button. 3. Admin sees detailed information about student and class. 4. Admin clicks the manage class button. 5. Admin clicks the class. 6. Admin finds the student. 7. Admin deletes the student. 8. Student gets removed by class. 9. System sends notification to Student Affairs.
Entry condition: Admin is logged into When The Ring Bells. Admin clicks request page button.
Exit conditions: When the system sends notification to Student Affairs, **OR** Student affairs receives an explanation indicating why Admin did not approved the removing student.
Quality requirements: Students removed by class within 1 day. System holds the data of student's attandance until student gets removed.

Use case name: Admin Approves Delete Student
Participant actors: Initiated by Admin
Flow of events: 1. Admin sees the request 2. Admin clicks the request button. 3. Admin sees detailed information about student. **4.** Admin clicks delete user button and find student name, surname, id and classes. 5. Admin confirms to delete student and delete the user. 6. Student gets deleted from classes and lists. 7. System sends notification to Student Affairs.
Entry condition: Admin is logged into When The Ring Bells. Admin clicks request page button.
Exit conditions: When the system sends notification to Student Affairs, **OR** Student affairs receives an explanation indicating why Admin did not approved the change request.
Quality requirements: Student gets deleted from data base. Student can not login after get deleted.

Use case name: Admin Approves Class of Student
Participant actors: Initiated by Admin
Flow of events: 1. Admin sees the request. 2. Admin clicks the request button. 3. Admin clicks the request. 4. Admin sees detailed informatin that contains student's name, surname, id and class name and id. 5. Admin clicks name of class. 6. Admin clicks the manage class button. 7. Admin enters the student's name, surname and id. 8. Admin clicks confirm. 9. System sends notification to Student Affairs.
Entry condition: Admin is logged into When The Ring Bells. Admin clicks request page button.
Exit conditions: When the system sends notification to Student Affairs, **OR** Student affairs receives an explanation indicating why Admin did not approved the change request.
Quality requirements: Student gets added to class list within 1 min after Admin approves change. Student's schedule upgraded within 1 min.

Use case name: Admin Approves Class of Teacher

Participant actors: Initiated by Admin

Flow of events:

1. Admin sees the request.
2. Admin clicks the request button.
3. Admin clicks the request.
4. Admin sees detailed informatin that contains teacher's name, surname, id and class name and id.
5. Admin clicks name of class.
6. Admin clicks the manage class button.
7. Admin clicks add teacher button.
8. Admin enters the teacher's name, surname and id
9. Admin clicks confirm.
10. System sends notification to Student Affairs and Teacher

Entry condition: Admin is logged into When The Ring Bells.
Admin clicks request page button.

Exit conditions:

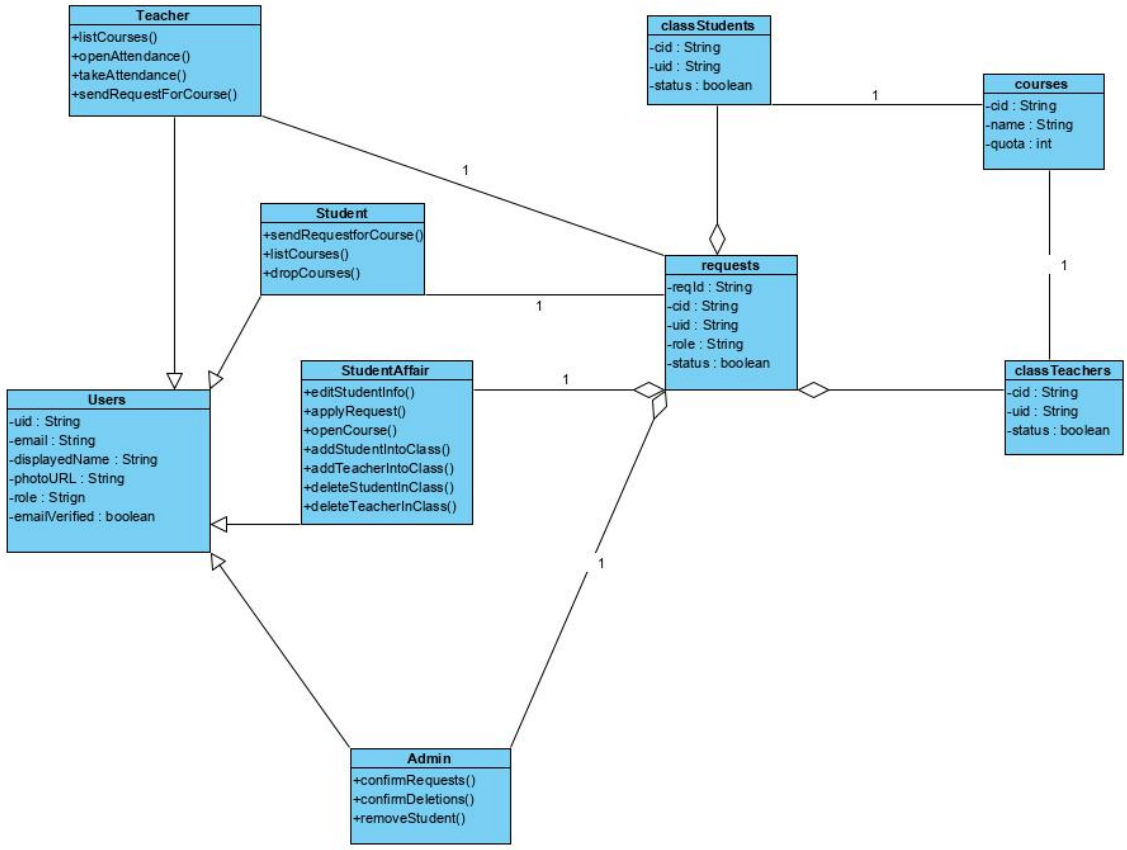
When the system sends notification to Student Affairs and Teacher, **OR**
Teacher or Student affairs receives an explanation indicating why Admin did not approved the change request.

Quality requirements:

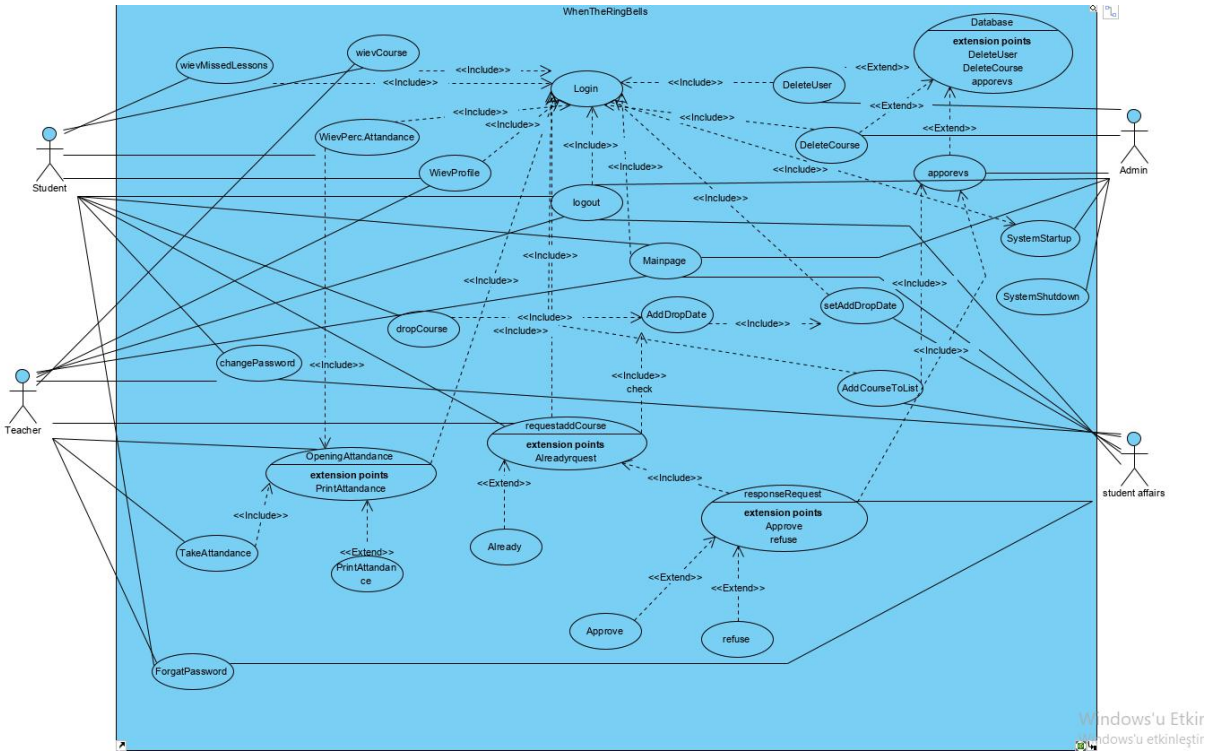
Teacher gets added to class list within 1 min after Admin approves change.
Teacher's class list upgraded within 1 min.
Teachers can not be add to class if this class already have teacher.

