

**MOTEL SYSTEM**

**Software Design Specification**

– Hanoi, May 2022 –

**Record of changeS**

| **Date** | **A\* M, D** | **In charge** | **Change Description** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

\*A - Added M - Modified D - Deleted

**Table of Contents**

[I. Overview](#_heading=h.gjdgxs) 5

[1. Code Packages](#_heading=h.30j0zll) 5

[2. Database Design](#_heading=h.1fob9te) 6

[a. Database Schema](#_heading=h.3znysh7) 6

[b. Table Description](#_heading=h.2et92p0) 7

[II. Code Designs](#_heading=h.tyjcwt) 8

1. Home9

[a. Class Diagram](#_heading=h.1t3h5sf) 9

[b. Class Specifications](#_heading=h.4d34og8) 9

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 10

[d. Database queries](#_heading=h.17dp8vu) 12

2. Login 10

[a. Class Diagram](#_heading=h.1t3h5sf) 10

[b. Class Specifications](#_heading=h.4d34og8) 10

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 11

[d. Database queries](#_heading=h.17dp8vu) 11

3[. Create Account](#_heading=h.3dy6vkm) 11

[a. Class Diagram](#_heading=h.1t3h5sf) 11

[b. Class Specifications](#_heading=h.4d34og8) 11

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 12

[d. Database queries](#_heading=h.17dp8vu) 13

4[. Change Password](#_heading=h.3dy6vkm) 92

[a. Class Diagram](#_heading=h.1t3h5sf) 93

[b. Class Specifications](#_heading=h.4d34og8) 94

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 94

[d. Database queries](#_heading=h.17dp8vu) 94

4[. Forgot password](#_heading=h.3dy6vkm) 13

[a. Class Diagram](#_heading=h.1t3h5sf) 13

[b. Class Specifications](#_heading=h.4d34og8) 14

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 15

[d. Database queries](#_heading=h.17dp8vu) 15

5[. Manage Account by Admin](#_heading=h.3dy6vkm) 16

[a. Class Diagram 16](#_heading=h.1t3h5sf)

[b. Class Specifications](#_heading=h.4d34og8) 16

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 17

[d. Database queries](#_heading=h.17dp8vu) 18

6[. Manage Motel by Admin](#_heading=h.3dy6vkm) 18

[a. Class Diagram](#_heading=h.1t3h5sf) 18

[b. Class Specifications](#_heading=h.4d34og8) 19

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 20

[d. Database queries](#_heading=h.17dp8vu) 20

7[. Manage Post by Admin](#_heading=h.3dy6vkm) 21

[a. Class Diagram](#_heading=h.1t3h5sf) 21

[b. Class Specifications](#_heading=h.4d34og8) 21

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 22

[d. Database queries](#_heading=h.17dp8vu) 23

8[. Manage Profile Account](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

10[. Post Detail](#_heading=h.3dy6vkm) 34

[a. Class Diagram](#_heading=h.1t3h5sf) 36

[b. Class Specifications](#_heading=h.4d34og8) 37

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 38

[d. Database queries](#_heading=h.17dp8vu) 38

11[. Manage Account by Admin/Delete](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

12[. Manage Account by Admin/Update](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

13[. Manage Account by Admin/Search](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

14[. Manage Account by Admin/Sort](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

15[. Manage Account by Admin/View detail](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

16[. Manage Motel by Admin/Delete](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

17[. Manage Motel by Admin/Update](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

18[. Manage Motel by Admin/Search](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

19[. Manage Motel by Admin/Sort](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

20[. Manage Motel by Admin/View detail](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

21[. Manage Post by Admin/Delete](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

22[. Manage Post by Admin/Update](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

23[. Manage Post by Admin/Search](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

24[. Manage Post by Admin/Sort](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

25[. Manage Post by Admin/View detail](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

26[. Manage Request by Admin](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

27[. Manage Request by Admin/Delete](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

28[. Manage Request by Admin/Update](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

29[. Manage Request by Admin/View detail](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

30[. Manage Request by Admin/Search](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

31[. Manage Request by Admin/Sort](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

32. [Manage Room By Host](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu) 25

33. [Manage Room By Host/Add](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu) 25

34. [Manage Room By Host/Update | ViewDetail](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu) 25

35. [Manage Room By Host/Delete](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu) 25

36. [Manage Room By Host/Search](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu) 25

37. [Manage Room By Host/Sort](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu) 25

38. [Manage Services By Host](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu) 25

39. [Manage Services By Host/Add](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu) 25

40. [Manage Services By Host/Update | ViewDetail](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu) 25

41. [Manage Services By Host/Delete](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu) 25

42. [Manage Services By Host/Search](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu) 25

43. [Manage Services By Host/Sort](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu) 25

52[. Manage Request By Host](#_heading=h.3dy6vkm) 94

[a. Class Diagram](#_heading=h.1t3h5sf) 95

[b. Class Specifications](#_heading=h.4d34og8) 95

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 96

[d. Database queries](#_heading=h.17dp8vu) 96

53[. Manage Request By Host/Delete](#_heading=h.3dy6vkm) 96

[a. Class Diagram](#_heading=h.1t3h5sf) 96

[b. Class Specifications](#_heading=h.4d34og8) 97

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 98

[d. Database queries](#_heading=h.17dp8vu) 98

54[. Manage Request By Host/Search](#_heading=h.3dy6vkm) 98

[a. Class Diagram](#_heading=h.1t3h5sf) 99

[b. Class Specifications](#_heading=h.4d34og8) 99

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 99

[d. Database queries](#_heading=h.17dp8vu) 100

55[. Manage Request By Host/Sort](#_heading=h.3dy6vkm) 98

[a. Class Diagram](#_heading=h.1t3h5sf) 99

[b. Class Specifications](#_heading=h.4d34og8) 99

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 99

[d. Database queries](#_heading=h.17dp8vu) 100

56[. Manage Request By Host/Update/Show](#_heading=h.3dy6vkm) 98

[a. Class Diagram](#_heading=h.1t3h5sf) 99

[b. Class Specifications](#_heading=h.4d34og8) 99

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 99

[d. Database queries](#_heading=h.17dp8vu) 100

57[. Filter Post](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

58[. Manage Post by Host](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

59[. Manage Post by Host/Add](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

60[. Manage Post by Host/Update](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

61[. Manage Post by Host/Delete](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

62[. Manage Post by Host/Search](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

63[. Manage Post by Host/Sort](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

64[. Manage Motel by Host/View](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

65[. Manage Motel by Host/Update](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

66[. Manage Motel by Host/Delete](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

67[. Manage Motel by Host/View detail](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

68[. Manage Motel by Host/Search](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

69[. Manage Motel by Host/Sort](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu) 119

70[. Manage Motel by Host/Add](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu) 119

71[. Manage Request by Customer](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

72[. Manage Request by Customer/Add](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

73[. Manage Request by Customer/Update](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

74[. Manage Request by Customer/Delete](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

75[. Manage Request by Customer/Search](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

76[. Manage Request by Customer/Sort](#_heading=h.3dy6vkm) 117

[a. Class Diagram](#_heading=h.1t3h5sf) 117

[b. Class Specifications](#_heading=h.4d34og8) 117

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 119

[d. Database queries](#_heading=h.17dp8vu)

77[. Manage Request by Customer/View detail](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

78[. Manage Message by Host](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

79[. Manage Message by Host/Send Message](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

80[. Manage Message by Host/Rep Message](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

81[. Manage Message by Host/Search](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

82[. Manage Message by Host/Sort](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

83[. Manage Message by Customer](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

84[. Manage Message by Customer/Send Message](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

85[. Manage Message by Customer/Rep Message](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

86[. Manage Message by Customer/Search](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

87[. Manage Message by Customer/Sort](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

8[. Manage Profile Account](#_heading=h.3dy6vkm) 23

[a. Class Diagram](#_heading=h.1t3h5sf) 23

[b. Class Specifications](#_heading=h.4d34og8) 23

[c. Sequence Diagram(s)](#_heading=h.2s8eyo1) 24

[d. Database queries](#_heading=h.17dp8vu)

# I. Overview

## 1. Code Packages

Graphical user interface

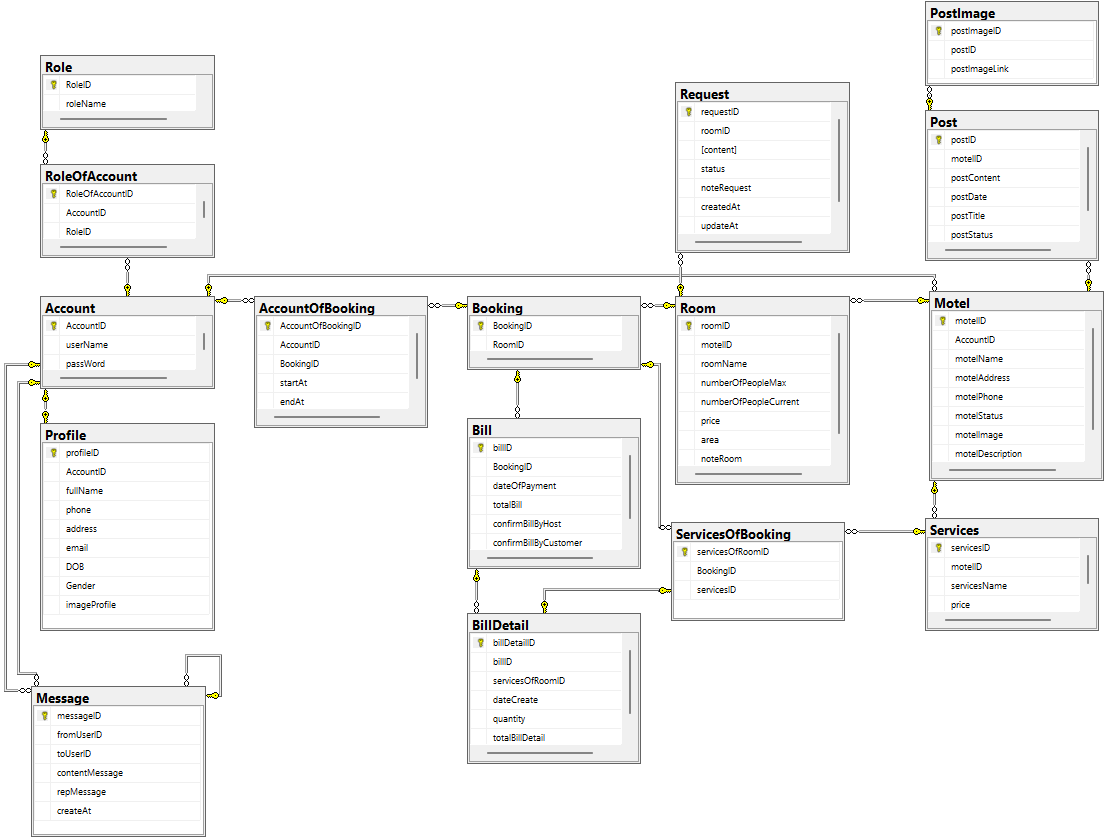
Description automatically generated

***Package descriptions & package class naming conventions***

| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | Web Pages | Contains front-end support packages |
| 02 | WEB-INF | Contains the config files of the web |
| 03 | css | Used to store cascading style sheets files |
| 04 | js | Used to store javascript files |
| 05 | page | Used to store display files |
| 06 | DAL | Used to store files that interact with the database |
| 07 | Model | Used to store data types |
| 08 | Controller | Contains package controllers by role |
| 09 | Controller.Admin | Contains controller classes according to the admin role |
| 10 | Controller.Host | Contains controller classes according to the host role |
| 11 | Controller.Customer | Contains controller classes according to the customer role |

## 2. Database Design

### a. Database Schema



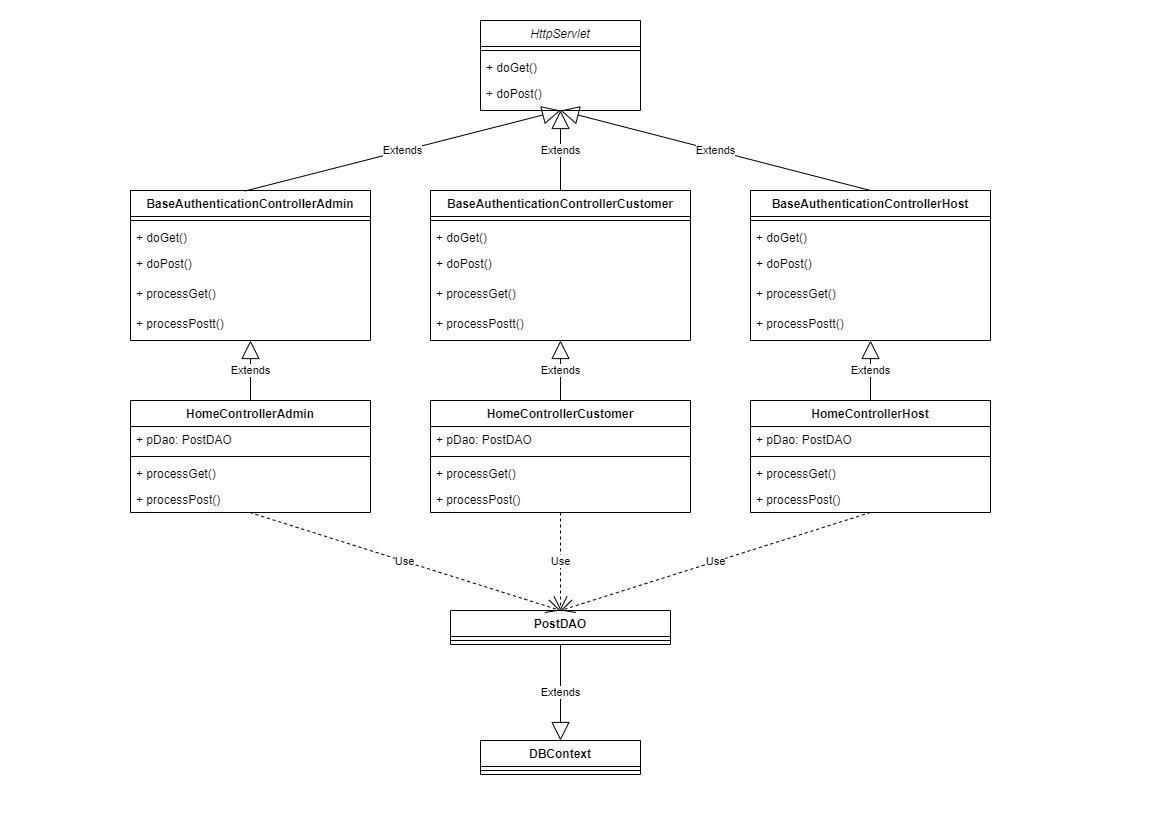
### b. Table Description

| **No** | **Table** | **Description** |
| --- | --- | --- |
| 1 | Account | Have Column: AccountID (PK), userName, passWord.  Storing user login information |
| 2 | Role | Have Column: RoleID (PK), roleName  Store user positions in the system |
| 3 | RoleOfAccount | Have Column: RoleOfAccount (PK), AccountID (FK), RoleID (FK).  Authorization for users |
| 4 | Profile | Have Column: profileID (PK), AccountID (FK), fullName, phone, address, email, DOB, Gender, imageProfile  Keep personal information of each account |
| 5 | Motel | Have Column: motelID (PK), AccountID (FK), motelName, motelAddress, motelPhone.  Retention of accommodation information |
| 6 | Post | Have Column: postID (PK), motelID (FK), postContent, postDate  Save the owner's posts about the inn |
| 7 | PostImage | Have Column: postImageID (PK), postID (FK), postImageLink.  Save the pictures of that inn. |
| 8 | Services | Have Column: servicesID (PK), servicesName, price  Save service name and price |
| 9 | ServicesOfMotel | Have Column: servicesOfMotelID (PK), motelID (FK), serviceID (FK).  Keep the name of the service and the price of that inn |
| 10 | Room | Have Column:roomID (PK), motelID (FK), roomName, numberOfPeopleMax, numberOfPeopleCurrent, price, area, noteRoom  Store room information |
| 11 | Request | Have Column:requestID (PK), roomID (FK), content, status, noteRequest, createAt, updateAt  Keep the requirements of that room with the inn |
| 12 | RoomOfAccount | Have Column: roomOfAccountID (PK), AccountID (FK), roomID (FK), startAt, endAt  Save the members of the room |
| 13 | ServicesOfRoom | Have Column: servicesOfRoomID (PK), roomID (FK), servicesID (FK)  Store the services of that room |
| 14 | Bill | Have Column:billID (PK), roomID (FK), dateOfPayment, totalBill, statusBill  Keep the receipts of the room with the inn. |
| 15 | BillDetail | Have Column: billDetailID (PK), billID (FK), servicesOfRoomID (FK), dateCreate, quantity, totalBillDetail  Keep the bill details of the room with the hostel. |

# II. Code Designs

## 1. Home

### a. Class Diagram



### b. Class Specifications

#### HomeControllerAdmin Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will return the login.jsp page according to the corresponding role |
| 02 | doPost | input is request and response. After receiving the request from the user, it will get the username and password and call the function getAccountByUsernamePasswordAndRole(username, password, role) of the AccountDAO class. |

#### HttpServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet |  |
| 02 | doPost | Using this HttpServlet object, we can build Java web applications to handle requests from clients with HTTP methods like GET, POST, HEAD, PUT, DELETE, TRACE and OPTIONS. When the client sends a request to the web server container, the service() method in the HttpServlet object processes the request. Depending on the HTTP method that the client sends, further processing will be done by the doGet(), doPost(), doHead(), doPut(), doDelete(), doTrace() and doOptions() methods of this object. receive. |
| 03 | doHead |
| 04 | doPut |
| 05 | doDelete |
| 06 | doOption |
| 07 | doTrace |

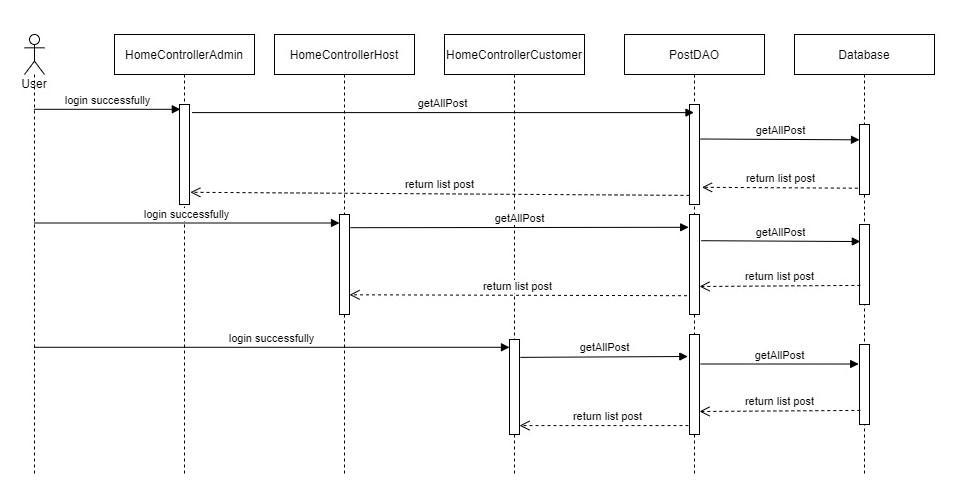
#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPost | Get all post in database |
| 02 | getAllPostByMotelName | Get all post in database which have similar to input motel name |

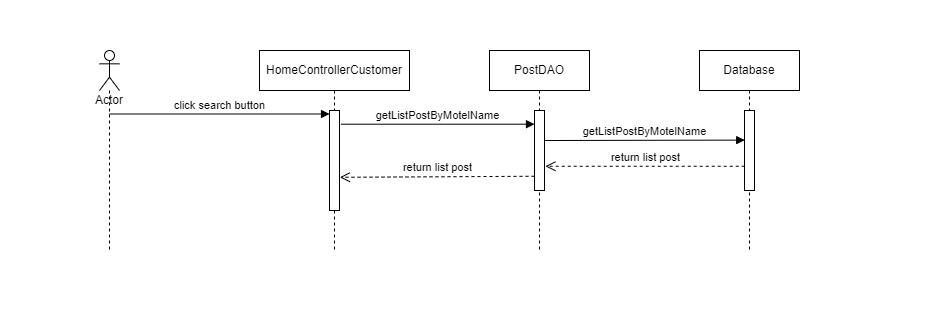
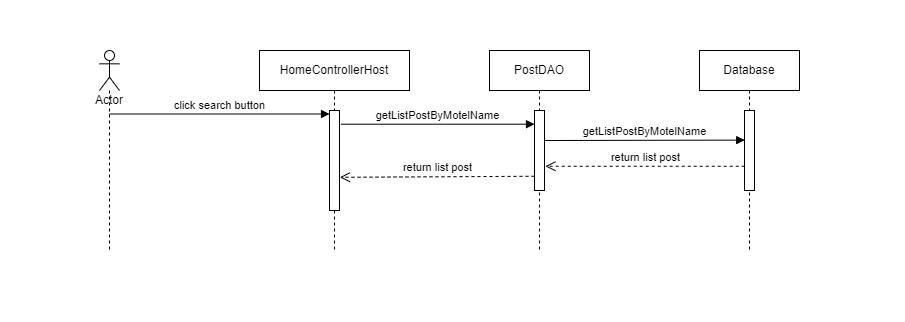
#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

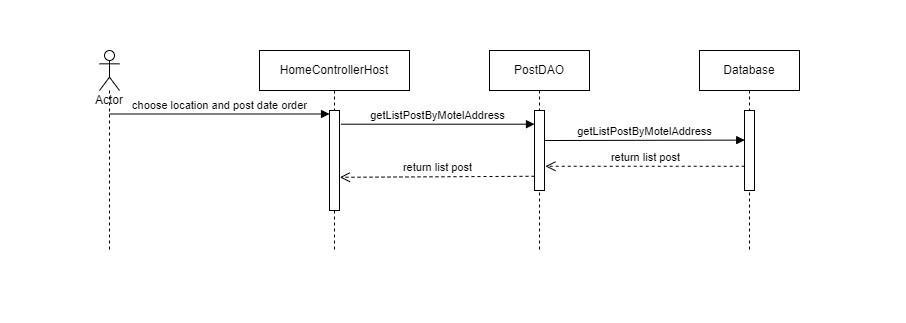
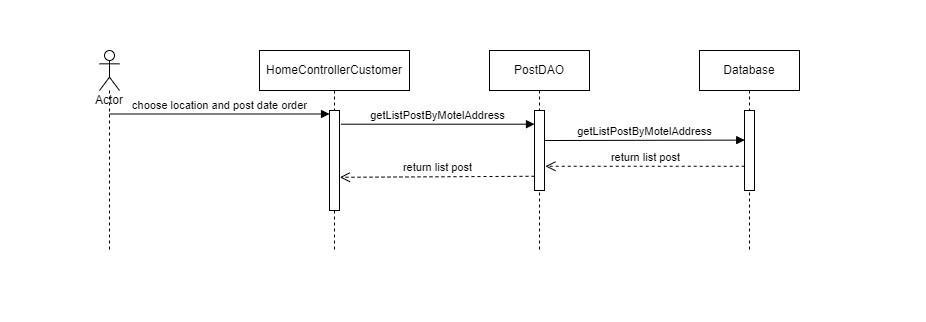
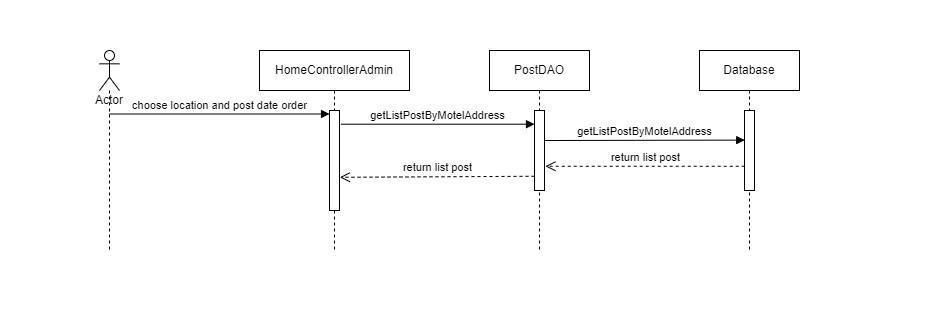
### c. Sequence Diagram(s)



1. Search post



1. Filter post



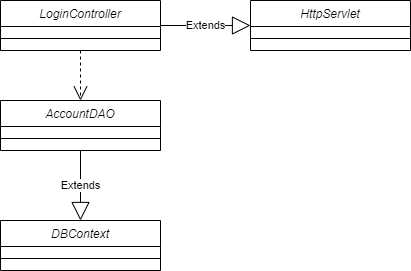
### d. Database queries

* select \* from Post;
* select pst.postID, pst.motelID, pst.postContent, pst.postDate, pst.postTitle, pst.postStatus, mtl.motelName, pfl.fullName from Post pst left join Motel mtl on mtl.motelID=pst.motelID left join Profile pfl on pfl.AccountID=mtl.AccountID where mtl.motelName like ?;
* select pst.postID, pst.motelID, pst.postContent, pst.postDate, pst.postTitle, pst.postStatus, mtl.motelName, pfl.fullName from Post pst left join Motel mtl on mtl.motelID=pst.motelID left join Profile pfl on pfl.AccountID=mtl.AccountID where upper(mtl.motelAddress) like upper(N'%" + motelAddress + "%') order by pst.postDate "+order+";

## 

## 2. Login

### a. Class Diagram



### b. Class Specifications

#### LoginController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will return the login.jsp page according to the corresponding role |
| 02 | doPost | input is request and response. After receiving the request from the user, it will get the username and password and call the function getAccountByUsernamePasswordAndRole(username, password, role) of the AccountDAO class. |

#### HttpServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet |  |
| 02 | doPost | Using this HttpServlet object, we can build Java web applications to handle requests from clients with HTTP methods like GET, POST, HEAD, PUT, DELETE, TRACE and OPTIONS. When the client sends a request to the web server container, the service() method in the HttpServlet object processes the request. Depending on the HTTP method that the client sends, further processing will be done by the doGet(), doPost(), doHead(), doPut(), doDelete(), doTrace() and doOptions() methods of this object. receive. |
| 03 | doHead |
| 04 | doPut |
| 05 | doDelete |
| 06 | doOption |
| 07 | doTrace |

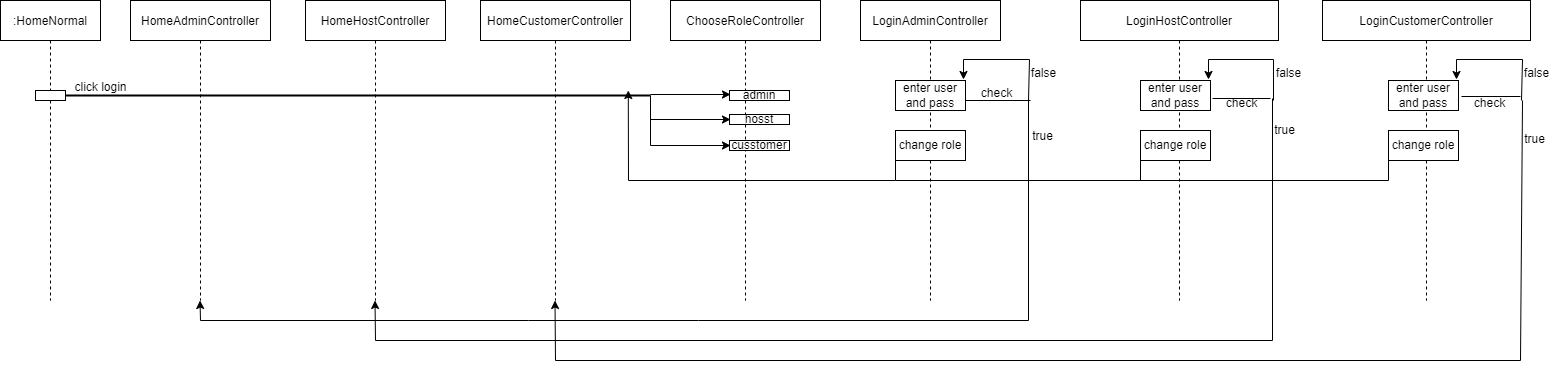
#### AccountDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAccountByUsernamePasswordAndRole | input is username and password. The function will perform to get the information of the account with the username and password as passed in the account, profile, role table |

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *DBContext* | *Make a connection to the database* |

### c. Sequence Diagram(s)

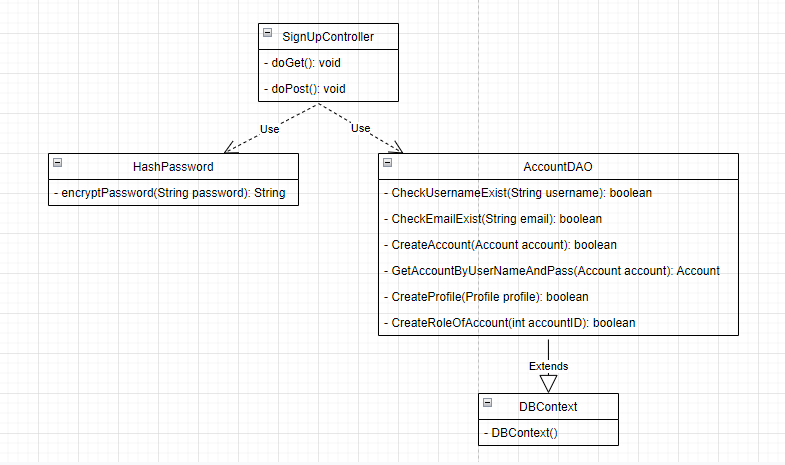


### d. Database queries

* select \* from account where username = ? and password = ?;
* select \* from Profile where AccountID = ?;
* select \* from RoleOfAccount where AccountID = ?
* select \* from role where RoleID = ?

## 3. Create Account/Register Account

### a. Class Diagram



### b. Class Specifications

#### SignUpController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doPost | Get data from “SignUpAccount” page and create new account and profile for this account |
| 02 | doGet | Sendirect to “SignUpAccount” page |

#### AccountDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | CreateAccount | Get information from object Account and create new account in database |
| 02 | GetAccountByUserNameAndPass | Get username and password from object Account and get Account that have same username and password from database |
| 03 | CreateProfile | Get information from object Profile and create profile for new account that newly created |
| 04 | CreateRoleOfAccount | Create role customer for new account that newly created |
| 05 | CheckUsernameExist | Check username exist in database |
| 06 | CheckEmailExist | Check email exist in database |

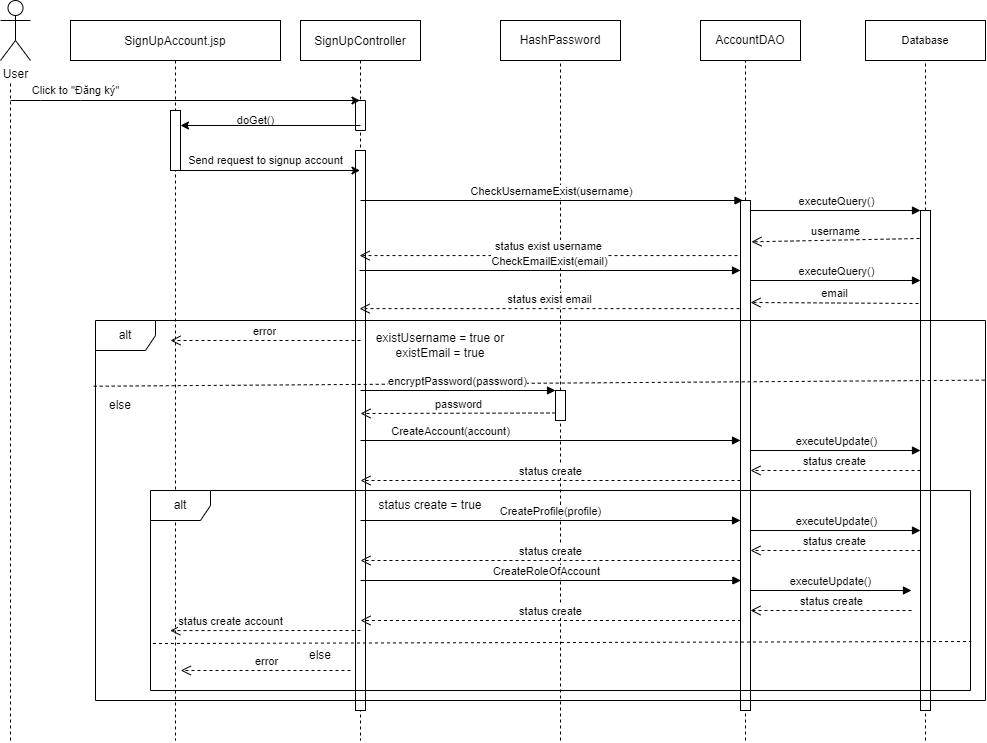
#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Connect SQL with Netbeans to get data from database to use |

#### HashPassword Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | encryptPassword | Hash code for password to increase security |

### c. Sequence Diagram(s)



### d. Database queries

+ select username from Account where username = ? and isDeleteAccount = 0

+ select email from Profile where email = ? and isDeleteProfile = 0

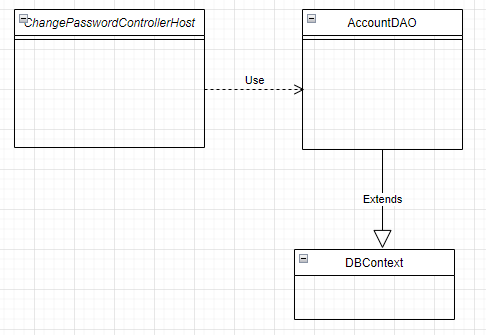
+INSERT INTO Account(userName,[passWord],isDeleteAccount) VALUES (?,?,0)

+ INSERT INTO [Profile](AccountID, fullName, DOB, Gender, [address], email, phone,isDeleteProfile) VALUES (?,?,?,?,?,?,?,0)

+ insert into RoleOfAccount values(?,3,0)

## 4.1 Change password

### a. Class Diagram



### b. Class Specifications

ChangePasswordControllerHost class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Check old password, new password and update password in database |

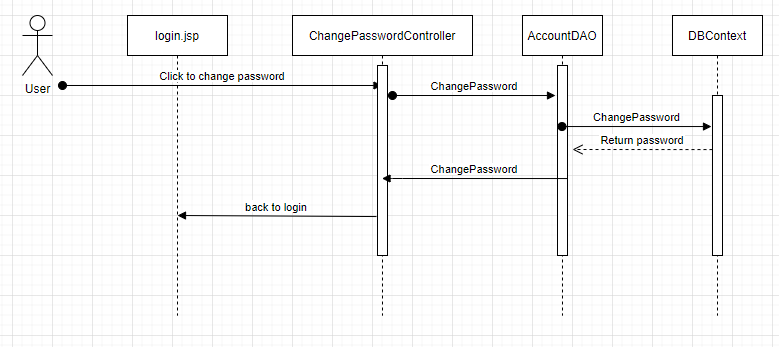
AccountDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | ChangePassword | Update password in database |

DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Connecting system with database |

### c. Sequence Diagram(s)



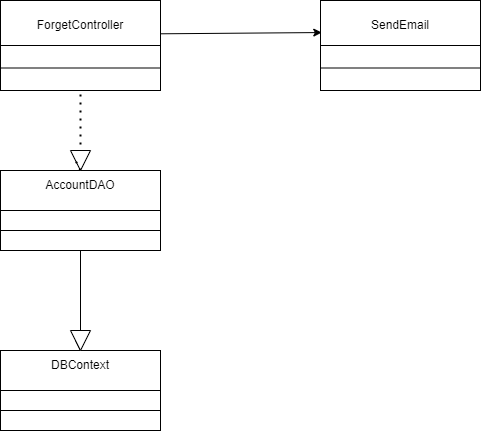
### d. Database queries

ChangePassword : update Account set [passWord] = ?

where AccountID = ?

## 4.2 Forgot Password

### a. Class Diagram

**

## 

### b. Class Specifications

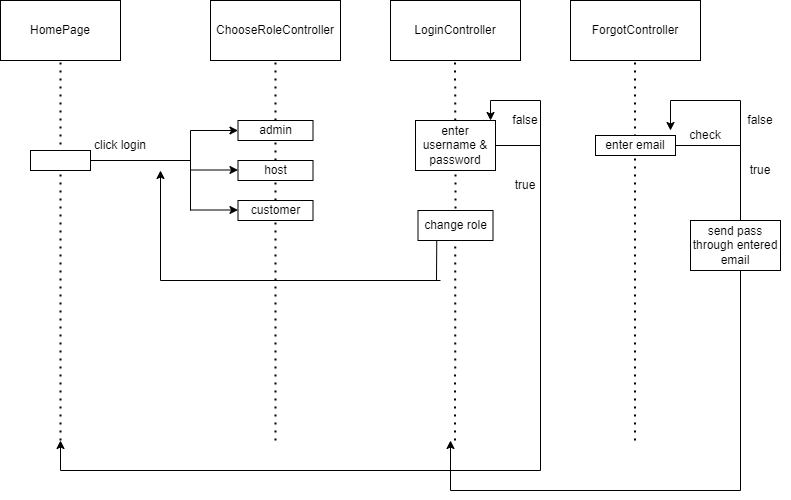
#### SendEmail.java

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | boolean sendEmail(String to, String password) | This function send that corresponding password to email that user enter (email must be signed up an account before) |

***AccountDAO.java***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | String getPasswordByEmail(String email) | get password from an account sign with entered email |
| 02 | ArrayList<String> getAllEmail() | get all email in database |

### c. Sequence Diagram(s)

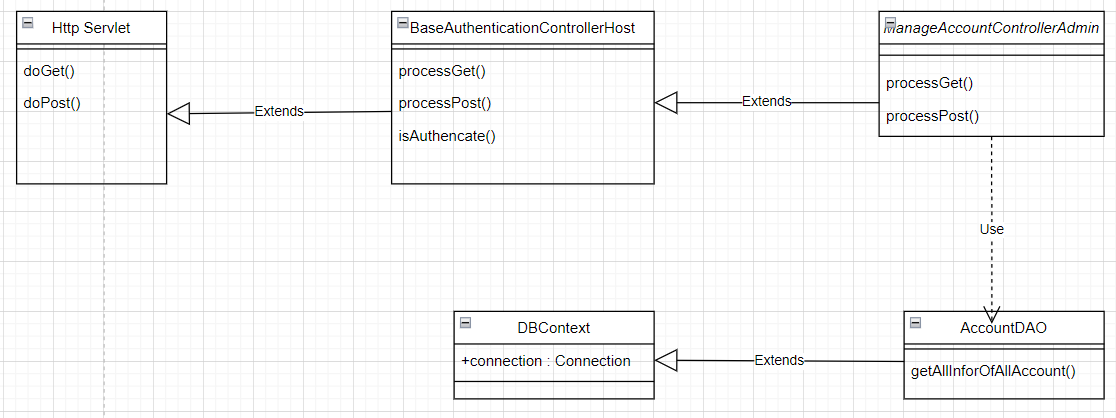


### d. Database queries

* select [passWord] from Account as a, Profile as p where p.email like ?
* select email from Profile

## 5. Manage Account by Admin

### a. Class Diagram



### b. Class Specifications

#### ManageAccountAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Input is request and response. After receiving the request from the user, it will call function getAllInforOfAllAccount to take all the information of all users and transfer it to ManageAdmin.jsp |

#### 

#### AccountDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllInforOfAllAccount | This function will retrieve all information of a user including account profile password .... |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

### 

### c. Sequence Diagram(s)

### 

### d. Database queries

* select a.AccountID,a.userName,p.phone,p.email,roa.RoleID,r.roleName from Account as a

left join [Profile] as p on a.AccountID = p.AccountID

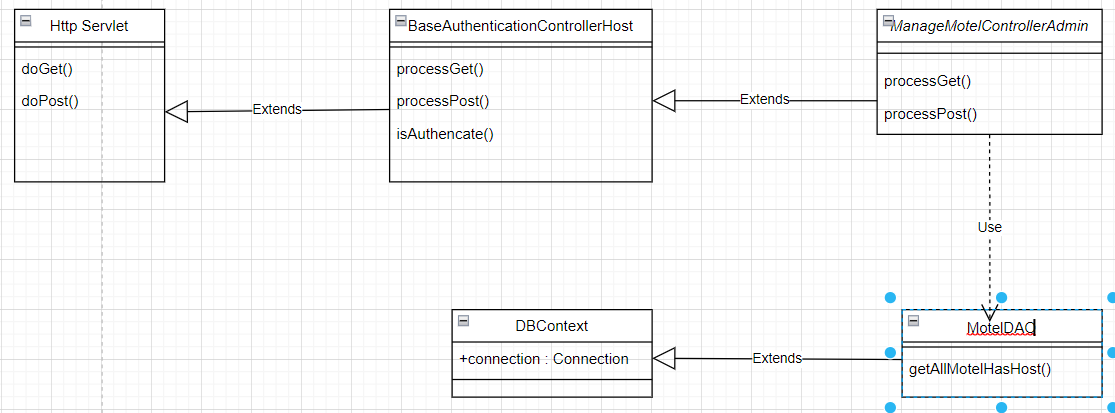
left join RoleOfAccount as roa on a.AccountID = roa.AccountID

left join [Role] as r on r.RoleID = roa.RoleID

where roa.isDeleteRoleOfAccount != 1 order by a.AccountID asc

## 6. Manage Motel by Admin

### a. Class Diagram



### b. Class Specifications

#### ManageMotelAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will call function getAllMotel to take all the information of all motel and transfer it to ManageMotel.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelHasHost() | This function will retrieve all information of all motel in database |

