# 3.2 Functional Requirements

NATHAN O’CONNOR – K00243135

## 3.2.1 List of Requirements

1. View course details
2. Modify course details
3. Add new course details
4. Delete course details
5. Update Student result (External)
6. View exam details by location
7. Process external exam (allocate examiner to centre)
8. Book an exam
9. Generate Result Sheet
10. Update a book
11. View book orders
12. Add a book
13. Delete a book
14. Purchase a book
15. Maintenance request form
16. View timetable (teacher)
17. Process timetable (teacher)
18. Book a course
19. Book a lesson
20. Process timetable (student fulltime)
21. View timetable (student)
22. Allocate teacher and room to student (private tuition)
23. Delete Student
24. View Student details
25. Change registered details (user)
26. Register Student
27. Update Student
28. Update staff/teacher
29. Delete staff/teacher
30. View staff details
31. Add staff/teacher
32. Remove items from teacher’s area
33. Update items in teacher’s area
34. Add items to teacher’s area
35. Update event details
36. View event details
37. Add event details
38. Book event
39. Delete event details
40. Generate annual sales report from all facets of business including book sales and box office sales
41. Generate monthly sales report from book sales
42. Display Dashboard
43. Generate monthly sales report for external exams
44. Generate monthly sales report from fulltime course
45. Generate monthly sales report from box office sales
46. Generate monthly sales report from private tuition
47. Login/Logout
48. Generate maintenance request list for mgt

# 3.3 Non-Functional Requirements

NATHAN O’CONNOR – K00243135

## 3.3.3 Performance

Performance is very important when creating a web application because it directly affects user experience when users are using the web application. If the site performs slowly it is more likely users won’t return and they won’t recommend the service to other people. It is important to create a web application that operates as fast as possible for users to have a good experience, but it is also important for when staff and admins want to run functions on the web application.

NATHAN O’CONNOR – K00243135

## 3.3.4 Supportability

Supportability is necessary to ensure the web application is up and running as much as possible. Support will be provided in-house by the admins of the web application. Supportability is needed for updates to the web application and changes like improving technologies being made available and scaling up of the web application. The main reason for supportability is for maintenance and bug fixing.

# 3.4 Use Case Model

## 3.4.1 Use Case Definition

|  |
| --- |
| **Use Case By:** NATHAN O’CONNOR – K00243135 |
| **Use Case Name:** Add Staff/Teacher |
| **Use Case Description:**  Use case to add new staff or new teachers to the system. |
| **Participating Actor(s):** Admin |
| **Entry Conditions:**  The admin is logged in.  The add staff/teacher function has been invoked. |
| **Flow of Events:**   1. The system responds by displaying the add staff/teacher screen and requests the admin to enter the staff/teachers details and press continue. 2. The admin enters the details and clicks on continue. 3. The system responds by displaying the entered details of the new staff/teacher and requests the admin to click confirm or cancel. 4. If the admin clicks confirm.    1. The system responds by displaying a message saying the staff/teacher has been added while showing the staff/teacher’s details. 5. If the admin clicks cancel.    1. The system responds by displaying a message saying the staff/teacher has not been added. |
| **Exit Condition (s):**  The staff or teacher has been added to the system.  The admin cancels the process of adding the staff or teacher. |
| **Exceptions:**  Invalid details added. |

|  |
| --- |
| **Use Case By:** NATHAN O’CONNOR – K00243135 |
| **Use Case Name:** Book a course |
| **Use Case Description:**  Use case to allow a user to book a full-time course in the Sullimar Academy of Music. |
| **Participating Actor(s):** User |
| **Entry Conditions:**  The user has registered.  The user is logged in.  The book a course function has been invoked. |
| **Flow of Events:**   1. The system responds by displaying the book a course screen and requests the applicant to choose what course they would like to attend then press continue. 2. The user chooses a course and clicks on continue. 3. The system responds by displaying a screen requesting the applicant to enter their contact details then press continue. 4. The user enters all of the required details and clicks on continue. 5. The system responds by displaying all the details on a screen back to the user and requests the user to click on finalise booking or cancel. 6. If the user clicks finalise booking.    1. The system responds by displaying a screen with fees and requests the user to enter payment details then click continue.    2. The user enters their payment details and clicks on continue.    3. The system responds by displaying the payment details back to the user and requests the user to click pay.    4. The user clicks on pay. 7. If the user clicks on cancel.    1. The system responds by displaying a message saying that the booking has been cancelled. |
| **Exit Condition (s):**  The user has booked a course.  The user has cancelled their booking. |
| **Exceptions:**  Invalid details added.  Invalid Payment details.  Payment declined. |