<When The Ring Bells>

Object model

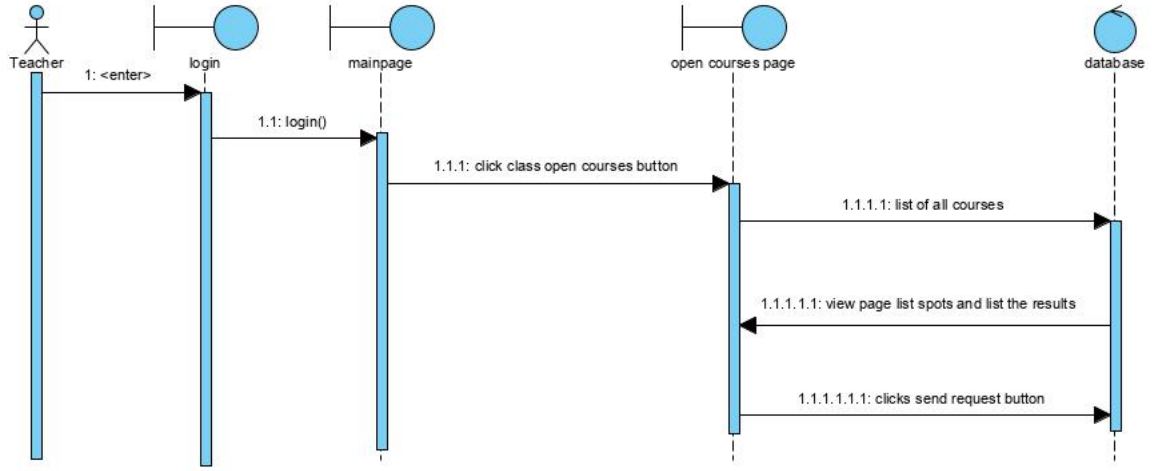


UseCaseDiagram

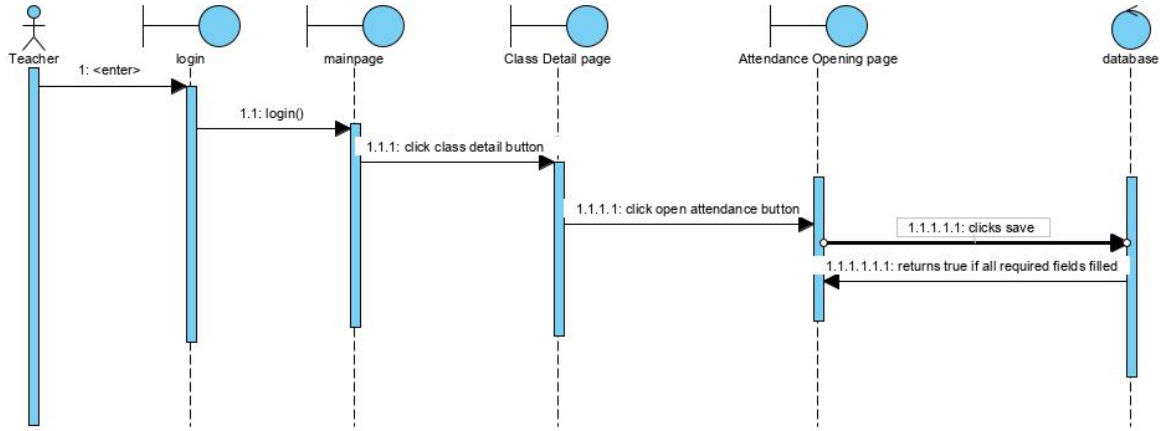


<When The Ring Bells>

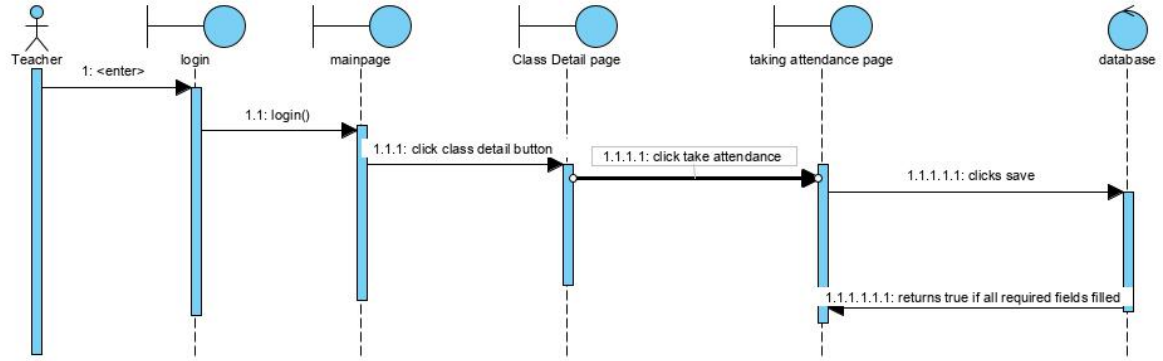
Model: Opening The Open Courses Page



Model: Seeing Class Detail And Student List Page

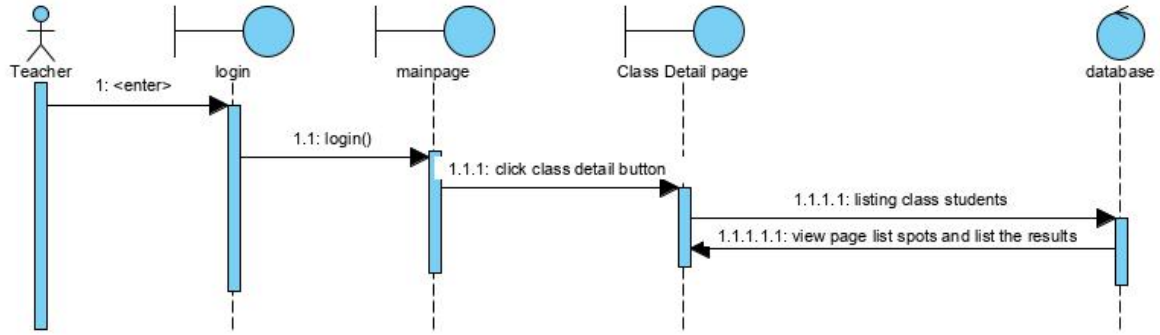