#### 

#### DBContext Class

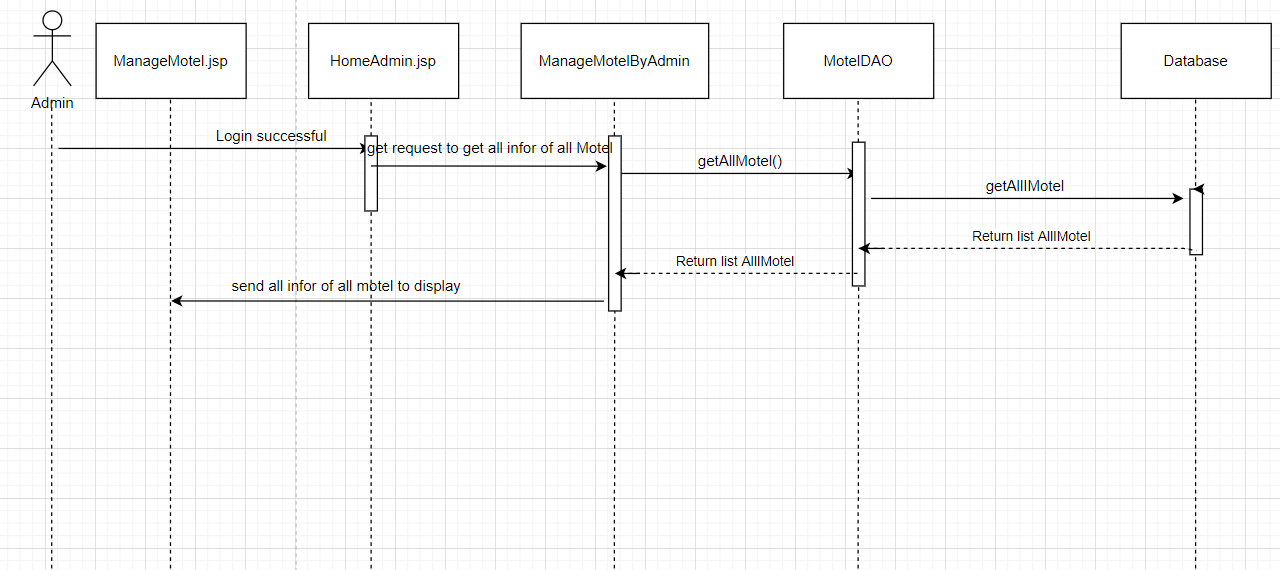
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

### 

### c. Sequence Diagram(s)

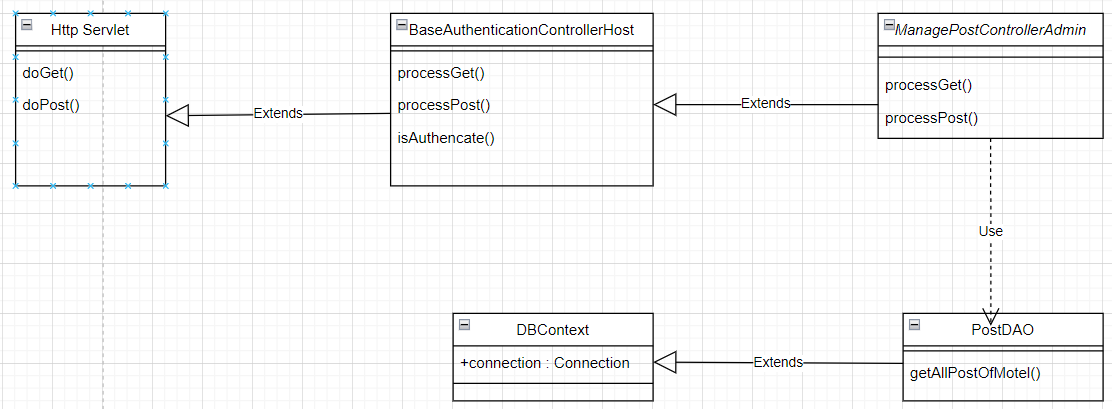


### d. Database queries

* select m.motelID,m.motelName, (select p.fullName from [Profile] as p where p.AccountID = m.AccountID),m.motelPhone,m.motelAddress,(select COUNT(\*) from Room as ro where ro.motelID = m.motelID), m.motelStatus,(select a.userName from Account as a where a.AccountID = m.AccountID) from motel as m

## 7. Manage Post by Admin

### a. Class Diagram



### b. Class Specifications

#### ManagePostAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will call function getAllPost to take all the information of all post and transfer it to ManagePost.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPostOfMotel | This function will retrieve all information of all postin database |

#### 

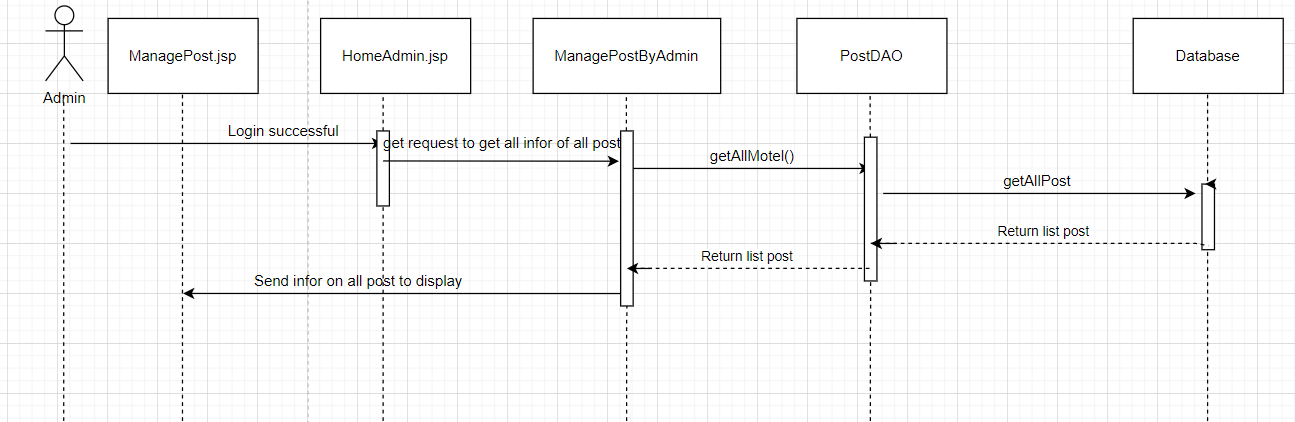
#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

### c. Sequence Diagram(s)



### d. Database queries

select p.postID, p.postDate,p.postTitle,m.motelName,pro.fullName,"

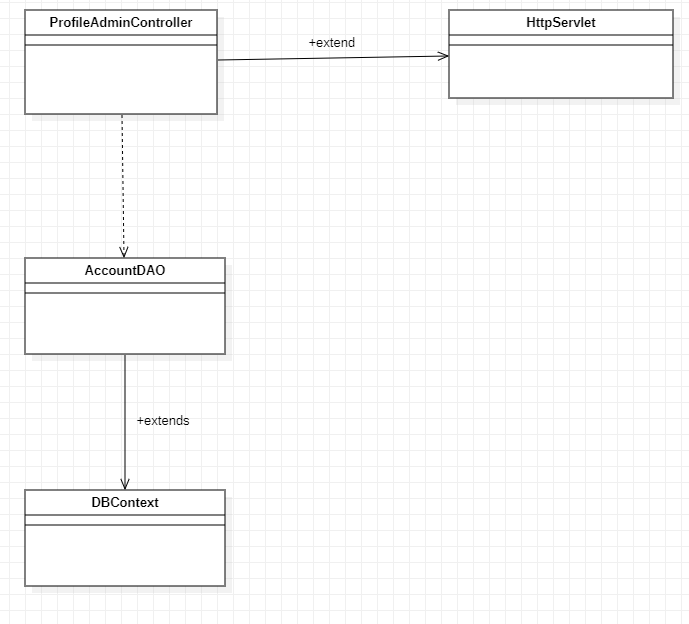
acc.userName,p.postStatus from Post as p left join Motel as m on p.motelID = m.motelID\n"

left join [Profile] as pro on pro.AccountID = m.AccountID \n"

left join Account as acc on acc.AccountID = m.AccountID

## **8.** Manage **Profile Account**

### Class ***Diagram***



### b. Class Specifications

#### ProfileAdminController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | link with file profile.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### AccountDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAccountByUsernamePasswordAndRole | Function get all the information of the account (full name, gender, role,...) |

#### 

#### DBContext Class

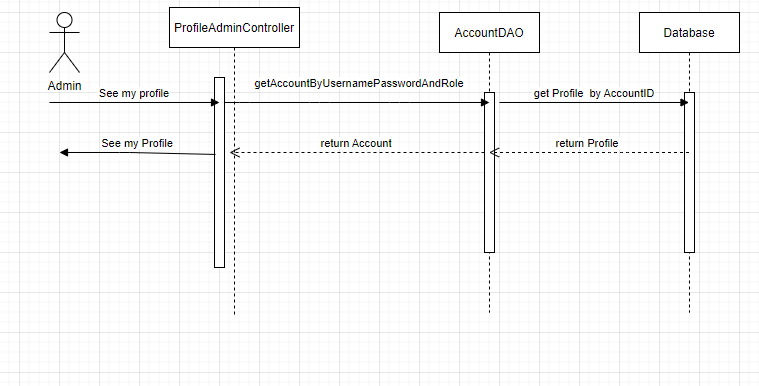
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

#### HttpServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet |  |
| 02 | doPost | Using this HttpServlet object, we can build Java web applications to handle requests from clients with HTTP methods like GET, POST, HEAD, PUT, DELETE, TRACE and OPTIONS. When the client sends a request to the web server container, the service() method in the HttpServlet object processes the request. Depending on the HTTP method that the client sends, further processing will be done by the doGet(), doPost(), doHead(), doPut(), doDelete(), doTrace() and doOptions() methods of this object. receive. |
| 03 | doHead |
| 04 | doPut |
| 05 | doDelete |
| 06 | doOption |
| 07 | doTrace |

#### 

### c. Sequence Diagram(s)



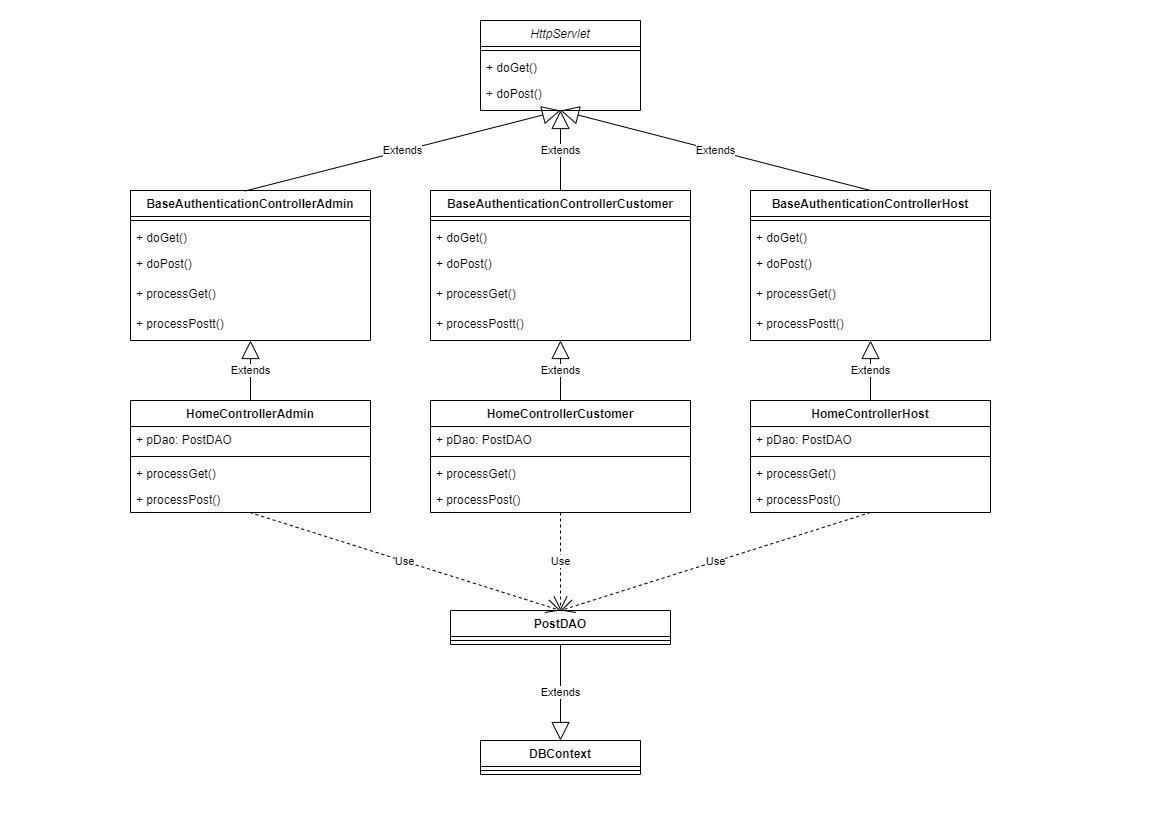
### d. Database queries

Select \* from Profile where AccountID = ?

## 

## 9. Search post

### a. Class Diagram



### b. Class Specifications

#### HomeControllerAdmin Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will return the login.jsp page according to the corresponding role |
| 02 | doPost | input is request and response. After receiving the request from the user, it will get the username and password and call the function getAccountByUsernamePasswordAndRole(username, password, role) of the AccountDAO class. |

#### HttpServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet |  |
| 02 | doPost | Using this HttpServlet object, we can build Java web applications to handle requests from clients with HTTP methods like GET, POST, HEAD, PUT, DELETE, TRACE and OPTIONS. When the client sends a request to the web server container, the service() method in the HttpServlet object processes the request. Depending on the HTTP method that the client sends, further processing will be done by the doGet(), doPost(), doHead(), doPut(), doDelete(), doTrace() and doOptions() methods of this object. receive. |
| 03 | doHead |
| 04 | doPut |
| 05 | doDelete |
| 06 | doOption |
| 07 | doTrace |

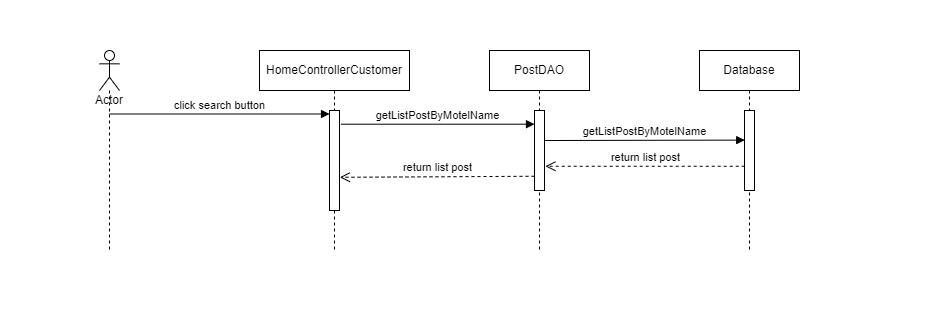
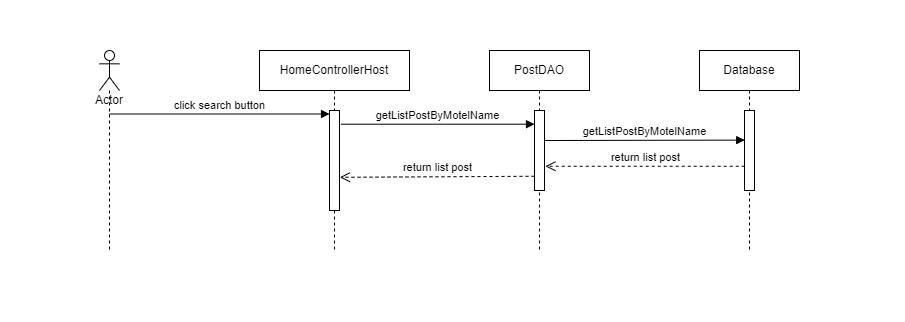
#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPost | Get all post in database |
| 02 | getAllPostByMotelName | Get all post in database which have similar to input motel name |

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)



### d. Database queries

* select pst.postID, pst.motelID, pst.postContent, pst.postDate, pst.postTitle, pst.postStatus, mtl.motelName, pfl.fullName from Post pst left join Motel mtl on mtl.motelID=pst.motelID left join Profile pfl on pfl.AccountID=mtl.AccountID where mtl.motelName like ?;

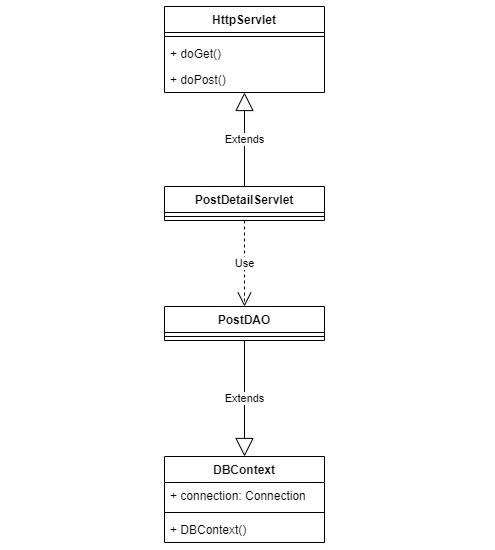
## 

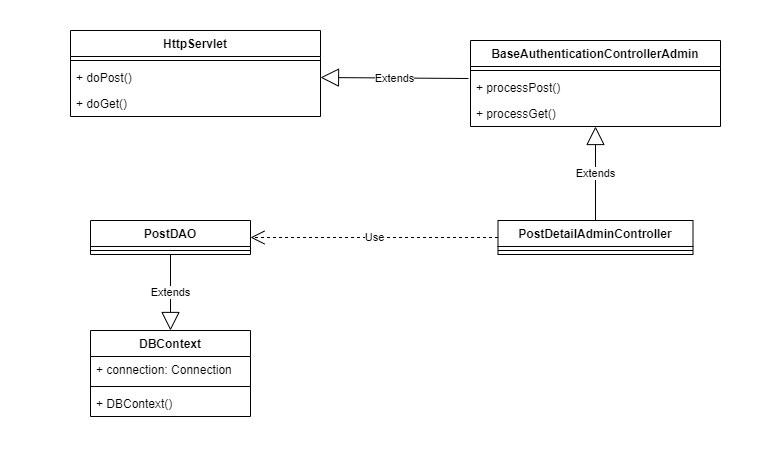
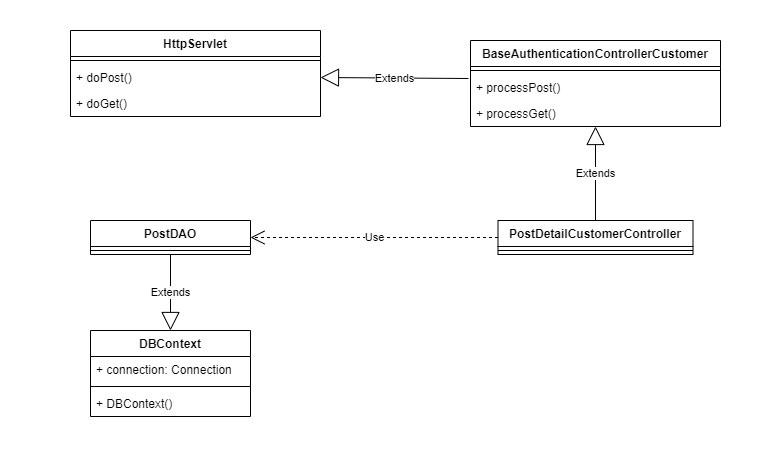
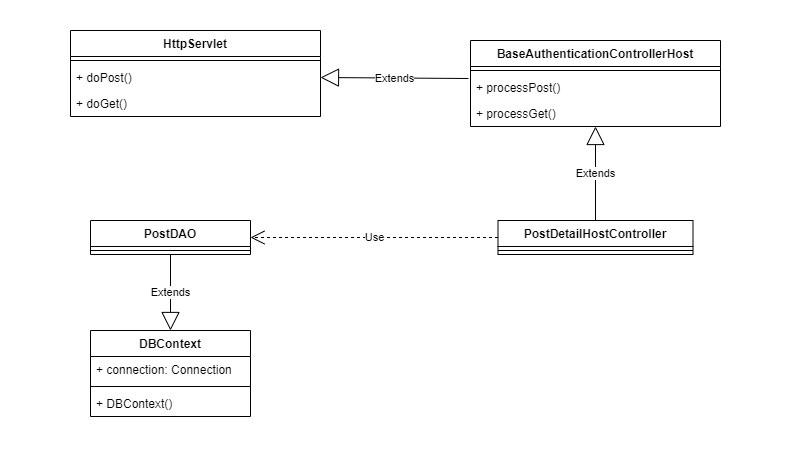
## 

## 10. Post Detail

### a. Class Diagram

* Normal post detail view



* Admin post detail
* Customer post detail
* Host post detail

### b. Class Specifications

PostDetailServlet class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Display content detail |

PostDAO class

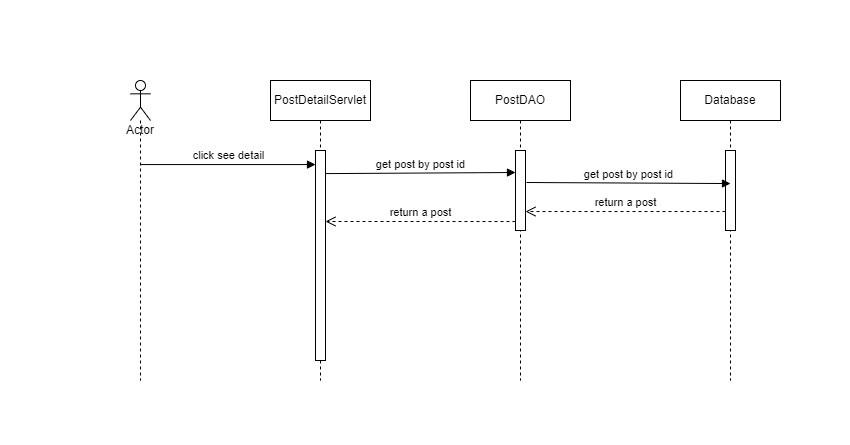
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getPostByPostID | Get post by post ID |

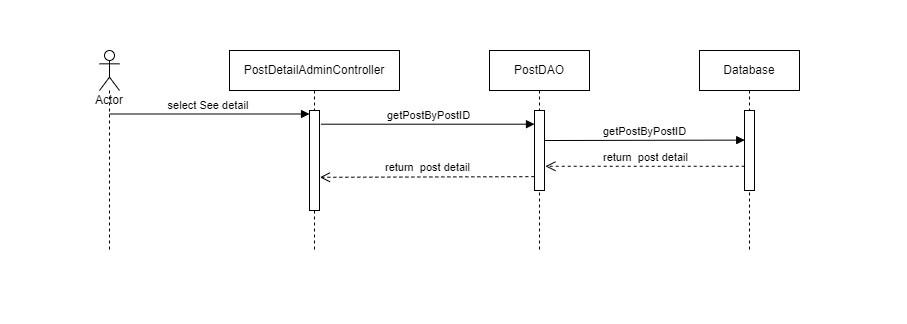
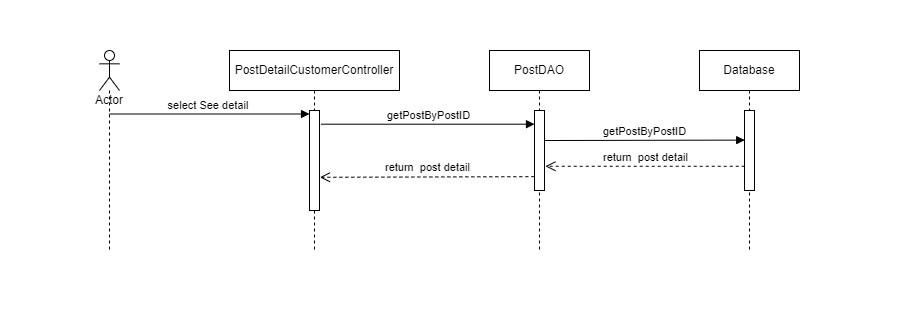
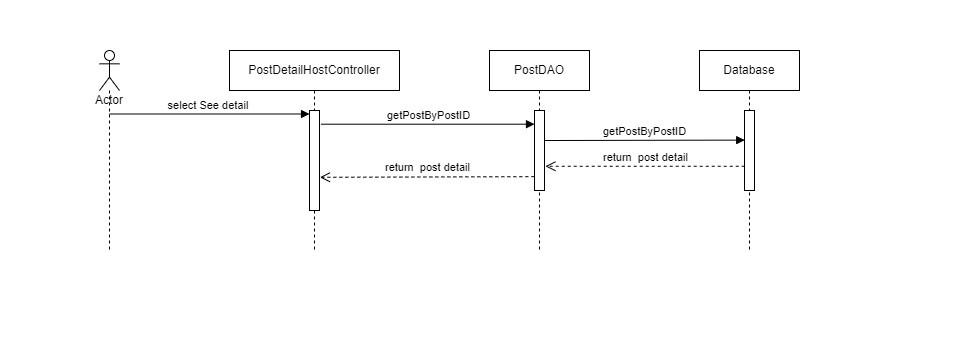
DBContext class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Connecting system with database |

### c. Sequence Diagram(s)

* Normal post detail view



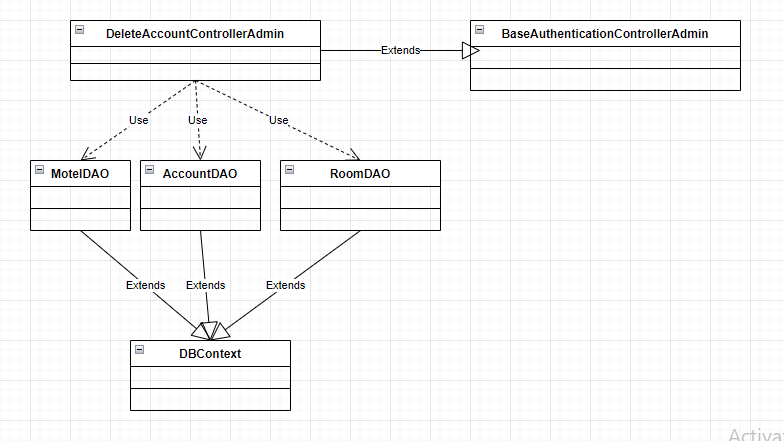
* Admin post detail
* Customer post detail
* Host post detail

### d. Database queries

* select \* from Post where postID = ?;

## **11. Manage Account by Admin/Delete**

### **a*.Class Diagram***

****

### b.Class Specifications

***DeleteAccountControllerAdmin class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Delete accounts with different roles except for the one that is logged in |

***MotelDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get list motels of account by AccountID |
| 02 | deleteMotel | Delete motel by accountID and list rooms of this motel |

***AccountDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllInforOfAccount | Get all account information |
| 02 | deleteAccountByAccountId | Delete the account violated by admin |
| 03 | deleteAccountOfBooking | Delete bookings by account |

***RoomDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getRoomidByMotelid | Get list room of motel by motelID |

***DBContext class***

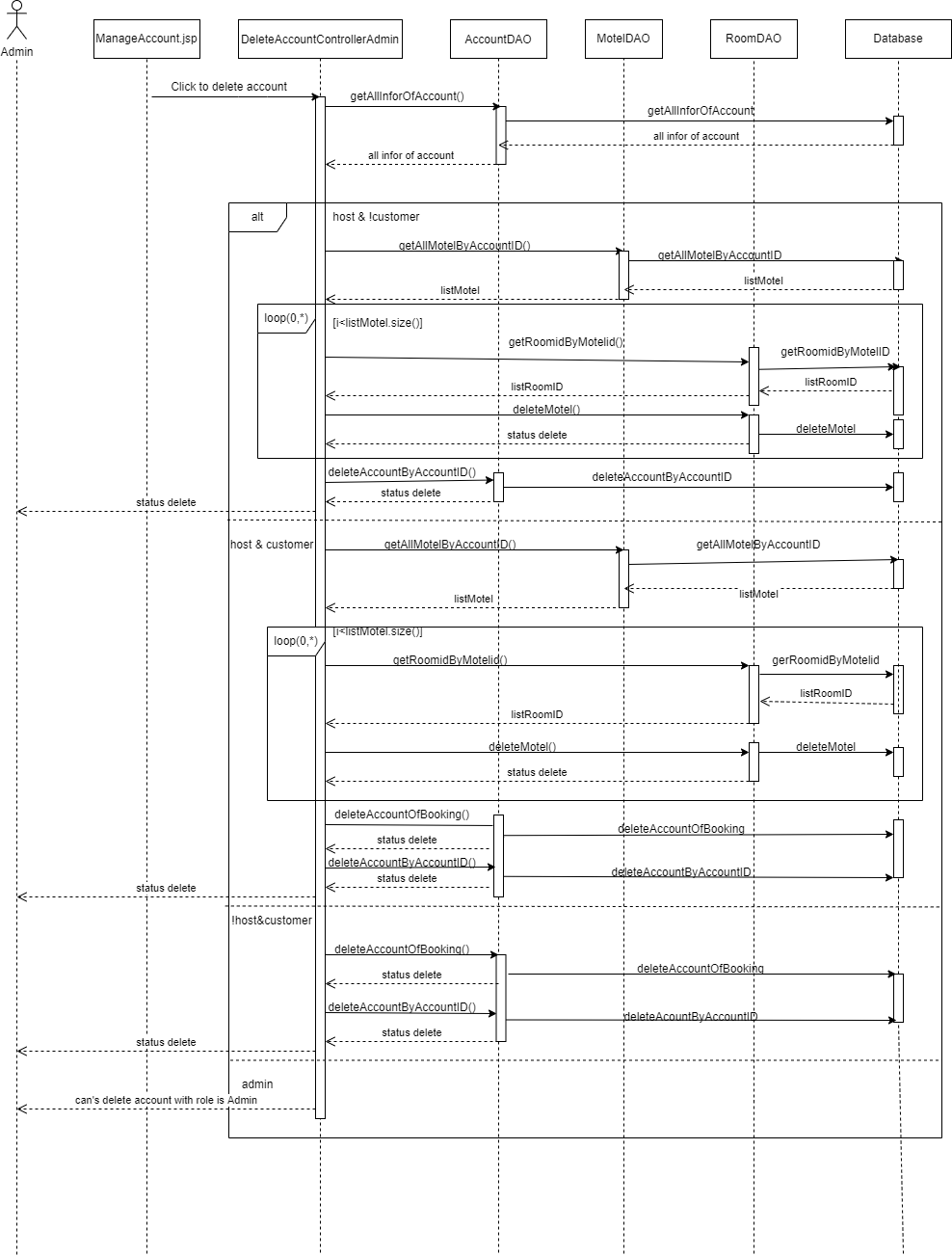
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the  role matches or not |

#### 

### c.Sequence Diagram(s)



### d.Database queries

select \* from Account where AccountID = ?

select \* from [Profile] where AccountID = ?

select \* from RoleOfAccount where AccountID = ?

select \* from [Role] where RoleID = ?

select \* from Motel where AccountID = ?

delete from RoleOfAccount where AccountID = ?

delete from [Profile] where AccountID = ?

delete from Account where AccountID = ?

select roomID from Room where motelID = ?

delete from AccountOfBooking where BookingID = ?

delete from BillDetail where billID = ?

delete from Bill where BookingID = ?\n delete from ServicesOfBooking where BookingID = ?

delete from Booking where RoomID = ? \n delete from Request where roomID = ?

delete from Room where motelID = ?\n delete from Services where motelID = ?

delete from PostImage where postID = ?

delete from Post where motelID =?

delete from Motel where motelID = ?

**12. Manage Account by Admin/Update**

[***a. Class Diagram***](#_heading=h.1t3h5sf)



[***b. Class Specifications***](#_heading=h.4d34og8)

***UpdateRoleOfAccountControllerAdmin class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Update roles of account by Admin |

***MotelDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get list motels of account by AccountID |
| 02 | deleteMotel | Delete motel by accountID and list rooms of this motel |

***AccountDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllInforOfAccount | Get all account information |
| 02 | updateRoleOfAccount | Add new role for account by Admin |
| 03 | deleteRoleOfAccount | Delete role of account by Admin |

***RoomDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getRoomidByMotelid | Get list room of motel by motelID |

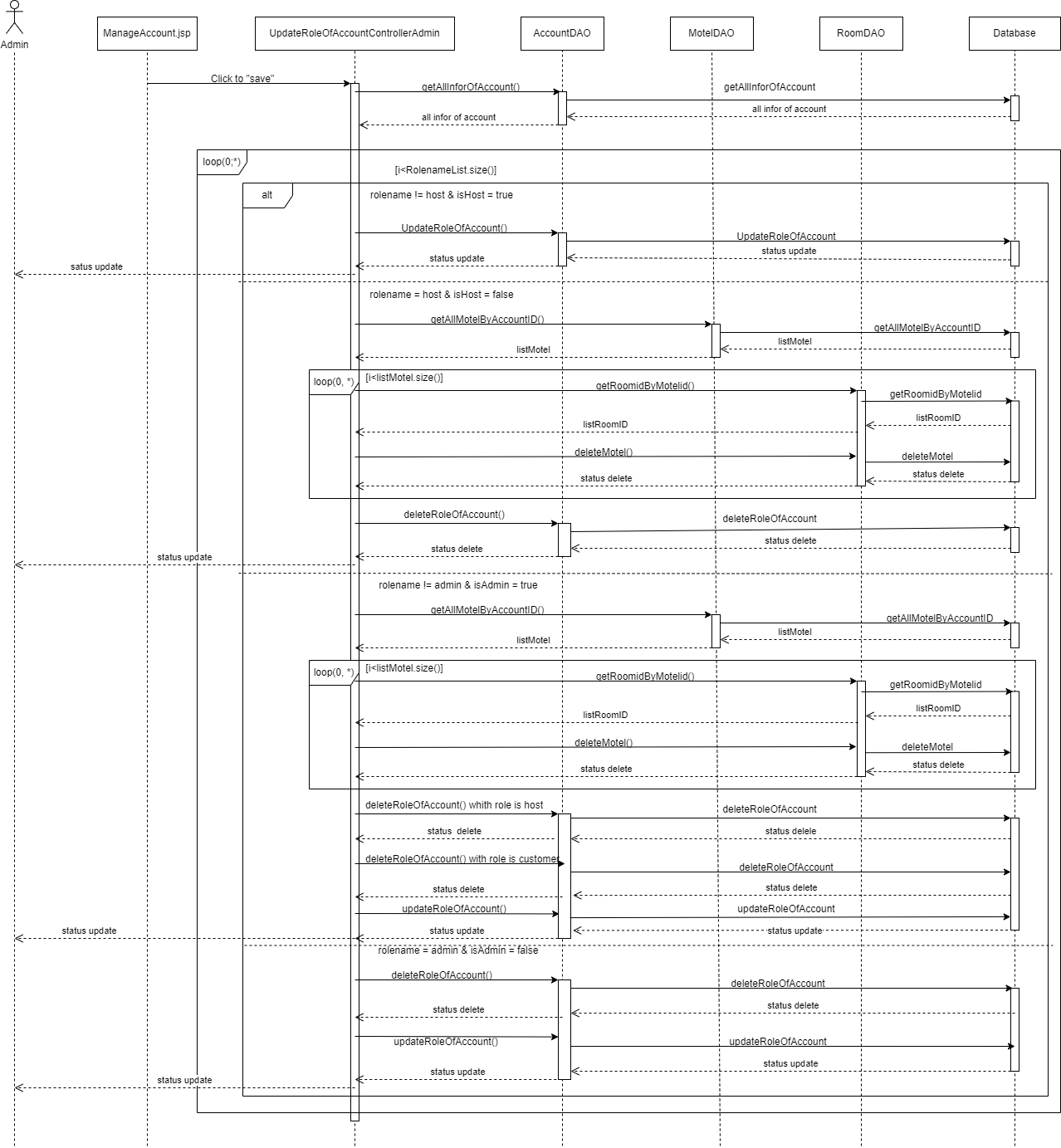
***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the  role matches or not |

[***c. Sequence Diagram(s)***](#_heading=h.2s8eyo1)



[***d. Database queries***](#_heading=h.17dp8vu)

select \* from Account where AccountID = ?

select \* from [Profile] where AccountID = ?

select \* from RoleOfAccount where AccountID = ?

select \* from [Role] where RoleID = ?

insert into RoleOfAccount values (?,?)

delete from RoleOfAccount where AccountID = ? and RoleID = ?

select \* from Motel where AccountID = ?

delete from AccountOfBooking where BookingID = ?

delete from BillDetail where billID = ?

delete from Bill where BookingID = ?\n delete from ServicesOfBooking where BookingID = ?

delete from Booking where RoomID = ? \n delete from Request where roomID = ?

delete from Room where motelID = ?\n delete from Services where motelID = ?

delete from PostImage where postID = ?

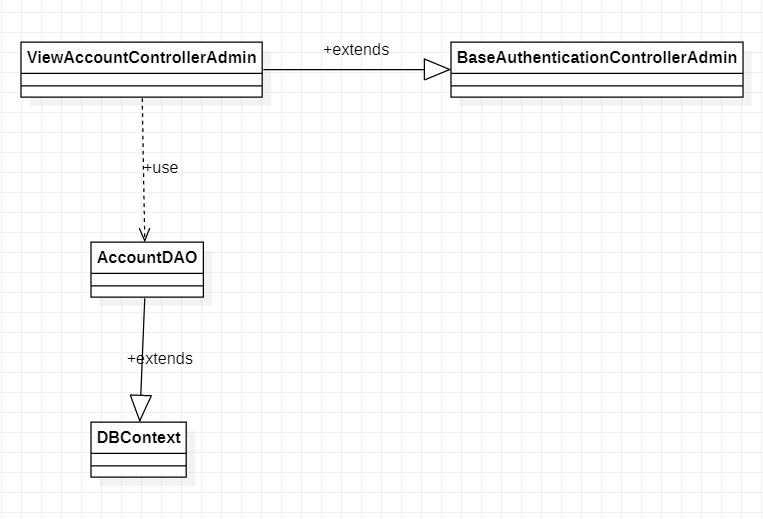
delete from Post where motelID =?

delete from Motel where motelID = ?

select roomID from Room where motelID = ?

## 13. Manage Account by Admin/Search

### Class Diagram



### b. Class Specifications

#### ManageAccountAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will call function getAllInforOfAllAccount to take all the information of all users and transfer it to ManageAdmin.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### AccountDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllInforOfAllAccount | This function will retrieve all information of a user including account profile password .... |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

#### 

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

### c. Sequence Diagram(s)

### 

### d. Database queries

select \* from Account

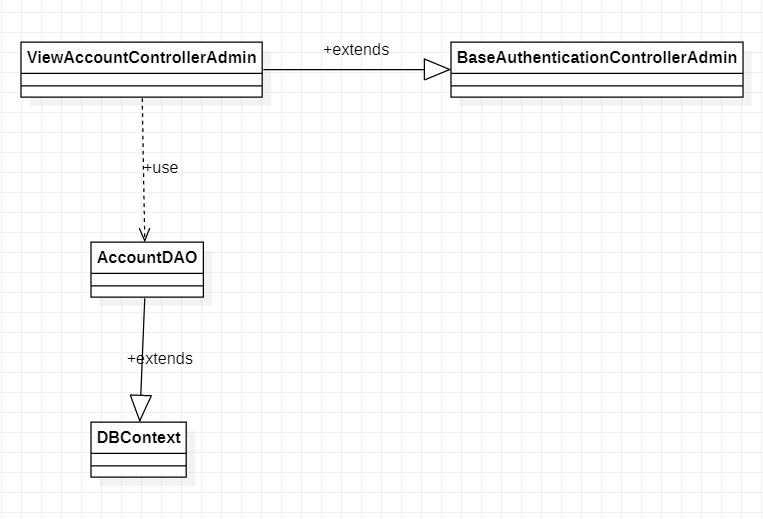
select \* from [Profile] where AccountID = ? (? = get id in list of account)

select \* from RoleOfAccount where AccountID = ?(? = get id in list of account)

select \* from [Role] where RoleID = ?

## 14. Manage Account by Admin/Sort

### a. Class Diagram



### b. Class Specifications

#### ManageAccountAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will call function getAllInforOfAllAccount to take all the information of all users and transfer it to ManageAdmin.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### AccountDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllInforOfAllAccount | This function will retrieve all information of a user including account profile password .... |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

### c. Sequence Diagram(s)

### 

### d. Database queries

select \* from Account

select \* from [Profile] where AccountID = ? (? = get id in list of account)

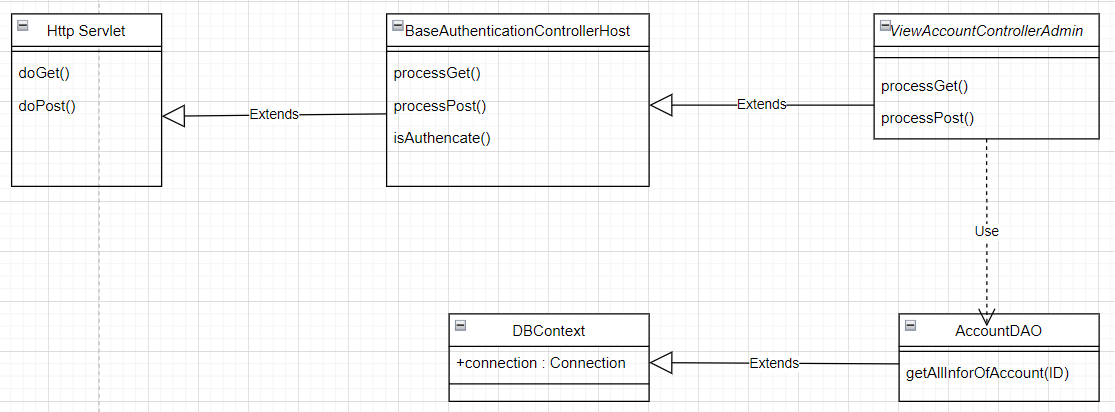
select \* from RoleOfAccount where AccountID = ?(? = get id in list of account)

select \* from [Role] where RoleID = ?