|  |
| --- |
| **Use Case By:** NATHAN O’CONNOR – K00243135 |
| **Use Case Name:** Book a lesson |
| **Use Case Description:**  Use case to allow a user to book a lesson or to allow a parent to book a lesson for their child. |
| **Participating Actor(s):** User |
| **Entry Conditions:**  The user has registered.  The user is logged in.  The book a lesson function has been invoked. |
| **Flow of Events:**   1. The system responds by displaying the book a lesson screen and requests the user to choose what lesson they would like to book then press continue. 2. The user chooses a lesson and clicks on continue. 3. The system responds by displaying a screen requesting the user to enter the applicants details then press continue. 4. The user enters all of the required details and clicks on continue. 5. The system responds by displaying a screen requesting the user to enter contact details then press continue. 6. The user enters the contact details and clicks on continue. 7. The system responds by displaying all the details on a screen back to the user and requests the user to click on finalise booking or cancel. 8. If the user clicks finalise booking.    1. The system responds by displaying a screen with lesson costs and requests the user to enter payment details then click continue.    2. The user enters their payment details and clicks on continue.    3. The system responds by displaying the payment details back to the user and requests the user to click pay.    4. The user clicks on pay. 9. If the user clicks on cancel.    1. The system responds by displaying a message saying that the booking has been cancelled. |
| **Exit Condition (s):**  The user has booked a lesson.  The user has cancelled their booking. |
| **Exceptions:**  Invalid details added.  Invalid Payment details.  Payment declined. |

|  |
| --- |
| **Use Case By:** NATHAN O’CONNOR – K00243135 |
| **Use Case Name:** Book an exam |
| **Use Case Description:**  Use case for when a person wants to book an exam. |
| **Participating Actor(s):** User |
| **Entry Conditions:**  The book an exam function has been invoked. |
| **Flow of Events:**   1. The system responds by displaying the book an exam screen and requests the user to enter all of their details then click continue. 2. The user enters all their details and clicks on continue. 3. The system responds by displaying a screen requesting the user to enter the details of the exam(s) they’d like to take, and the details of the preferred examination centre then click continue. 4. The user enters all of the exam details and clicks on continue. 5. The system responds by displaying all the details on a screen back to the user and requests the user to click on finalise booking or cancel. 6. If the user clicks finalise booking.    1. The system responds by displaying a screen with the price and requests the user to enter payment details then click continue.    2. The user enters their payment details and clicks on continue.    3. The system responds by displaying the payment details back to the user and requests the user to click pay.    4. The user clicks on pay. 7. If the user clicks on cancel.    1. The system responds by displaying a message saying that the booking has been cancelled. |
| **Exit Condition (s):**  The user has booked an exam.  The user has cancelled their booking. |
| **Exceptions:**  Invalid details added.  Invalid Payment details.  Payment declined.  Exam attendance full. |

|  |
| --- |
| **Use Case By:** NATHAN O’CONNOR – K00243135 |
| **Use Case Name:** Change Registered Details (User) |
| **Use Case Description:**  Use case to allow a user to update their registered details. |
| **Participating Actor(s):** User |
| **Entry Conditions:**  The user has registered.  The user is logged in.  The change registered details function has been invoked. |
| **Flow of Events:**   1. The system responds by displaying the user’s account details and requests the user to make changes then click confirm or cancel. 2. If the user makes changes to their details, then clicks confirm.    1. The system responds by updating their account details then displays a screen back to the user with the updated account details. 3. If the user clicks on cancel.    1. The system responds by displaying a message saying the user details have not been changed. |
| **Exit Condition (s):**  The user has changed their registered details.  The user has cancelled changing their details. |
| **Exceptions:**  Invalid details added. |

|  |
| --- |
| **Use Case By:** NATHAN O’CONNOR – K00243135 |
| **Use Case Name:** Delete a Book |
| **Use Case Description:**  Use case for the admin to select a book and delete the book from the system. |
| **Participating Actor(s):** Admin |
| **Entry Conditions:**  The admin is logged in.  The delete a book function has been invoked. |
| **Flow of Events:**   1. The system responds by listing all the books on the system and requests the admin to select a book to delete. 2. The admin selects a book to delete. 3. The system responds by displaying the books details and prompts the admin to delete or cancel. 4. If the admin clicks delete.    1. The book is removed from the system and a message is displayed saying the book has been deleted. 5. If the admin clicks cancel.    1. The system responds by returning to the admin page. |
| **Exit Condition (s):**  The admin has deleted a book.  The admin has cancelled deleting a book. |
| **Exceptions:**  The book could not be removed from the system. |