Model: Opening New Attendance

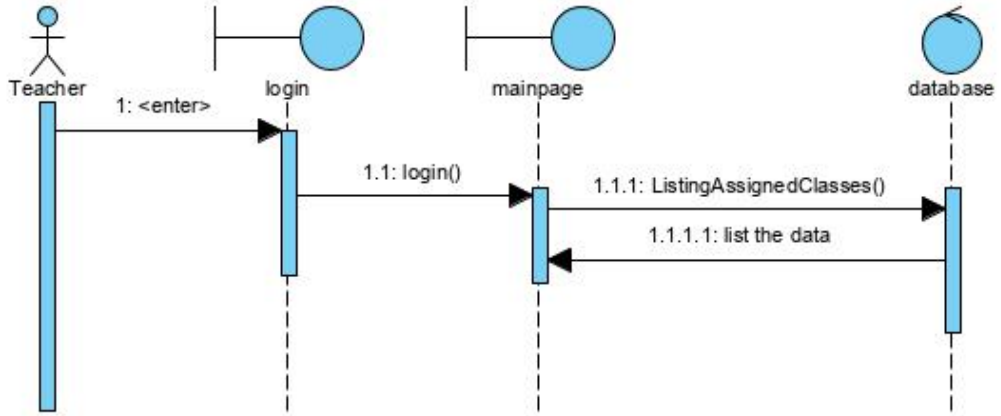


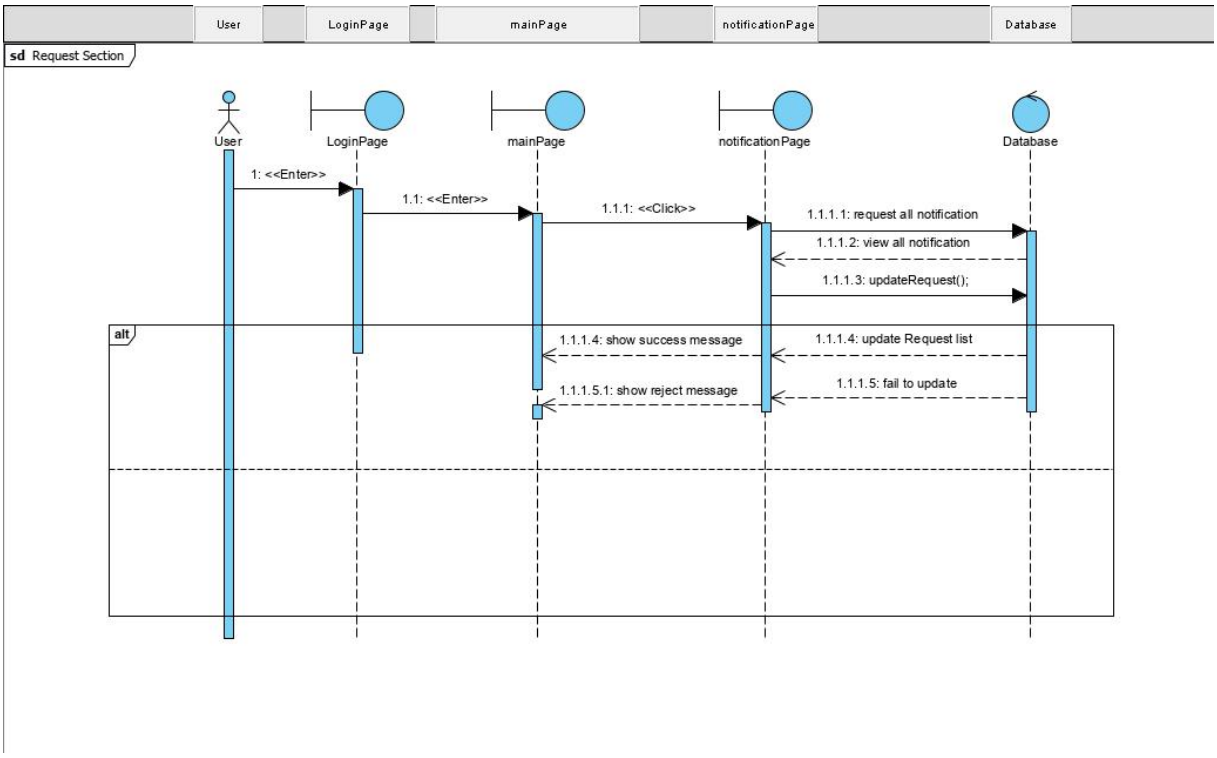
<When The Ring Bells>

Model: Listing Student In Desired Class

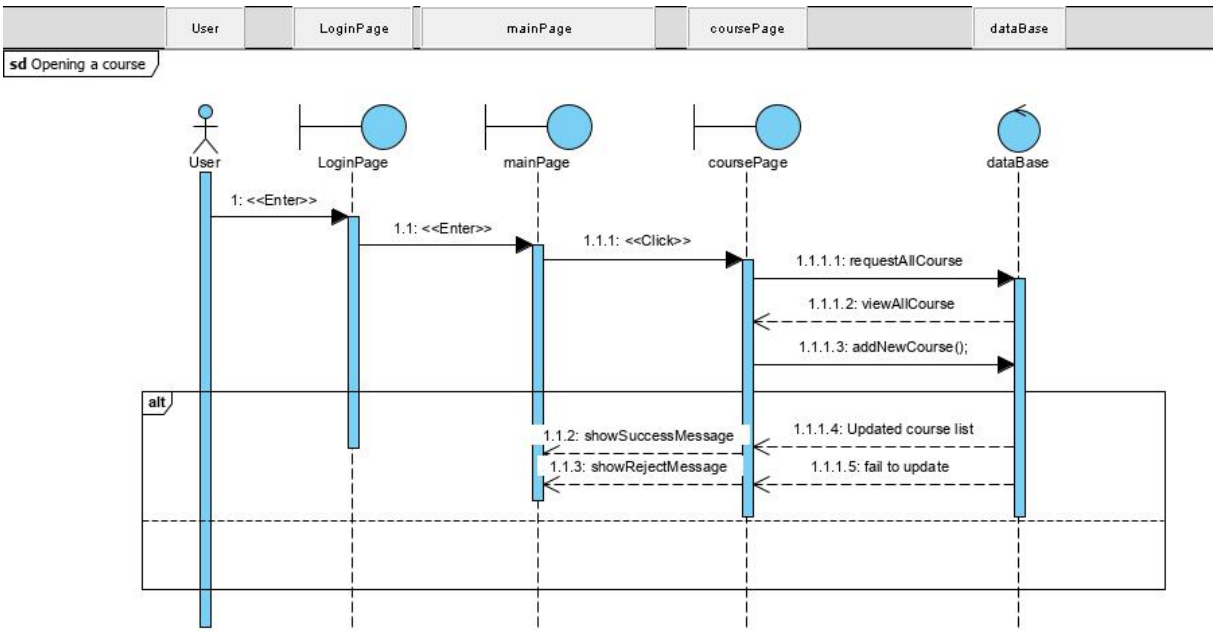


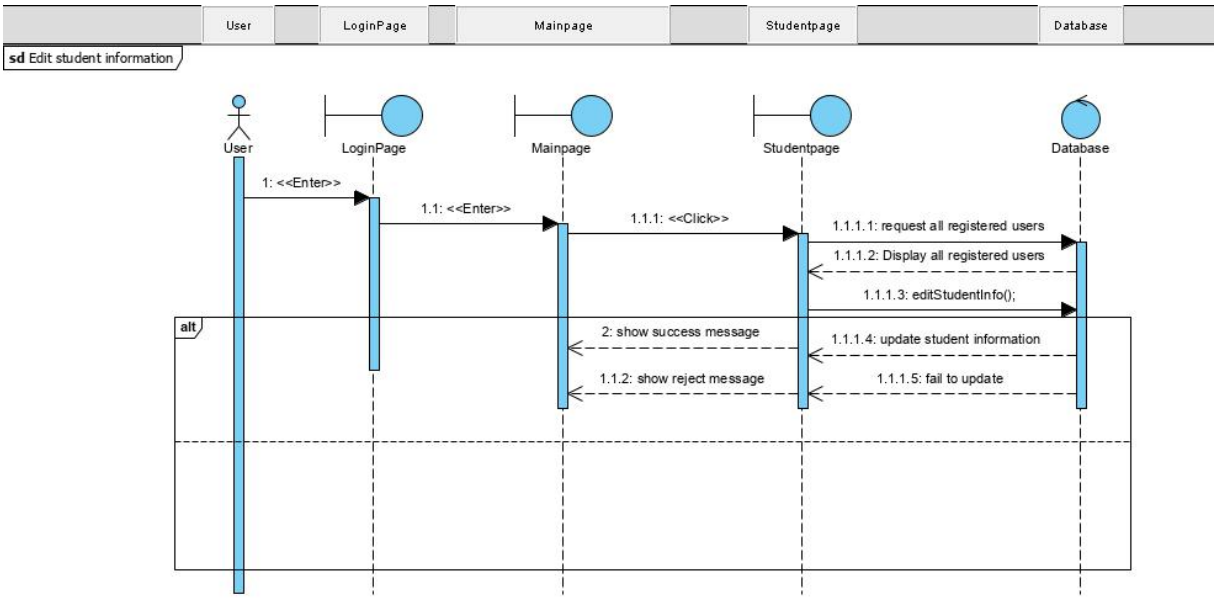
Model: Listing Assigned Classes Of Logged Teacher