### 

## 15. Manage Account by Admin/View Detail

### a. Class Diagram



### b. Class Specifications

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

#### ViewAccountControllerAdmin

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | get all the information about the account then send to viewAccount.jsp page |

#### 

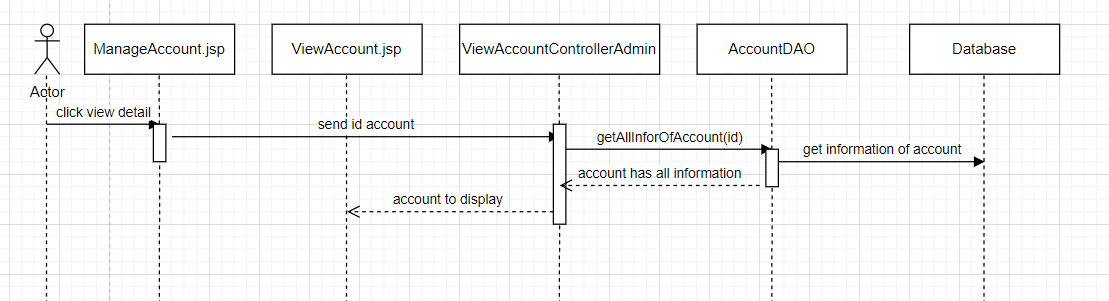
#### AccountDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllInforOfAccount | get all information of one account by accountID |

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)



### d. Database queries

select p.fullName,a.userName,p.DOB,p.Gender,p.phone,p.address,

p.email,roa.RoleID,r.roleName from Account as a

left join [Profile] as p on a.AccountID = p.AccountID

left join RoleOfAccount as roa on a.AccountID = roa.AccountID

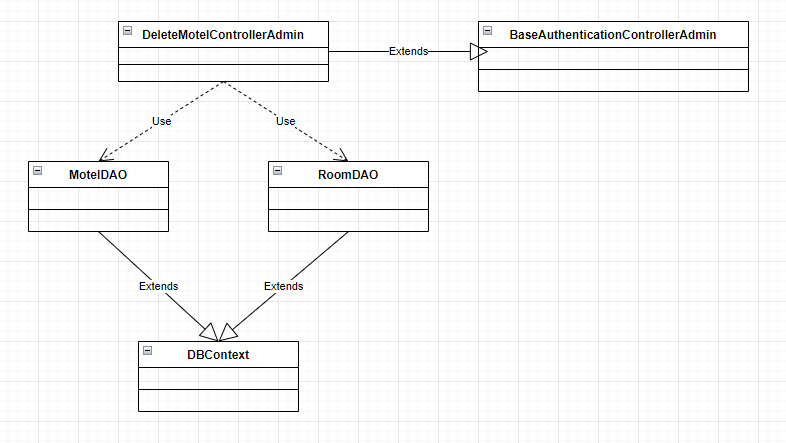
left join [Role] as r on r.RoleID = roa.RoleID

where roa.isDeleteRoleOfAccount != 1 and a.AccountID = ? order by r.RoleID asc

## 16. Manage Motel by Admin/Delete

### 

### a. Class Diagram



### b. Class Specifications

***DeleteMotelControllerAdmin class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Delete Motel violated by admin |

***MotelDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMotelByMotelID | Get a motel by motelID |
| 02 | deleteMotel | Delete motel by accountID and list rooms of this motel |
| 03 | deleteMotelByMotelID | Delete motel by MotelID |

***RoomDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getRoomidByMotelid | Get list room of motel by motelID |

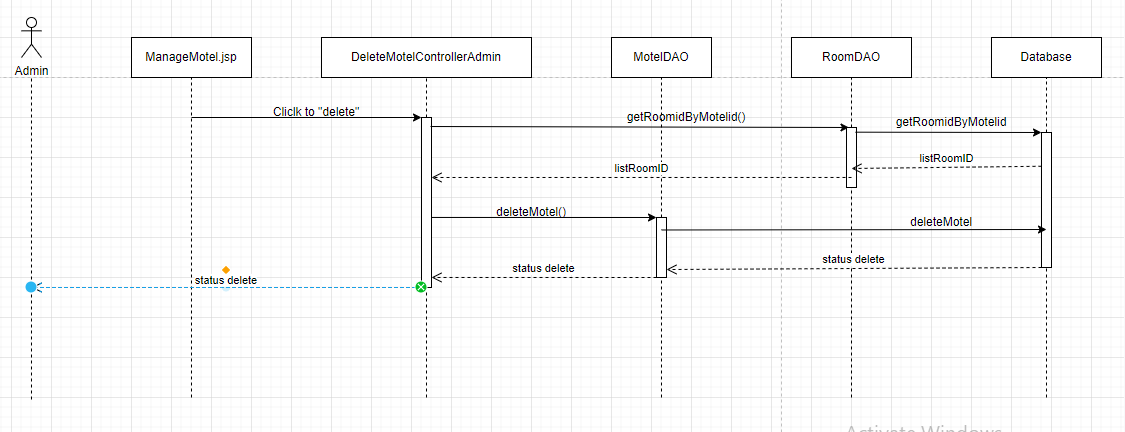
***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the  role matches or not |

### c. Sequence Diagram(s)



### d. Database queries

select \* from Motel where motelID = ?

delete Motel where motelID = ?

delete from AccountOfBooking where BookingID = ?

delete from BillDetail where billID = ?

delete from Bill where BookingID = ?\n delete from ServicesOfBooking where BookingID = ?

delete from Booking where RoomID = ? \n delete from Request where roomID = ?

delete from Room where motelID = ?\n delete from Services where motelID = ?

delete from PostImage where postID = ?

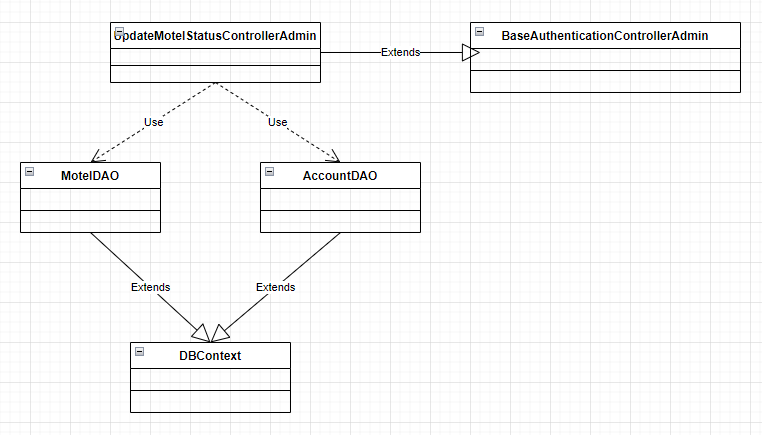
delete from Post where motelID =?

delete from Motel where motelID = ?

select roomID from Room where motelID = ?

## 17. Manage Motel by Admin/Update

### a. Class Diagram



### b. Class Specifications

***UpdateMotelStatusControllerAdmin class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Update status of motel to active by Admin |

***MotelDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMotelByMotelID | Get a motel by motelID |
| 02 | updateMotelStatus | Update status of motel by Admin |

***AccountDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllInforOfAccount | Get all account information by accountID |
| 02 | updateRoleOfAccount | Add new role for account by Admin |

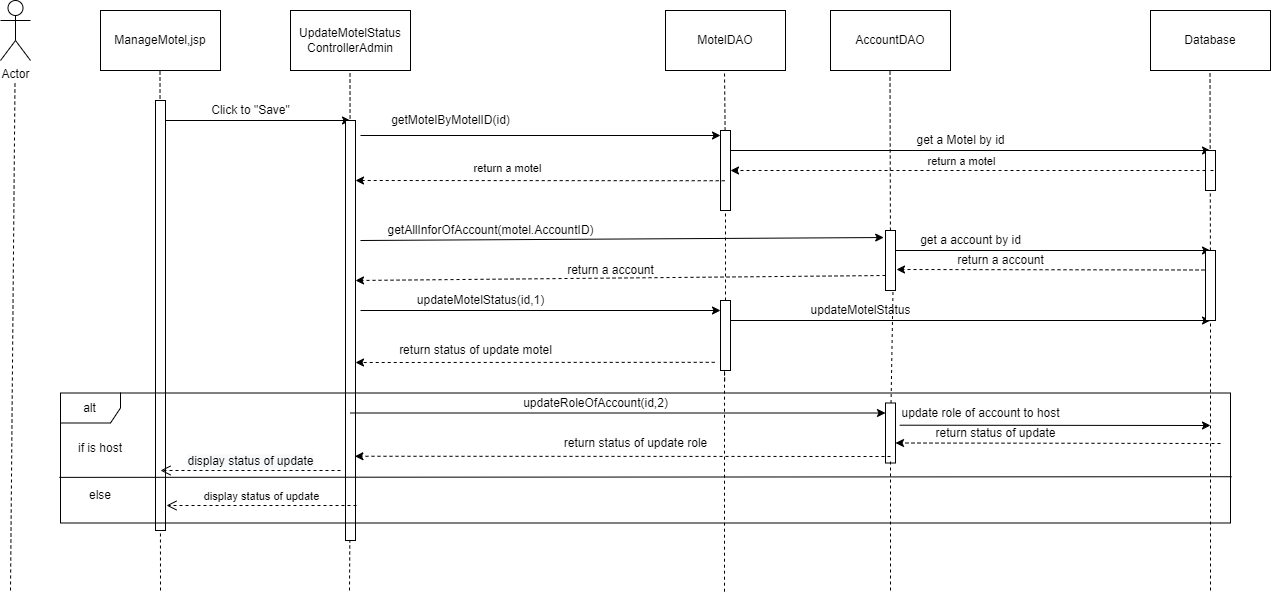
***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the  role matches or not |

### c. Sequence Diagram(s)



### d. Database queries

select \* from Motel where motelID = ?

select \* from Account where AccountID = ?

select \* from [Profile] where AccountID = ?

select \* from RoleOfAccount where AccountID = ?

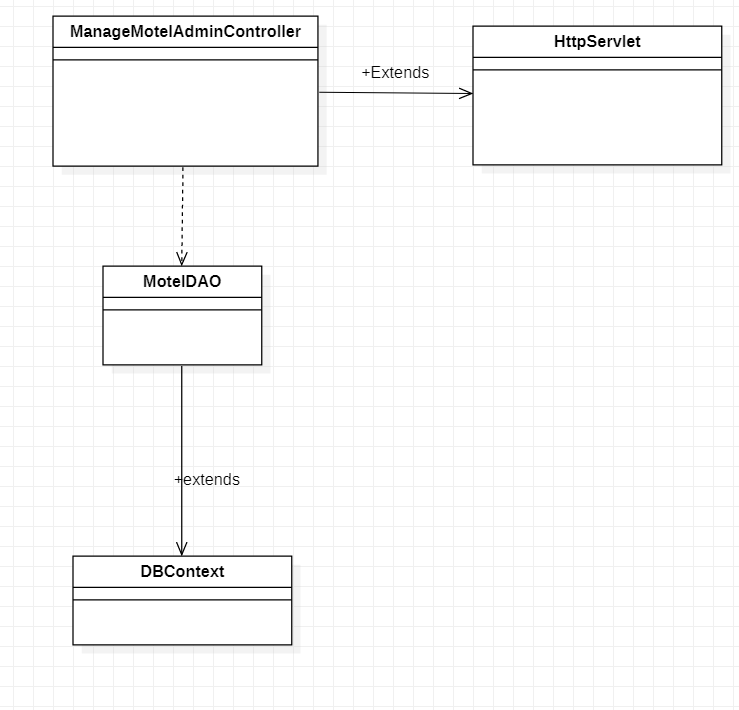
select \* from [Role] where RoleID = ?

update Motel set motelStatus = ? where motelID = ?

insert into RoleOfAccount values (?,?)

## 18. Manage Motel by Admin/Search

### a. Class Diagram



### b. Class Specifications

#### ManageMotelAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will call function getAllMotel to take all the information of all motel and transfer it to ManageMotel.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotel | This function will retrieve all information of all motel in database |
| 02 | getNumberOfRoomOneMotelByMotelID | input is motelID , this will calculate number of room in a motel and return number of room |
| 03 | getMotelByMotelID | input is motelID, it will get all information of a motel by motelID |

#### 

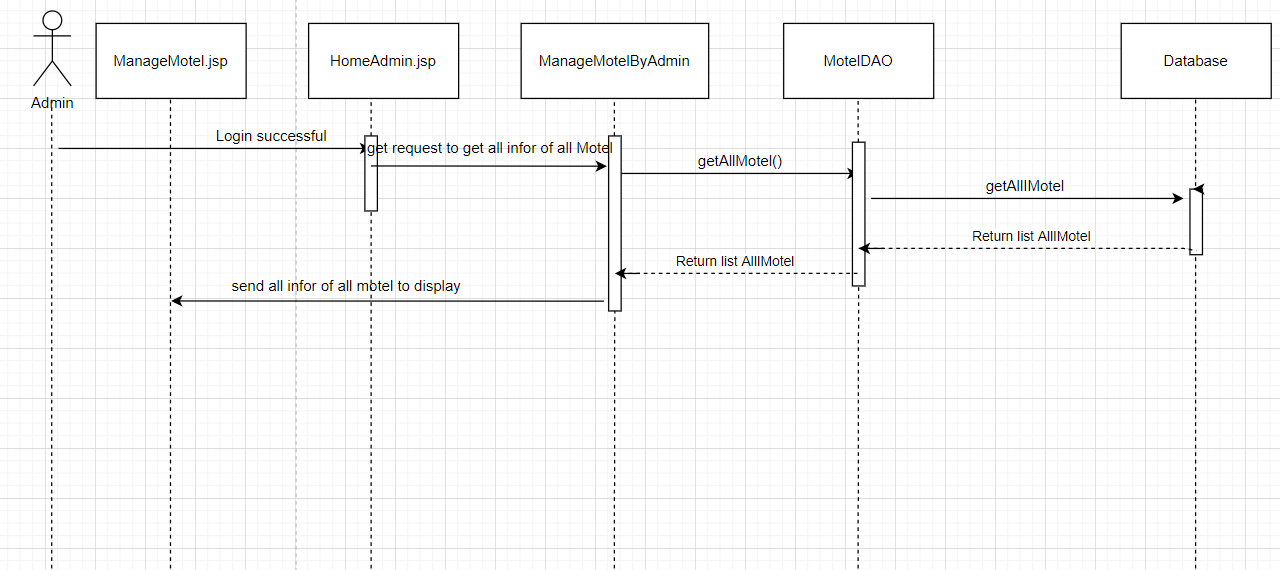
#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

### c. Sequence Diagram(s)



### d. Database queries

select \* from motel

select count(\*) from Room where motelID = ?

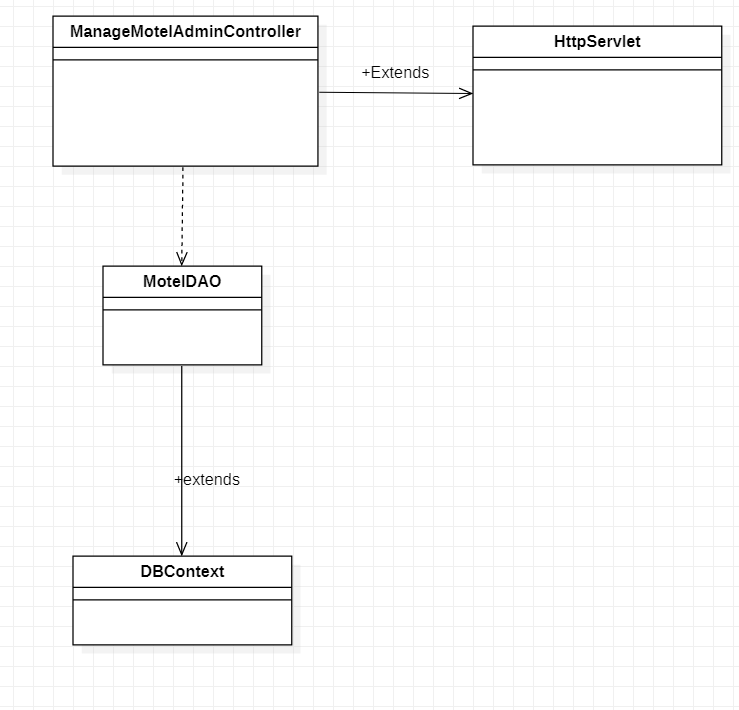
select \* from Motel where motelID = ?

### 

## 

## 19. Manage Motel by Admin/Sort

### a. Class Diagram



### b. Class Specifications

#### ManageMotelAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will call function getAllMotel to take all the information of all motel and transfer it to ManageMotel.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotel | This function will retrieve all information of all motel in database |
| 02 | getNumberOfRoomOneMotelByMotelID | input is motelID , this will calculate number of room in a motel and return number of room |
| 03 | getMotelByMotelID | input is motelID, it will get all information of a motel by motelID |

#### 

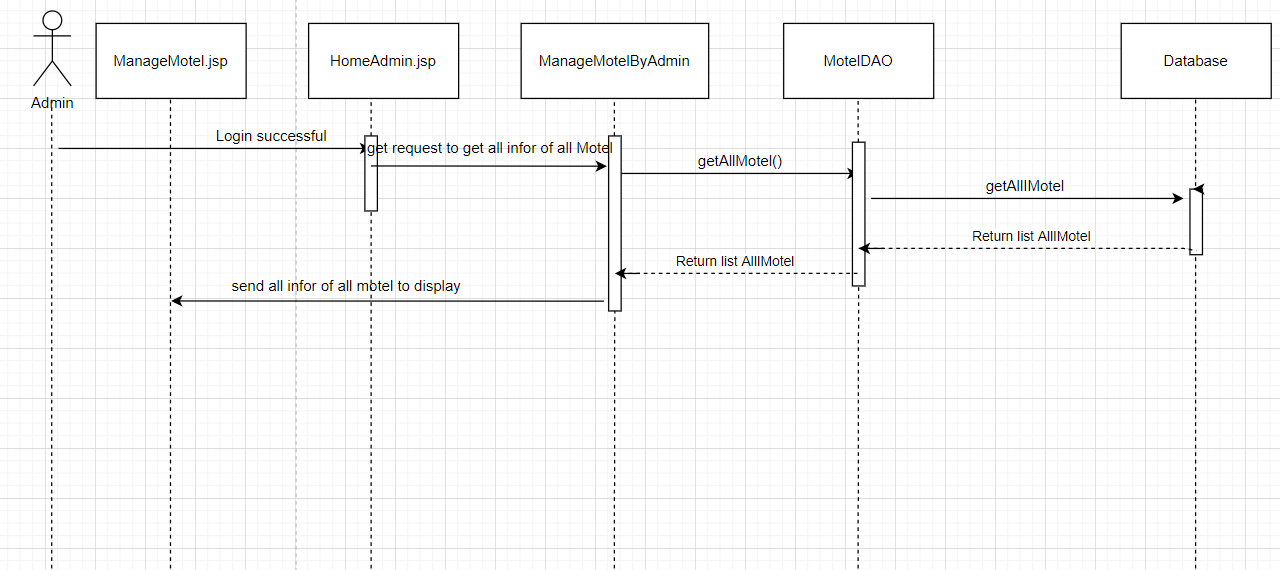
#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

### c. Sequence Diagram(s)



### d. Database queries

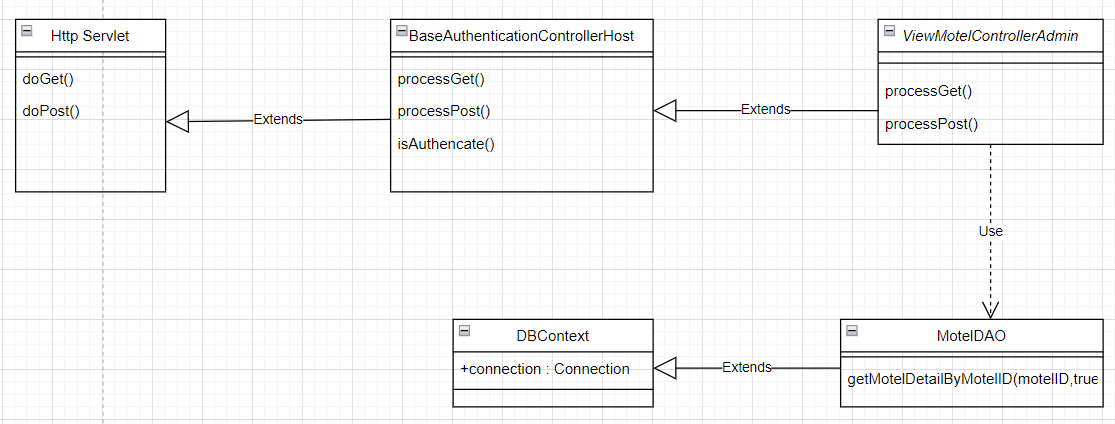
select \* from motel

select count(\*) from Room where motelID = ?

select \* from Motel where motelID = ?

## 20. Manage Motel by Admin/View Detail

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

**ViewMoteControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | get all information of motel then go to ViewMotel.jsp |

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMotelDetailByMotelID(motelID,true) | Get all information of motel in database |

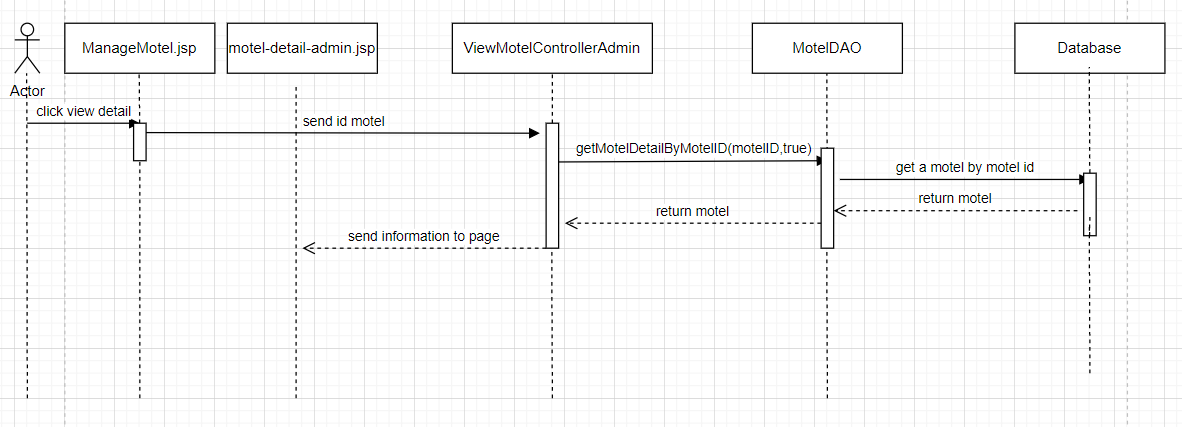
#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)



### d. Database queries

select m.motelID, m.accountID,(select p.fullName from

[Profile] as p where p.AccountID = m.AccountID), m.motelName,

(select count(\*) from Room as r where r.motelID = m.motelID),

m.motelAddress, m.motelPhone, m.motelStatus,

(select MAX(r.price) from Room as r where r.motelID = m.motelID),

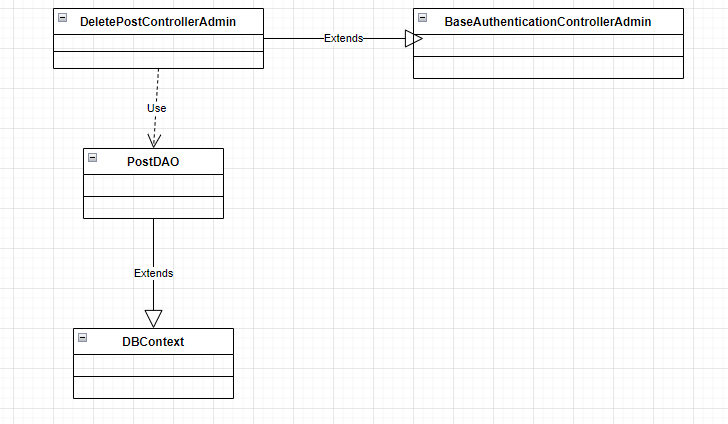
(select MIN(r.price) from Room as r where r.motelID = m.motelID),

m.motelDescription,m.motelImage, img.imageLink from Motel as m left

join MotelImage as img on img.motelID = m.motelID where m.motelID = ?

## 21. Manage Post by Admin/Delete

### a. Class Diagram



### b. Class Specifications

***DeletePostControllerAdmin class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Delete post violate by Admin |

***PostDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deletePostByPostID | Delete post by postID |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the  role matches or not |

### c. Sequence Diagram(s)



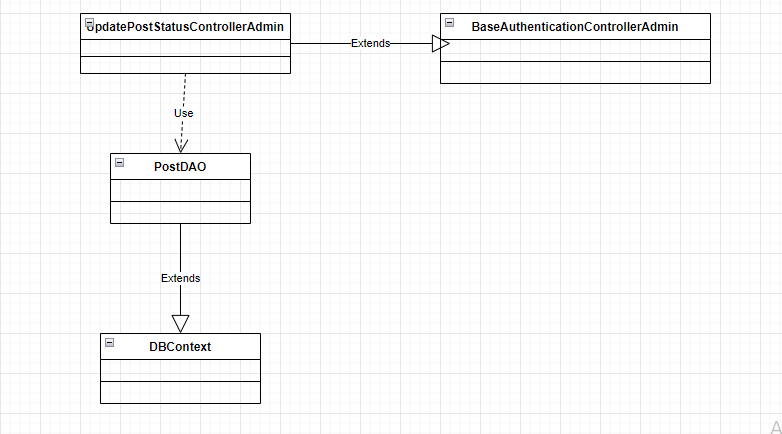
### d. Database queries

delete PostImage where postID = ?

delete Post where postID = ?

## 22. Manage Post by Admin/Update

### a. Class Diagram



### b. Class Specifications

***UpdatePostStatusControllerAdmin class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Update status of post by Admin |

***PostDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updatePostStatus | Update status of post by postID |

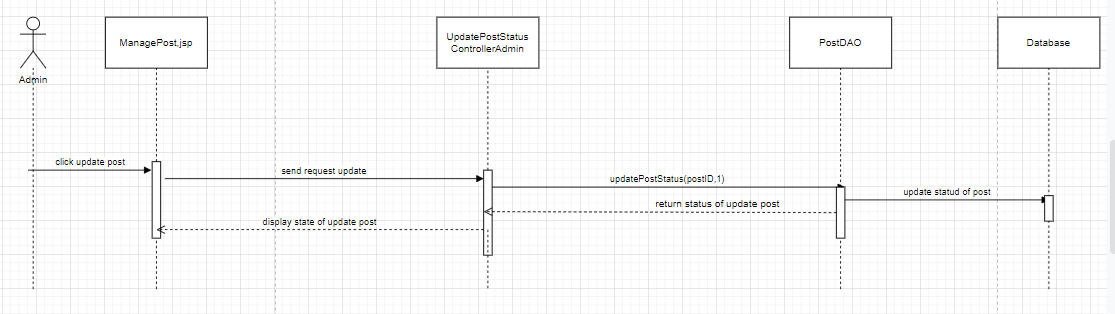
***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the  role matches or not |

### c. Sequence Diagram(s)

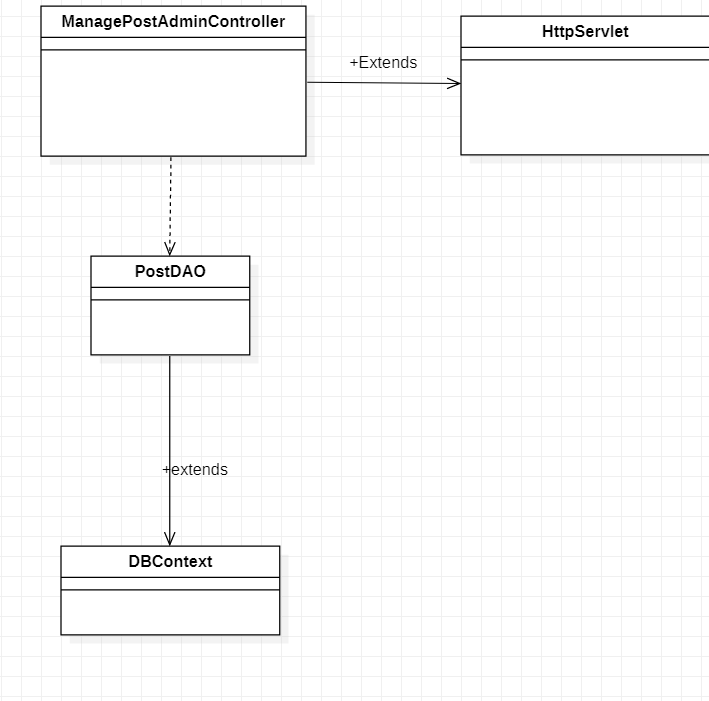


### d. Database queries

update Post set postStatus = ? where postID = ?

## 23. Manage Post by Admin/Search

### a. Class Diagram



### b. Class Specifications

#### ManagePostAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will call function getAllPost to take all the information of all post and transfer it to ManagePost.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPost | This function will retrieve all information of all postin database |

#### 

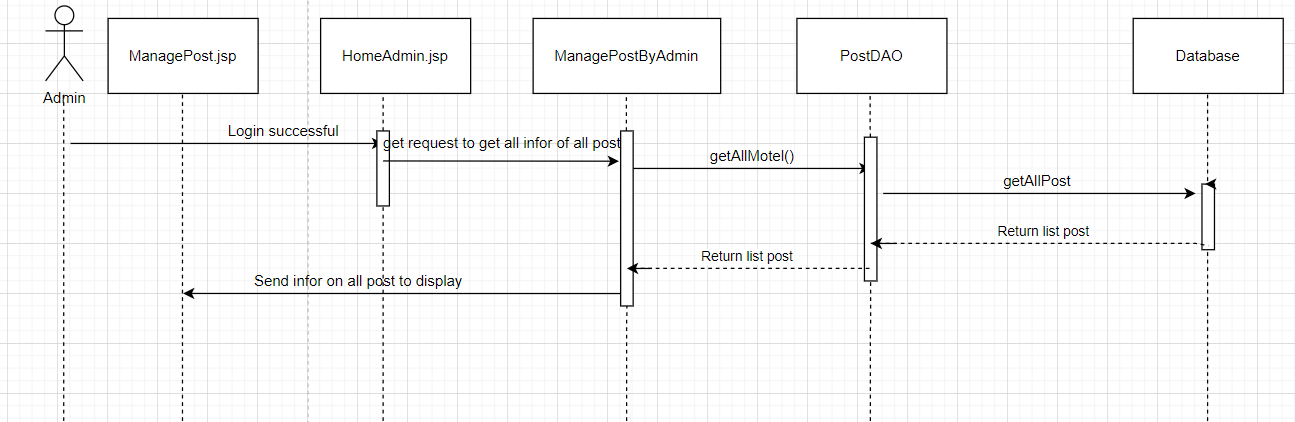
#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

### c. Sequence Diagram(s)



### d. Database queries

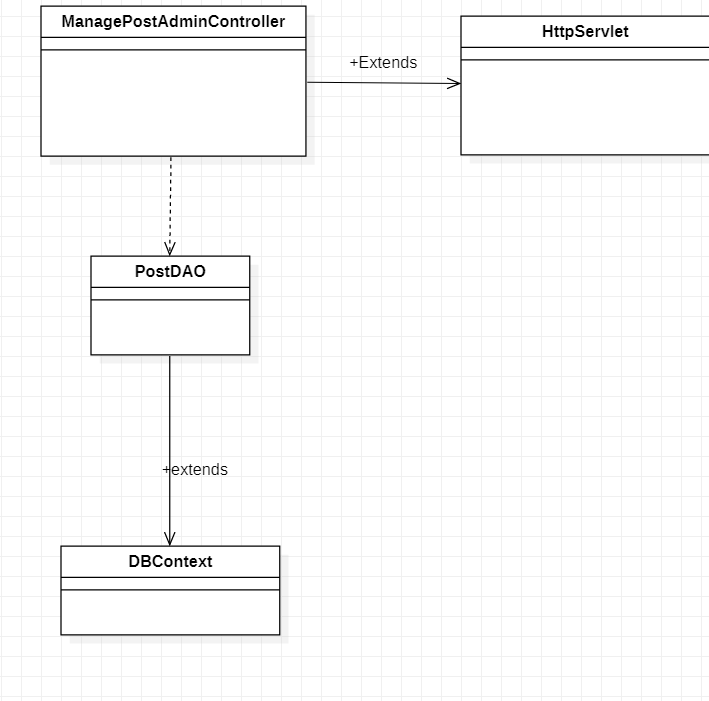
select \* from Post

## 

## 

## 24. Manage Post by Admin/Sort

### a. Class Diagram



### b. Class Specifications

#### ManagePostAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will call function getAllPost to take all the information of all post and transfer it to ManagePost.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPost | This function will retrieve all information of all postin database |

#### 

#### DBContext Class

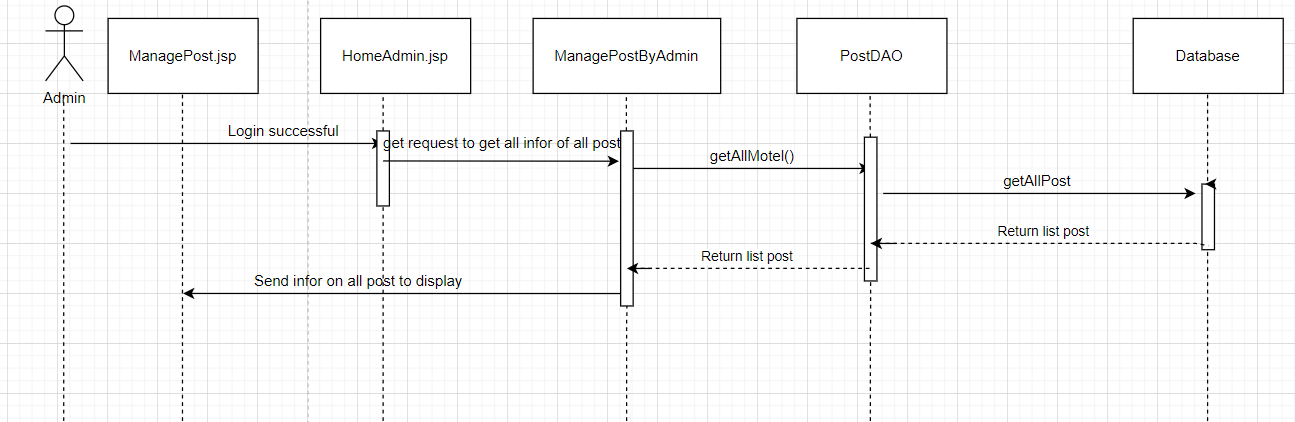
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

#### 

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

### c. Sequence Diagram(s)

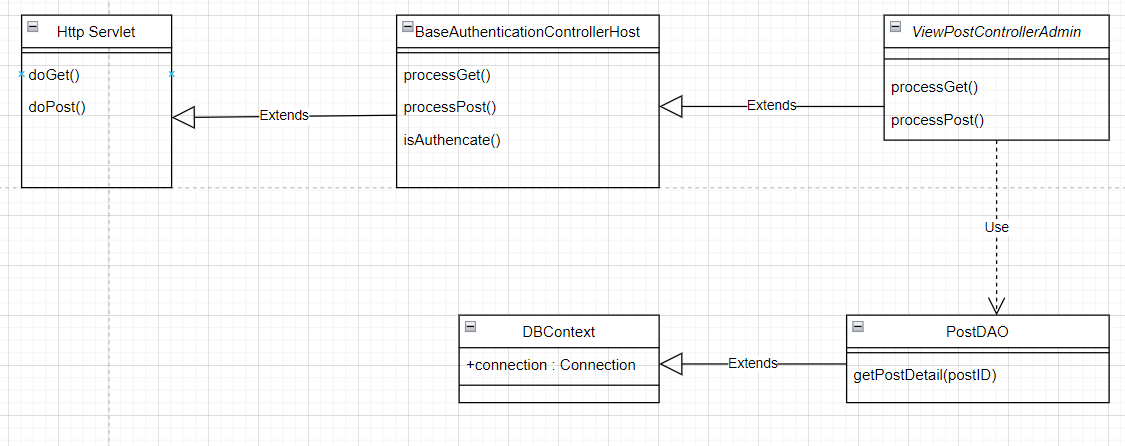


### d. Database queries

select \* from Post

## 25. Manage Post by Admin/View Detail

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

**ViewPostControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | get all information of post then go to ViewPost.jsp |

#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getPostDetail(postID) | Get all information of one post in database |

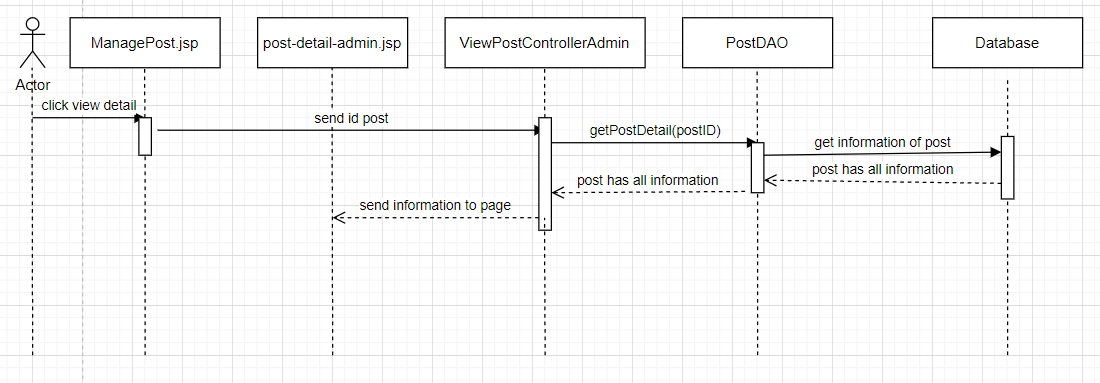
#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)

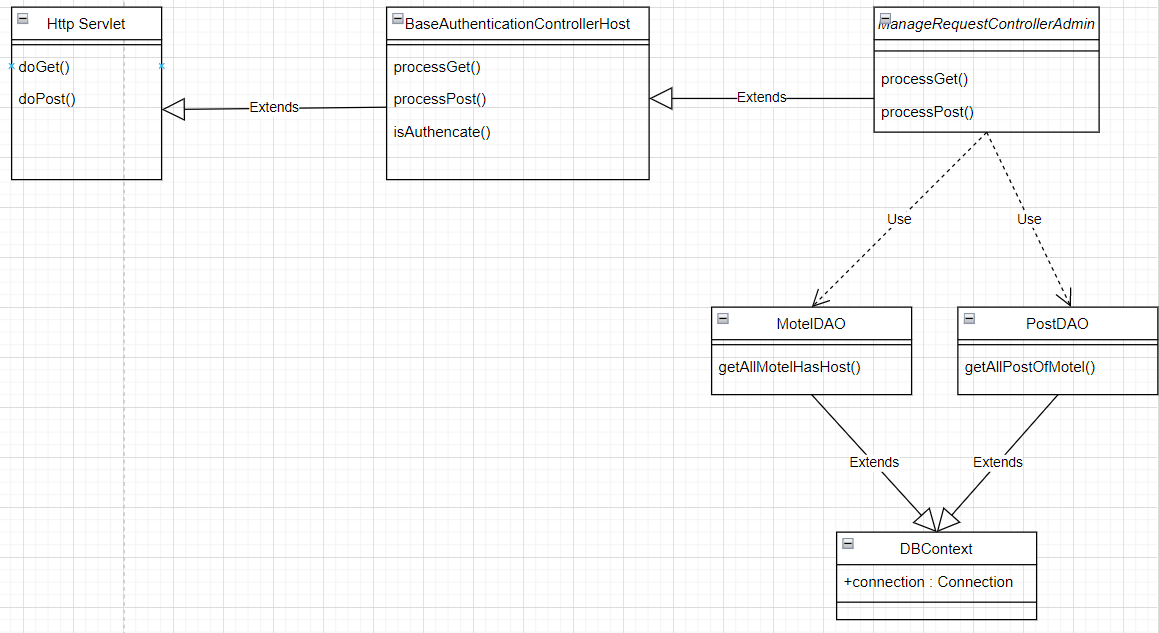


### d. Database queries

* select pt.postID, pt.motelID, mt.AccountID, mt.motelName, pt.postTitle, pt.postContent, (select max(price) from Room where motelID = pt.motelID) as maxPrice, (select min(price) from Room where motelID = pt.motelID) as minPrice, mt.motelPhone, mt.motelAddress, pt.postDate, pt.postStatus, pim.postImageID, pim.postImageLink from Post pt left join Motel mt on pt.motelID = mt.motelID left join PostImage pim on pim.postID = pt.postID where pt.postID = ?;

## 26. Manage Request by Admin

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

**ManageequestControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | get all information of post and motel then go to Manageequest.jsp |

#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPostOfMotel | Get all information of all post in database |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelHasHost | Get all information of all motel in database |

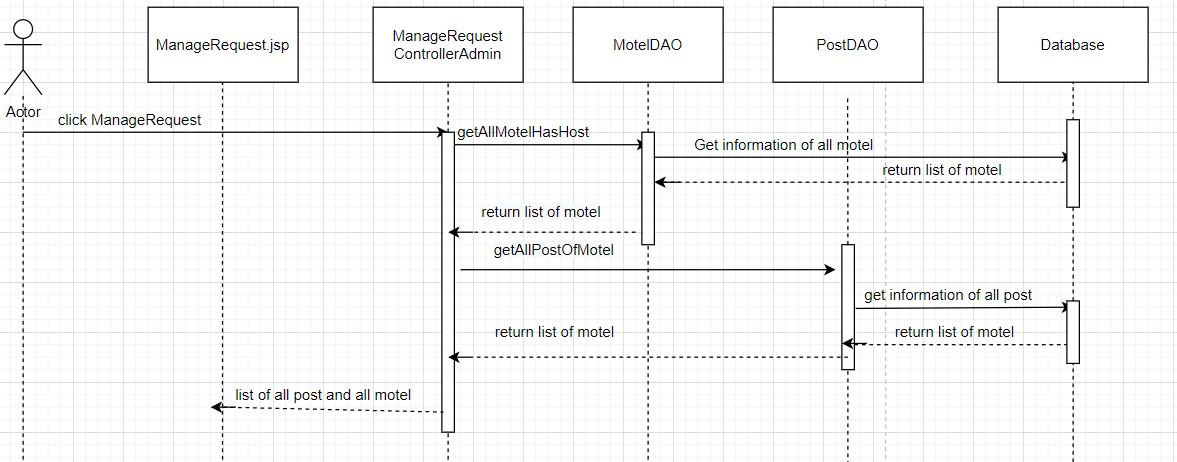
#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)