|  |
| --- |
| **Use Case By:** NATHAN O’CONNOR – K00243135 |
| **Use Case Name:** Generate monthly sales report from box office sales |
| **Use Case Description:**  Use case to generate a monthly sales report from the box office sales. Displays the details of the box office sales for that month on the sales report. |
| **Participating Actor(s):** Admin |
| **Entry Conditions:**  The admin has logged in.  The generate monthly sales report from box office sales function has been invoked. |
| **Flow of Events:**   1. The system responds by generating a monthly sales report from box office sales and displays the report to the admin (Detailing daily sales, daily booking amounts, percentage of sales for that month, total sales and time period) then requests the admin to click continue. 2. The admin clicks continue. 3. The system responds by saving the report and displaying a message stating the report has been saved. |
| **Exit Condition (s):**  The admin has generated a monthly sales report from box office sales. |
| **Exceptions:**  The system fails to generate a report. |

|  |
| --- |
| **Use Case By:** NATHAN O’CONNOR – K00243135 |
| **Use Case Name:** Generate monthly sales report from private tuition |
| **Use Case Description:**  Use case to generate a monthly sales report from private tuition. Displays the details of the private tuition sales for that month on the sales report. |
| **Participating Actor(s):** Admin |
| **Entry Conditions:**  The admin has logged in.  The generate monthly sales report from private tuition function has been invoked. |
| **Flow of Events:**   1. The system responds by generating a monthly sales report from private tuition and displays the report to the admin (Detailing daily sales, amount of private tuition lessons per day, percentage of sales for that month, total sales and time period) then requests the admin to click continue. 2. The admin clicks continue. 3. The system responds by saving the report and displaying a message stating the report has been saved. |
| **Exit Condition (s):**  The admin has generated a monthly sales report from private tuition. |
| **Exceptions:**  The system fails to generate a report. |

|  |
| --- |
| **Use Case By:** NATHAN O’CONNOR – K00243135 |
| **Use Case Name:** Process External Exam |
| **Use Case Description:**  Use case to process an external exam and allocate an examiner to a centre. |
| **Participating Actor(s):** Admin |
| **Entry Conditions:**  The admin has logged in.  The process external exam function has been invoked. |
| **Flow of Events:**   1. The system responds by displaying a screen requesting the admin to enter the exam details (subject/instrument, grade, start time, end time, date, price) then press continue. 2. The admin enters the exam details and presses continue. 3. The system responds by displaying a list of available exam centres and prompts the admin to select a centre. 4. The admin selects a centre. 5. The system responds by displaying a list of available examiners to allocate to the exam and prompts the admin to select an examiner. 6. The admin selects an examiner to allocate to the exam. 7. The system responds by displaying a screen detailing the exam details, exam centre and examiner details then prompts the admin to click process or cancel. 8. If the admin selects process.    1. The system responds by processing the exam. 9. If the admin selects cancel.    1. The system responds by returning the admin to the admin page. |
| **Exit Condition (s):**  The admin has processed an external exam.  The admin has cancelled the processing of an external exam. |
| **Exceptions:**  Invalid exam details.  Failed to add exam centre.  Failed to allocate examiner. |

|  |
| --- |
| **Use Case By:** NATHAN O’CONNOR – K00243135 |
| **Use Case Name:** Process Timetable (Teacher) |
| **Use Case Description:**  Use case to process the timetable for a teacher. |
| **Participating Actor(s):** Admin |
| **Entry Conditions:**  The admin has logged in.  The process timetable function has been invoked. |
| **Flow of Events:**   1. The system responds by displaying a screen listing all the teachers and requests the admin to select a teacher. 2. The admin selects a teacher. 3. The system responds by displaying the teachers details and    1. FOR each subject/instrument the teacher teaches the system requests the admin to select a room, time and day then click continue.    2. The admin enters the details and clicks continue.    3. The system responds by checking availability and requests the admin to enter re-enter details if the availability check fails. 4. The admin entered the last subject/instrument timetable details and clicked continue. 5. The system responds by displaying the teachers details and the timetable for the teacher then prompts the admin to click process or cancel. 6. If the admin clicks process.    1. The system responds by processing the timetable. 7. If the admin clicks cancel.    1. The system responds by returning the admin to the admin page. |
| **Exit Condition (s):**  The admin has processed the timetable.  The admin cancelled processing the timetable. |
| **Exceptions:**  No rooms available. |