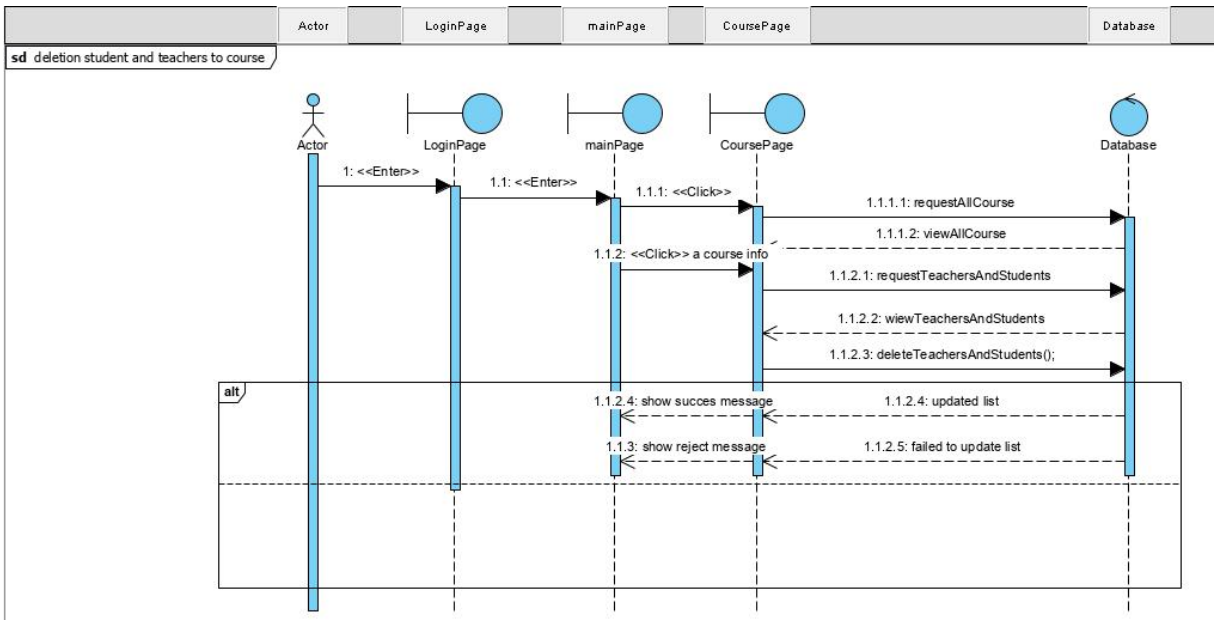


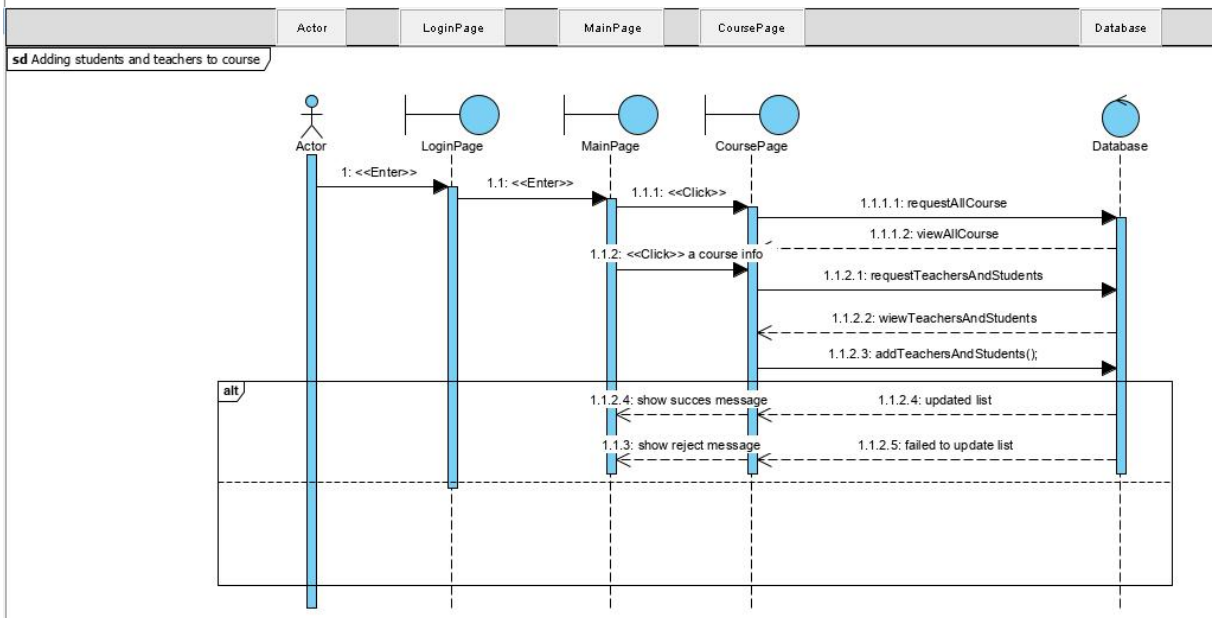
<When The Ring Bells>





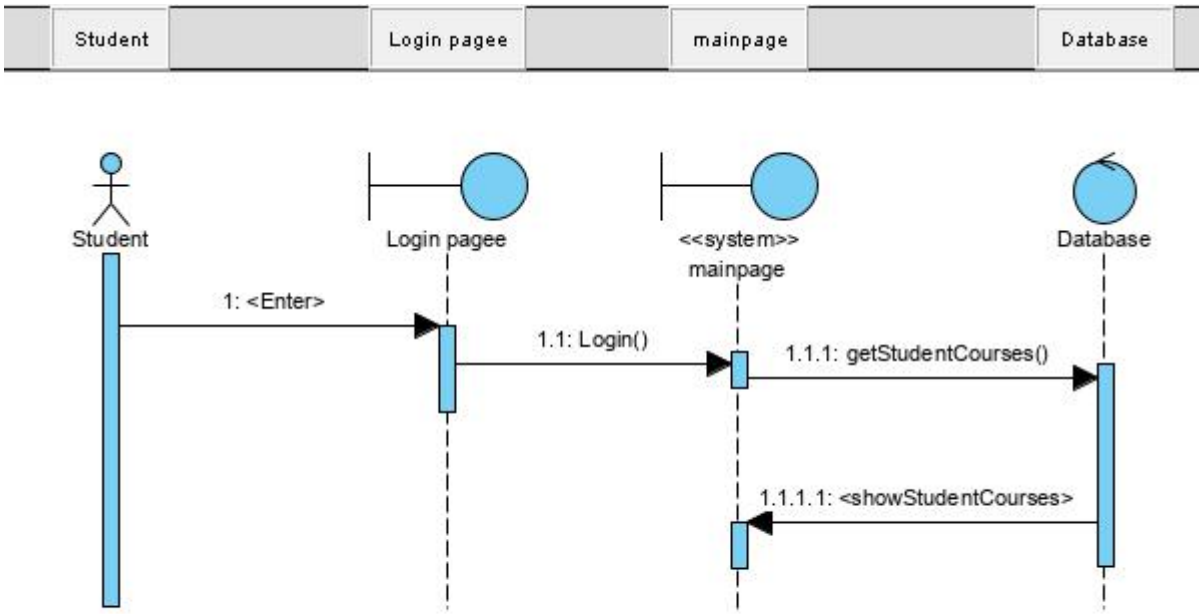
<When The Ring Bells>



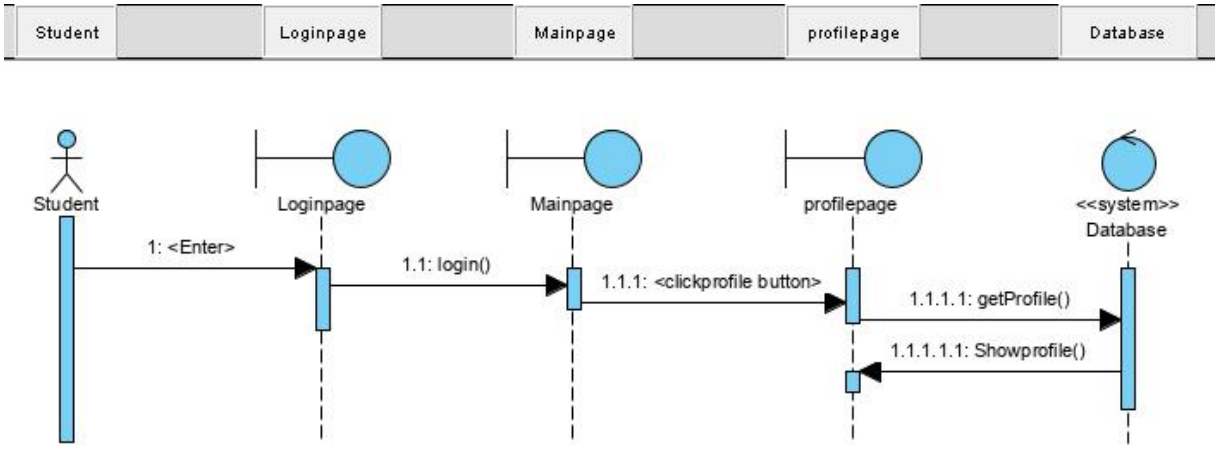


<When The Ring Bells>

Model : List Student’s Courses

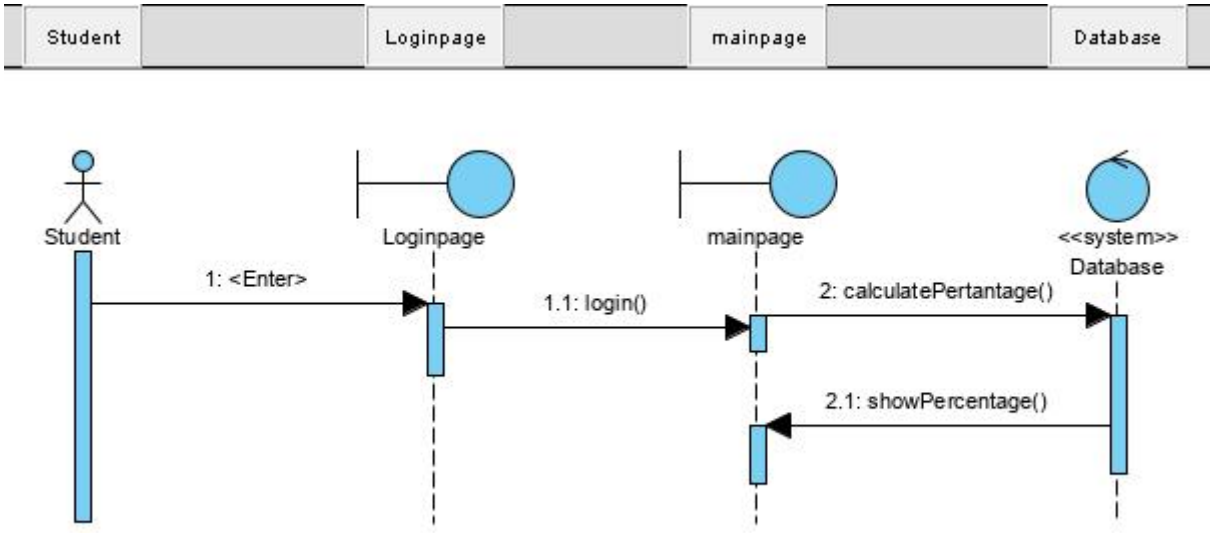


Model : Student Show Profile

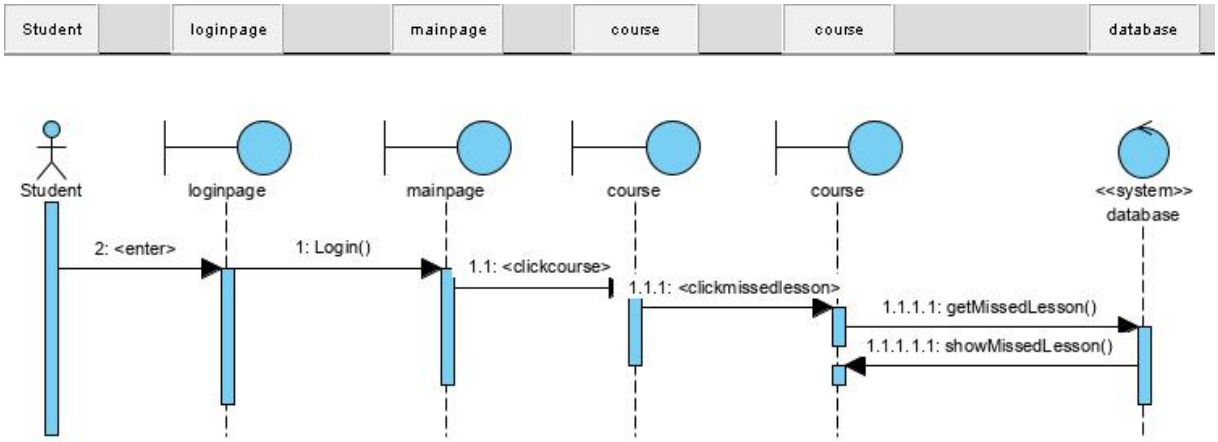


<When The Ring Bells>

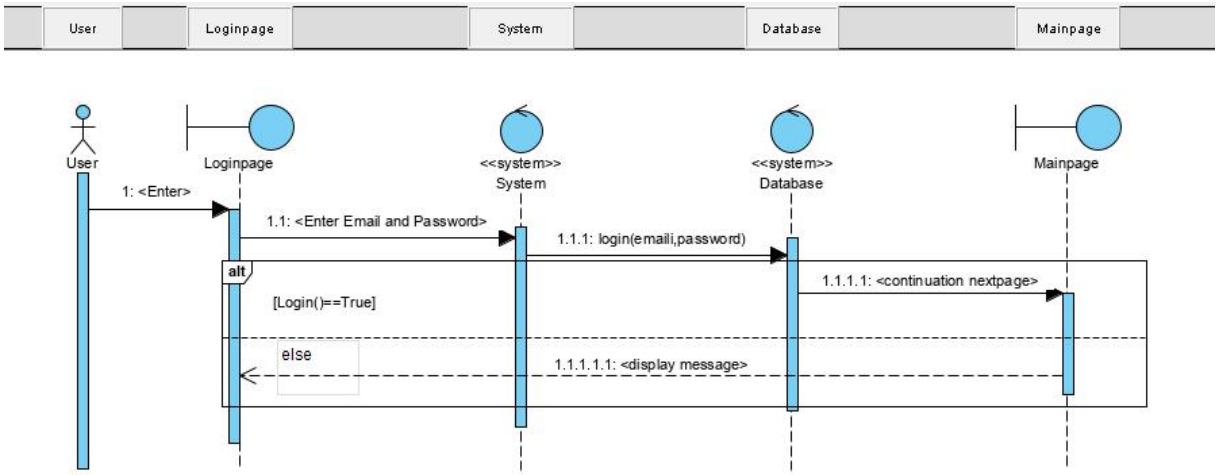
Model :Show Percentage Attandence



Model : Missed Lessons

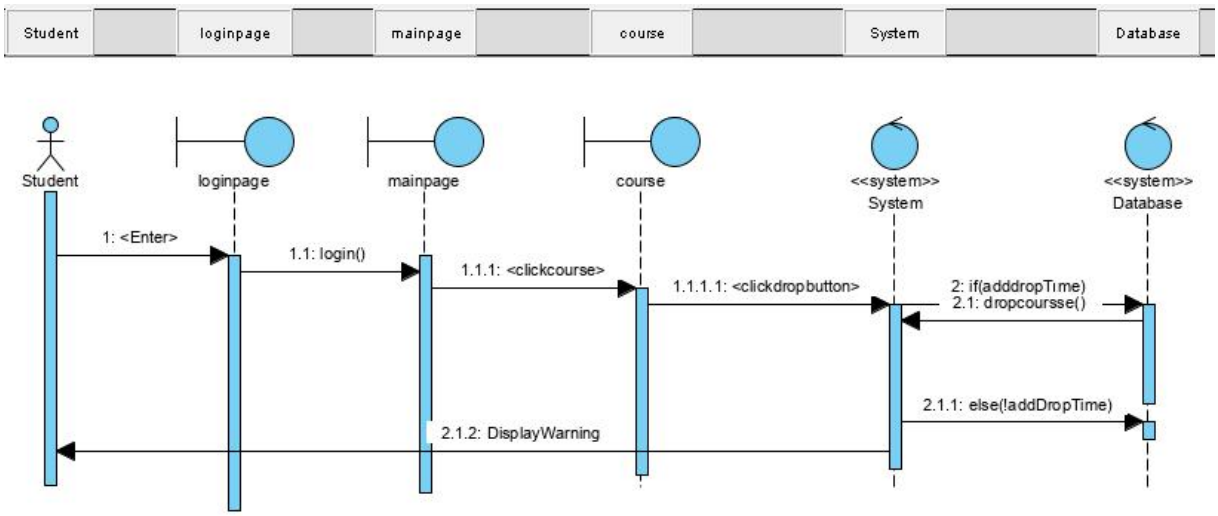


Model : Login

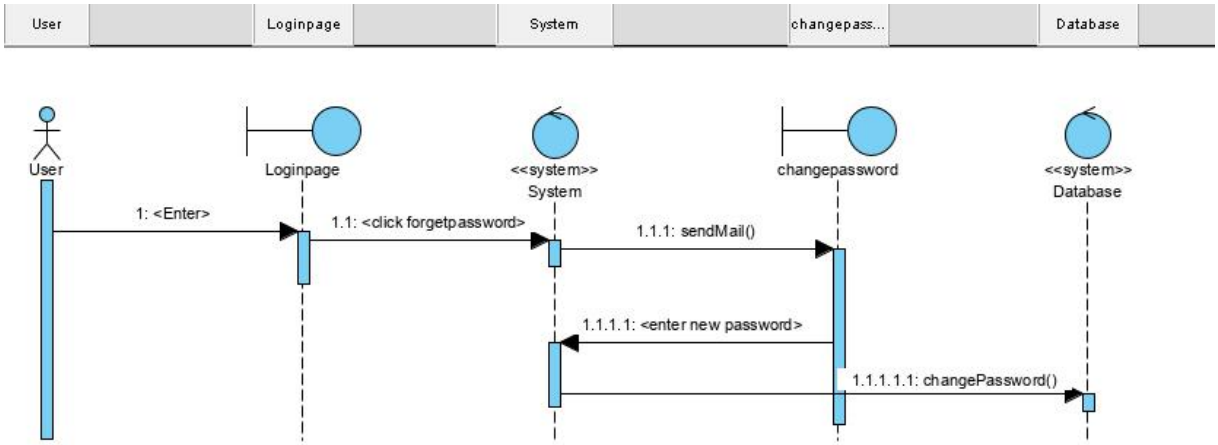


<When The Ring Bells>

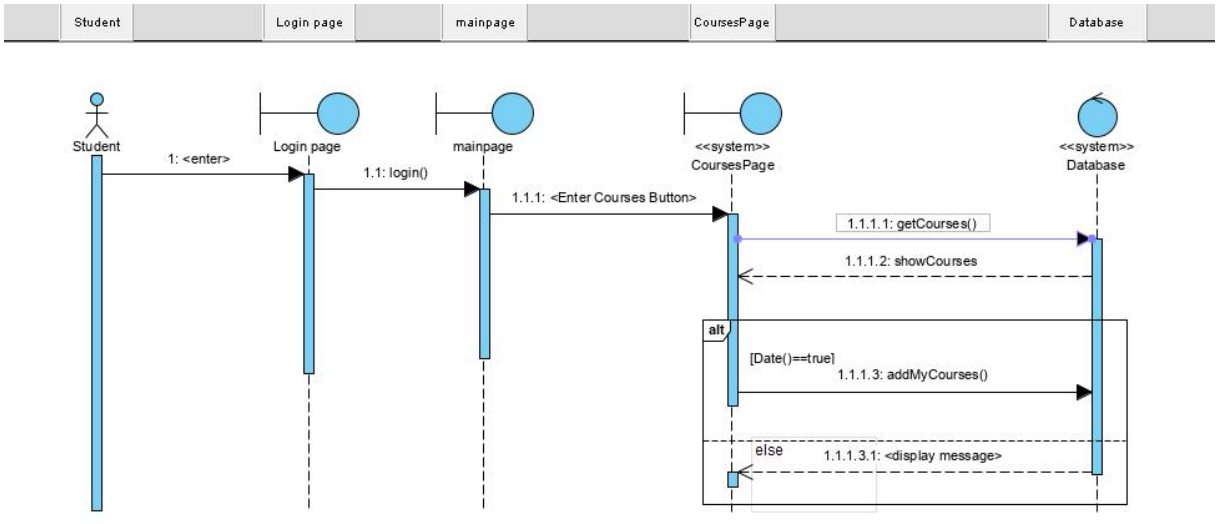
Model : Student Drop Course



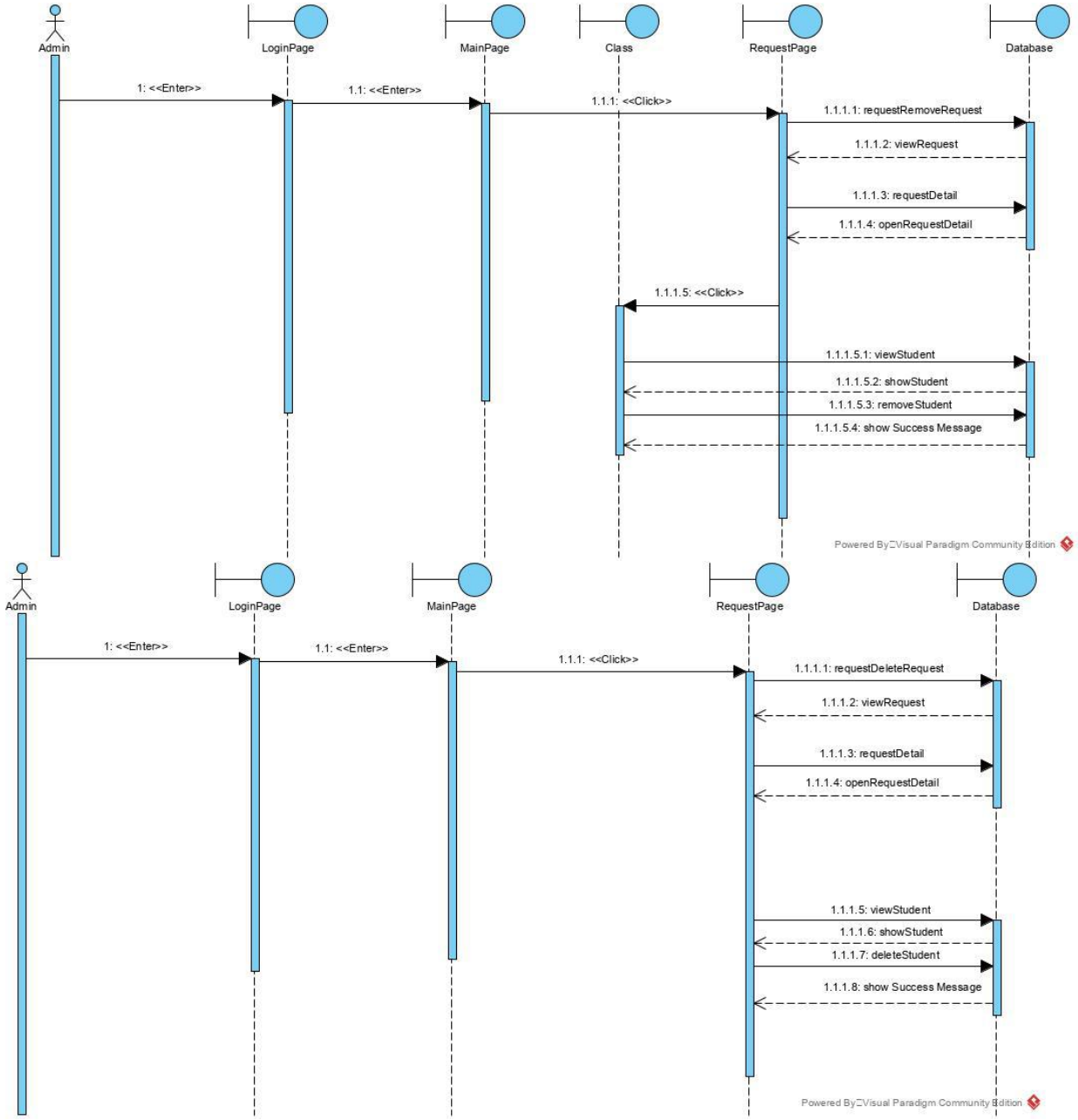
Model : Forget Password