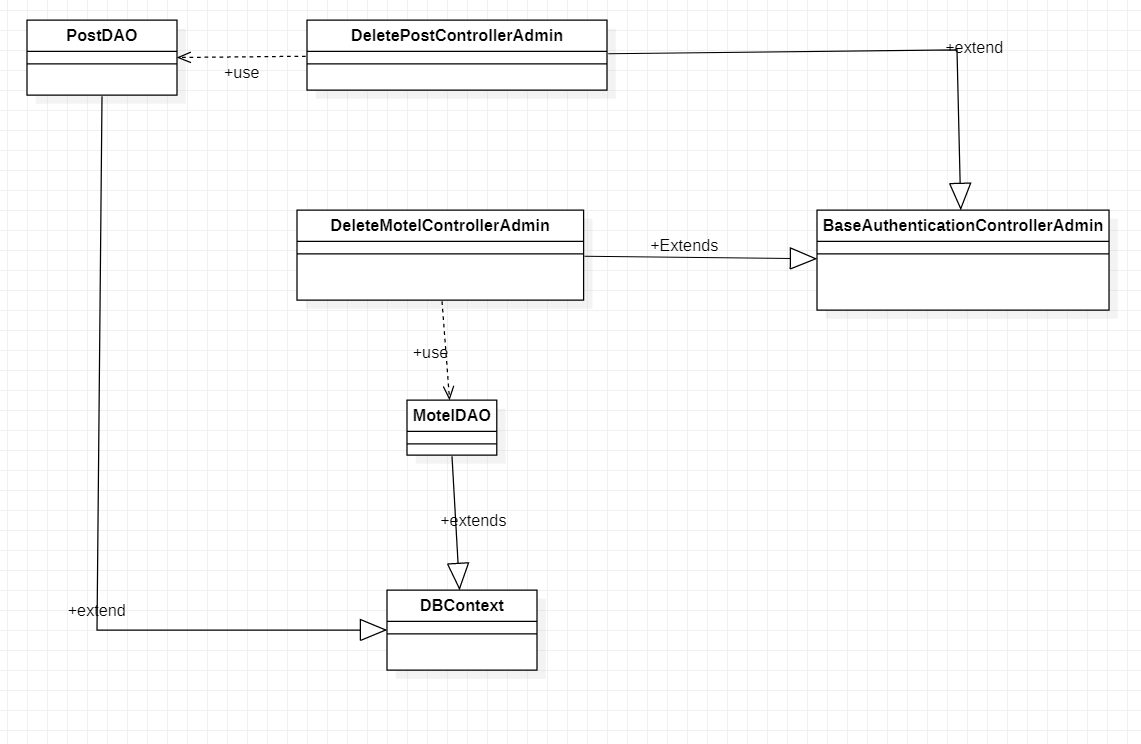


### d. Database queries

* select m.motelID,m.motelName, (select p.fullName from [Profile] as p where p.AccountID = m.AccountID), m.motelPhone,m.motelAddress, (select COUNT(\*) from Room as ro where ro.motelID = m.motelID) , m.motelStatus,(select a.userName from Account as a where a.AccountID = m.AccountID) from motel as m
* select p.postID, p.postDate,p.postTitle,m.motelName,pro.fullName, acc.userName,p.postStatus from Post as p left join Motel as m on p.motelID = m.motelID left join [Profile] as pro on pro.AccountID = m.AccountID left join Account as acc on acc.AccountID = m.AccountID

## 27. Manage Request by Admin/Delete

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

#### DeleteMotelControllerAdmin

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | delete motel that have not accept by motel id and then go to Manageequest.jsp or manageMotel.jsp base on link |

#### 

#### DeletePostControllerAdmin

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | delete post that have not accept by postID then go to Manageequest.jsp or managePost.jsp base on link |

#### 

#### PostDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deletePostByPostID | delete post by id |

#### MotelDAO Class

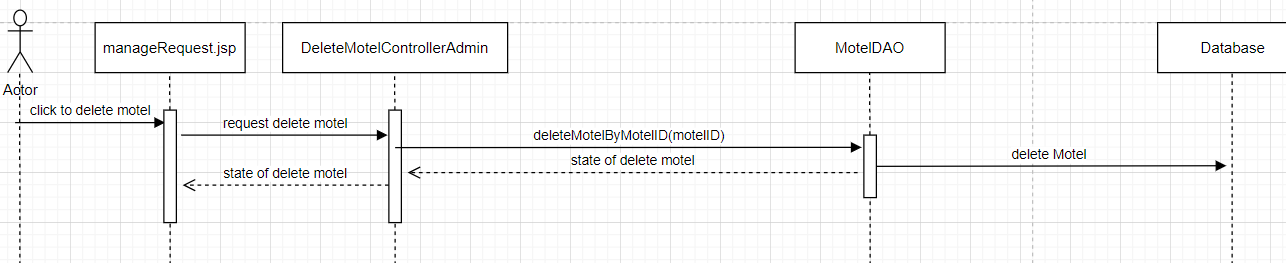
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deleteMotelByMotelID | delete motel by id |

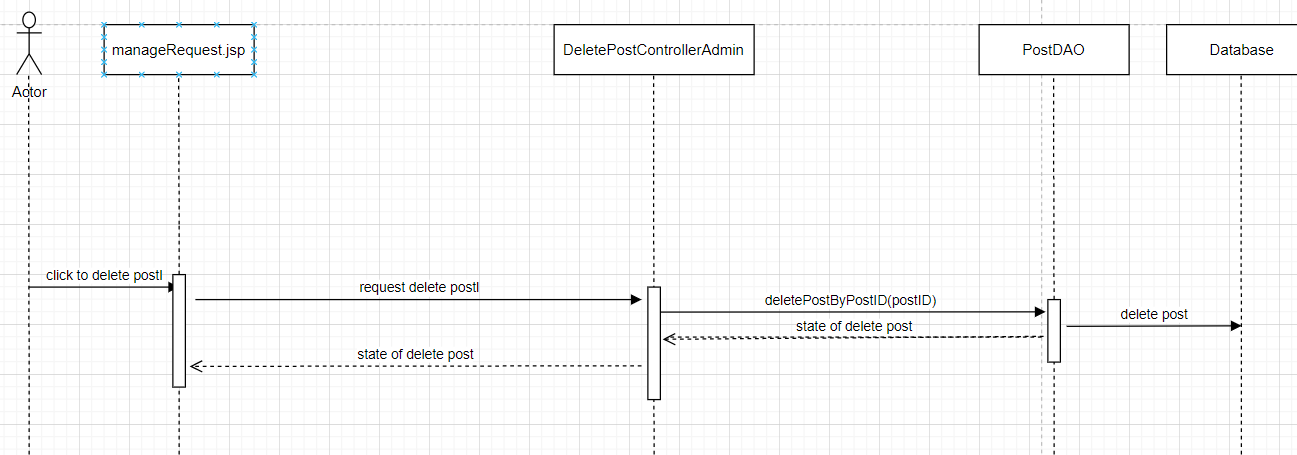
#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)





### d. Database queries

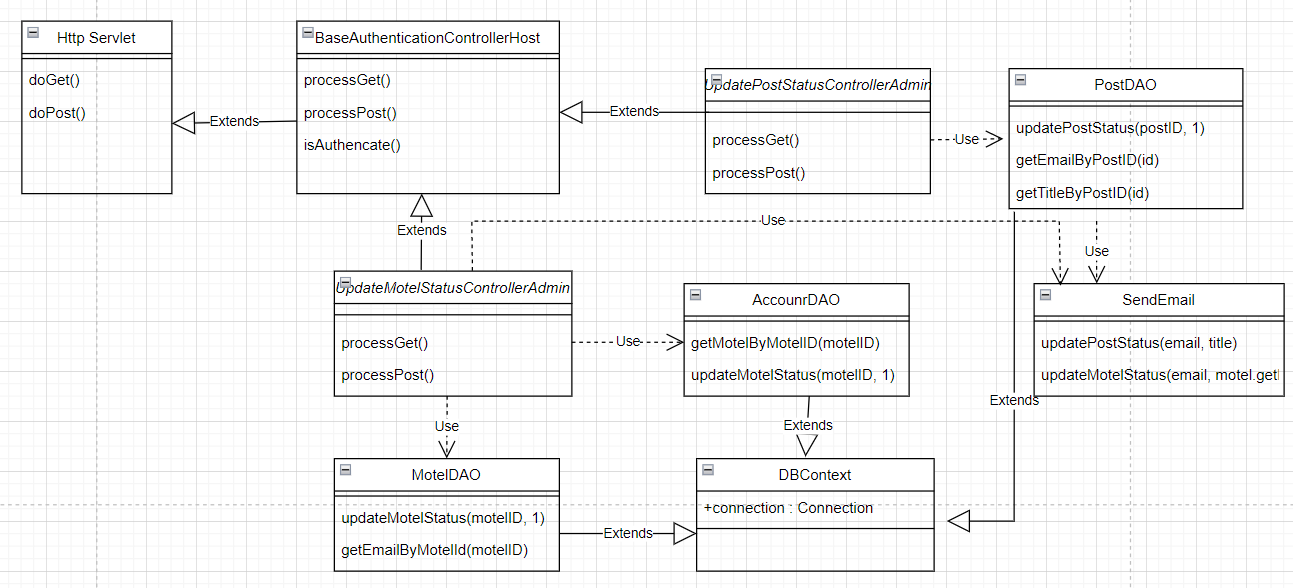
delete Motel where motelID = ?

delete PostImage where postID = ?

delete Post where postID = ?

## 28. Manage Request by Admin/Update

### a. Class Diagram



### b. Class Specifications

### BaseAuthenticationControllerAdmin

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

#### UpdateMotelStatusControllerAdmin

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | check the status of the motel is changed by admin or not if yes then update role of account that changed status of motel and status of motel after that go to Manageequest.jsp |

#### 

#### UpdatePostStatusControllerAdmin

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | check the status of the post is changed by admin or not if yes  update status of post after that go to Manageequest.jsp |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMotelByMotelID | get all information of one motel by motelID |
| 02 | updateMotelStatus | change status of motel (bit 0 → 1) |

#### 

#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updatePostStatus | change status of post(bit 0 → 1) |

#### 

#### AccountDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateRoleOfAccount | insert new row into table RoleOfAccount in database |
| 02 | checkIsAccountHasRole("host", motel.getAccountID()) | check that account has role host or not |

#### 

#### SendEmail Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateMotelStatus(email, motel.getMotelName()) | send email for account that updated |
| 02 | updatePostStatus(email, title) | send email for account has motel send post |

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### 

### 

### 

### 

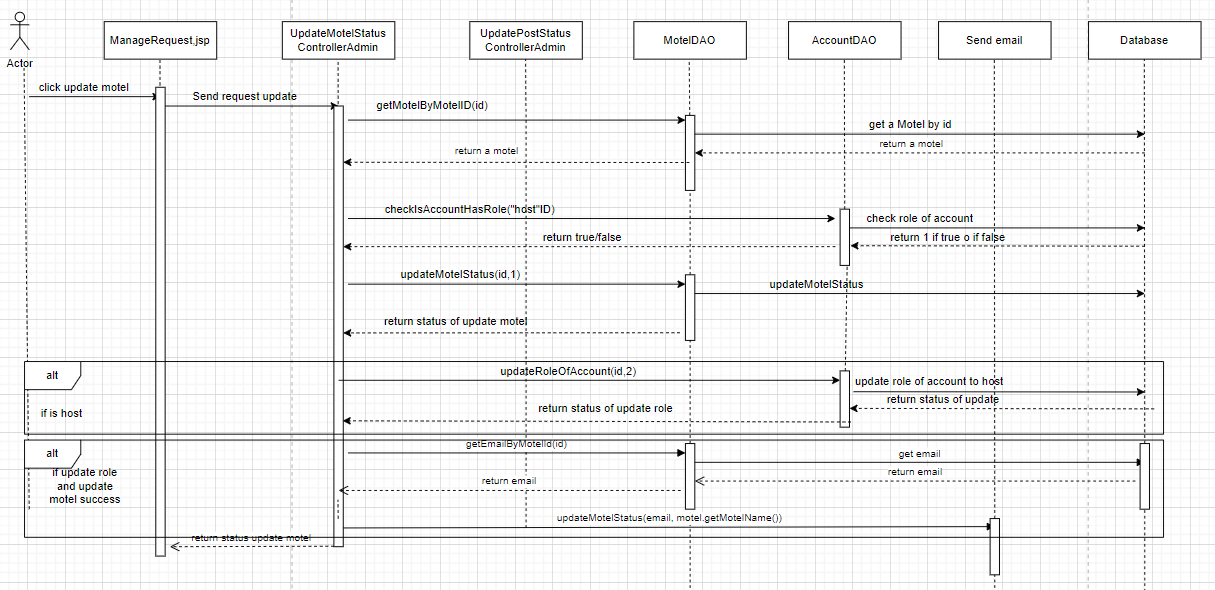
### 

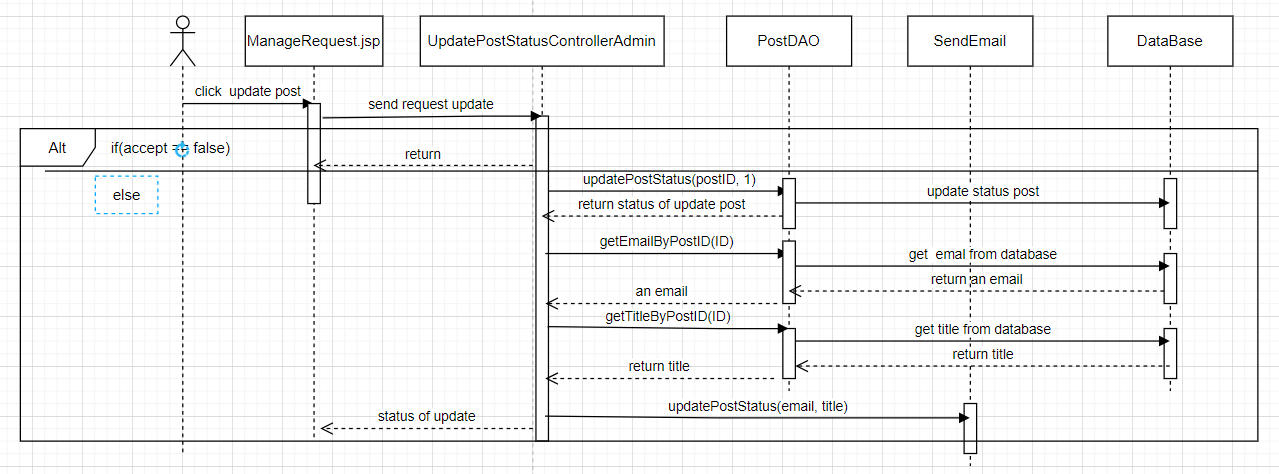
### 

### 

### 

### c. Sequence Diagram(s)





### d. Database queries

* select \* from Motel where motelID = ?
* if (?) in (select r.roleName from RoleOfAccount as RoA join [Role] as r on r.RoleID = RoA.RoleID where RoA.AccountID = ?) \n"

+ "select 1;\n"

+ "else\n"

+ "select 0;

* update Motel set motelStatus = ? where motelID = ?
* insert into RoleOfAccount values (?,?,0)
* select p.email

from Motel as m, Account as a, Profile as p

where m.AccountID = a.AccountID and a.AccountID = p.AccountID

and m.motelID = ?

* update Post set postStatus = ? where postID = ?
* select pr.email

from Post as p, Motel as m, Account as a, Profile as pr

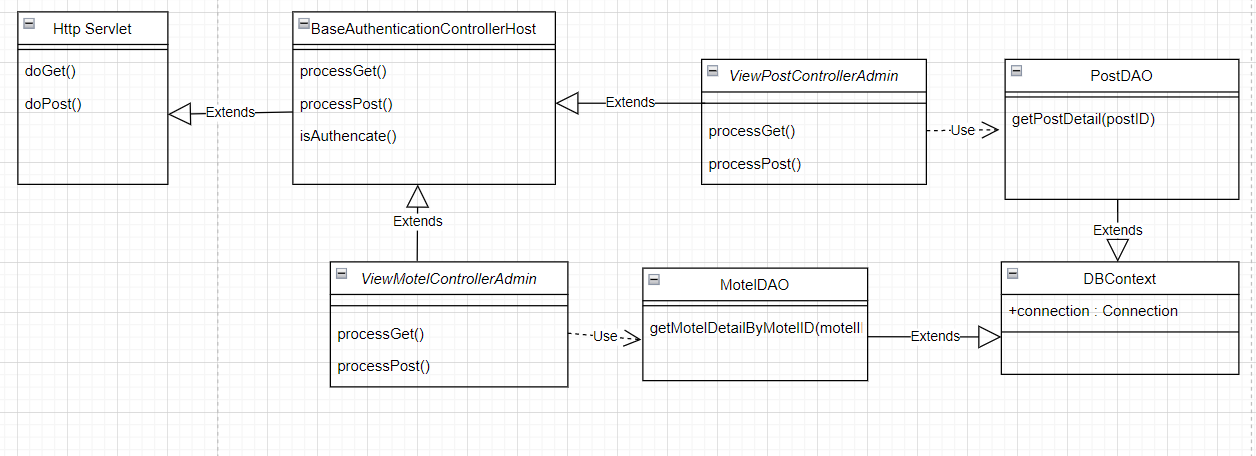
where p.motelID = m.motelID and m.AccountID = a.AccountID

and a.AccountID = pr.AccountID and p.postID = ?

* select postTitle from Post where postID = ?

## 29. Manage Request by Admin/ViewDetail

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

#### ViewMotelControllerAdmin

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | get all information of motel and send to motel-detail-admin.jsp |
| 02 | processPost | input is request and response. do nothing |

#### 

#### ViewPostControllerAdmin

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | get all information of post and send to post-detail-admin |

#### 

#### MotelDAOClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMotelDetailByMotelID(motelID,true) | get motel bt motel id |

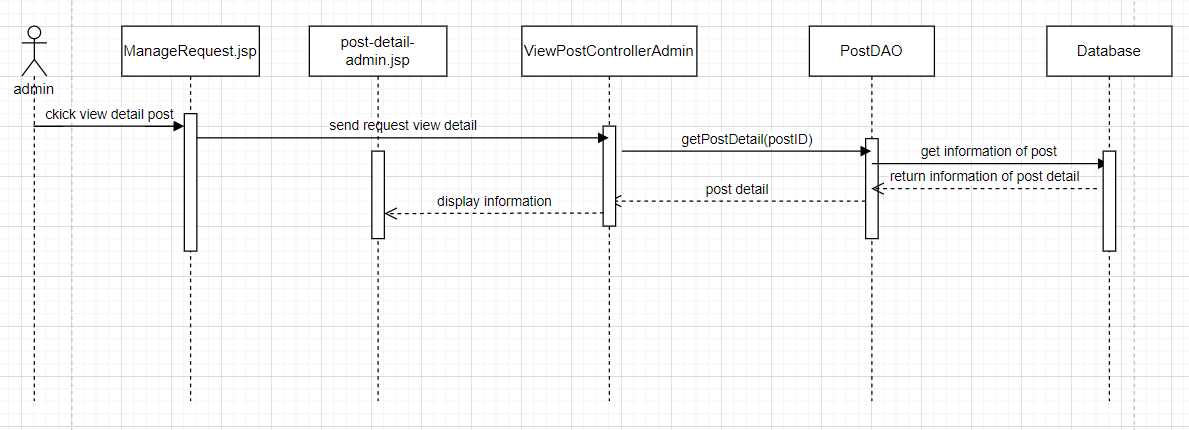
#### PostDAOClass

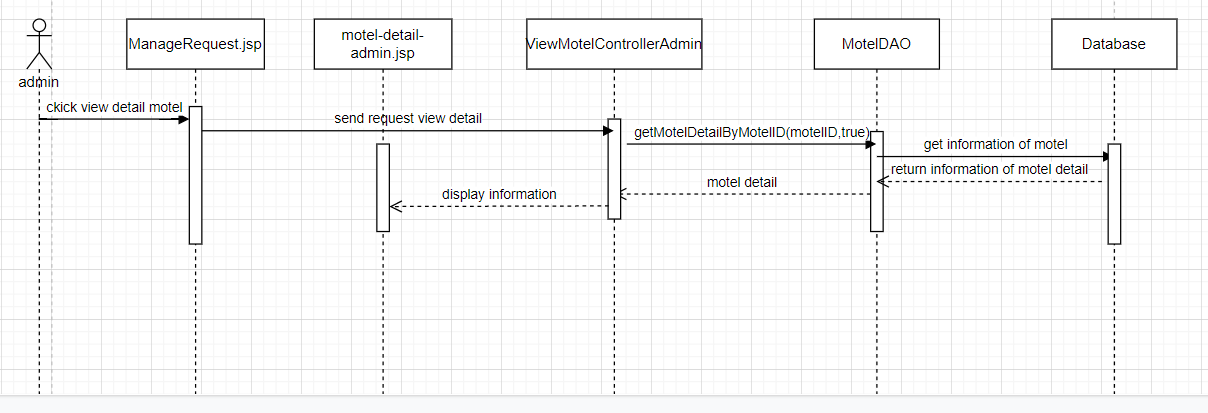
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getPostDetail(postID) | get post by post id |

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)





### d. Database queries

select m.motelID, m.accountID,(select p.fullName from "

+ "[Profile] as p where p.AccountID = m.AccountID), m.motelName,\n"

+ "(select count(\*) from Room as r where r.motelID = m.motelID),"

+ "m.motelAddress, m.motelPhone, m.motelStatus,\n"

+ "(select MAX(r.price) from Room as r where r.motelID = m.motelID),\n"

+ "(select MIN(r.price) from Room as r where r.motelID = m.motelID),"

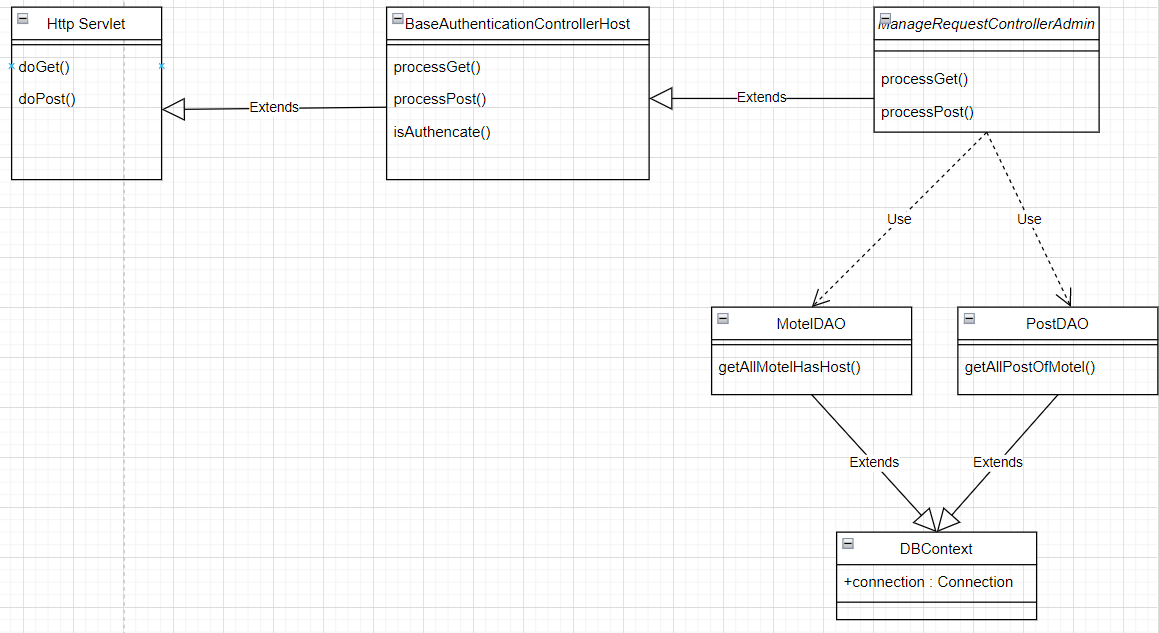
+ " m.motelDescription,m.motelImage, img.imageLink from Motel as m left "

+ "join MotelImage as img on img.motelID = m.motelID where m.motelID = ?

* select pt.postID, pt.motelID, mt.AccountID, mt.motelName, pt.postTitle, pt.postContent, (select max(price) from Room where motelID = pt.motelID) as maxPrice, (select min(price) from Room where motelID = pt.motelID) as minPrice, mt.motelPhone, mt.motelAddress, pt.postDate, pt.postStatus, pim.postImageID, pim.postImageLink from Post pt left join Motel mt on pt.motelID = mt.motelID left join PostImage pim on pim.postID = pt.postID where pt.postID = ?;

## 30. Manage Request by Admin/Search

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

**ManageequestControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | get all information of post and motel then go to Manageequest.jsp |

#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPostOfMotel | Get all information of all post in database |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelHasHost | Get all information of all motel in database |

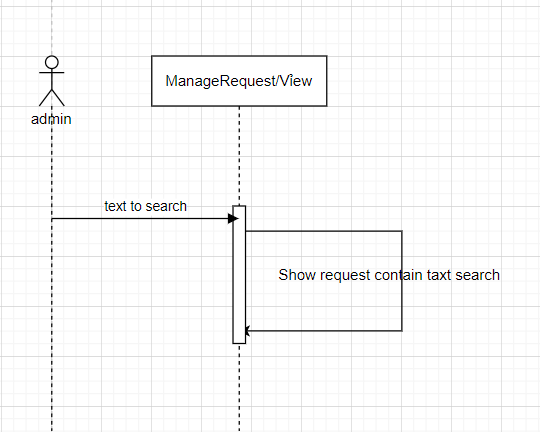
#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

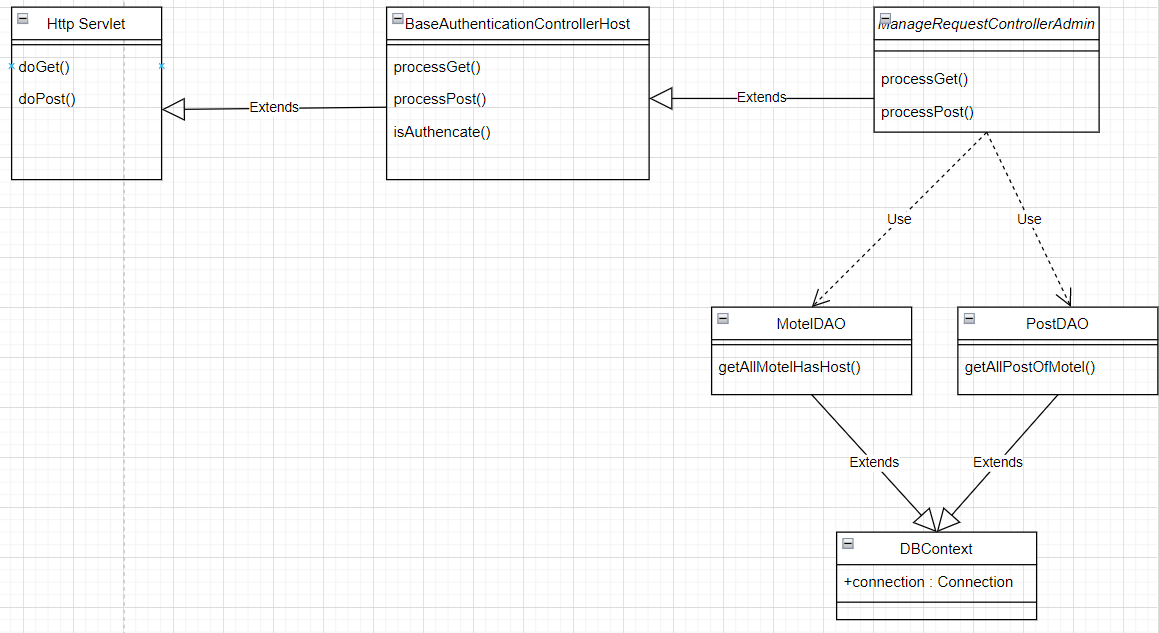
### c. Sequence Diagram(s)



### d. Database queries

## 31. Manage Request by Admin/Sort

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

**ManageequestControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | get all information of post and motel then go to Manageequest.jsp |

#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPostOfMotel | Get all information of all post in database |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelHasHost | Get all information of all motel in database |

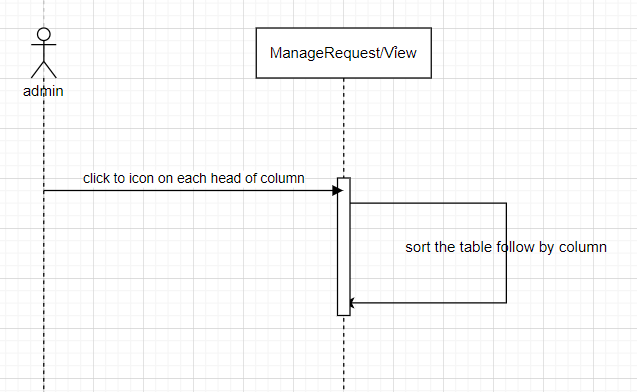
#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)



### d. Database queries

## **32. Manage Room By Host**

### **Class *Diagram***

### Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageoomControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the rooms of the account on the session and go to Manageoom.jsp |

***MotelDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of account. The parameter passed is the id of account |

***RoomDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRoomOfAccount | Get all room of account. The parameters passed are the id of motel |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

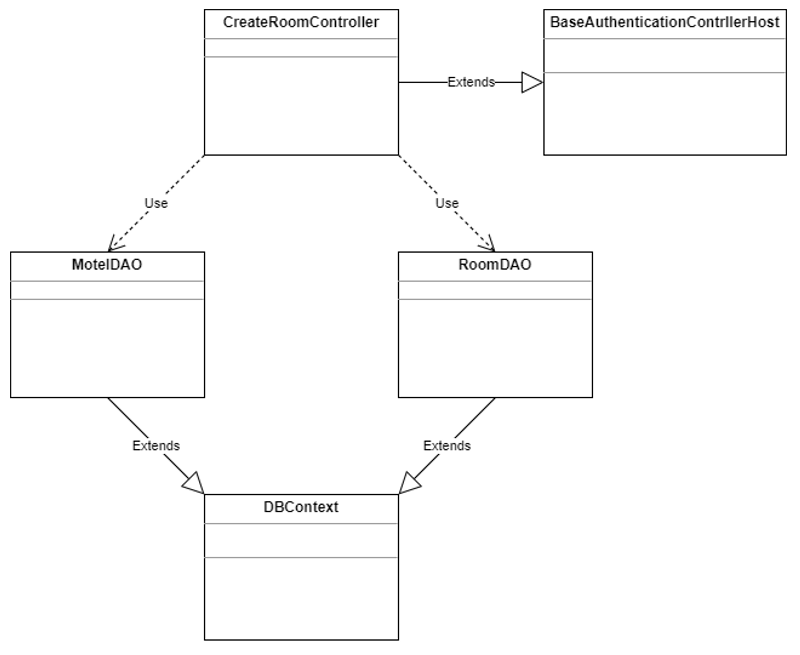
### Sequence Diagram(s)

1. ***Database queries***

* *getAllMotelByAccountID:* select \* from Motel where AccountID = ?
* *getAllRoomOfAccount*: select \* from Room where motelID = ?

## **3**3**. Manage Room By Host/Add**

### ***Class Diagram***

******

### Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***CreateRoomControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Receive request from user, get all rooms and motels of account. Forward to createRoom.jsp page (used to enter the information of the room) |
| 02 | processPost | Get information from createRoom.jsp page and add to database |

***MotelDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of account. The parameter passed is the id of motel |

***RoomDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRoomOfAccount | Get all room of account. The parameters passed are the id of account |
| 02 | CreateRoom | Create new room |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)

1. ***Database queries***

* *getAllMotelByAccountID:* select \* from Motel where AccountID = ?
* *getAllRoomOfAccount*: select \* from Room where motelID = ?
* *CreateRoom: insert into Room values(?,?,?,?,?,?,?)*

## **3**4**. Manage Room By Host/Update | Viewdetails**

### ***Class Diagram***

### Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***UpdateRoomControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Receive request from user, get all services and motels of account. go to updateRoom.jsp page (The page is used to update the information and display all the information of the selected room) |
| 02 | processPost | Get information from updateeRoom.jsp page and update to database |

***MotelDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of account. The parameter passed is the id of account |

***RoomDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRoomOfAccount | Get all room of account. The parameters passed are the id of motel |
| 02 | getRoomByRoomID | Get all information of room by room id |
| 03 | UpdateRoom | Update information of room |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)

### ***Database queries***

* *getAllMotelByAccountID:* select \* from Motel where AccountID = ?
* *getAllRoomOfAccount*: select \* from Room where motelID = ?
* *getRoomByRoomID: Select \* from Room where roomID = ?*
* *UpdateRoom: update room set motelID = ?, roomName = ?, numberOfPeopleMax = ?, numberOfPeopleCurrent = ?, price = ?, area = ?, noteRoom =? where roomID = ?*

## 

## 

## 

## **3**5**. Manage Room By Host/Delete**

### ***Class Diagram***

### Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***DeleteRoomControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Revice id need to delete from user request and call the function DeleteRoomByID |

***RoomDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DeleteRoomByID | delete room by room id |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)

### ***Database queries***

* *DeleteRoomByID:* delete Room where RoomID = ?

## **3**6**. Manage Room By Host/Search**

### ***Class Diagram***

### Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageoomControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the rooms of the account on the session and go to Manageoom.jsp |

***MotelDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of account. The parameter passed is the id of account |

***RoomDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRoomOfAccount | Get all room of account. The parameters passed are the id of account |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)

### ***Database queries***

* *getAllMotelByAccountID:* select \* from Motel where AccountID = ?
* *getAllRoomOfAccount*: select \* from Room where motelID = ?

## **3**7**. Manage Room By Host/Sort**

### ***Class Diagram***

### Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageoomControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the rooms of the account on the session and go to Manageoom.jsp |

***MotelDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of account. The parameter passed is the id of account |

***RoomDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRoomOfAccount | Get all room of account. The parameters passed are the id of account |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)

### ***Database queries***

* *getAllMotelByAccountID:* select \* from Motel where AccountID = ?
* *getAllRoomOfAccount*: select \* from Room where motelID = ?

## **38. Manage Services By Host**

### ***Class Diagram***

### Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageServicesControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the services of the account on the session and go to manageServices.jsp |

***MotelDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of account. The parameter passed is the id of account |

***ServicesDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllServicesOfAccount | Get all services of account. The parameters passed are the id of motel |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)

### ***Database queries***

* *getAllMotelByAccountID:* select \* from Motel where AccountID = ?
* *getAllServicesOfAccount*: select \* from Services where motelID = ?

## **39. Manage Services By Host/Add**

### ***Class Diagram***

### Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***CreateServicesControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Receive request from user, get all services and motel of account. go to createServices.jsp page (used to enter the information of the service) |
| 02 | processPost | Get information from createServices.jsp page and add to database |

***MotelDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of account. The parameter passed is the id of account |

***ServicesDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRoomOfAccount | Get all services of account. The parameters passed are the id of motel |
| 02 | CreateServices | Create new services |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)

### ***Database queries***

* *getAllMotelByAccountID:* select \* from Motel where AccountID = ?
* getAllServicesOfAccount: select \* from Services where motelID = ?
* *insert into Services(motelID, servicesName, price) values(?, ?, ?)*

## 40**. Manage Services By Host/Update | ViewDetail**

### ***Class Diagram***

### Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageServicesControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processPost | Get all informations of services want to update from page manageServices.jsp and call function UpdateServices |

***MotelDAO***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of account. The parameter passed is the id of account |

***ServicesDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllServicesOfAccount | Get all services of account. The parameters passed are the id of motel |
| 03 | UpdateServices | Update information of services |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)

### ***Database queries***

* *getAllMotelByAccountID:* select \* from Motel where AccountID = ?
* *getAllServicesOfAccount*: select \* from Services where motelID = ?
* *UpdateServices:* UPDATE Services set servicesName = ?, price = ? where servicesID = ? and motelID = ?

## 41**. Manage Services By Host/Delete**

### c***lass Diagram***

### Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***DeleteRoomControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Revice id need to delete from user request and call the function DeleteServicesByID |

***RoomDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DeleteServicesByID | delete services by services id |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

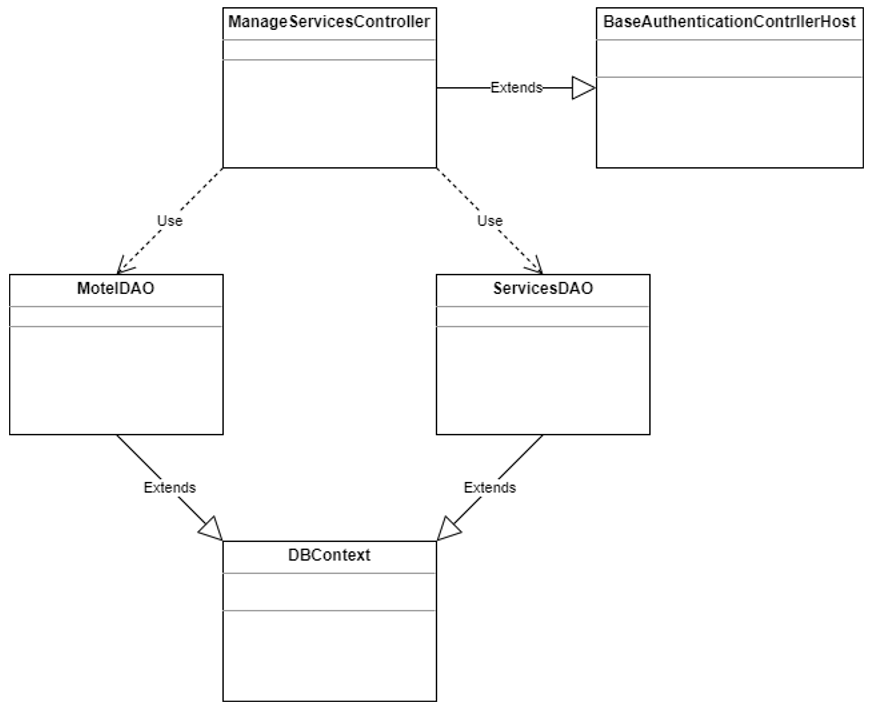
### Sequence Diagram(s)

### ***Database queries***

* *DeleteServicesByID:* delete Services where servicesID = ?

## **42. Manage Services By Host/Search**

### ***Class Diagram***

******

### Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageServicesControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the services of the account on the session and go to manageServices.jsp |

***MotelDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of account. The parameter passed is the id of account |

***ServicesDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllServicesOfAccount | Get all services of account. The parameters passed are the id of motel |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

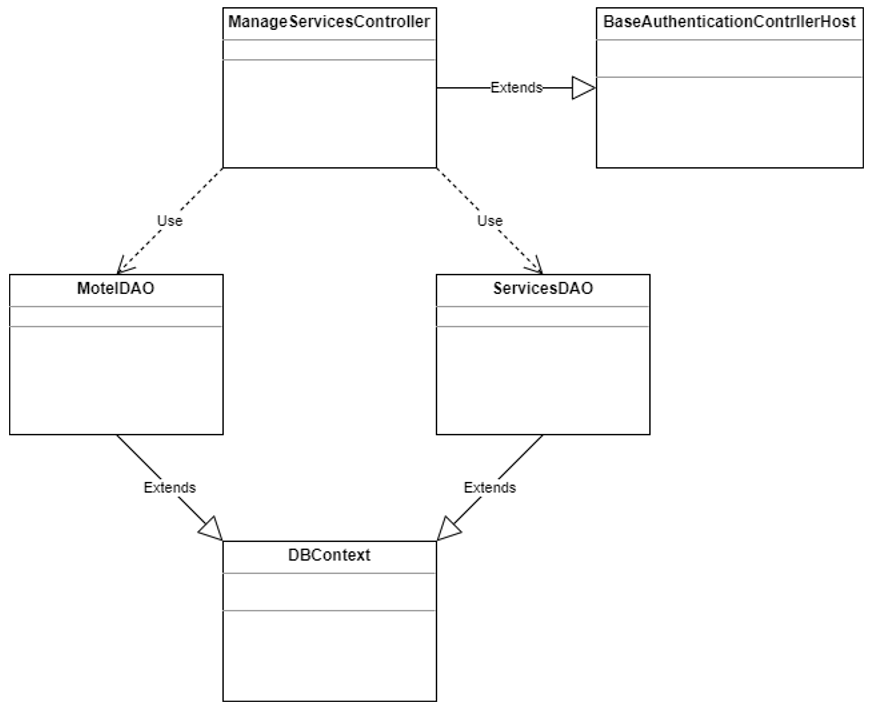
### Sequence Diagram(s)

### ***Database queries***

* *getAllMotelByAccountID:* select \* from Motel where AccountID = ?
* *getAllServicesOfAccount*: select \* from Services where motelID = ?

## **4**3**. Manage Services By Host/Sort**

### ***Class Diagram***

******

### Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageServicesControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the services of the account on the session and go to manageServices.jsp |

***MotelDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of account. The parameter passed is the id of account |

***ServicesDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllServicesOfAccount | Get all services of account. The parameters passed are the id of motel |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)

### ***Database queries***

* *getAllMotelByAccountID:* select \* from Motel where AccountID = ?
* *getAllServicesOfAccount*: select \* from Services where motelID = ?

## 44. Manage bill by host / view

### Class Diagram



### Class Specifications

***ManageBillController class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | get | Get all bill from db and send them to view |

***BillDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllBillByMotelId | get all bill in motel |

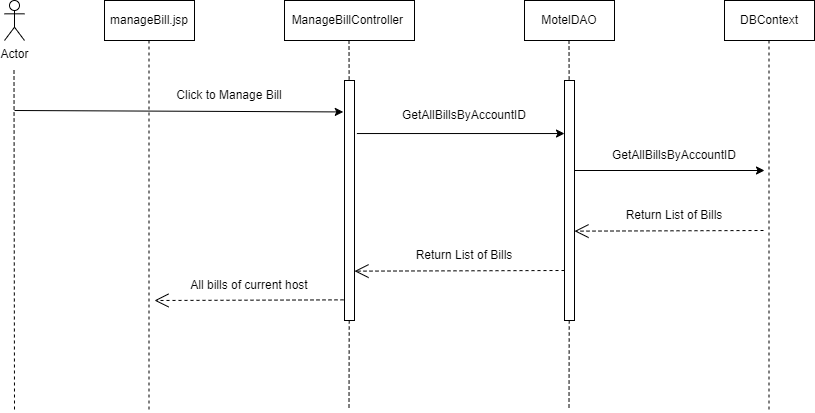
***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)

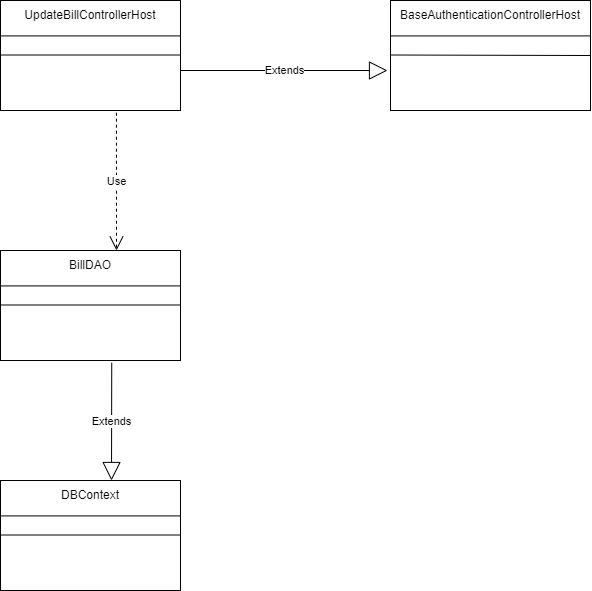


### Database queries

select a.billID, d.motelName, c.roomName, a.totalBill, a.statusBill, a.dateOfPayment, b.BookingID, c.roomID, d.motelID from Bill as a, Booking as b, Room as c, Motel as d where a.BookingID = b.BookingID and b.RoomID = c.roomID and c.motelID = d.motelID

## 45. Manage bill by host / update

### a. Class Diagram



### Class Specifications

***UpdateBillController class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | get | Get changed confirmation from user and update to bill |

***BillDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateBill | Update bill |

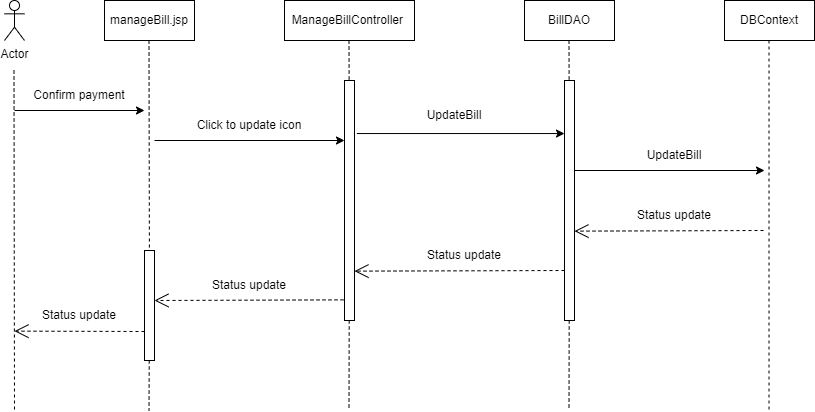
***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)

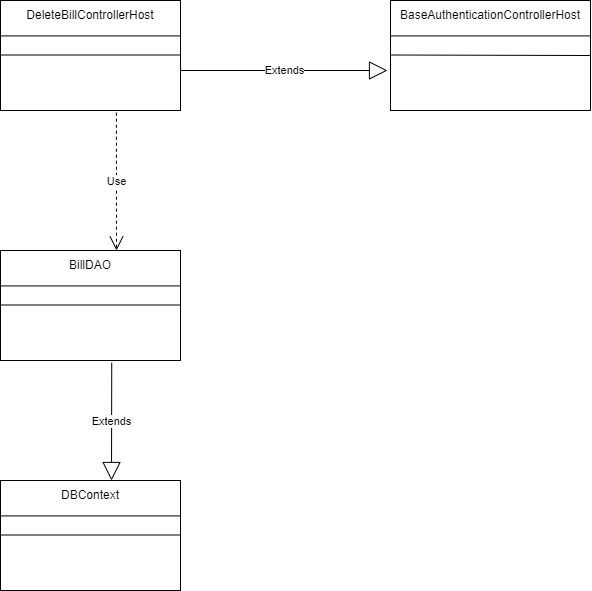


### Database queries

update Bill set statusBill = 'paid', dateOfPayment = GETDATE() where billID = ?

## 46. Manage bill by host / delete

### a. Class Diagram



### Class Specifications

***DeleteBillController class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | get | Delete bill that user choose |

***BillDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deleteBill | Delete bill |

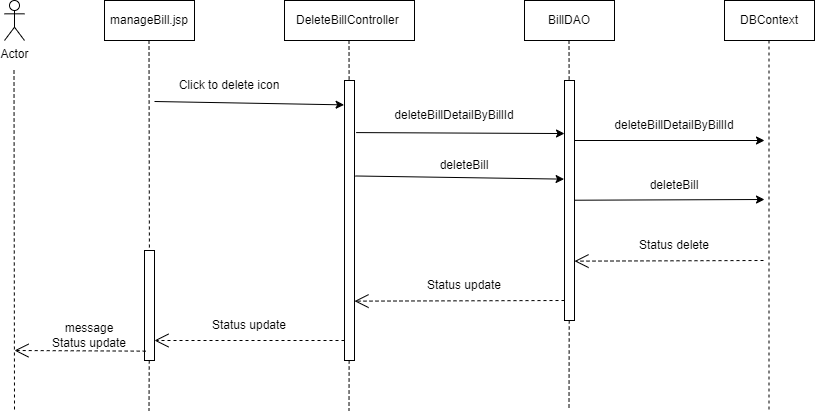
***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)



### Database queries

delete from BillDetail where billID = ?

delete from Bill where billID = ?

## 47. Manage bill by host / search

### a. Class Diagram



### Class Specifications

***ManageBillController class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | get | Get all bill from db and send them to view |

***BillDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllBillByMotelId | get all bill in motel |

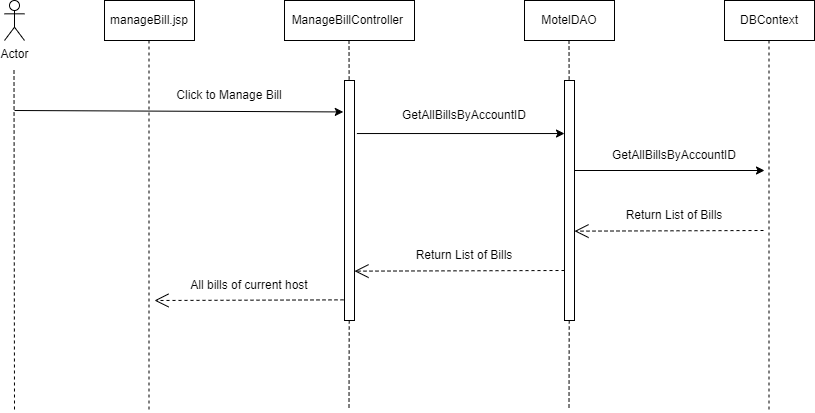
***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)



### Database queries

select a.billID, d.motelName, c.roomName, a.totalBill, a.statusBill, a.dateOfPayment, b.BookingID, c.roomID, d.motelID from Bill as a, Booking as b, Room as c, Motel as d where a.BookingID = b.BookingID and b.RoomID = c.roomID and c.motelID = d.motelID

## 48. Manage bill by host / sort

### a. Class Diagram



### Class Specifications

***ManageBillController class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | get | Get all bill from db and send them to view |

***BillDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllBillByMotelId | get all bill in motel |

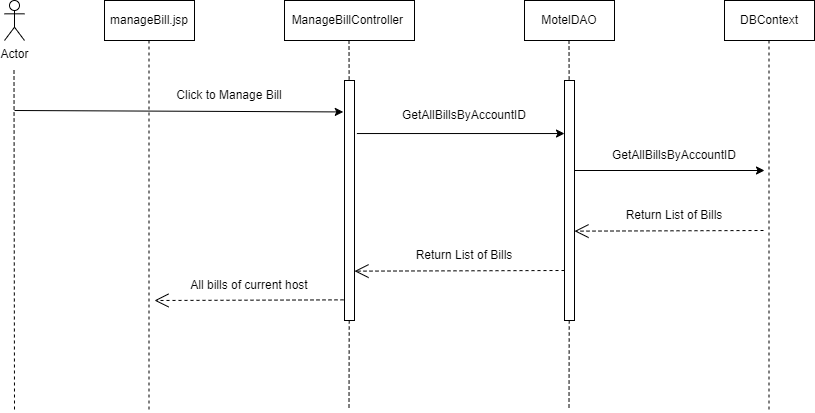
***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)

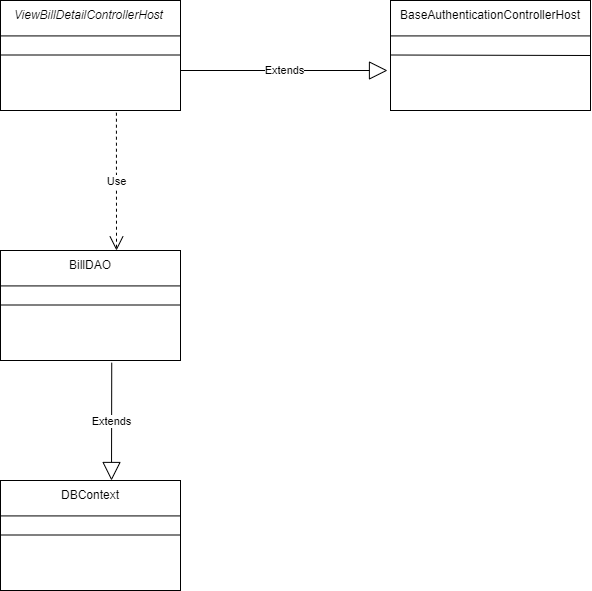


### Database queries

select a.billID, d.motelName, c.roomName, a.totalBill, a.statusBill, a.dateOfPayment, b.BookingID, c.roomID, d.motelID from Bill as a, Booking as b, Room as c, Motel as d where a.BookingID = b.BookingID and b.RoomID = c.roomID and c.motelID = d.motelID

## 49. Manage bill by host / view detail

### a. Class Diagram



### Class Specifications

***ViewBillDetailControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | get | Receive click from user and show all bill detail corresponding with choosed bill |

***BillDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getBillDetailsByBillId | Get all corresponding bill detail (services) from database |

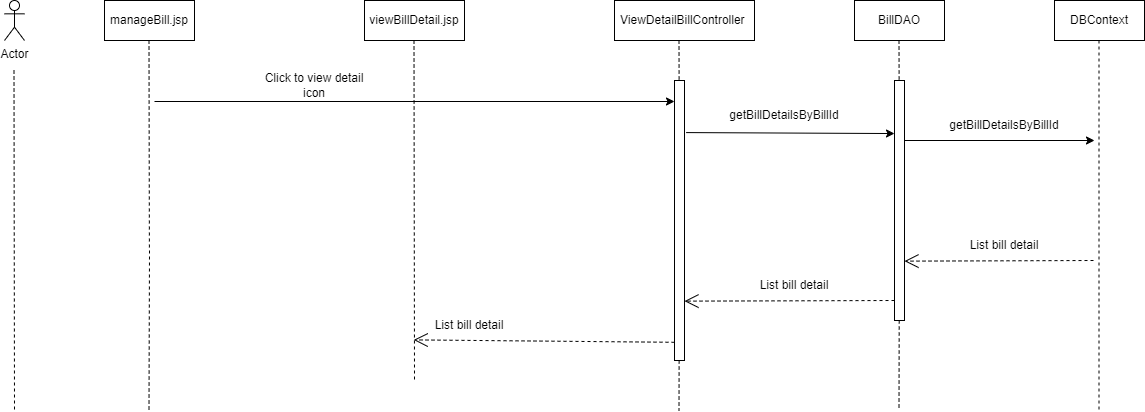
***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)

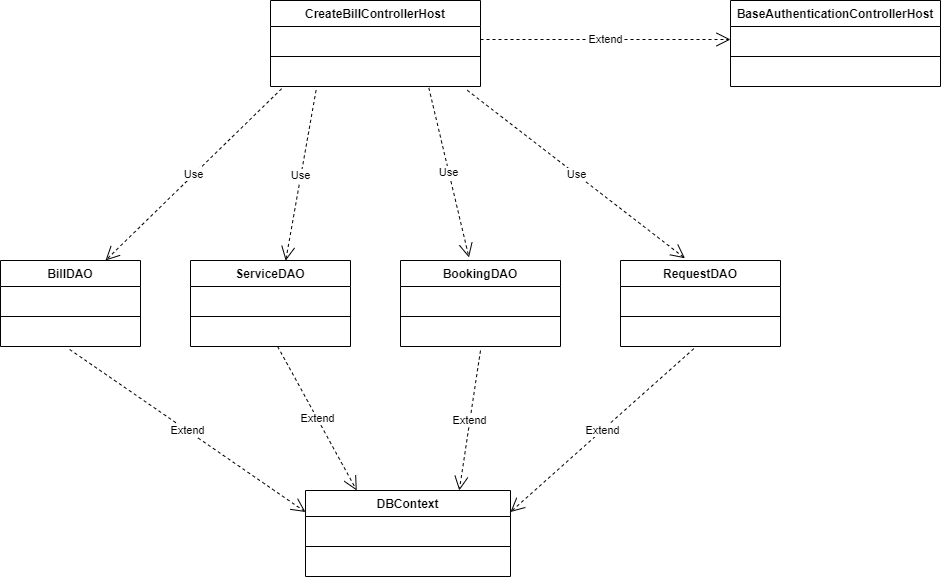


### Database queries

select b.billDetailID, d.servicesName, d.price, b.quantity, b.dateCreate, .totalBillDetail, c.servicesOfRoomID, d.servicesID from BillDetail as b, ServicesOfBooking as c, Services as d where b.billID = ? and b.servicesOfRoomID = c.servicesOfRoomID and c.servicesID = d.servicesID

## 50. Manage bill by host / create new bill

### a. Class Diagram



### Class Specifications

***CreateBillControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | get | Get all information to make a new bill |

***BillDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createBill(bookingId, totalBill) | insert a new bill to db |
| 02 | createBillDetailSOB() | insert a new bill detail to db |
| 03 | createBillDetailRq() | insert a new bill detail to db |

***BookingDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createBooking(roomID) | insert a new booking |
| 02 | getLastestBookingId() | get the current booking id |

***ServiceDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createServiceOfBooking() | insert a new bill to db |
| 02 | getLastestServiceOfBookingID() | get current SORID |

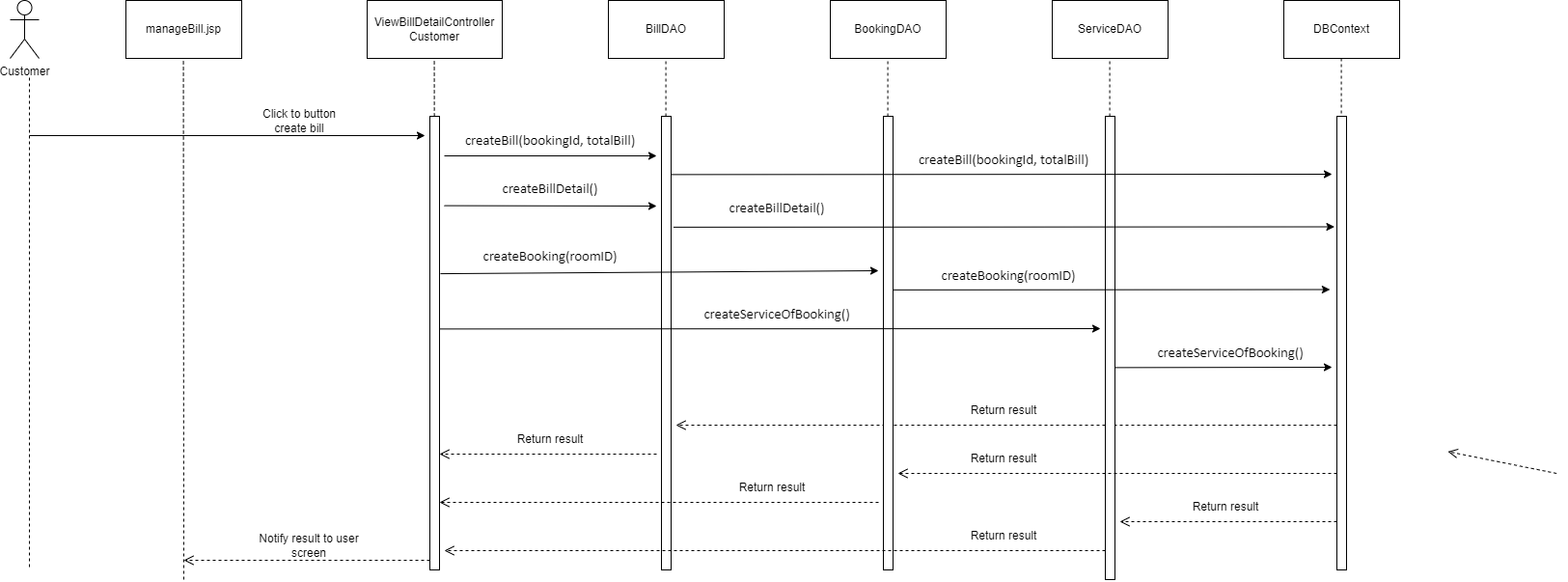
***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### Sequence Diagram(s)



### Database queries

* insert into Booking(RoomID,isDeleteBooking,isDraft) values(?,0,0)
* select MAX(BookingID) from Booking
* insert into ServicesOfBooking(BookingID, servicesID, isDeleteServicesOfBooking,isDraft) values(?,?,0,0)
* insert into BillDetail(billID, servicesOfRoomID, quantity, totalBillDetail,isDeleteBillDetail,isDraft,price) values(?,?,?,?,0,0,?)
* insert into BillDetail(billID, requestID, quantity, totalBillDetail,isDeleteBillDetail,isDraft,price) values(?,?,?,?,0,0,?)

## 52. Manager Request By Host

### a. Class Diagram



### b. Class Specifications

ManageRequestControllerHost class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Display all list request from room in motel |
| 02 | doPost | Display all list request from room in each motel |

MotelDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | All list motel by Account |

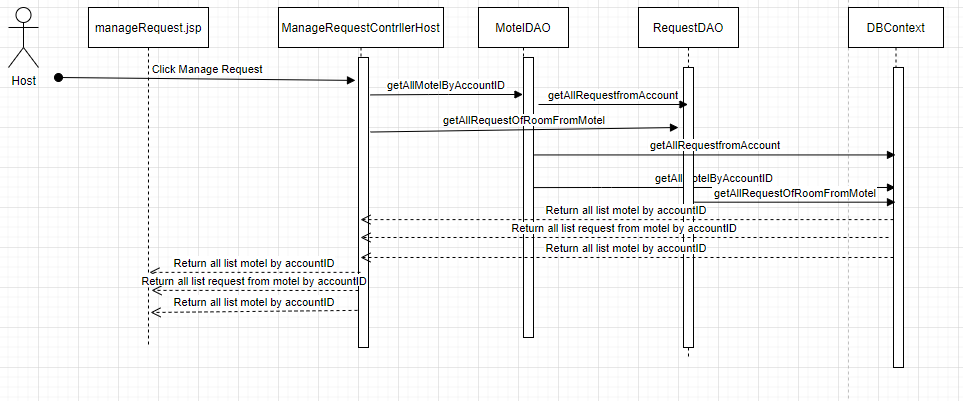
RequestDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRequestfromAccount | get all request from room in motel by Account |
| 02 | getAllRequestOfRoomFromMotel | get all request from room in each motel |

DBContext class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Connecting system with database |

### c. Sequence Diagram(s)



### d. Database queries

getAllMotelByAccountID: select \* from Motel where AccountID = ?

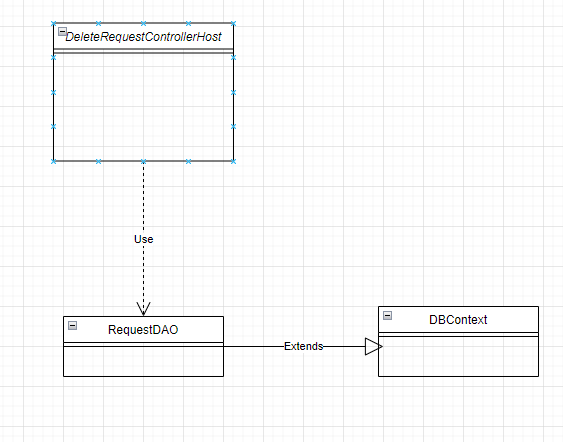
getAllRequestfromAccount: select \* from Room r inner join Request re on re.roomID = r.roomID

where motelID = ?   
getAllRequestOfRoomFromMotel: select \* from Room r inner join Request re on re.roomID = r.roomID

where motelID = ?

## 53. Manager Request By Host/Delete

### a. Class Diagram



### b. Class Specifications

DeleteRequestControllerHost class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Delete request by requestID |

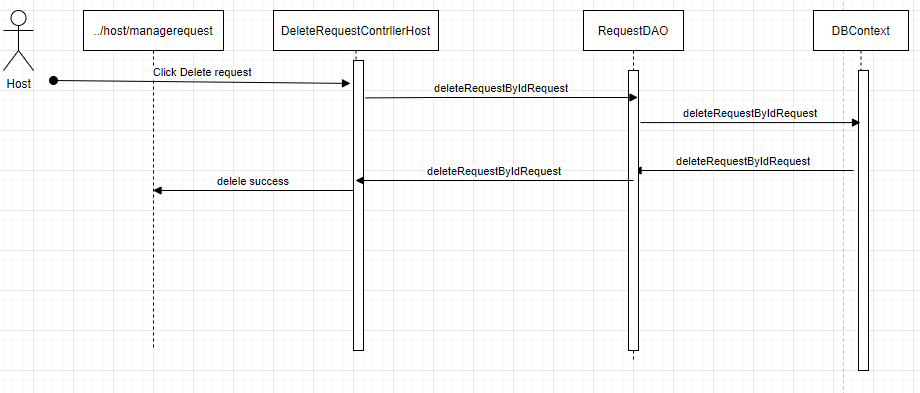
RequestDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deleteRequestByIdRequest | Delete request by requestID |

DBContext class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Connecting system with database |

### c. Sequence Diagram(s)



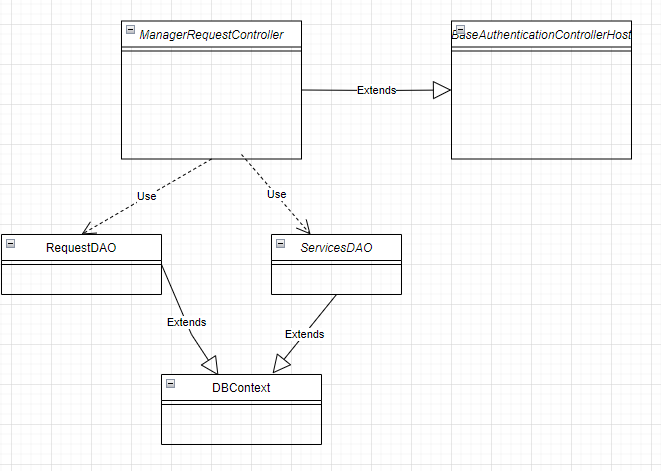
### d. Database queries

deleteRequestByIdRequest: delete from Request

where requestID = ?

## 54. Manager Request By Host/Search

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageServicesControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the request of the account on the session and go to manageRequest.jsp |

***RequestDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRequestByAccountID | Get all request from room in motel of account. The parameter passed is the id of account |

***ServicesDAO class***

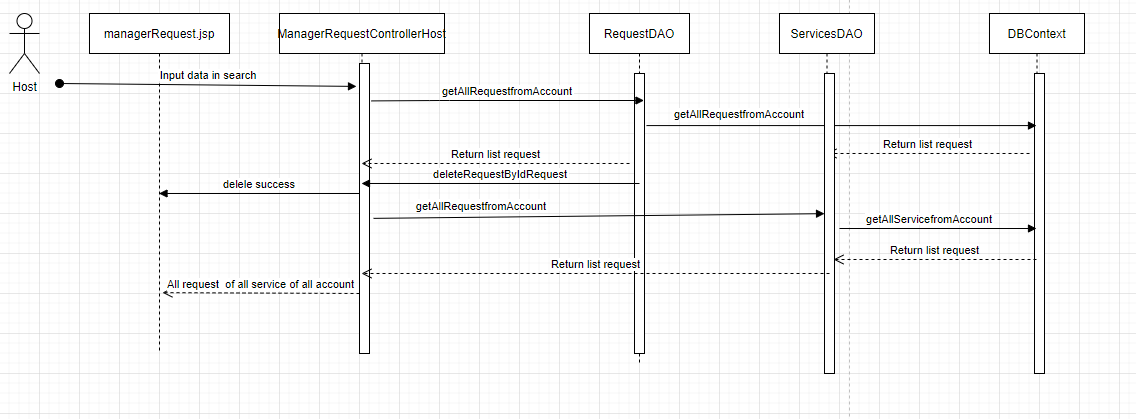
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllServicesOfAccount | Get all services of account. The parameters passed are the id of motel |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)

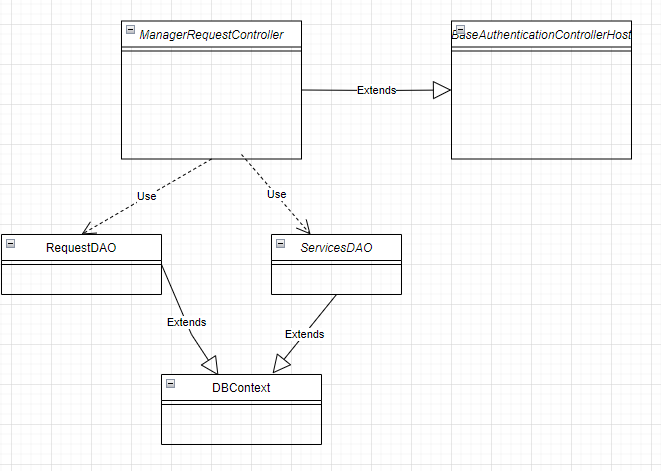


### d. Database queries

* *getAllRequestByAccount:* select \* from Room r inner join Request re on re.roomID = r.roomID where motelID = ?
* *getAllServicesOfAccount*: select \* from Request where requestID = ?

## 55. Manager Request By Host/Sort

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageServicesControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the request of the account on the session and go to manageRequest.jsp |

***RequestDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRequestByAccountID | Get all request from room in motel of account. The parameter passed is the id of account |

***ServicesDAO class***

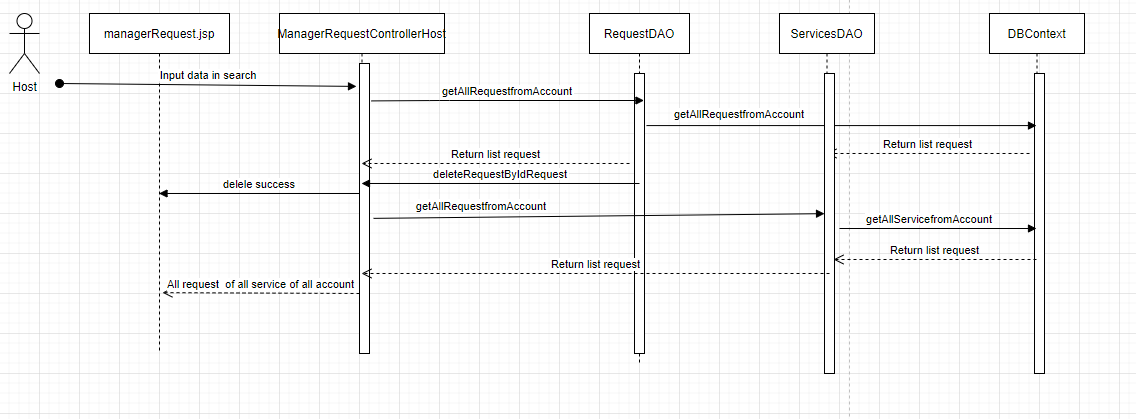
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllServicesOfAccount | Get all services of account. The parameters passed are the id of motel |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)

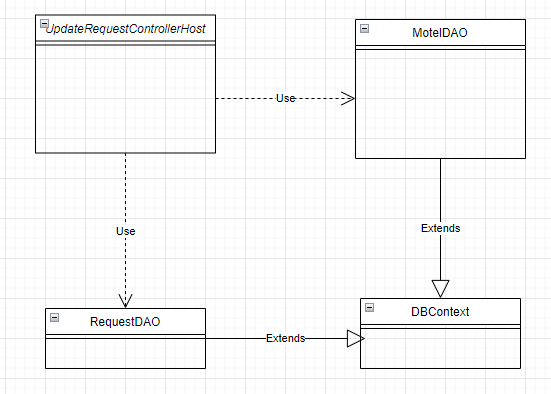


### d. Database queries

* *getAllRequestByAccount:* select \* from Room r inner join Request re on re.roomID = r.roomID where motelID = ?
* *getAllServicesOfAccount*: select \* from Request where requestID = ?

## 56. Manager Request By Host/Update/Show

### a. Class Diagram



### b. Class Specifications

UpdateRequestControllerHost class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Display content detail |
| 02 | doPost | Update request by host |

MotelDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | All list motel by Account |
| 02 | getNameMotelByRoomID | Get name motel by roomID |

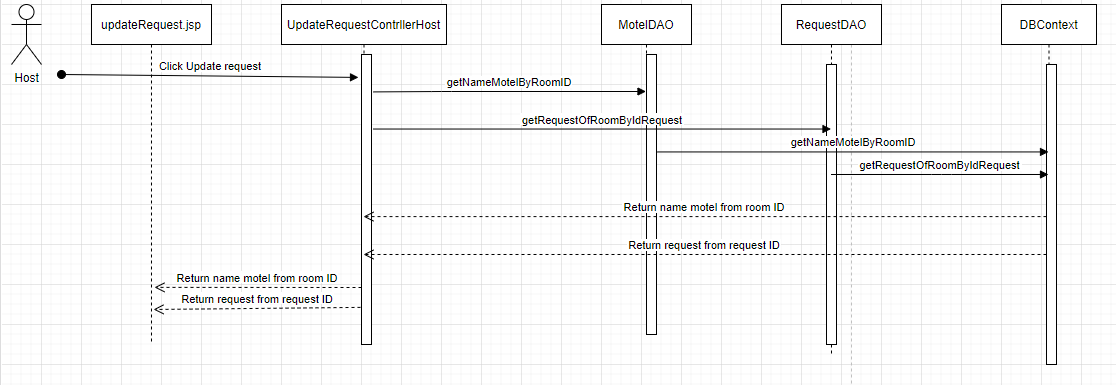
RequestDAO class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getRequestOfRoomByIdRequest | get request from room by id request |
| 02 | getAllRequestfromAccount | get all request from room in motel by Account |

DBContext class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Connecting system with database |

### c. Sequence Diagram(s)



### d. Database queries

getNameMotelByRoomID: select m.motelName from Motel m inner join Room r on m.motelID = r.motelID where roomID = ?

updateRequestOfRoomByIdRequest: update Request set content = ?, [status] = ?, noteRequest = ?, updateAt = getdate() where requestID = ?

getAllRequestfromAccount: select \* from Room r inner join Request re on re.roomID = r.roomID

where motelID = ?

## 

## 

## 

## 

## 

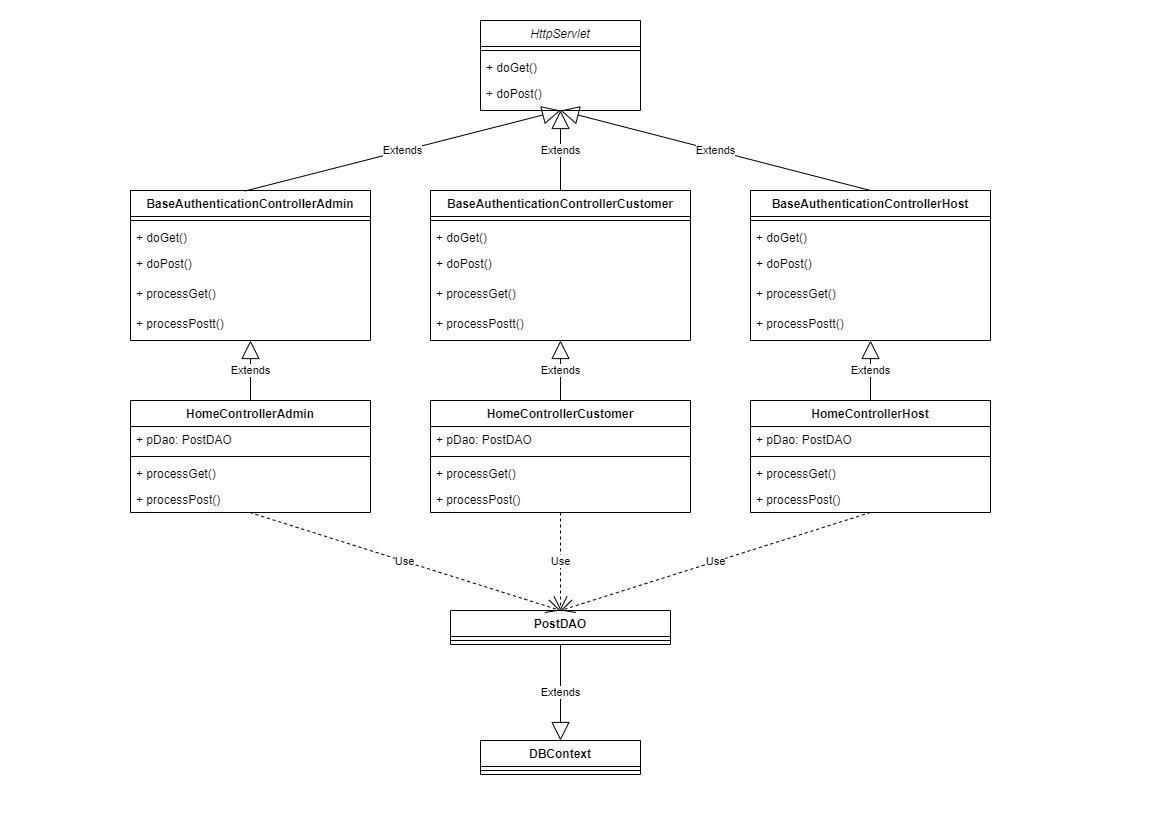
## 

## 

## 

## 57. Filter Post

### a. Class Diagram



### b. Class Specifications

#### HomeControllerAdmin Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will return the login.jsp page according to the corresponding role |
| 02 | doPost | input is request and response. After receiving the request from the user, it will get the username and password and call the function getAccountByUsernamePasswordAndRole(username, password, role) of the AccountDAO class. |

#### HttpServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet |  |
| 02 | doPost | Using this HttpServlet object, we can build Java web applications to handle requests from clients with HTTP methods like GET, POST, HEAD, PUT, DELETE, TRACE and OPTIONS. When the client sends a request to the web server container, the service() method in the HttpServlet object processes the request. Depending on the HTTP method that the client sends, further processing will be done by the doGet(), doPost(), doHead(), doPut(), doDelete(), doTrace() and doOptions() methods of this object. receive. |
| 03 | doHead |
| 04 | doPut |
| 05 | doDelete |
| 06 | doOption |
| 07 | doTrace |

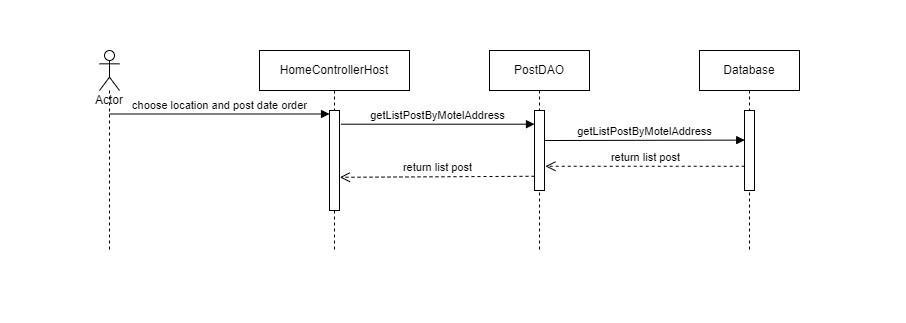
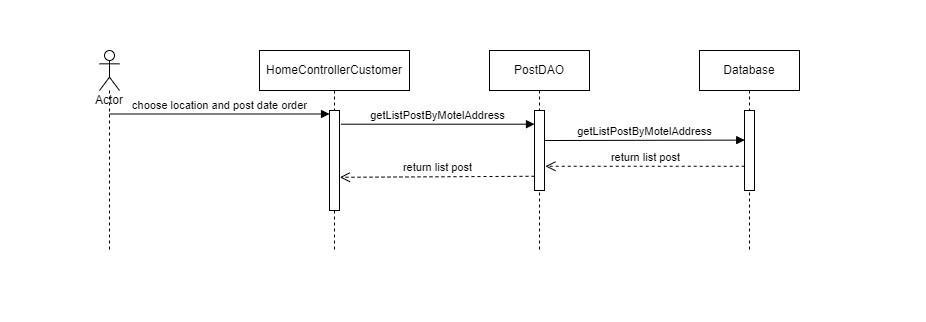
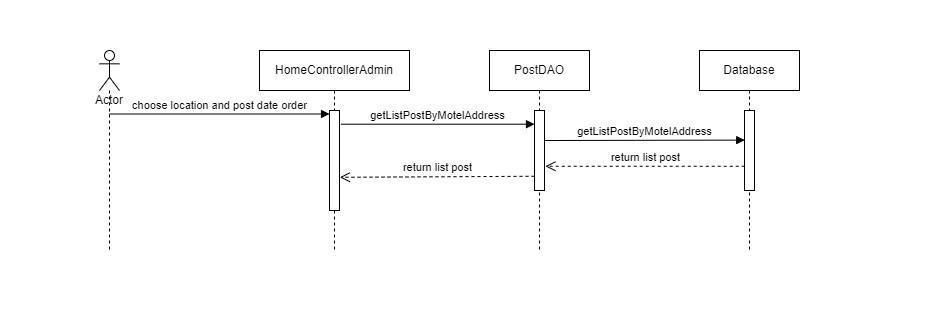
#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPost | Get all post in database |
| 02 | getAllPostByMotelName | Get all post in database which have similar to input motel name |

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)

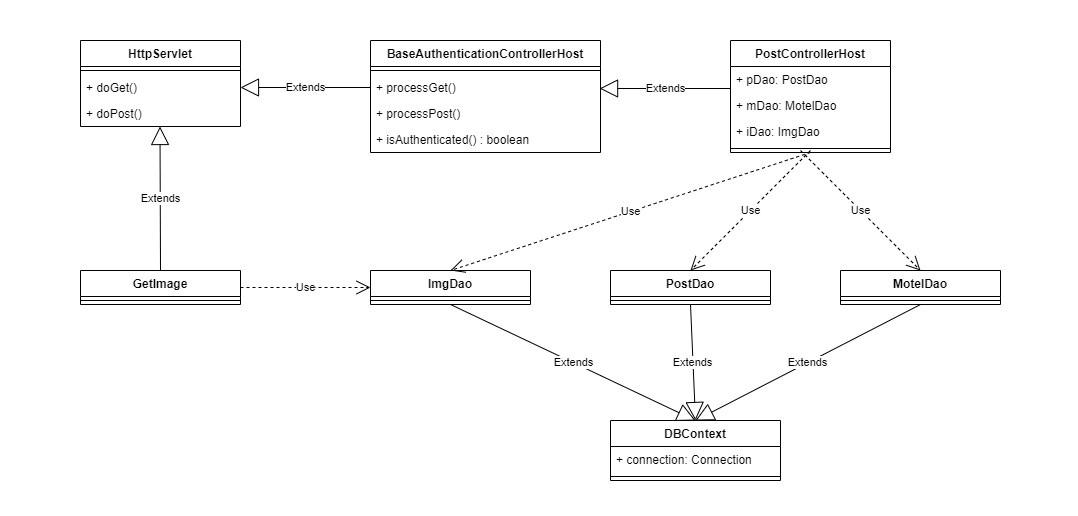


### d. Database queries

* select pst.postID, pst.motelID, pst.postContent, pst.postDate, pst.postTitle, pst.postStatus, mtl.motelName, pfl.fullName from Post pst left join Motel mtl on mtl.motelID=pst.motelID left join Profile pfl on pfl.AccountID=mtl.AccountID where upper(mtl.motelAddress) like upper(N'%" + motelAddress + "%') order by pst.postDate "+order+";

## 58. Manage Post by Host

### a. Class Diagram



### b. Class Specifications

#### PostControllerHost Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will return the login.jsp page according to the corresponding role |
| 02 | doPost | input is request and response. After receiving the request from the user, it will get the username and password and call the function getAccountByUsernamePasswordAndRole(username, password, role) of the AccountDAO class. |

#### HttpServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet |  |
| 02 | doPost | Using this HttpServlet object, we can build Java web applications to handle requests from clients with HTTP methods like GET, POST, HEAD, PUT, DELETE, TRACE and OPTIONS. When the client sends a request to the web server container, the service() method in the HttpServlet object processes the request. Depending on the HTTP method that the client sends, further processing will be done by the doGet(), doPost(), doHead(), doPut(), doDelete(), doTrace() and doOptions() methods of this object. receive. |
| 03 | doHead |
| 04 | doPut |
| 05 | doDelete |
| 06 | doOption |
| 07 | doTrace |

#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPostOfHost | Get all post in database by host id |
| 02 | getPostDetailByPostID | Get all post in database which have similar to input post id |
| 03 | deletePostByID | Delete post by post id |
| 04 | getListImageIDOfPost | Get all image id of a post |
| 05 | getInsertedPostID | Get lastest post id in database |
| 06 | addNewPost | Insert new post to database |
| 07 | updatePostByPostID | Update post in database |

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of owner by accountID |
| 02 | getMotelByMotelID | Get motel by motel id |

#### ImgDAO Class

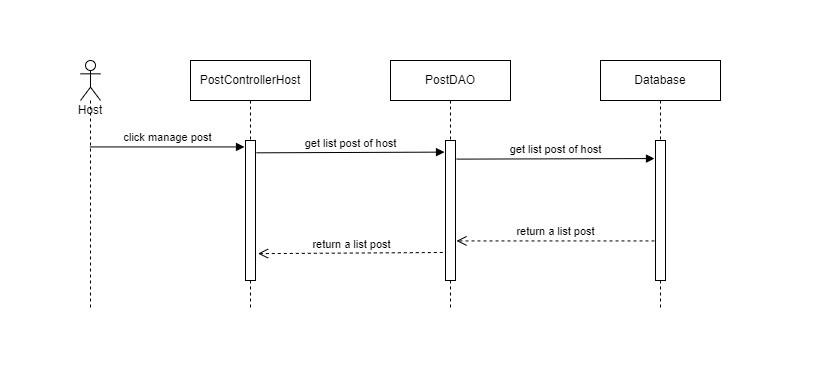
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | insertImg | Insert image into database |
| 02 | deleteImageByImageID | delete image in database by image id |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)