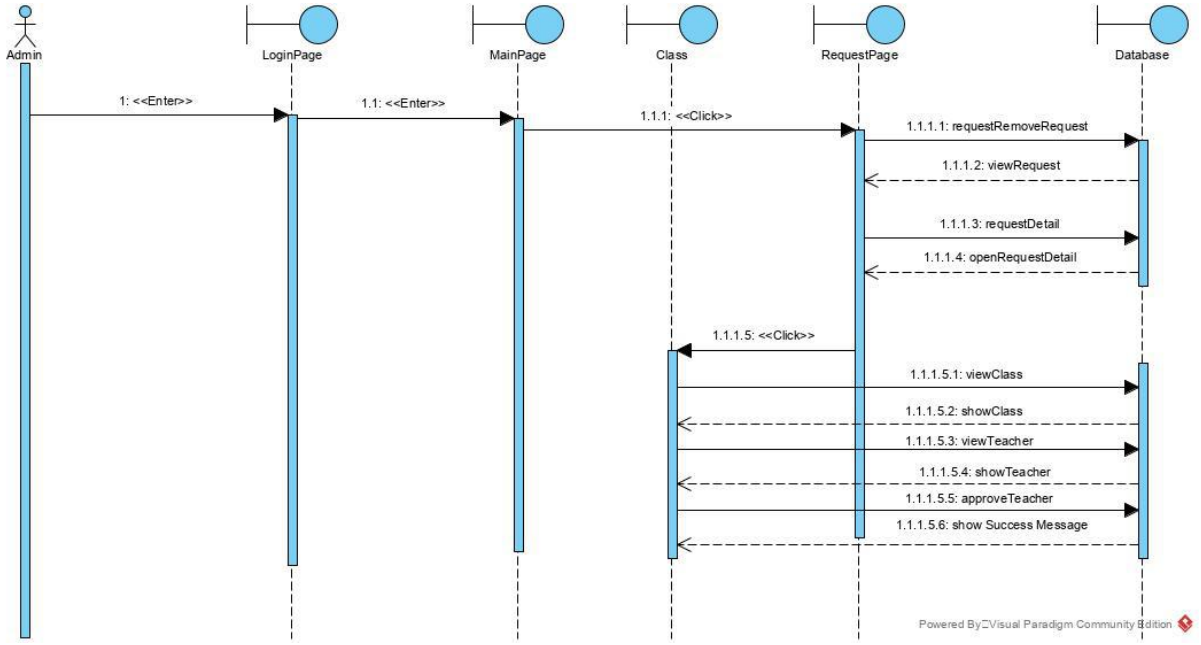
Model : Student Add course



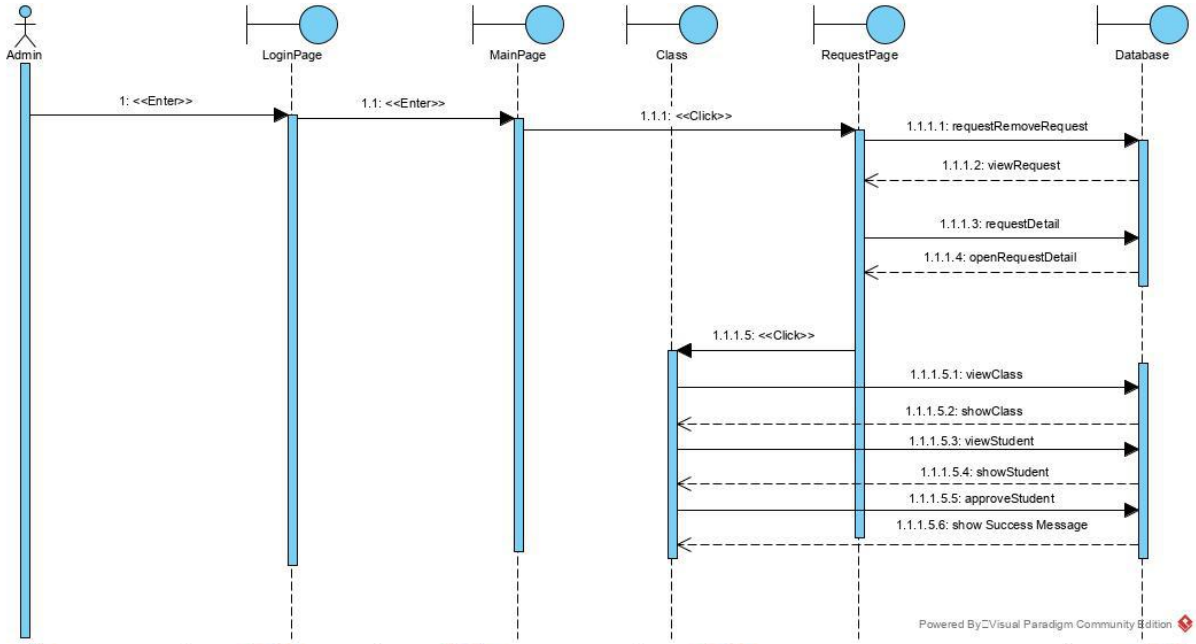
<When The Ring Bells>

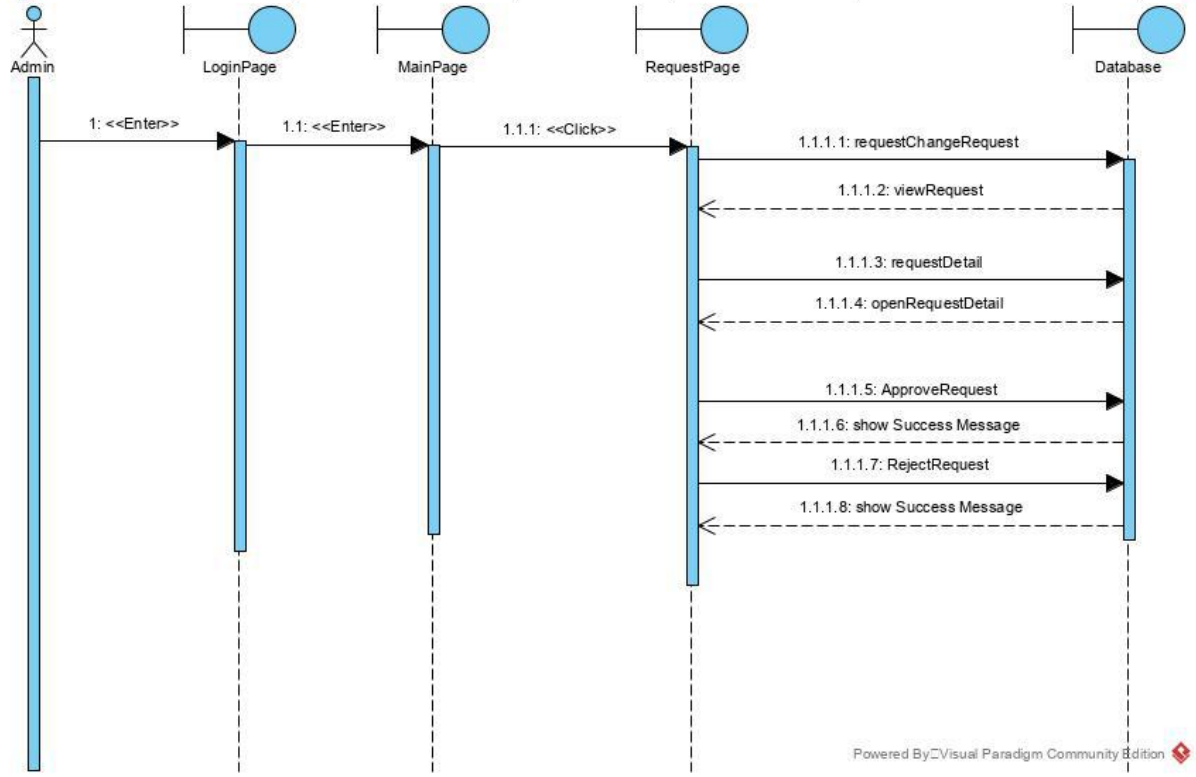


<When The Ring Bells>

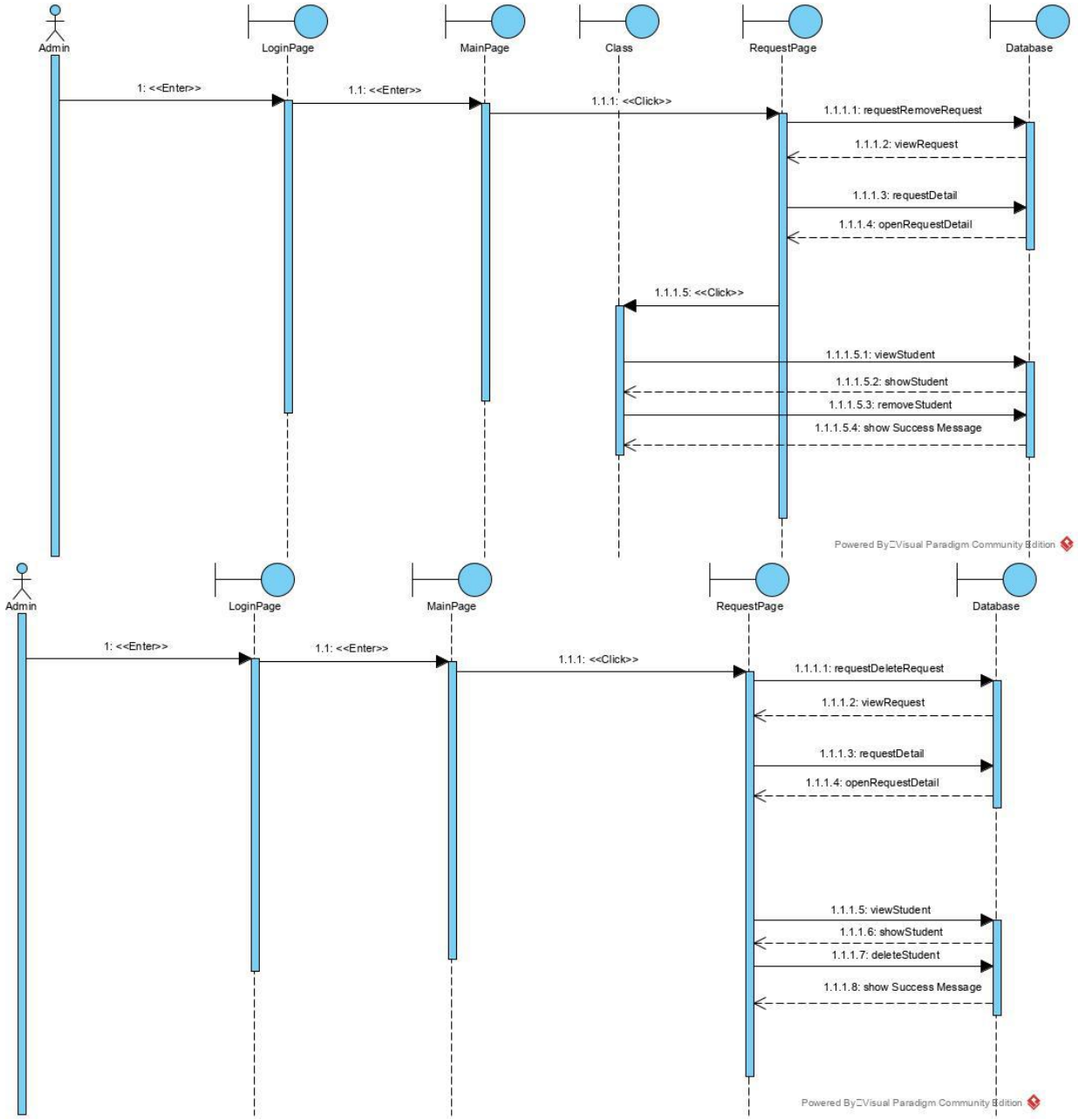


<When The Ring Bells>

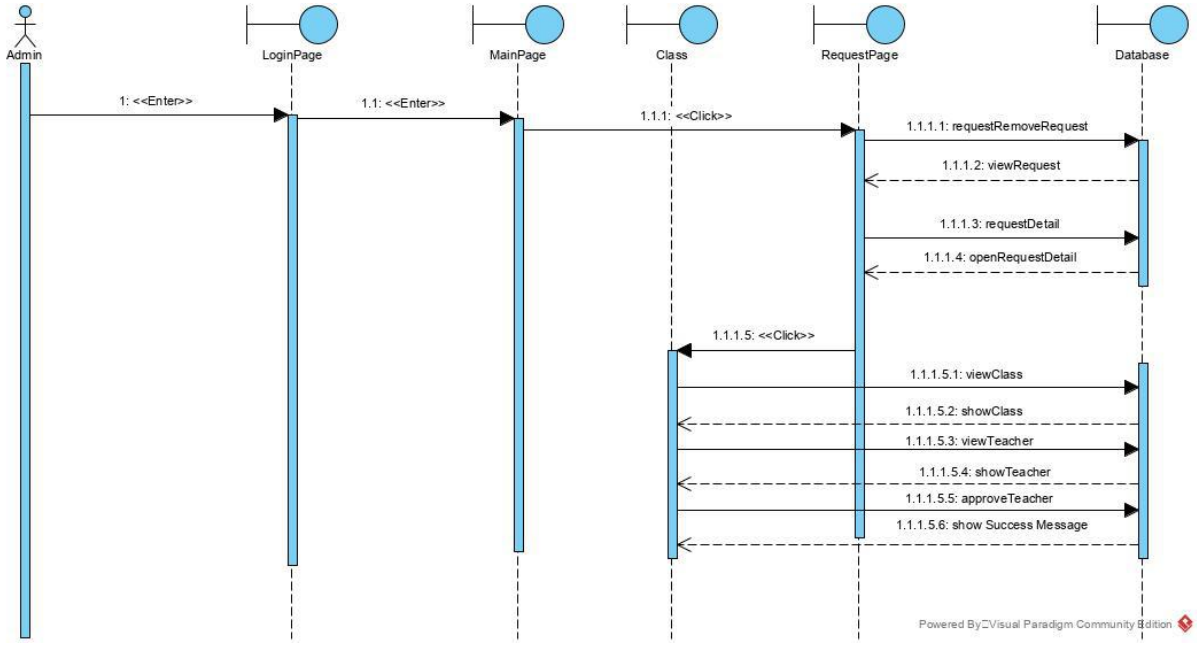




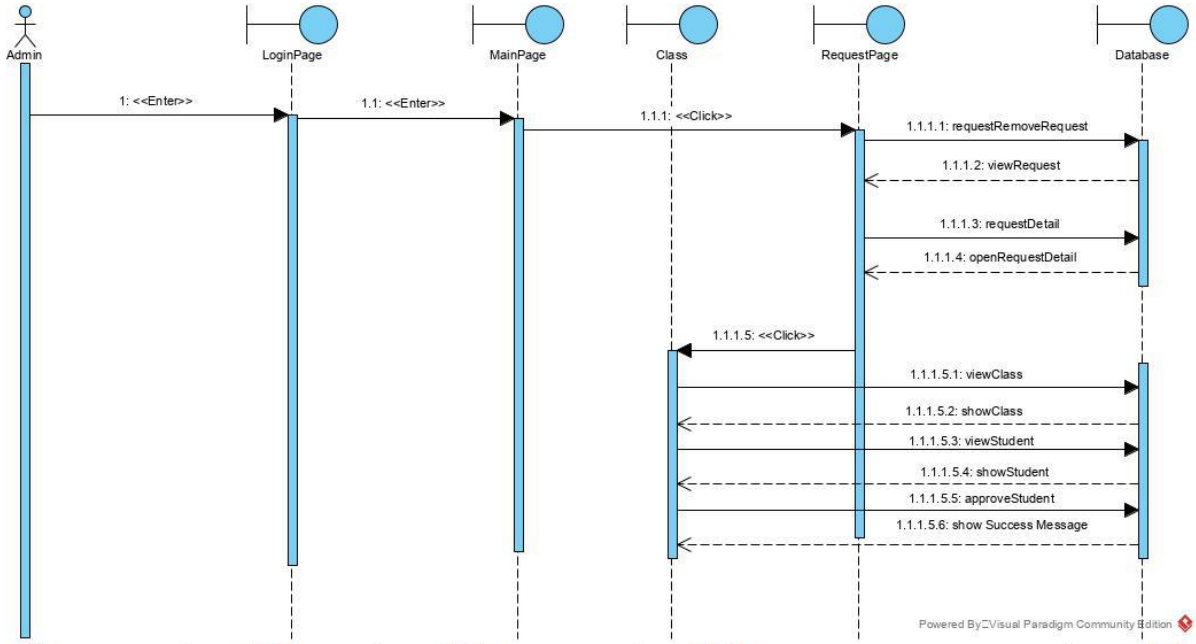
<When The Ring Bells>

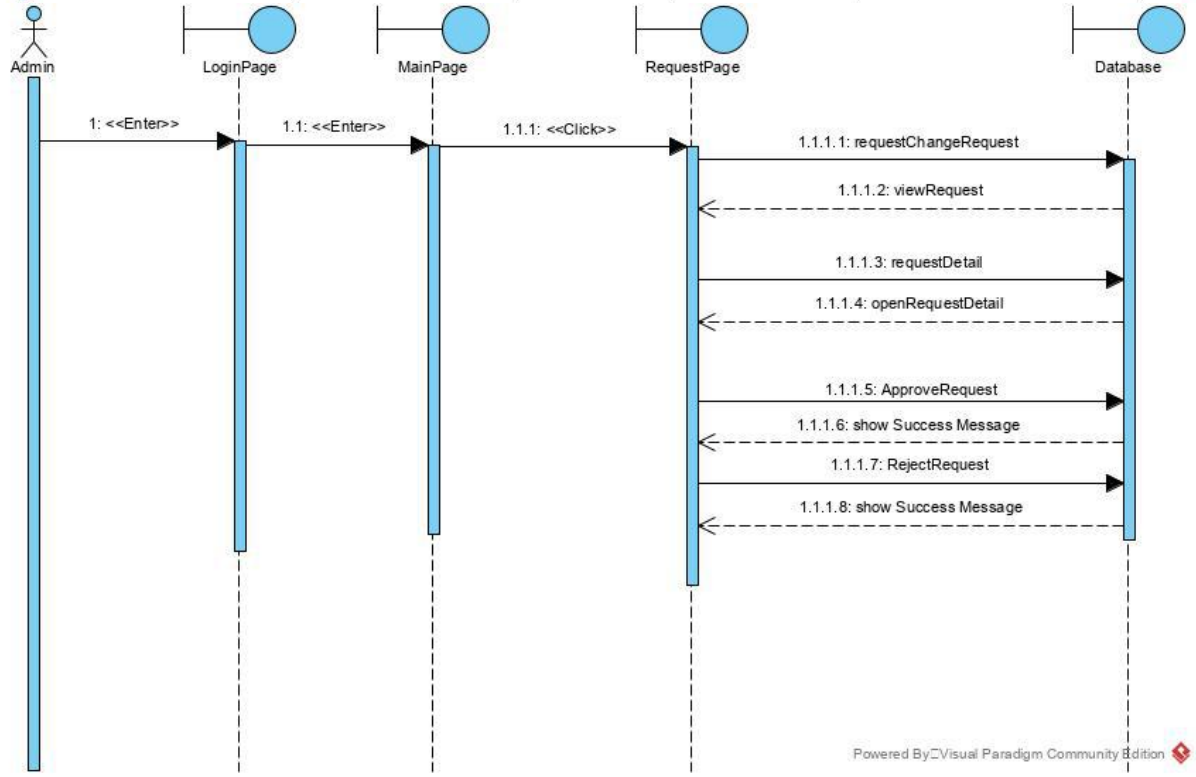


<When The Ring Bells>



<When The Ring Bells>



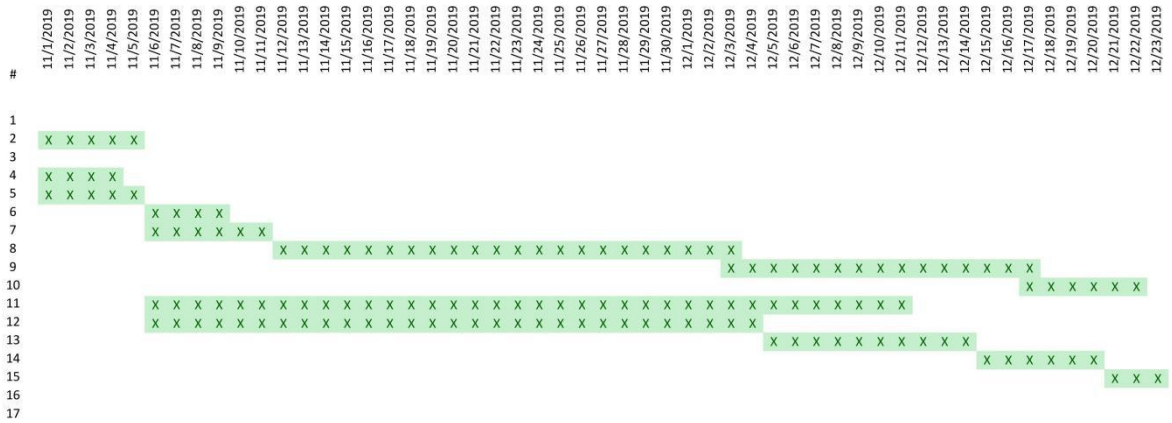


<When The Ring Bells>

3.5. Project Schedule

Project Name	Project Duration	Start Date	End Date
When The Ring Bells	84	10/1/2019	12/23/2019

#	Task Name	Duration(Days)	Start Date	End Date	Resource Names
1	Analyze Requerinment	14	10/1/2019	10/15/2019	Ersin, Serdar
2	Project Planning	5	10/15/2019	10/20/2019	Ersin
3	Scenerio Writing	10	10/20/2019	10/30/2019	Çağatay, Zafer
4	Prepare RAD	21	10/15/2019	11/5/2019	Ersin, Serdar, Çağatay, Zafer
5	Redesign Existing Components	11	10/20/2019	10/31/2019	Ersin, Serdar, Çağatay, Zafer
6	Use Case Diagrams	3	11/1/2019	11/4/2019	Zafer
7	Sequence Diagram	4	11/1/2019	11/5/2019	Çağatay
8	Class Diagram	3	11/6/2019	11/9/2019	Serdar, Zafer
9	Sytem Requerinment Control	5	11/6/2019	11/11/2019	Ersin, Serdar, Çağatay, Zafer
10	Object Design Document	21	11/12/2019	12/3/2019	Ersin, Serdar, Çağatay, Zafer
11	System Design Document	14	12/3/2019	12/17/2019	Ersin, Serdar, Çağatay, Zafer
12	Fix	5	12/17/2019	12/22/2019	Ersin, Zafer
13	Implementation	35	11/6/2019	12/11/2019	Ersin, Serdar, Çağatay, Zafer
14	Development	28	11/6/2019	12/4/2019	Ersin, Serdar, Çağatay, Zafer
15	Alpha/Quality Test	9	12/5/2019	12/14/2019	Ersin, Serdar
16	Beta Test	5	12/15/2019	12/20/2019	Serdar, Zafer
17	Release	2	12/21/2019	12/23/2019	Ersin



4. Glossary

Admin: Admin is the person who can manages all the content. Admin can apply request, confirm process, delete user profile.

Teacher: Teacher is the person who can make the main purpose of the system. Opening attendance, taking attendance etc.

Student: Student is the person who send request to add courses and follow the relationship between courses.

Profile: Shows the information of users.

Database: Database is a system which have users, courses and roles.

Student Affairs: Student Affairs can open courses and add students and teachers to a course.

<When The Ring Bells>

Attendance: Attendance is opened by the teacher and filled in by the teacher.

Login: Procedure used to entered to attendance system with a specific registered account.

Logout: Procedure used to exited to attendance system with a specific registered account.

Alert Dialog: Alert dialog is a view which shows feedback messages to users.

Add Drop Time: Add Drop Time is the time for students to add or remove lessons.

List Course: List Course shows the courses that students and teachers can choose.

5. References

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. Lecture presentations of the course (the presentations were provided by the Instructor who is Emine Ekin and Deniz Yigitbasi).