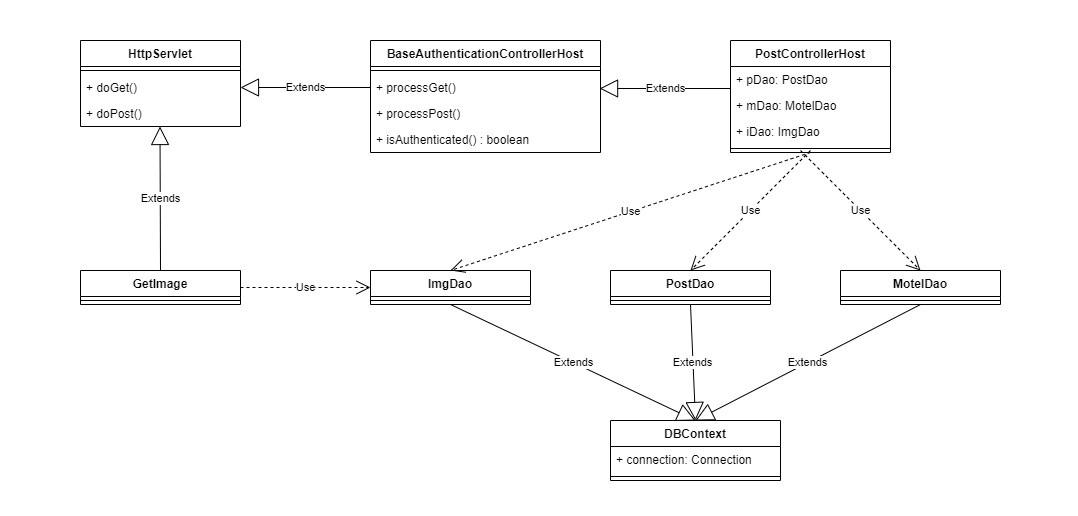


### d. Database queries

* select pt.postID,pt.motelID,mt.accountID,pt.postTitle,pt.postContent,pt.postDate, pt.postStatus from Post pt left join Motel mt on pt.motelID = mt.motelID where mt.AccountID=?;
* select \* from Motel where AccountID = ?;

## 59. Manage Post by Host/Add new post

### a. Class Diagram



### b. Class Specifications

#### PostControllerHost Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will return the login.jsp page according to the corresponding role |
| 02 | doPost | input is request and response. After receiving the request from the user, it will get the username and password and call the function getAccountByUsernamePasswordAndRole(username, password, role) of the AccountDAO class. |

#### HttpServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet |  |
| 02 | doPost | Using this HttpServlet object, we can build Java web applications to handle requests from clients with HTTP methods like GET, POST, HEAD, PUT, DELETE, TRACE and OPTIONS. When the client sends a request to the web server container, the service() method in the HttpServlet object processes the request. Depending on the HTTP method that the client sends, further processing will be done by the doGet(), doPost(), doHead(), doPut(), doDelete(), doTrace() and doOptions() methods of this object. receive. |
| 03 | doHead |
| 04 | doPut |
| 05 | doDelete |
| 06 | doOption |
| 07 | doTrace |

#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPostOfHost | Get all post in database by host id |
| 02 | getPostDetailByPostID | Get all post in database which have similar to input post id |
| 03 | deletePostByID | Delete post by post id |
| 04 | getListImageIDOfPost | Get all image id of a post |
| 05 | getInsertedPostID | Get lastest post id in database |
| 06 | addNewPost | Insert new post to database |
| 07 | updatePostByPostID | Update post in database |

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of owner by accountID |
| 02 | getMotelByMotelID | Get motel by motel id |

#### ImgDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | insertImg | Insert image into database |
| 02 | deleteImageByImageID | delete image in database by image id |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)

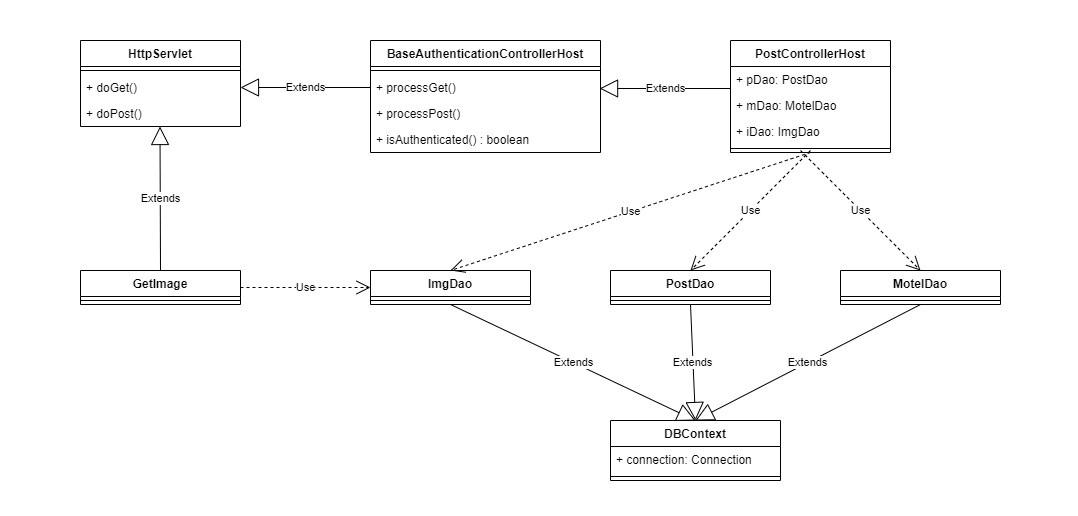


### d. Database queries

* insert into Post(motelID,postTitle,postContent,postDate,postStatus) values (?,?,?,?,?);

## 60. Manage Post by Host/Update post

### a. Class Diagram



### b. Class Specifications

#### PostControllerHost Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will return the login.jsp page according to the corresponding role |
| 02 | doPost | input is request and response. After receiving the request from the user, it will get the username and password and call the function getAccountByUsernamePasswordAndRole(username, password, role) of the AccountDAO class. |

#### HttpServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet |  |
| 02 | doPost | Using this HttpServlet object, we can build Java web applications to handle requests from clients with HTTP methods like GET, POST, HEAD, PUT, DELETE, TRACE and OPTIONS. When the client sends a request to the web server container, the service() method in the HttpServlet object processes the request. Depending on the HTTP method that the client sends, further processing will be done by the doGet(), doPost(), doHead(), doPut(), doDelete(), doTrace() and doOptions() methods of this object. receive. |
| 03 | doHead |
| 04 | doPut |
| 05 | doDelete |
| 06 | doOption |
| 07 | doTrace |

#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPostOfHost | Get all post in database by host id |
| 02 | getPostDetailByPostID | Get all post in database which have similar to input post id |
| 03 | deletePostByID | Delete post by post id |
| 04 | getListImageIDOfPost | Get all image id of a post |
| 05 | getInsertedPostID | Get lastest post id in database |
| 06 | addNewPost | Insert new post to database |
| 07 | updatePostByPostID | Update post in database |

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of owner by accountID |
| 02 | getMotelByMotelID | Get motel by motel id |

#### ImgDAO Class

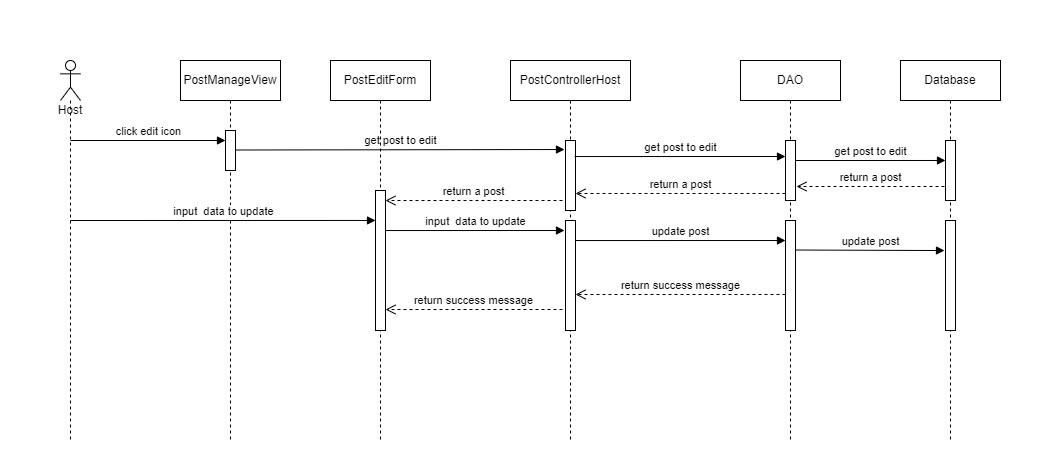
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | insertImg | Insert image into database |
| 02 | deleteImageByImageID | delete image in database by image id |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)

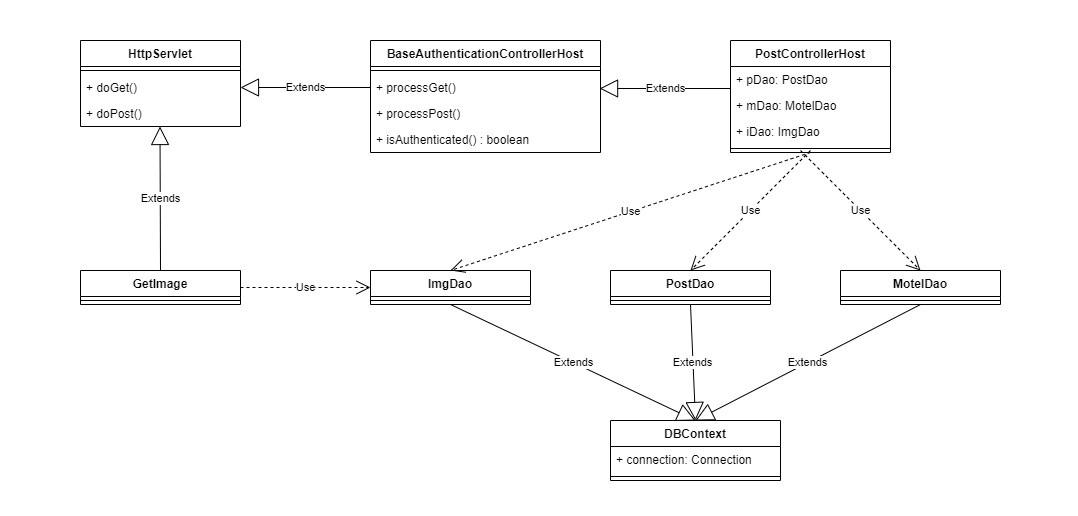


### d. Database queries

* update Post set postTitle = ?, postContent = ?, postStatus = 0 where postID = ?;

## 61. Manage Post by Host/Delete post

### a. Class Diagram



### b. Class Specifications

#### PostControllerHost Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will return the login.jsp page according to the corresponding role |
| 02 | doPost | input is request and response. After receiving the request from the user, it will get the username and password and call the function getAccountByUsernamePasswordAndRole(username, password, role) of the AccountDAO class. |

#### HttpServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet |  |
| 02 | doPost | Using this HttpServlet object, we can build Java web applications to handle requests from clients with HTTP methods like GET, POST, HEAD, PUT, DELETE, TRACE and OPTIONS. When the client sends a request to the web server container, the service() method in the HttpServlet object processes the request. Depending on the HTTP method that the client sends, further processing will be done by the doGet(), doPost(), doHead(), doPut(), doDelete(), doTrace() and doOptions() methods of this object. receive. |
| 03 | doHead |
| 04 | doPut |
| 05 | doDelete |
| 06 | doOption |
| 07 | doTrace |

#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPostOfHost | Get all post in database by host id |
| 02 | getPostDetailByPostID | Get all post in database which have similar to input post id |
| 03 | deletePostByID | Delete post by post id |
| 04 | getListImageIDOfPost | Get all image id of a post |
| 05 | getInsertedPostID | Get lastest post id in database |
| 06 | addNewPost | Insert new post to database |
| 07 | updatePostByPostID | Update post in database |

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of owner by accountID |
| 02 | getMotelByMotelID | Get motel by motel id |

#### ImgDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | insertImg | Insert image into database |
| 02 | deleteImageByImageID | delete image in database by image id |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)

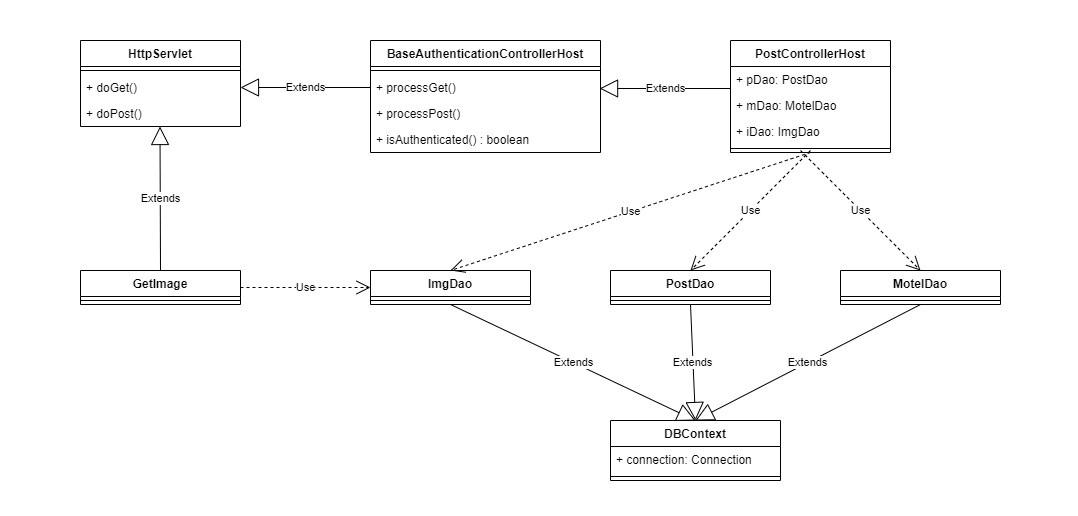


### d. Database queries

* delete from Post where postID = ?;

## 62. Manage Post by Host/Search

### a. Class Diagram



### b. Class Specifications

#### PostControllerHost Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will return the login.jsp page according to the corresponding role |
| 02 | doPost | input is request and response. After receiving the request from the user, it will get the username and password and call the function getAccountByUsernamePasswordAndRole(username, password, role) of the AccountDAO class. |

#### HttpServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet |  |
| 02 | doPost | Using this HttpServlet object, we can build Java web applications to handle requests from clients with HTTP methods like GET, POST, HEAD, PUT, DELETE, TRACE and OPTIONS. When the client sends a request to the web server container, the service() method in the HttpServlet object processes the request. Depending on the HTTP method that the client sends, further processing will be done by the doGet(), doPost(), doHead(), doPut(), doDelete(), doTrace() and doOptions() methods of this object. receive. |
| 03 | doHead |
| 04 | doPut |
| 05 | doDelete |
| 06 | doOption |
| 07 | doTrace |

#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPostOfHost | Get all post in database by host id |
| 02 | getPostDetailByPostID | Get all post in database which have similar to input post id |
| 03 | deletePostByID | Delete post by post id |
| 04 | getListImageIDOfPost | Get all image id of a post |
| 05 | getInsertedPostID | Get lastest post id in database |
| 06 | addNewPost | Insert new post to database |
| 07 | updatePostByPostID | Update post in database |

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of owner by accountID |
| 02 | getMotelByMotelID | Get motel by motel id |

#### ImgDAO Class

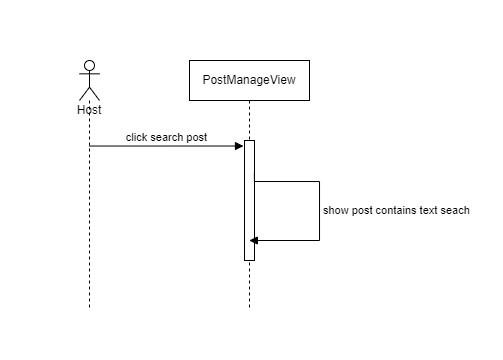
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | insertImg | Insert image into database |
| 02 | deleteImageByImageID | delete image in database by image id |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)

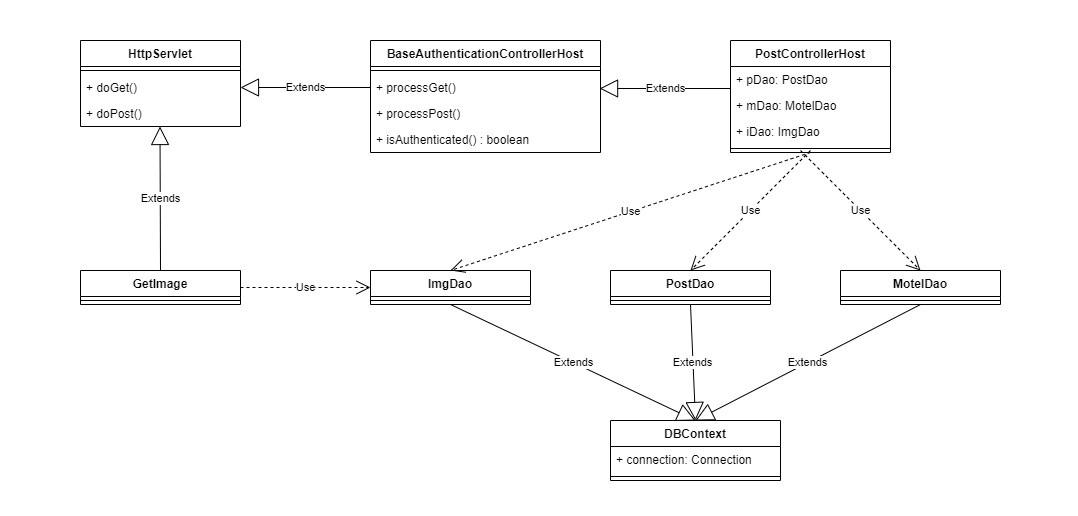


### d. Database queries

* select pt.postID,pt.motelID,mt.accountID,pt.postTitle,pt.postContent,pt.postDate, pt.postStatus from Post pt left join Motel mt on pt.motelID = mt.motelID where mt.AccountID=?;
* select \* from Motel where AccountID = ?

## 63. Manage Post by Host/Sort

### a. Class Diagram



### b. Class Specifications

#### PostControllerHost Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will return the login.jsp page according to the corresponding role |
| 02 | doPost | input is request and response. After receiving the request from the user, it will get the username and password and call the function getAccountByUsernamePasswordAndRole(username, password, role) of the AccountDAO class. |

#### HttpServlet Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet |  |
| 02 | doPost | Using this HttpServlet object, we can build Java web applications to handle requests from clients with HTTP methods like GET, POST, HEAD, PUT, DELETE, TRACE and OPTIONS. When the client sends a request to the web server container, the service() method in the HttpServlet object processes the request. Depending on the HTTP method that the client sends, further processing will be done by the doGet(), doPost(), doHead(), doPut(), doDelete(), doTrace() and doOptions() methods of this object. receive. |
| 03 | doHead |
| 04 | doPut |
| 05 | doDelete |
| 06 | doOption |
| 07 | doTrace |

#### PostDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllPostOfHost | Get all post in database by host id |
| 02 | getPostDetailByPostID | Get all post in database which have similar to input post id |
| 03 | deletePostByID | Delete post by post id |
| 04 | getListImageIDOfPost | Get all image id of a post |
| 05 | getInsertedPostID | Get lastest post id in database |
| 06 | addNewPost | Insert new post to database |
| 07 | updatePostByPostID | Update post in database |

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelByAccountID | Get all motel of owner by accountID |
| 02 | getMotelByMotelID | Get motel by motel id |

#### ImgDAO Class

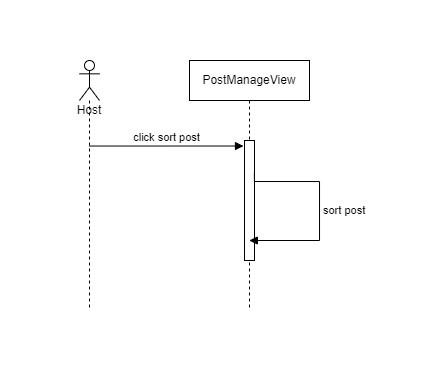
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | insertImg | Insert image into database |
| 02 | deleteImageByImageID | delete image in database by image id |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)

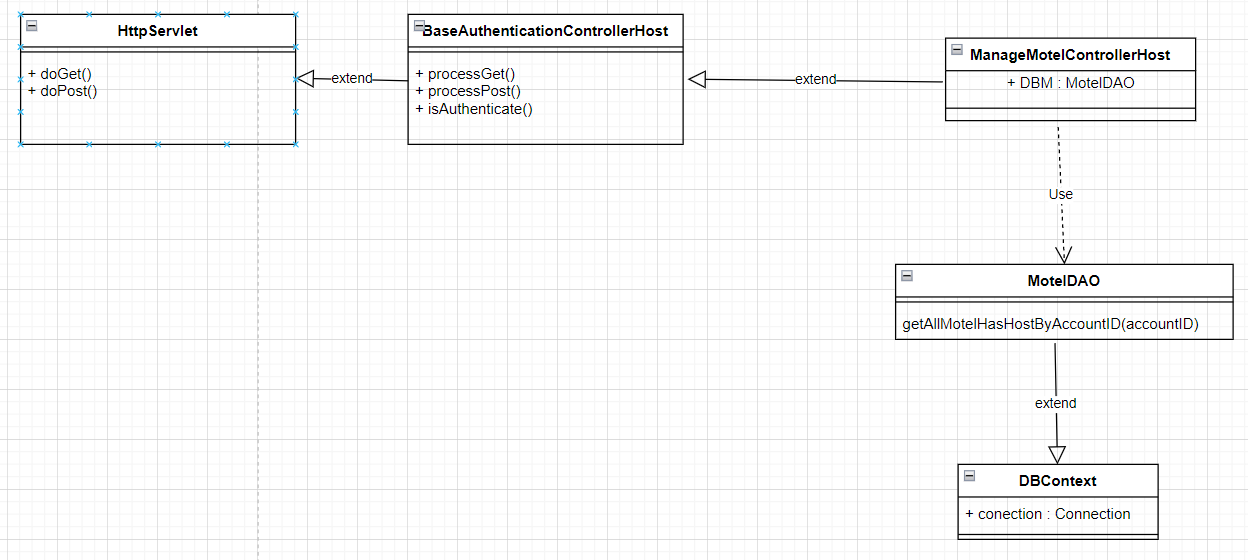


### d. Database queries

* select pt.postID,pt.motelID,mt.accountID,pt.postTitle,pt.postContent,pt.postDate, pt.postStatus from Post pt left join Motel mt on pt.motelID = mt.motelID where mt.AccountID=?;
* select \* from Motel where AccountID = ?

## 64. Manage Motel by Host /View

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

**ManageMotelControllerHost**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | get all information of motel belong to one host then go to ManageMotel.jsp |

#### 

#### MotelDAO Class

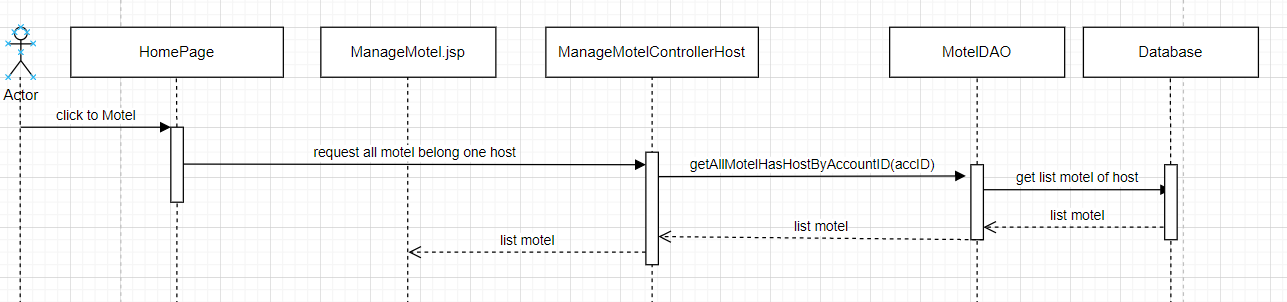
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelHasHostByAccountID(accountID) | Get all information of all motel belong to one host have accountID in database |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)



### d. Database queries

* select m.motelID,m.motelName,m.motelAddress,m.motelPhone,m.motelStatus,

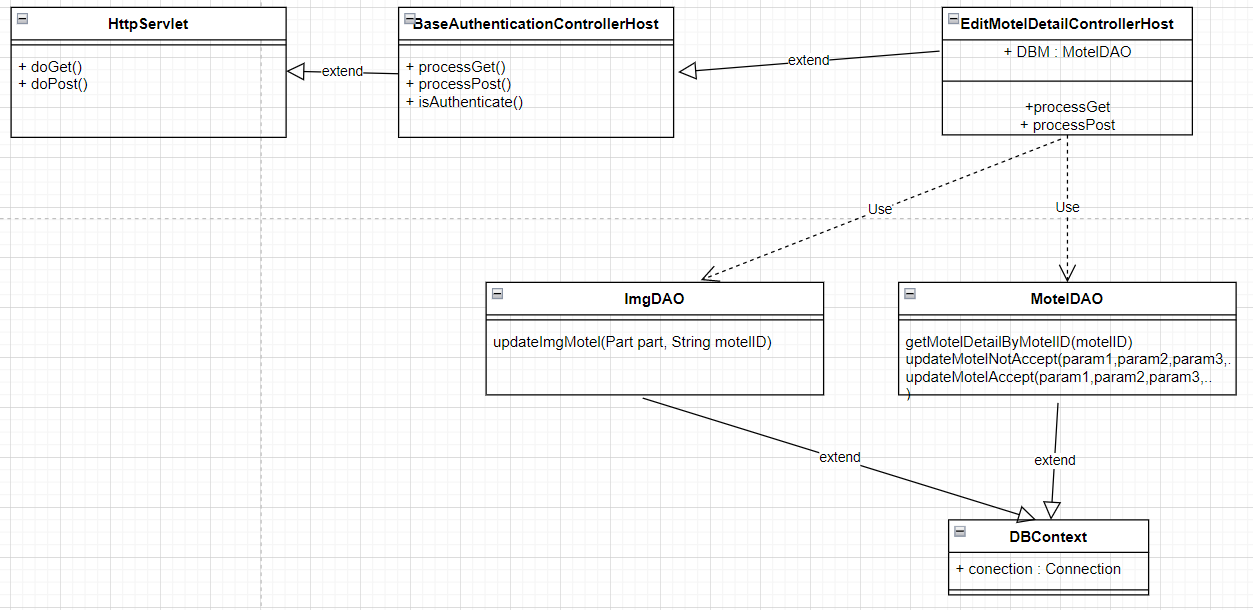
(select count(\*) from Room as r where r.motelID = m.motelID),

(select p.fullName from [Profile] as p where p.AccountID = m.AccountID)

from Motel as m where m.AccountID = ?

## 65.Manage Motel by Host /Update

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

**EditMotelControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | get all information of post and motel then go to Manageequest.jsp |
| 02 | processPost | receive infor that user update then update to database |

#### 

**ImgDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateImgMotel(Part part, String motelID) | Change image of motel |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMotelDetailByMotelID | Get all information of one motel in database |
| 02 | updateMotelAccept(motelName, motelPhone, motelDescription, motelID) | update information one motel that accept by admin |
| 03 | updateMotelNotAccept(motelName, motelPhone,motelAddress, motelDescription, motelID) | update information one motel not yet that accept by admin |

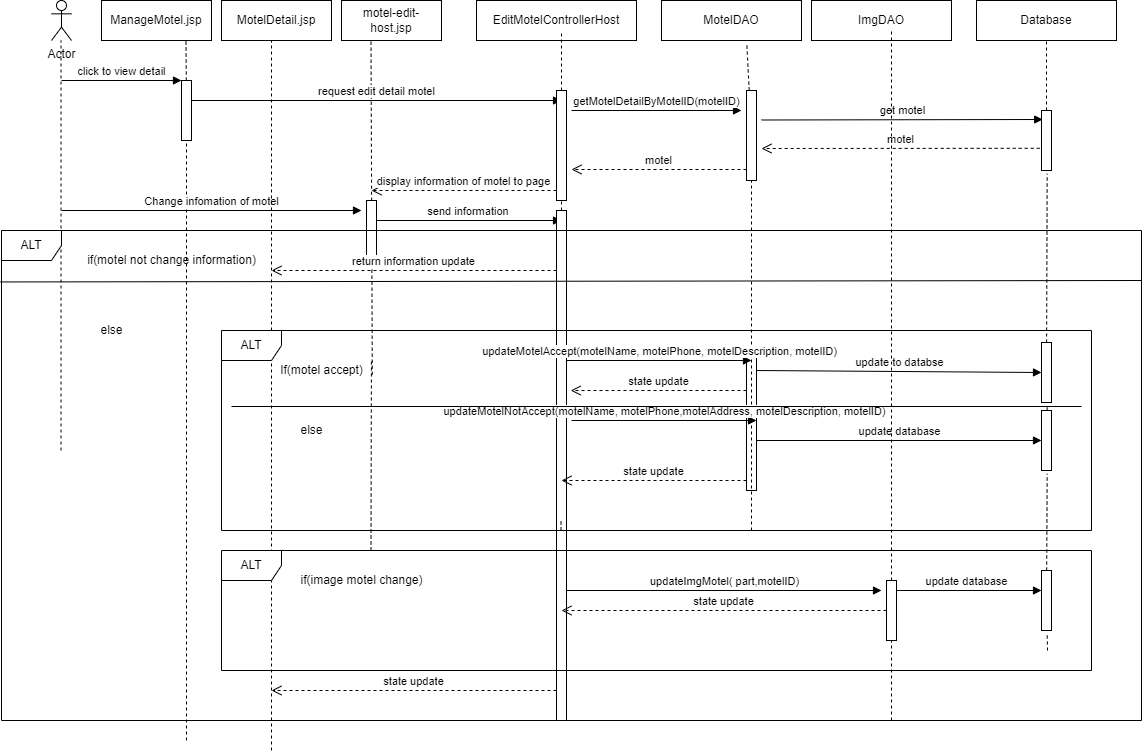
#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)

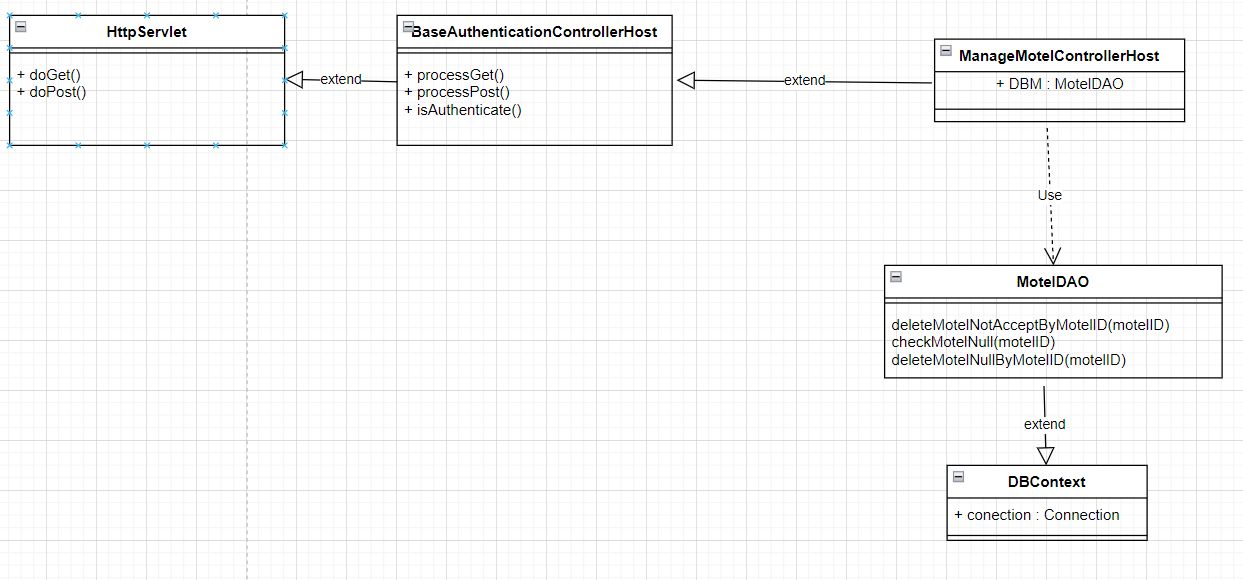


### d. Database queries

* select m.motelID, m.accountID,(select p.fullName from [Profile] as p where p.AccountID = m.AccountID), m.motelName, (select count(\*) from Room as r where r.motelID = m.motelID),m.motelAddress, m.motelPhone, m.motelStatus, (select MAX(r.price) from Room as r where r.motelID = m.motelID), (select MIN(r.price) from Room as r where r.motelID = m.motelID), m.motelDescription from Motel as m where m.motelID = ?
* update Motel set motelName = ?, motelPhone = ?, motelDescription = ? where motelID = ?
* update Motel set motelName = ?, motelPhone = ?,motelAddress = ?, motelDescription = ? where motelID = ?
* update Motel set motelImage = ? where motelID= ?

## 66. Manage Motel by Host / Delete

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

**DeleteMotelControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Delete motel that has customer use or not |

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deleteMotelNotAcceptByMotelID(Integer.parseInt(motelID) | Delete motel that not yet accept from admin |
| 02 | checkMotelNull(motelID) | Check if the motel has customer use or not |
| 03 | deleteMotelNullByMotelID(motelID) | Delete motel that do not has customer use |

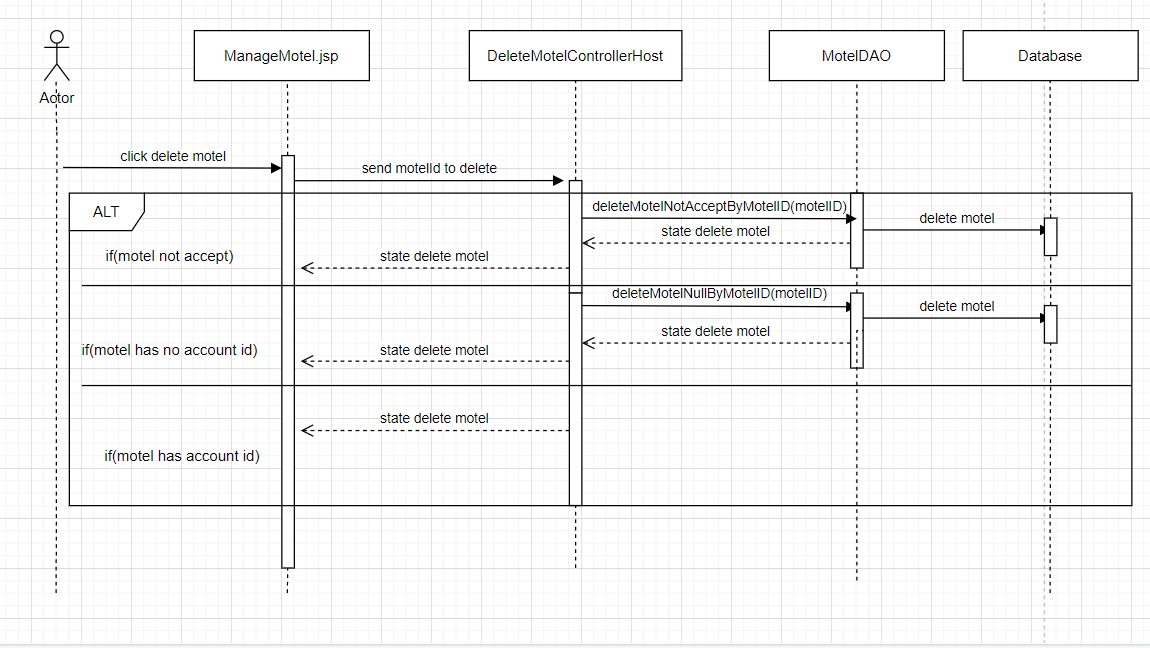
#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)

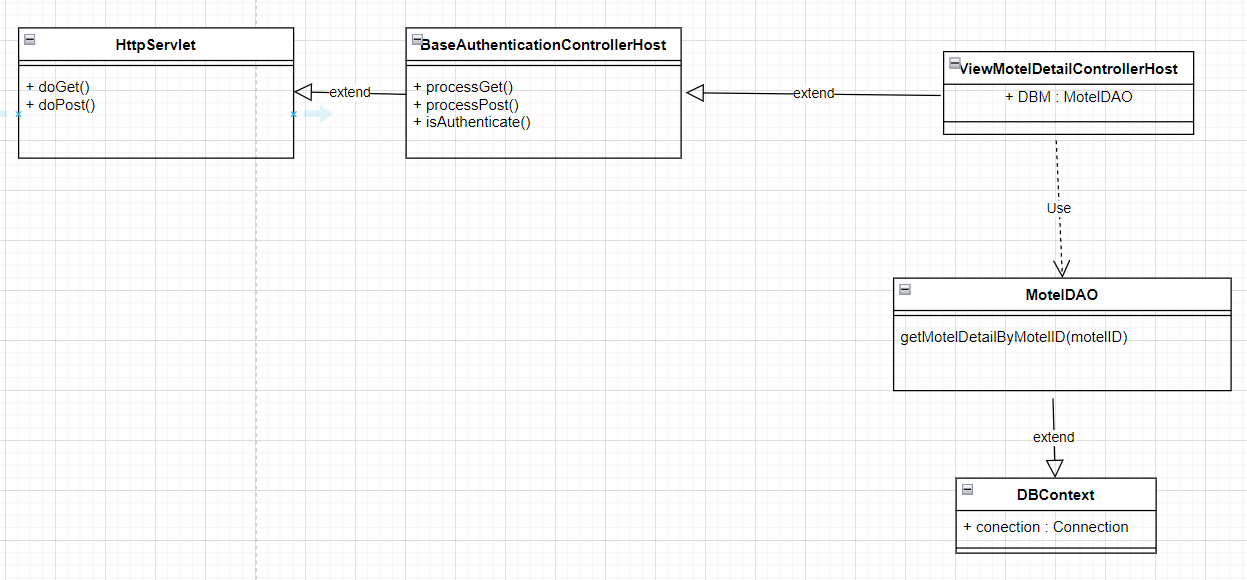


### d. Database queries

* delete Motel where motelID = ?
* delete PostImage where postID in (select p.postID from Post as p where p.motelID = ?)
* delete Post where motelID = ?

## 67. Manage Motel by Host / View Detail

### a. Class Diagram



### b. Class Specifications

### BaseAuthenticationControllerAdmin

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

**ManageequestControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | get all information of motel then go to Motel-Detail-Host.jsp |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMotelDetailByMotelID(motelID) | Get all information of all motel in database by motelID |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)

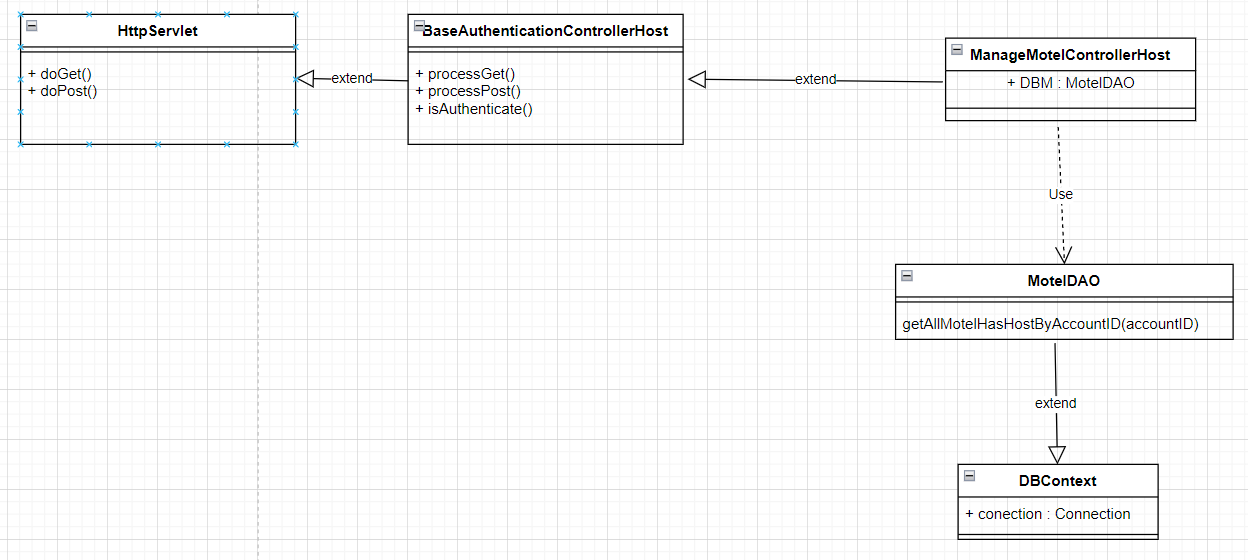


### d. Database queries

* select m.motelID, m.accountID,(select p.fullName from [Profile] as p where p.AccountID = m.AccountID), m.motelName,(select count(\*) from Room as r where r.motelID = m.motelID),m.motelAddress, m.motelPhone, m.motelStatus,(select MAX(r.price) from Room as r where r.motelID = m.motelID), (select MIN(r.price) from Room as r where r.motelID = m.motelID), m.motelDescription from Motel as m where m.motelID = ?

## 68. Manage Motel by Host / Search

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

**ManageMotelControllerHost**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | get all information of motel belong to one host then go to ManageMotel.jsp |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelHasHostByAccountID(accountID) | Get all information of all motel belong to one host have accountID in database |

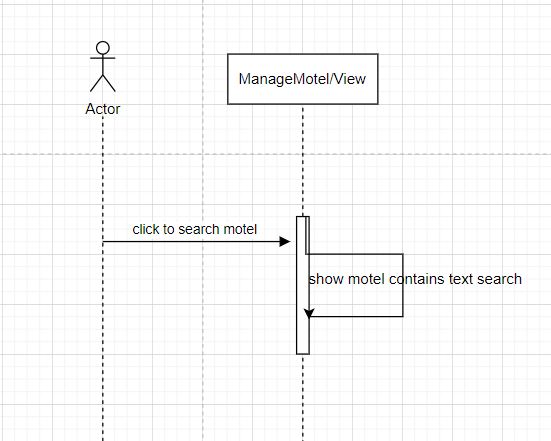
#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)



### d. Database queries

* select m.motelID,m.motelName,m.motelAddress,m.motelPhone,m.motelStatus,

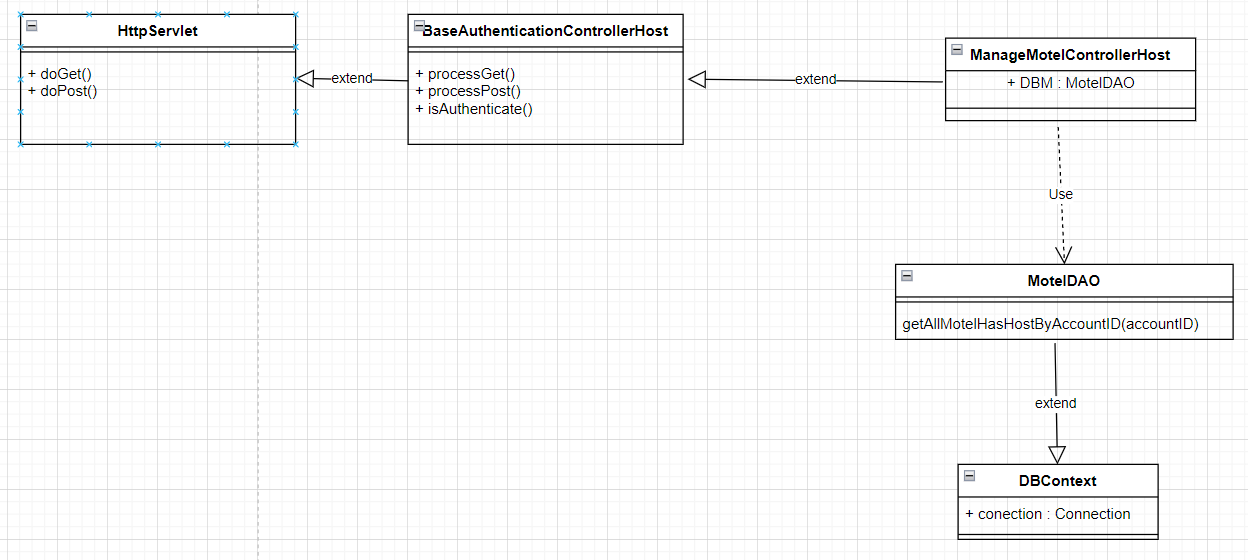
(select count(\*) from Room as r where r.motelID = m.motelID),

(select p.fullName from [Profile] as p where p.AccountID = m.AccountID)

from Motel as m where m.AccountID = ?

## 69. Manage Motel by Host / Sort

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

**ManageMotelControllerHost**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | navigate to register-motel-host.jsp page |
| 02 | processPost | add new motel into database |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMotelIDByAccountIDAndMotelName(ownerID, name) | get motelID of motel base on name and account id |
| 02 | insertNewMotel(ownerID, name, phone, address, description, item, motelCreateAt) | add new motel into database |
| 03 | insertImgMotelVerification(part, motelID) | add image verification into database |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)



### d. Database queries

select m.motelID,m.motelName,m.motelAddress,m.motelPhone,m.motelStatus,

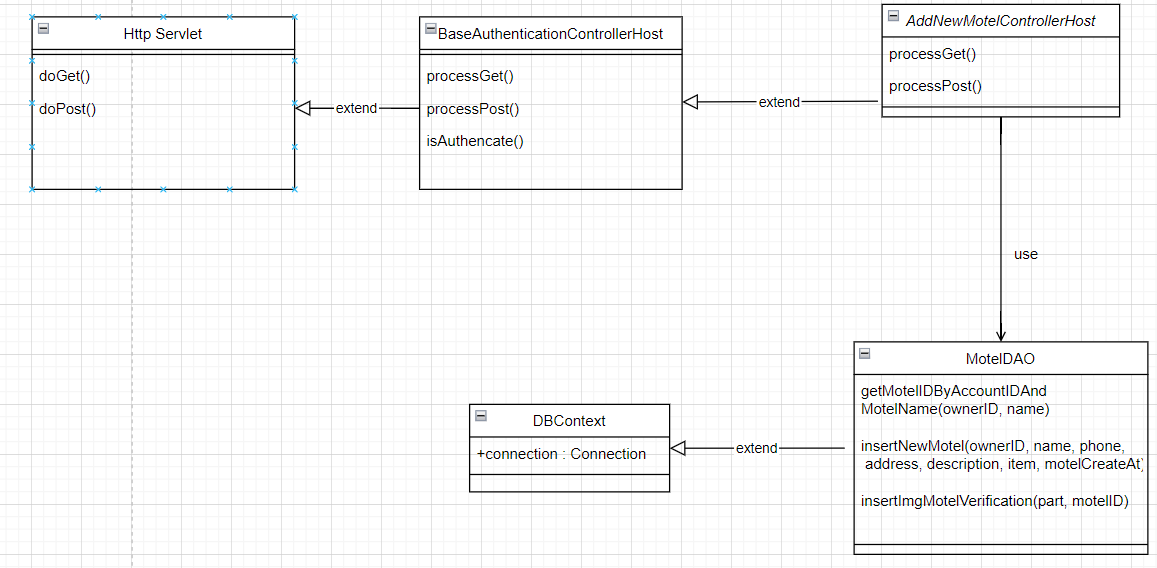
(select count(\*) from Room as r where r.motelID = m.motelID),

(select p.fullName from [Profile] as p where p.AccountID = m.AccountID)

from Motel as m where m.AccountID = ?

## 70. Register New Motel by Host

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerHost**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

**ManageMotelControllerHost**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | navigate from home to jsp page |
| 02 | processPost | add new motel into database |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMotelIDByAccountIDAndMotelName(ownerID, name) | get motel id bt account id and motel name |
| 02 | insertNewMotel(ownerID, name, phone, address, description, item, motelCreateAt) | add new motel into database |
| 03 | insertImgMotelVerification(part, motelID) | add new image verification of motel into database |

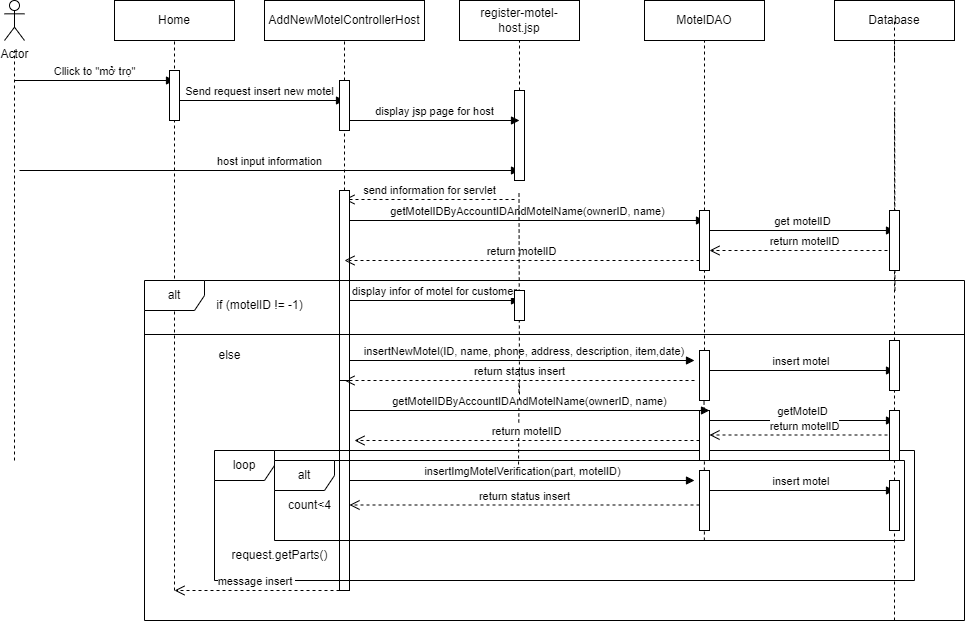
#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)



### d. Database queries

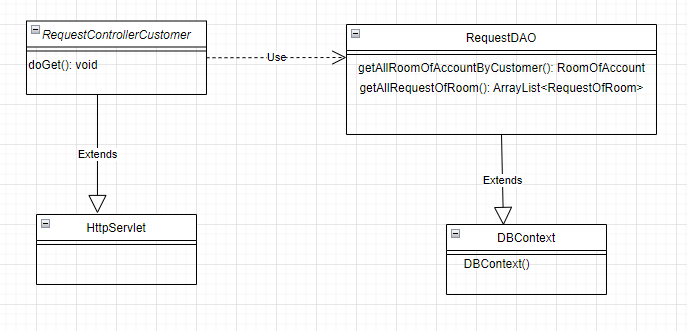
select m.motelID,m.AccountID from Motel as m where m.AccountID = ? and m.motelName = ?

insert into Motel (AccountID ,motelName ,motelAddress ,motelPhone ,motelStatus ,motelDescription , motelCreateAt,motelImage,isDeleteMotel) values(?,?,?,?,0,?,'" + motelCreateAt + "',?,0)

insert into MotelImage(motelID,imageLink) values(?,?)

## 71. Home/Manage Request

### a. Class Diagram



### b. Class Specifications

**RequestControllerCustomer**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet() | Get all information of request and pass it to request.jsp |

**RequestDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRoomOfAccountByCustomer() | Get the rooms that the account has rented |
| 02 | getAllRequestOfRoom() | Get all requests of that motel room |

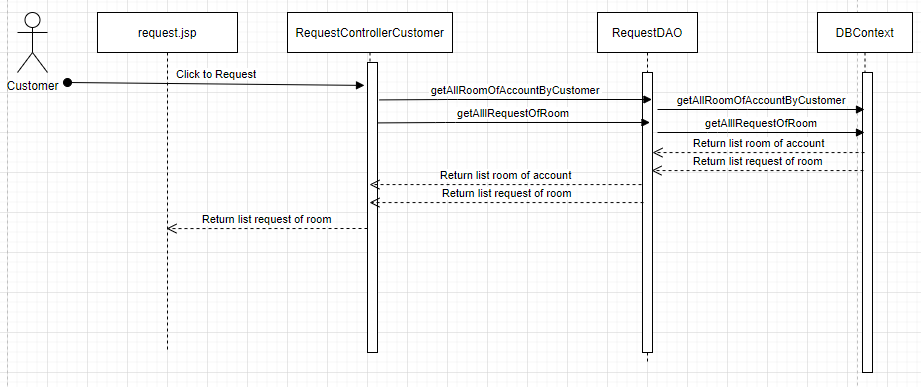
#### 

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)



### d. Database queries

**getAllRoomFromMotelOfAccountByCustomer():**

select r.roomID, r.motelID, r.roomName, r.numberOfPeopleMax, r.numberOfPeopleCurrent, r.price, r.area, r.noteRoom from Room r inner join RoomOfAccount ra on r.roomID = ra.roomID

where r.motelID = ? and ra.AccountID = ?

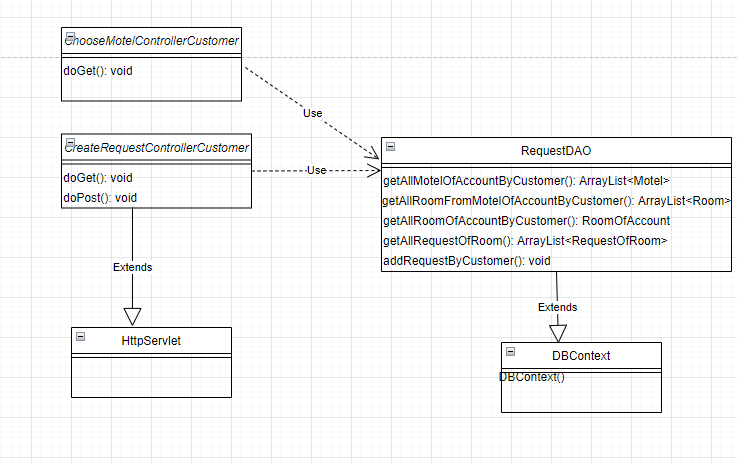
**getAllRequestOfRoom():**

select re.requestID, r.roomID, r.roomName, re.content, re.status, re.noteRequest,re.createdAt, re.updateAt from Room r inner join Request re on re.roomID = r.roomID

where r.roomID = ?

## 72. Manage Request By Customer/Add

### a. Class Diagram



### b. Class Specifications

**ChooseMotelControllerCustomer**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet() | Get all information of motel and pass it to createRequest.jsp |

**ChooseMotelControllerCustomer**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet() | Get all information of room and pass it to createRequest.jsp |
| 02 | doPost() | Add request from Customer |

**RequestDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMotelOfAccountByCustomer() | Get all motel from account |
| 02 | getAllRoomFromMotelOfAccountByCustomer() | Get all room from motel of account |
| 03 | getAllRoomOfAccountByCustomer() | Get all room from account |
| 04 | getAllRequestOfRoom() | Get all request from room of account |
| 05 | addRequestByCustomer() | Add request from customer |

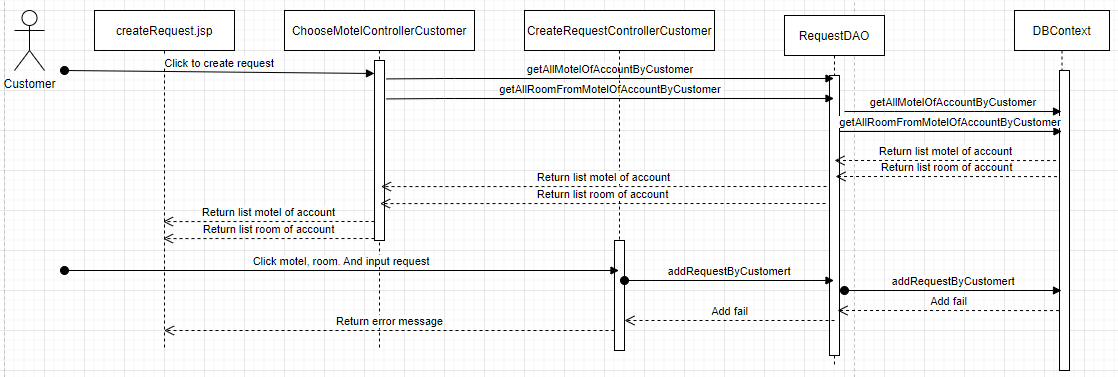
#### 

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)



### d. Database queries

getAllMotelOfAccountByCustomer():

select roomID from Account a inner join RoomOfAccount ra on a.AccountID = ra.AccountID

where a.AccountID = ?

select distinct m.motelID, m.motelName from Room r inner join Motel m on r.motelID = m.motelID

where r.roomID = ?

getAllRoomFromMotelOfAccountByCustomer():

select r.roomID, r.motelID, r.roomName, r.numberOfPeopleMax, r.numberOfPeopleCurrent, r.price, r.area, r.noteRoom from Room r inner join RoomOfAccount ra on r.roomID = ra.roomID

where r.motelID = ? and ra.AccountID = ?

getAllRoomOfAccountByCustomer():

select roomID from Account a inner join RoomOfAccount ra on a.AccountID = ra.AccountID

where a.AccountID = ?

select \* from RoomOfAccount ra inner join Room r on ra.roomID = r.roomID

where r.roomID = ?

getAllRequestOfRoom():

select re.requestID, r.roomID, r.roomName, re.content, re.status, re.noteRequest,re.createdAt, re.updateAt from Room r inner join Request re on re.roomID = r.roomID

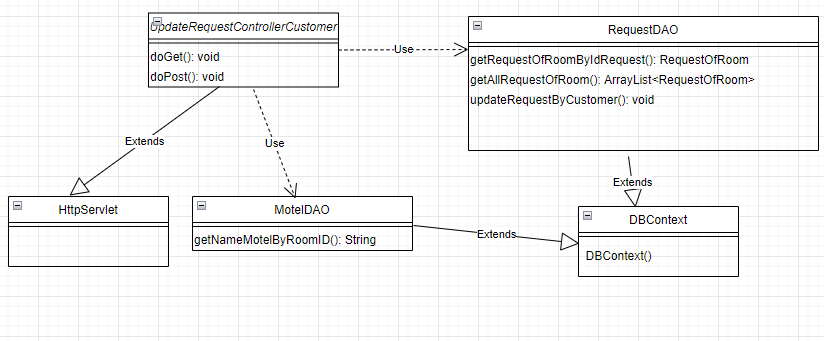
where r.roomID = ?

addRequestByCustomer():

insert into Request(roomID,content,status,createdAt) values (?,?,'0',getDate())

## 73. Manage Request By Customer/Update

### a. Class Diagram



### b. Class Specifications

**UpdateRequestControllerCustomer**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet() | Get all information of request and pass it to updateRequest.jsp |
| 02 | doPost() | Update request from Customer |

**MotelDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getNameMotelByRoom() | Get name motel of room by room id |

**RequestDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getRequestOfRoomByIdRequest() | Get request of room by request id |
| 02 | getAllRoomOfAccountByCustomer() | Get all room from account |
| 03 | getAllRequestOfRoom() | Get all request from room of account |
| 04 | updateRequestByCustomer() | Update request from customer |

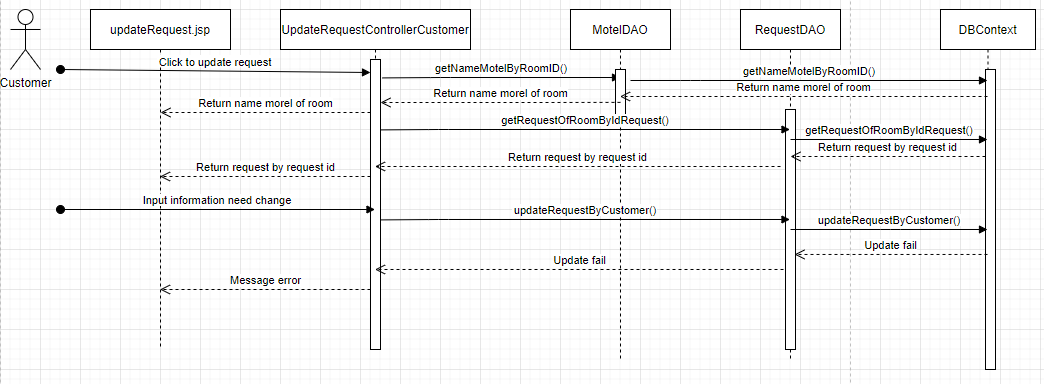
#### 

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)



### d. Database queries

getNameMotelByRoomID():

select m.motelName from Motel m inner join Room r on m.motelID = r.motelID

where roomID = ?

getRequestOfRoomByIdRequest():

select \* from Room r inner join Request re on re.roomID = r.roomID

where requestID = ?

getAllRequestOfRoom():

select re.requestID, r.roomID, r.roomName, re.content, re.status, re.noteRequest,re.createdAt, re.updateAt from Room r inner join Request re on re.roomID = r.roomID

where r.roomID = ?

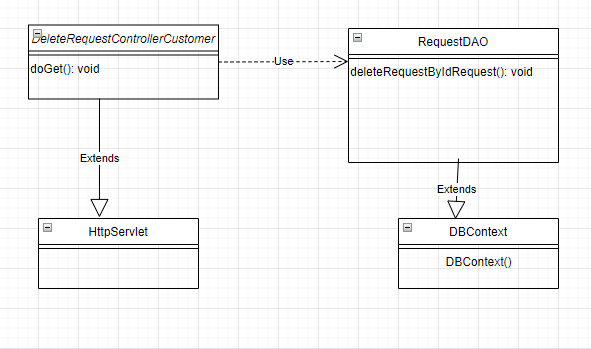
updateRequestByCustomer():

update Request set content = ?, noteRequest = ?, updateAt = getdate()

where requestID = ?

## 74. Manage Request By Customer/Delete

### a. Class Diagram



### b. Class Specifications

**DeleteRequestControllerCustomer**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet() | Delete request by request id |

**RequestDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | deleteRequestByIdReuqest() | Delete request by request id |

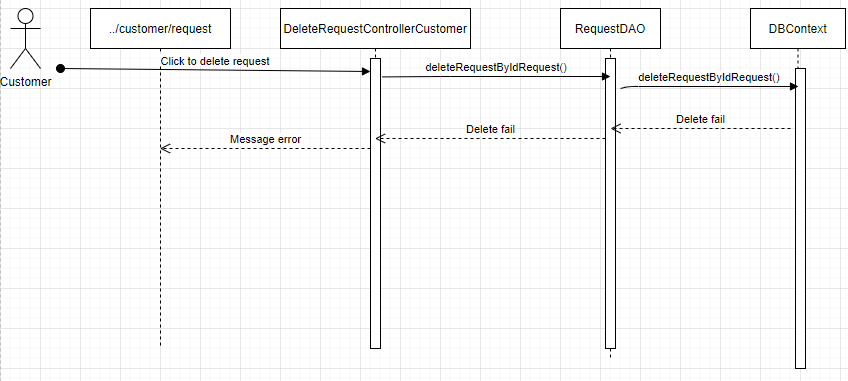
#### 

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)

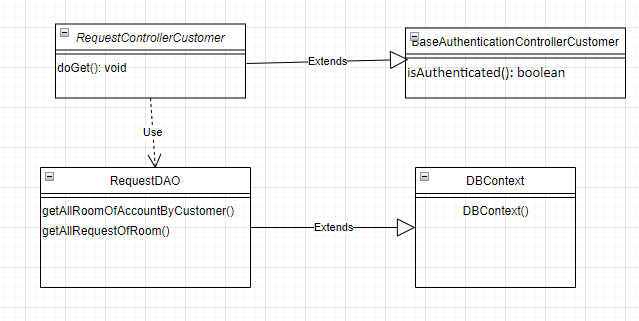


### d. Database queries

* deleteRequestByIdReuqest():
* delete from Request where requestID = ?

## 75. Manage Request By Customer/Search

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***RequestControllerCustomer***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doget() | Search all information in list request |

***RequestDAO***

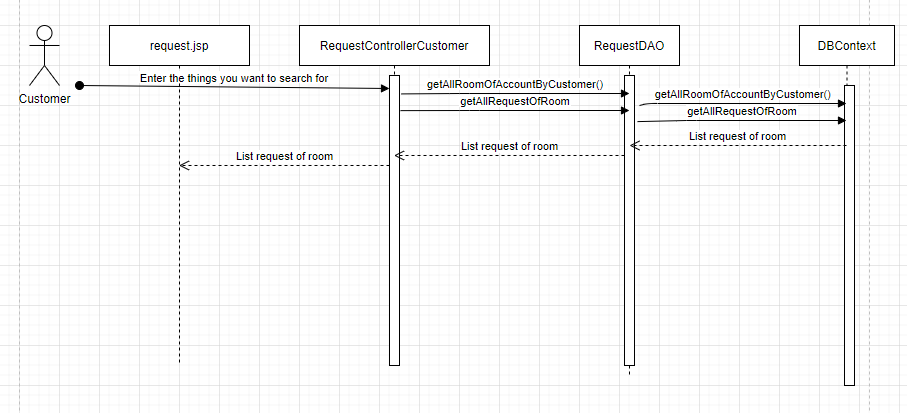
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRoomOfAccountByCustomer() | Get all room of account |
| 02 | getAllRequestOfRoom() | Get all requests that the room has requested |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)



### d. Database queries

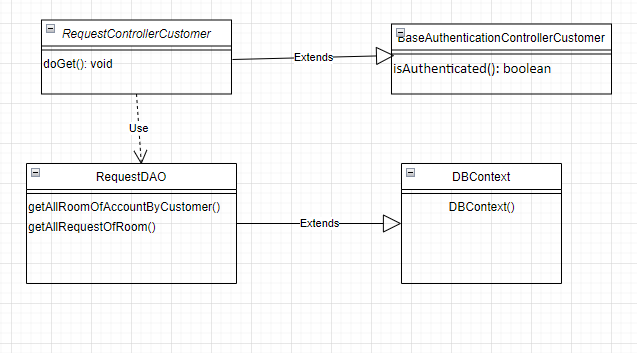
* getAllRoomOfAccountByCustomer():
* select roomID from Account a inner join RoomOfAccount ra on a.AccountID = ra.AccountID where a.AccountID = ?
* select \* from RoomOfAccount ra inner join Room r on ra.roomID = r.roomID

where r.roomID = ?

* getAllRequestOfRoom():
* select re.requestID, r.roomID, r.roomName, re.content, re.status, re.noteRequest,re.createdAt, re.updateAt from Room r inner join Request re on re.roomID = r.roomID where r.roomID = ?

## 76. Manage Request By Customer/Sort

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***RequestControllerCustomer***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doget() | Search all information in list request |

***RequestDAO***

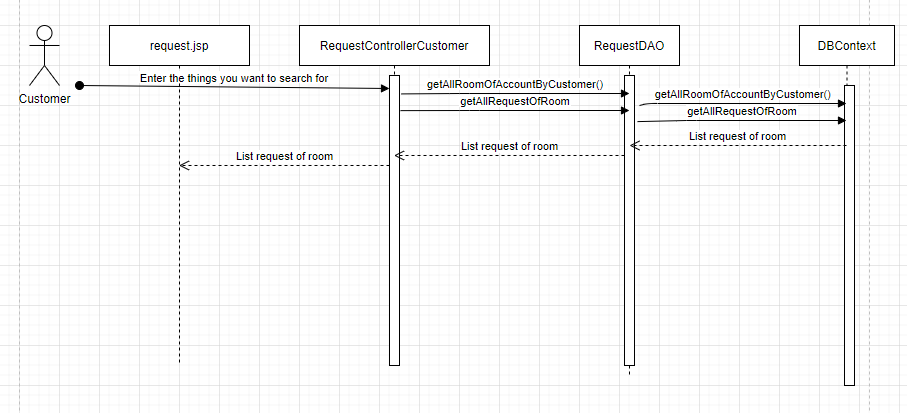
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRoomOfAccountByCustomer() | Get all room of account |
| 02 | getAllRequestOfRoom() | Get all requests that the room has requested |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)



### d. Database queries

* getAllRoomOfAccountByCustomer():
* select roomID from Account a inner join RoomOfAccount ra on a.AccountID = ra.AccountID where a.AccountID = ?
* select \* from RoomOfAccount ra inner join Room r on ra.roomID = r.roomID

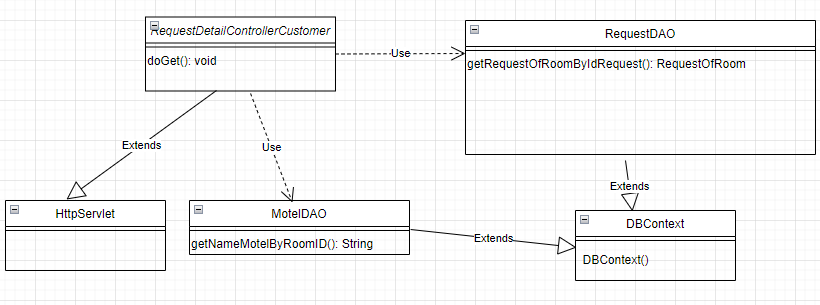
where r.roomID = ?

* getAllRequestOfRoom():
* select re.requestID, r.roomID, r.roomName, re.content, re.status, re.noteRequest,re.createdAt, re.updateAt from Room r inner join Request re on re.roomID = r.roomID

where r.roomID = ?

## 77. Manage Request By Customer/View Detail

### a. Class Diagram



### b. Class Specifications

**RequestDetailControllerCustomer**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet() | Get information of request by id request and pass it to requestDetail.jsp |

**MotelDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getNameMotelByRoom() | Get name motel of room by room id |

**RequestDAO**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getRequestOfRoomByIdRequest() | Get request of room by request id |

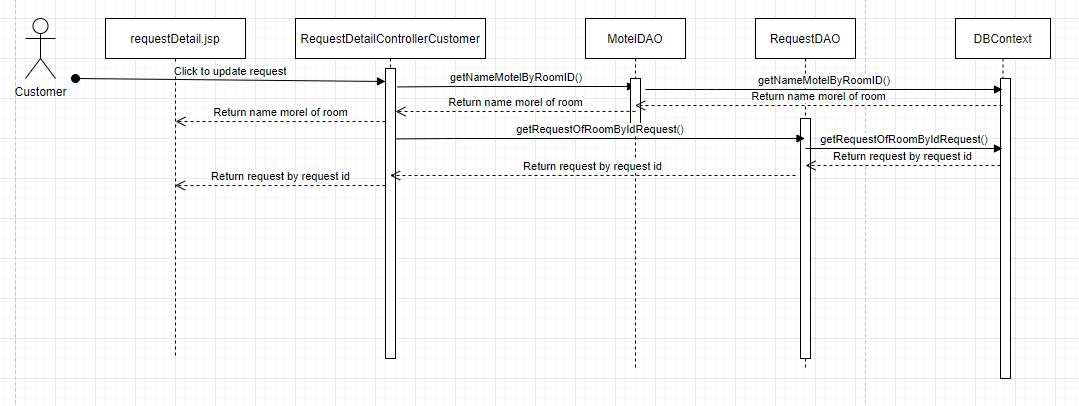
#### 

#### DBContext

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)



### d. Database queries

* getNameMotelByRoomID():
* select m.motelName from Motel m inner join Room r on m.motelID = r.motelID

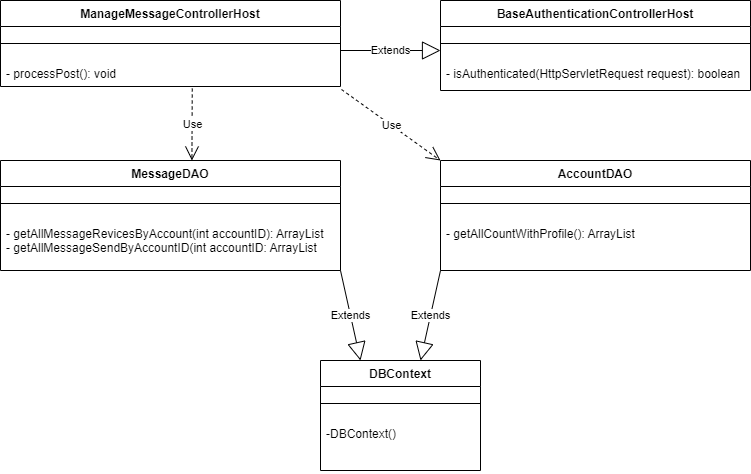
where roomID = ?

* getRequestOfRoomByIdRequest():
* select \* from Room r inner join Request re on re.roomID = r.roomID

where requestID = ?

## 78. Manager Message By Host

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageMessageControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the message received or sent of the account on the session and go to manageRequest.jsp |

***MessageDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMessageReceiveByAccountID | Get all message receive from other account . The parameter passed is the id of account |
| 02 | getAllMessageSendByAccountID | Get all message sent to other account . The parameter passed is the id of account |

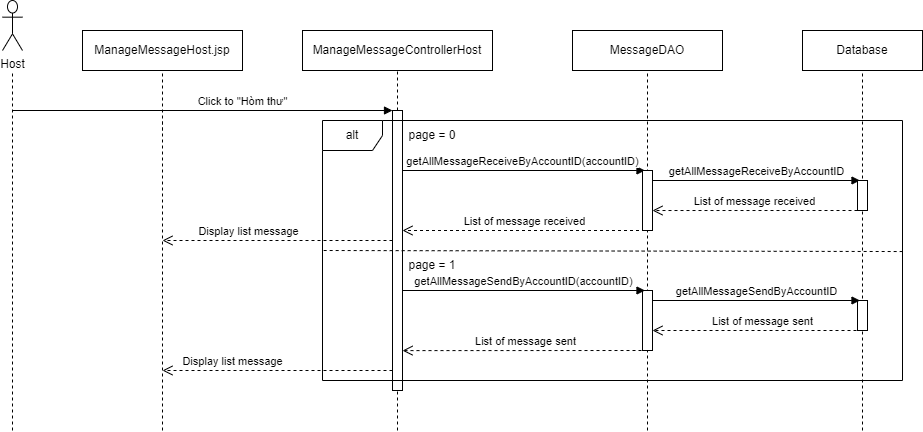
***AccountDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllAccountWithProfile | Get accountID, username, fullName of all account |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)



### d. Database queries

* select a.messageID, b.fullName, c.contentMessage, a.contentMessage, a.createAt

from [Message] as a inner join [Profile] as b on a.toUserID = b.AccountID

left join [Message] as c on a.messageID = c.repMessage where a.fromUserID = ?

* select a.messageID, b.fullName, c.contentMessage, a.contentMessage, a.createAt

from [Message] as a join [Profile] as b on a.fromUserID = b.AccountID

left join [Message] as c on a.messageID = c.repMessage where a.toUserID =?

* select a.AccountID, a.userName, p.fullName

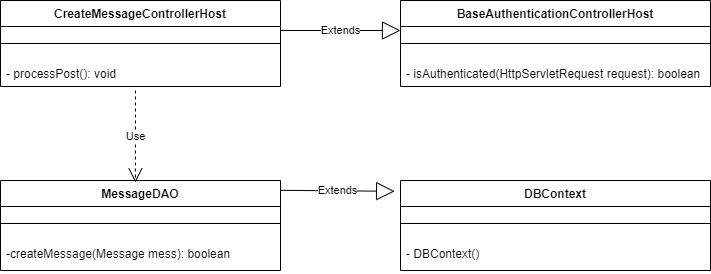
from Account as A join Profile as P on a.AccountID = p.AccountID

where a.AccountID in (select AccountID from RoleOfAccount where RoleID

in (select RoleID from Role where roleName != 'Admin'

## 79. Manager Message By Host/Send Message

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***CreateMessageControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processPost | Get all information of message from request and call function createMessage of MessageDAO |

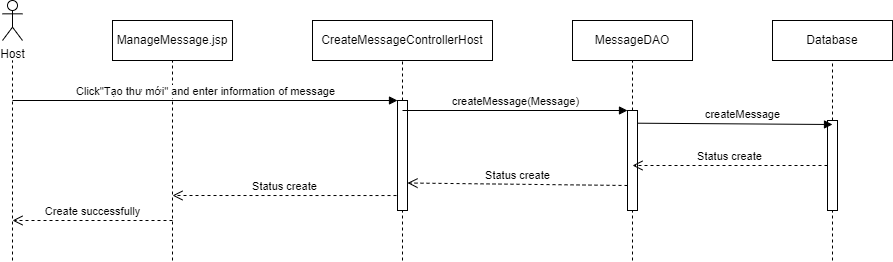
***MessageDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createMessage | insert Message to database |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)

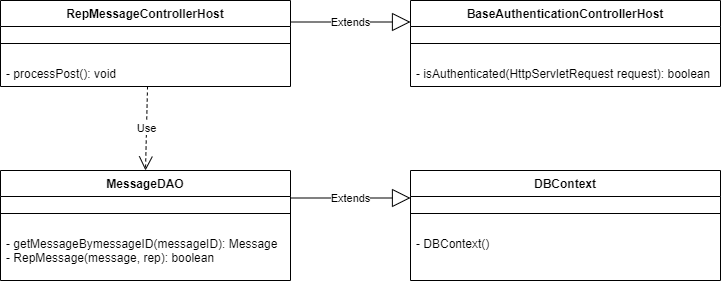


### d. Database queries

* insert into Message(fromUserID, toUserID, contentMessage, createAt) values(?,?,?,?)

## 80. Manager Message By Host/Rep Message

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageMessageControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processPost | Reply message received from other user |

***MessageDAO class***

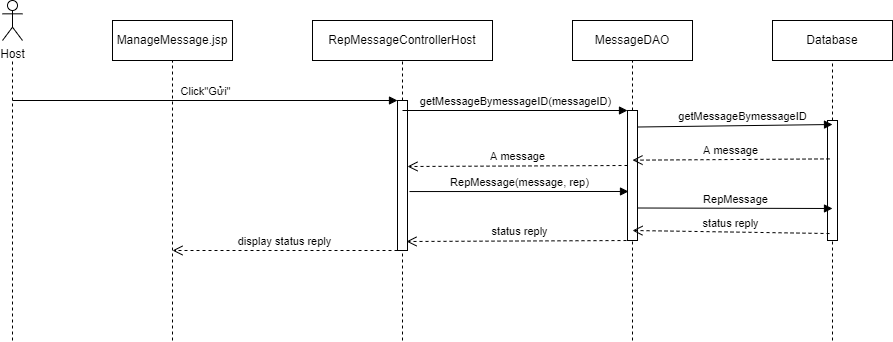
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMessageBymessageID | Get message receive and user want to reply from other account . The parameter passed is the id of message |
| 02 | RepMessage | Reply message sent by other user . The parameter passed is object message and string reply |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)

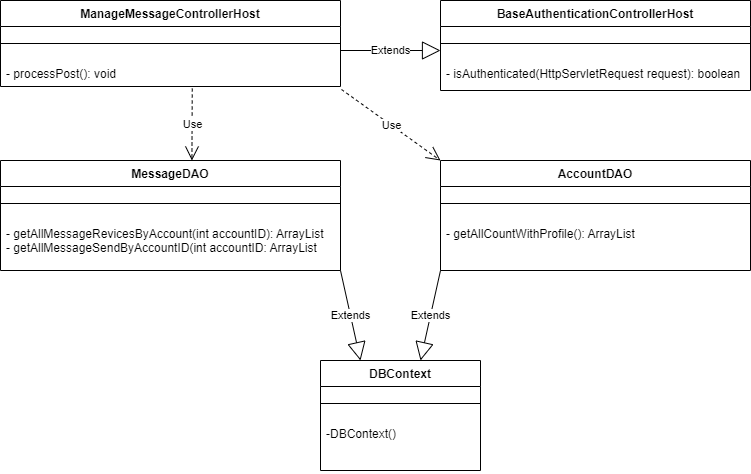


### d. Database queries

* select \* from [Message] where messageID = ?
* insert into Message(fromUserID, toUserID, contentMessage, createAt, repMessage) values(?,?,?,?,?)

## 81. Manager Message By Host/Search

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageMessageControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the message received or sent of the account on the session and go to manageRequest.jsp |

***MessageDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMessageReceiveByAccountID | Get all message receive from other account . The parameter passed is the id of account |
| 02 | getAllMessageSendByAccountID | Get all message sent to other account . The parameter passed is the id of account |

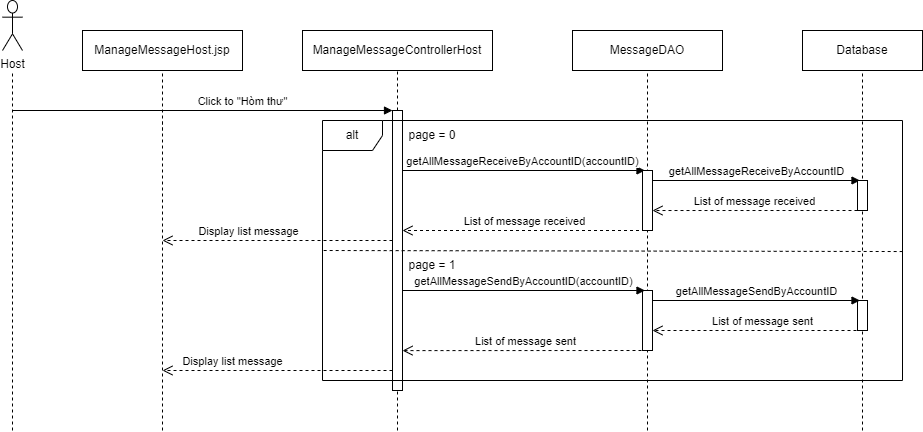
***AccountDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllAccountWithProfile | Get accountID, username, fullName of all account |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)



### d. Database queries

* select a.messageID, b.fullName, c.contentMessage, a.contentMessage, a.createAt

from [Message] as a inner join [Profile] as b on a.toUserID = b.AccountID

left join [Message] as c on a.messageID = c.repMessage where a.fromUserID = ?

* select a.messageID, b.fullName, c.contentMessage, a.contentMessage, a.createAt

from [Message] as a join [Profile] as b on a.fromUserID = b.AccountID

left join [Message] as c on a.messageID = c.repMessage where a.toUserID =?

* select a.AccountID, a.userName, p.fullName

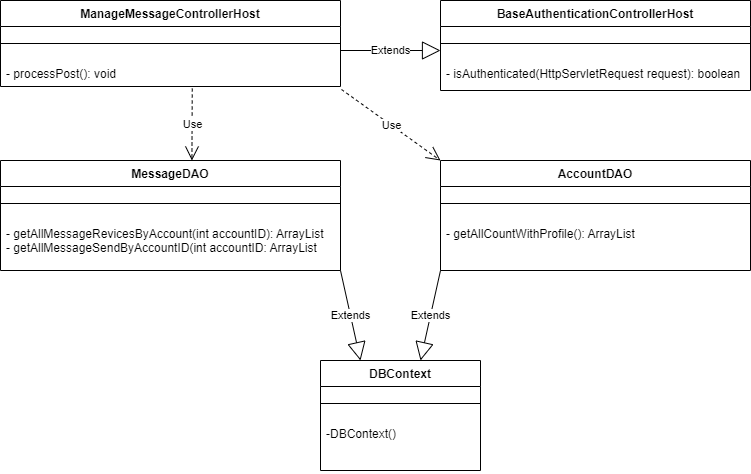
from Account as A join Profile as P on a.AccountID = p.AccountID

where a.AccountID in (select AccountID from RoleOfAccount where RoleID

in (select RoleID from Role where roleName != 'Admin'))

## 82. Manager Message By Host/Sort

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageMessageControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the message received or sent of the account on the session and go to manageRequest.jsp |

***MessageDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMessageReceiveByAccountID | Get all message receive from other account . The parameter passed is the id of account |
| 02 | getAllMessageSendByAccountID | Get all message sent to other account . The parameter passed is the id of account |

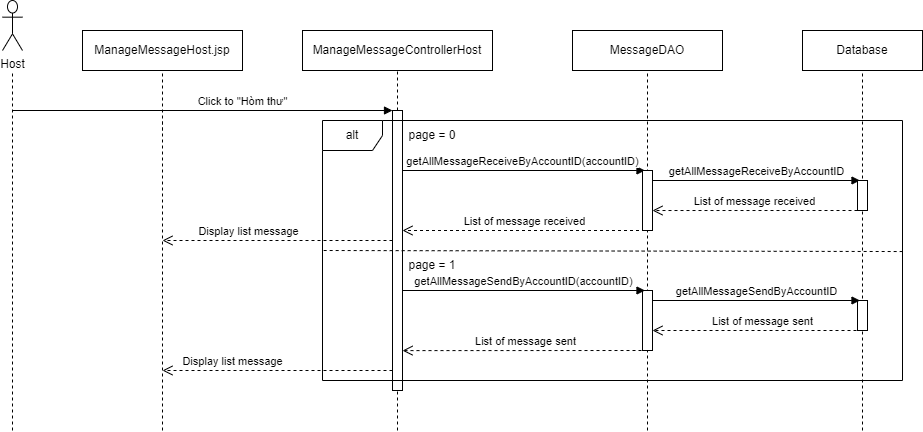
***AccountDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllAccountWithProfile | Get accountID, username, fullName of all account |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)



### d. Database queries

* select a.messageID, b.fullName, c.contentMessage, a.contentMessage, a.createAt

from [Message] as a inner join [Profile] as b on a.toUserID = b.AccountID

left join [Message] as c on a.messageID = c.repMessage where a.fromUserID = ?

* select a.messageID, b.fullName, c.contentMessage, a.contentMessage, a.createAt

from [Message] as a join [Profile] as b on a.fromUserID = b.AccountID

left join [Message] as c on a.messageID = c.repMessage where a.toUserID =?

* select a.AccountID, a.userName, p.fullName

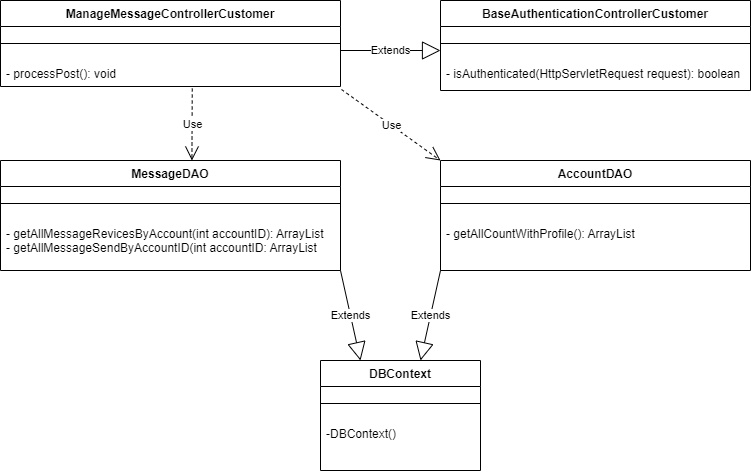
from Account as A join Profile as P on a.AccountID = p.AccountID

where a.AccountID in (select AccountID from RoleOfAccount where RoleID

in (select RoleID from Role where roleName != 'Admin'))

## 83. Manager Message By Customer

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageMessageControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the message received or sent of the account on the session and go to manageRequest.jsp |

***MessageDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMessageReceiveByAccountID | Get all message receive from other account . The parameter passed is the id of account |
| 02 | getAllMessageSendByAccountID | Get all message sent to other account . The parameter passed is the id of account |

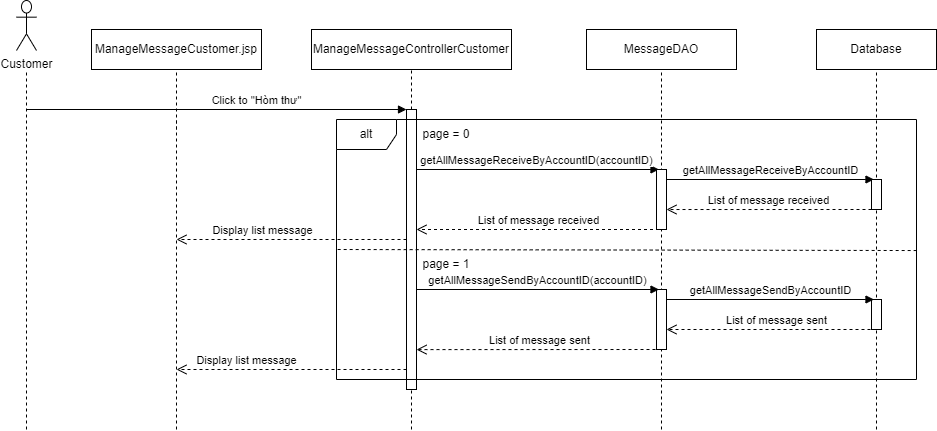
***AccountDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllAccountWithProfile | Get accountID, username, fullName of all account |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)



### d. Database queries

* select a.messageID, b.fullName, c.contentMessage, a.contentMessage, a.createAt

from [Message] as a inner join [Profile] as b on a.toUserID = b.AccountID

left join [Message] as c on a.messageID = c.repMessage where a.fromUserID = ?

* select a.messageID, b.fullName, c.contentMessage, a.contentMessage, a.createAt

from [Message] as a join [Profile] as b on a.fromUserID = b.AccountID

left join [Message] as c on a.messageID = c.repMessage where a.toUserID =?

* select a.AccountID, a.userName, p.fullName

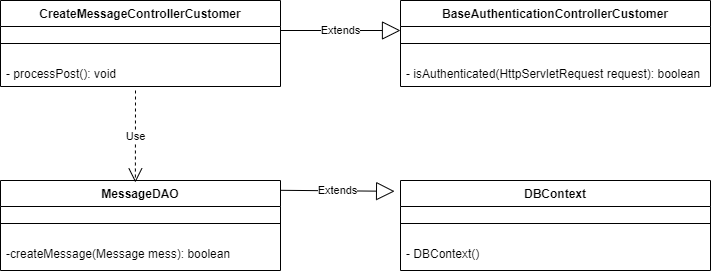
from Account as A join Profile as P on a.AccountID = p.AccountID

where a.AccountID in (select AccountID from RoleOfAccount where RoleID

in (select RoleID from Role where roleName != 'Admin'))

## 84. Manager Message By Customer/Send Message

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***CreateMessageControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processPost | Get all information of message from request and call function createMessage of MessageDAO |

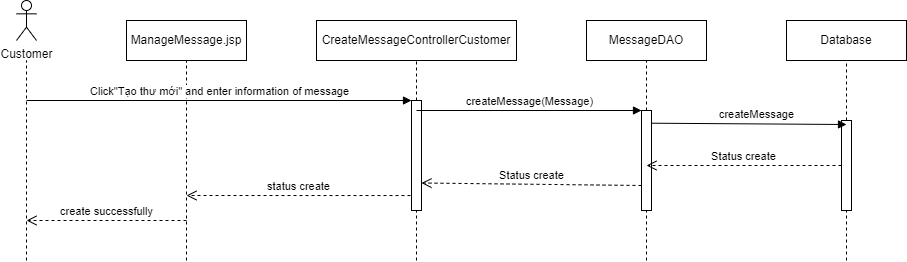
***MessageDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createMessage | insert Message to database |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)

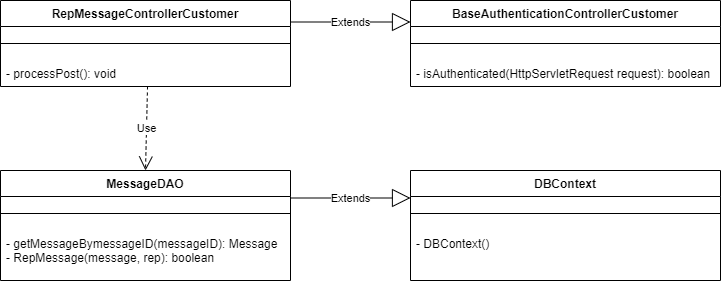


### d. Database queries

* insert into Message(fromUserID, toUserID, contentMessage, createAt) values(?,?,?,?)

## 85. Manager Message By Customer/Rep Message

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageMessageControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processPost | Reply message received from other user |

***MessageDAO class***

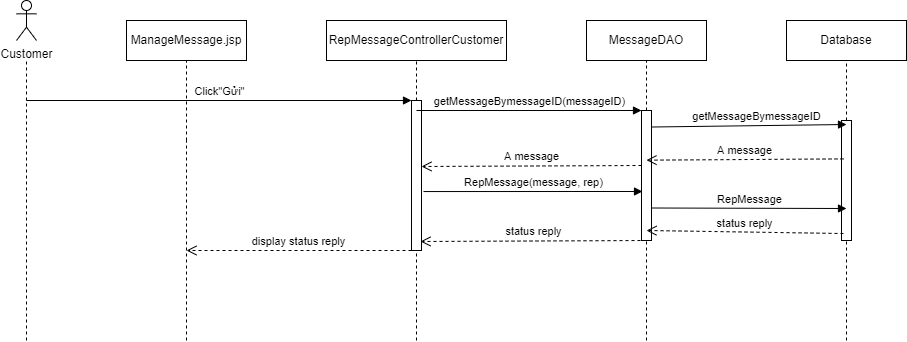
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMessageBymessageID | Get message receive and user want to reply from other account . The parameter passed is the id of message |
| 02 | RepMessage | Reply message sent by other user . The parameter passed is object message and string reply |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)

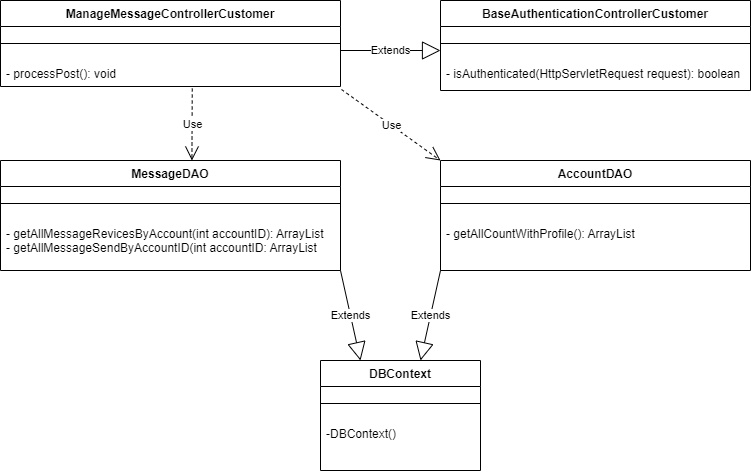


### d. Database queries

* select \* from [Message] where messageID = ?
* insert into Message(fromUserID, toUserID, contentMessage, createAt, repMessage) values(?,?,?,?,?)

## 86. Manager Message By Customer/Search

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageMessageControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the message received or sent of the account on the session and go to manageRequest.jsp |

***MessageDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMessageReceiveByAccountID | Get all message receive from other account . The parameter passed is the id of account |
| 02 | getAllMessageSendByAccountID | Get all message sent to other account . The parameter passed is the id of account |

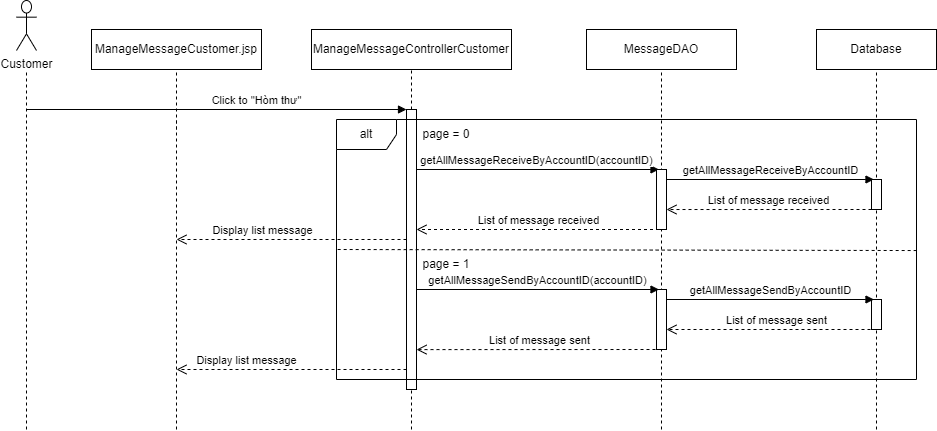
***AccountDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllAccountWithProfile | Get accountID, username, fullName of all account |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)



### d. Database queries

* select a.messageID, b.fullName, c.contentMessage, a.contentMessage, a.createAt

from [Message] as a inner join [Profile] as b on a.toUserID = b.AccountID

left join [Message] as c on a.messageID = c.repMessage where a.fromUserID = ?

* select a.messageID, b.fullName, c.contentMessage, a.contentMessage, a.createAt

from [Message] as a join [Profile] as b on a.fromUserID = b.AccountID

left join [Message] as c on a.messageID = c.repMessage where a.toUserID =?

* select a.AccountID, a.userName, p.fullName

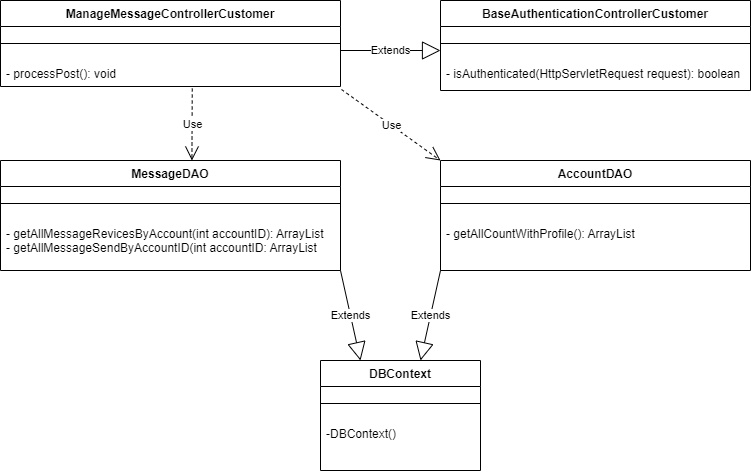
from Account as A join Profile as P on a.AccountID = p.AccountID

where a.AccountID in (select AccountID from RoleOfAccount where RoleID

in (select RoleID from Role where roleName != 'Admin'))

## 87. Manager Message By Customer/Sort

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageMessageControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all the message received or sent of the account on the session and go to manageRequest.jsp |

***MessageDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllMessageReceiveByAccountID | Get all message receive from other account . The parameter passed is the id of account |
| 02 | getAllMessageSendByAccountID | Get all message sent to other account . The parameter passed is the id of account |

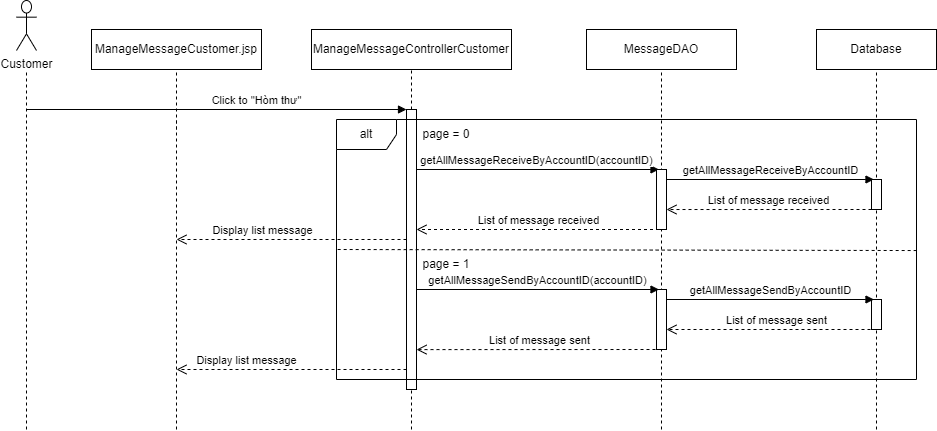
***AccountDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllAccountWithProfile | Get accountID, username, fullName of all account |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)



### d. Database queries

* select a.messageID, b.fullName, c.contentMessage, a.contentMessage, a.createAt

from [Message] as a inner join [Profile] as b on a.toUserID = b.AccountID

left join [Message] as c on a.messageID = c.repMessage where a.fromUserID = ?

* select a.messageID, b.fullName, c.contentMessage, a.contentMessage, a.createAt

from [Message] as a join [Profile] as b on a.fromUserID = b.AccountID

left join [Message] as c on a.messageID = c.repMessage where a.toUserID =?

* select a.AccountID, a.userName, p.fullName

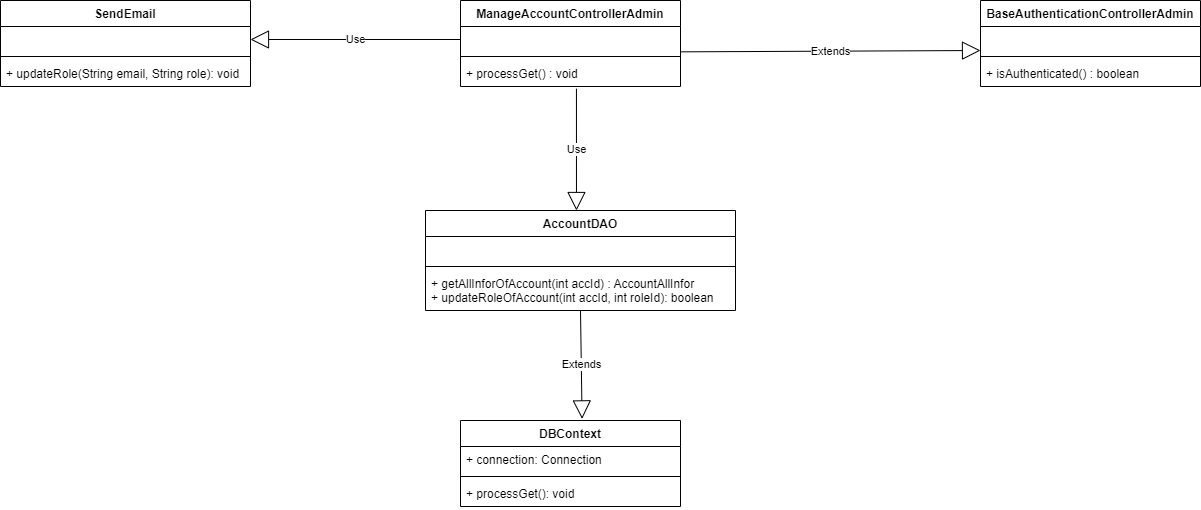
from Account as A join Profile as P on a.AccountID = p.AccountID

where a.AccountID in (select AccountID from RoleOfAccount where RoleID

in (select RoleID from Role where roleName != 'Admin'))

## 88. Send Email Update Role Of Account By Admin

### a. Class Diagram



### b. Class Specifications

#### ManageAccountAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will call function getAllInforOfAllAccount to take all the information of all users and transfer it to ManageAdmin.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### AccountDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllInforOfAllAccount | This function will retrieve all information of a user including account profile password .... |
| 02 | updateRoleOfAccount | This function to update role of edited account to database |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

### 

**SendEmail**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateRole | Send email to corresponding to notify updating |

### 

### c. Sequence Diagram(s)

### 

### d. Database queries

insert into RoleOfAccount values (?,?)

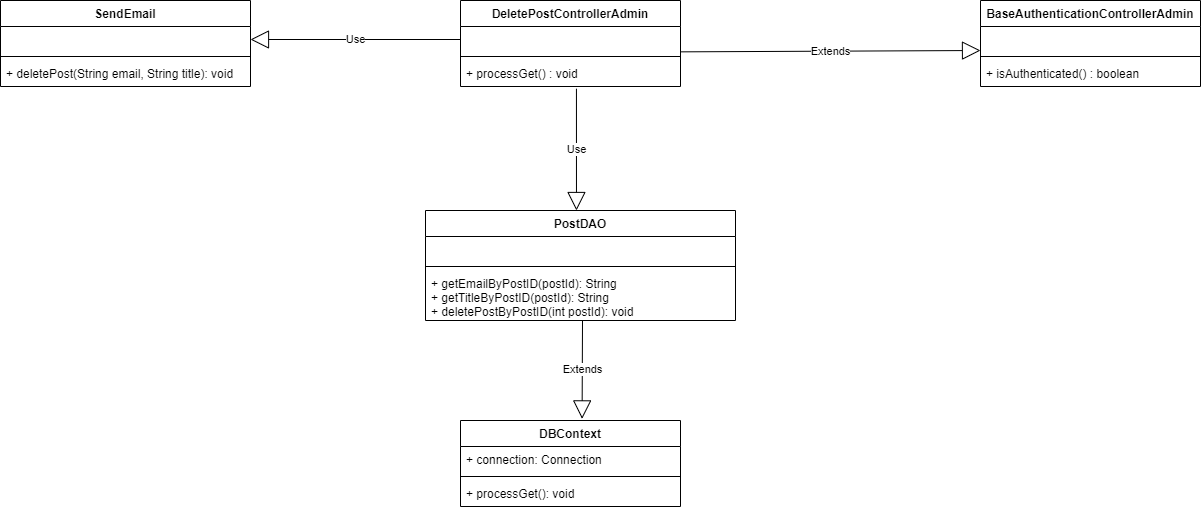
delete from RoleOfAccount where AccountID = ? and RoleID = ?

select roomID from Room where motelID = ?

delete from AccountOfBooking where AccountID = ?

## 89. Send Email Delete Post By Admin

### a. Class Diagram



### b. Class Specifications

#### ManageAccountAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will call function getAllInforOfAllAccount to take all the information of all users and transfer it to ManageAdmin.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### AccountDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllInforOfAllAccount | This function will retrieve all information of a user including account profile password .... |
| 02 | updateRoleOfAccount | This function to update role of edited account to database |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

### 

**SendEmail**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateRole | Send email to corresponding to notify updating |

### 

### c. Sequence Diagram(s)

### 

### d. Database queries

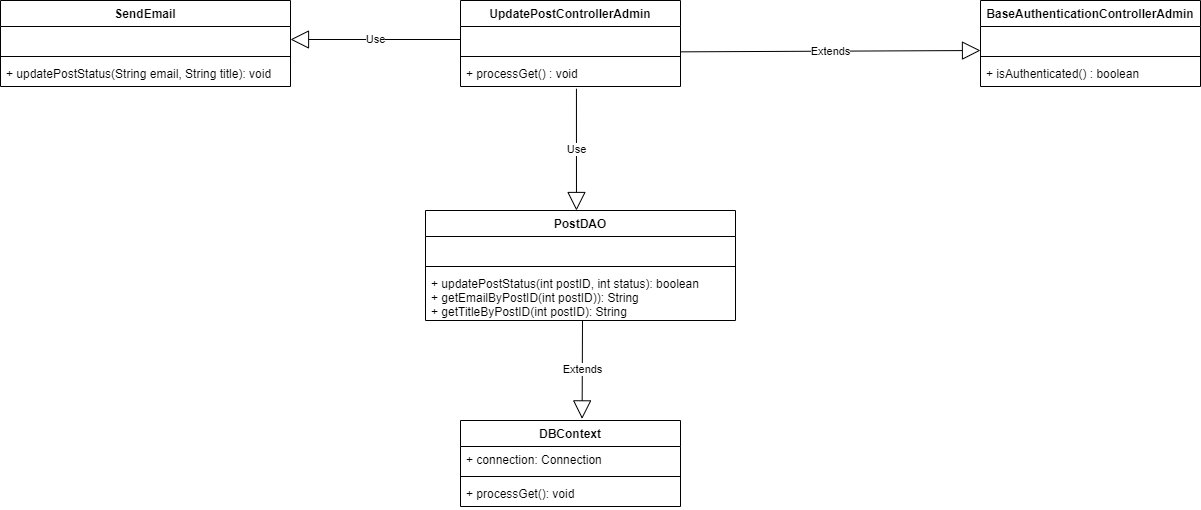
select pr.email from Post as p, Motel as m, Account as a, Profile as pr where p.motelID = m.motelID and m.AccountID = a.AccountID and a.AccountID = pr.AccountID and p.postID = ?

select postTitle from Post where postID = ?

delete PostImage where postID = ?

## 90. Send Email Update Post Status By Admin

### a. Class Diagram



### b. Class Specifications

#### ManageAccountAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will call function getAllInforOfAllAccount to take all the information of all users and transfer it to ManageAdmin.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### AccountDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllInforOfAllAccount | This function will retrieve all information of a user including account profile password .... |
| 02 | updateRoleOfAccount | This function to update role of edited account to database |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

### 

**SendEmail**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateRole | Send email to corresponding to notify updating |

### 

### c. Sequence Diagram(s)

### 

### d. Database queries

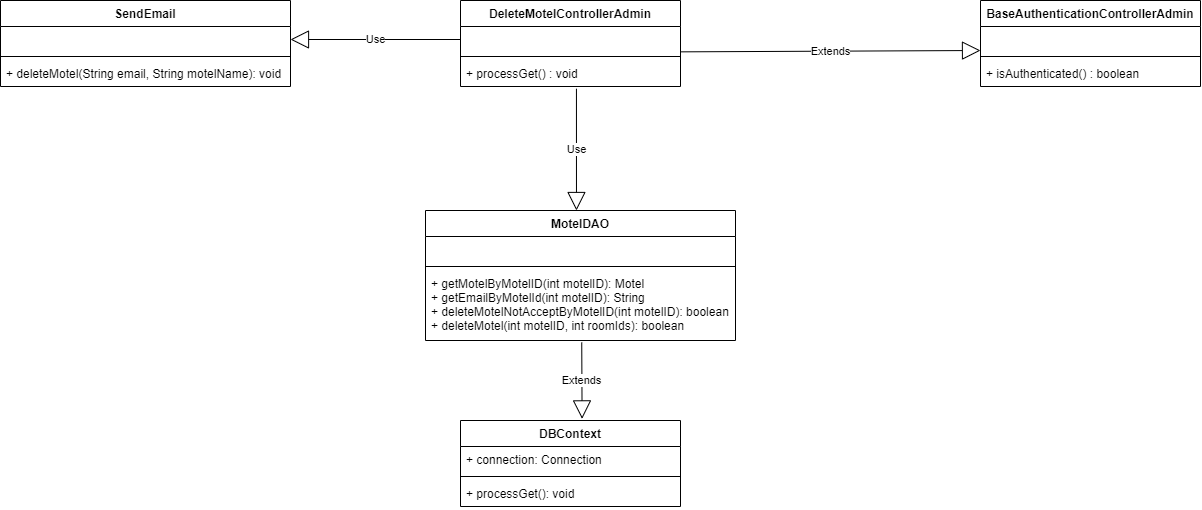
update Post set postStatus = ? where postID = ?

select pr.email from Post as p, Motel as m, Account as a, Profile as pr where p.motelID = m.motelID and m.AccountID = a.AccountID and a.AccountID = pr.AccountID and p.postID = ?

select postTitle from Post where postID = ?

## 91. Send Email Delete Motel By Admin

### a. Class Diagram



### b. Class Specifications

#### ManageAccountAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will call function getAllInforOfAllAccount to take all the information of all users and transfer it to ManageAdmin.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### AccountDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllInforOfAllAccount | This function will retrieve all information of a user including account profile password .... |
| 02 | updateRoleOfAccount | This function to update role of edited account to database |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

### 

**SendEmail**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateRole | Send email to corresponding to notify updating |

### 

### c. Sequence Diagram(s)

### 

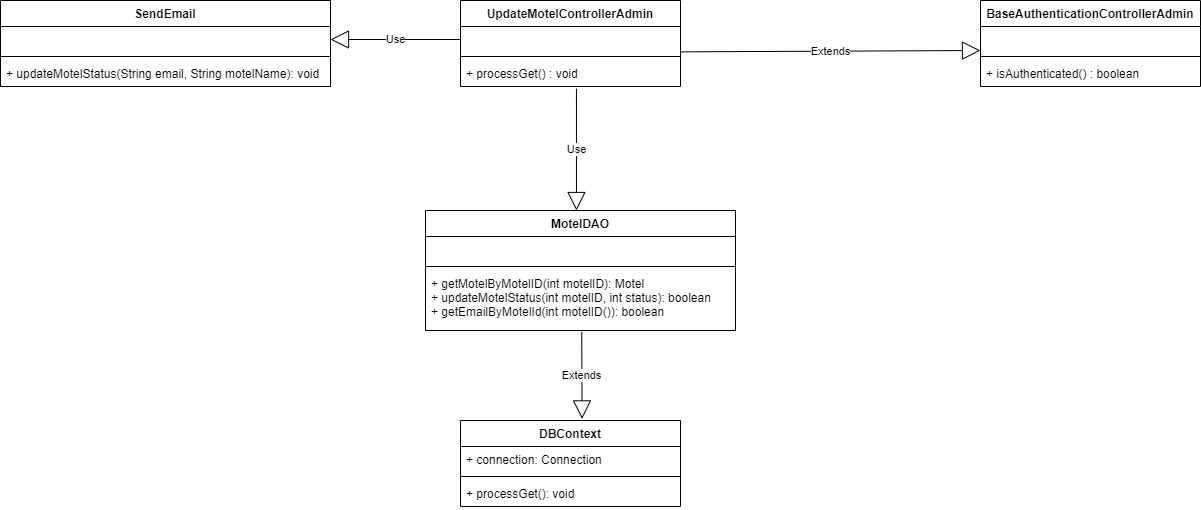
### d. Database queries

select p.email from Motel as m, Account as a, Profile as p where m.AccountID = a.AccountID and a.AccountID = p.AccountID and m.motelID = ?

delete Motel where motelID = ?

## 92. Send Email Update Motel Status By Admin

### a. Class Diagram



### b. Class Specifications

#### ManageAccountAdminControllerClass

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Input is request and response. After receiving the request from the user, it will call function getAllInforOfAllAccount to take all the information of all users and transfer it to ManageAdmin.jsp |
| 02 | doPost | input is request and response. do nothing |

#### 

#### AccountDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllInforOfAllAccount | This function will retrieve all information of a user including account profile password .... |
| 02 | updateRoleOfAccount | This function to update role of edited account to database |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

**BaseAuthencationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

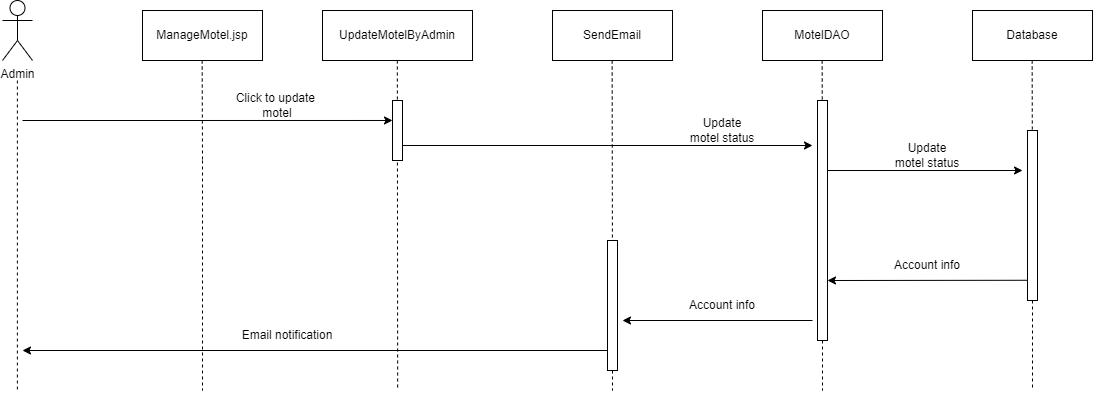
### 

**SendEmail**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateRole | Send email to corresponding to notify updating |

### 

### c. Sequence Diagram(s)



### 

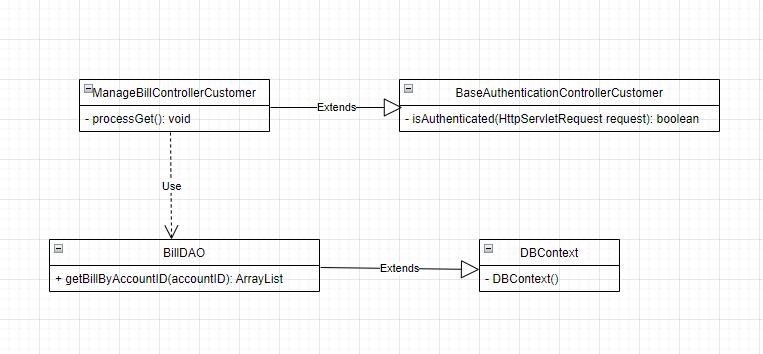
### d. Database queries

update Motel set motelStatus = ? where motelID = ?

select p.email from Motel as m, Account as a, Profile as p where m.AccountID = a.AccountID and a.AccountID = p.AccountID and m.motelID = ?

## 88. Manager Bill By Customer

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageBillControllerCustomer class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all bill of the account on the session and go to manageBillCus.jsp |

***BillDAO class***

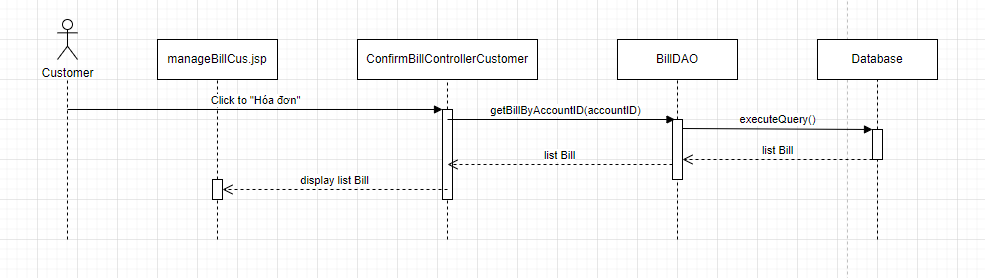
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getBillByAccountID | Get all bill of account . The parameter passed is the id of account |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)



### d. Database queries

select a.billID, a.confirmBillByCustomer, a.confirmBillByHost,a.dateCreateBill, a.dateOfPayment, a.totalBill, c.roomName, e.motelName

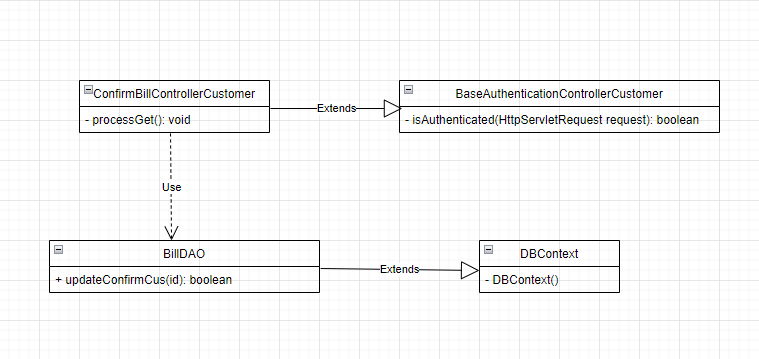
from Bill as a join Booking as b on a.BookingID = b.BookingID join Room as c on b.RoomID = c.roomID

join AccountOfBooking as d on b.BookingID = d.BookingID join Motel as e on c.motelID = e.motelID

where d.AccountID = ? order by a.dateCreateBill desc

## 90. Manager Bill By Customer/Confirm payment

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ManageBillControllerCustomer class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all bill of the account on the session and go to manageBillCus.jsp |

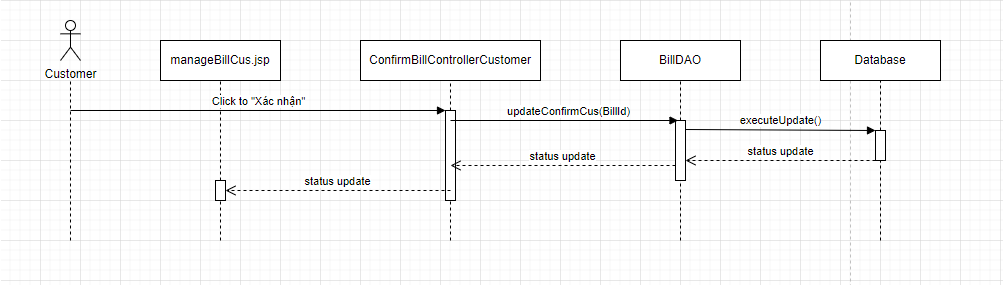
***BillDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateConfirmCus | Update status of bill is confirm payment . The parameter passed is the id of bill |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)

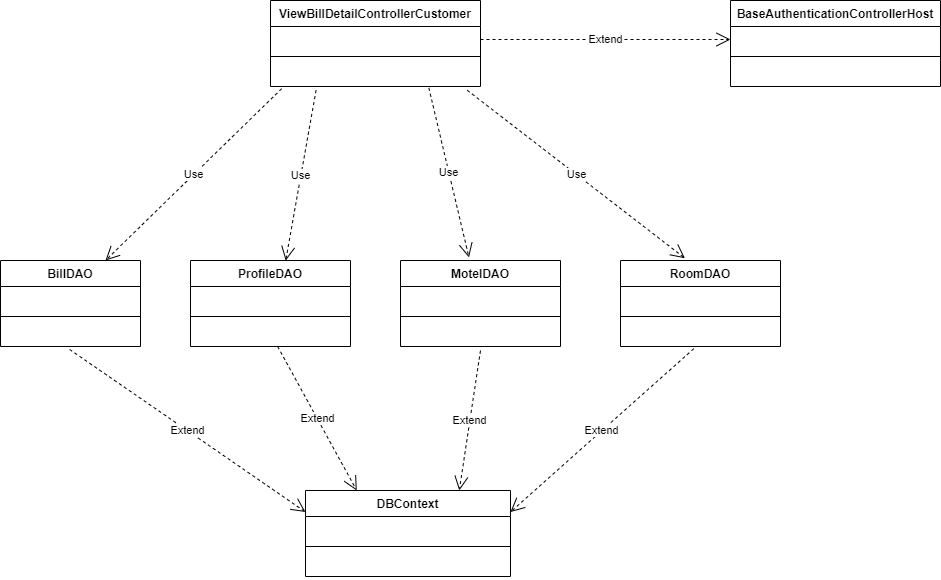


### d. Database queries

* update Bill set confirmBillByCustomer = 1, dateOfPayment = GETDATE() where billID = ?

## 91. Manager Bill By Customer/ View detail

### a. Class Diagram



### b. Class Specifications

***BaseAuthenticationControllerHost class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

***ViewBillDetailControllerCustomer class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | Get all information of a current bill to show to screen |

***BillDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getBillDetailsByBillId(billId) | Get all the bill detail of the bill user click view |

***ProfileDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getHostNameByMotelId(motelID) | Get the owner of the motel |

***MotelDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMotelIDByRoomID(roomID) | Get motel base on room id |

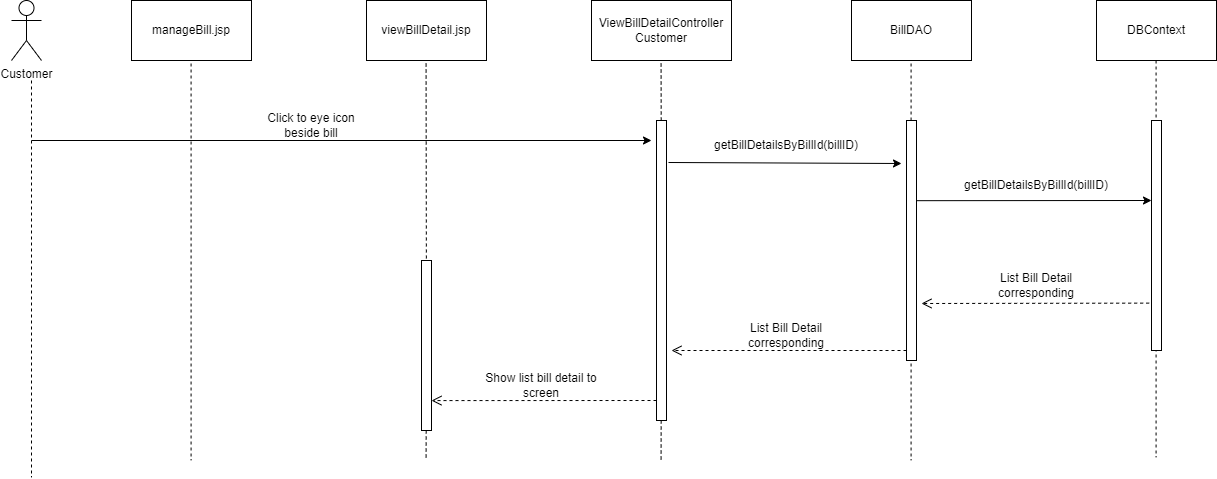
***RoomDAO class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getRoomIdByBookingId(bookingID) | Get room base on booking id |

***DBContext class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### c. Sequence Diagram(s)



### d. Database queries

* select bdt.billDetailID, bdt.billID, s.servicesID ,s.servicesName, s.price, bdt.quantity, bdt.totalBillDetail,bi.confirmBillByCustomer,bi.confirmBillByHost, bdt.isDeleteBillDetail from Bill as bi, BillDetail as bdt, ServicesOfBooking as sob, [Services] as s where bi.billID = ? and bi.billID = bdt.billID and bdt.servicesOfRoomID = sob.servicesOfRoomID and sob.servicesID = s.servicesID
* select c.fullName from Motel as a, Account as b, Profile as c where a.AccountID = b.AccountID and b.AccountID = c.AccountID and a.motelID = ?
* select RoomID from Booking where BookingID = ?
* select motelID from Room where roomID = ?

## 92. Manager Request By Host

### a. Class Diagram

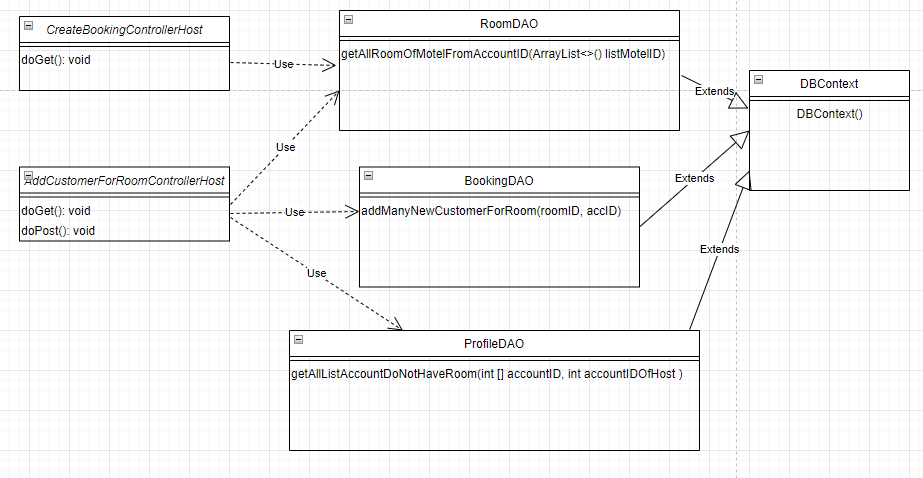
### b. Class Specifications

### c. Sequence Diagram(s)

### d. Database queries

## 94. Manager Booking By Host/ Add

### a. Class Diagram



### b. Class Specifications

#### CreateBookingControllerHost

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Display all information related to the room and inn of the username as the host and submit the information to createBooking.jsp |

#### 

#### AddCustomerForRoomControllerHost

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet | Display all the information of the account in the system (except for the admin, the owner, and the person already in the room) |
| 02 | doPost | Take action add new account to room |

#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

#### RoomDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRoomOfMotelFromAccountID | Get all the rooms of the account logged in as the innkeeper |

#### BookingDAO

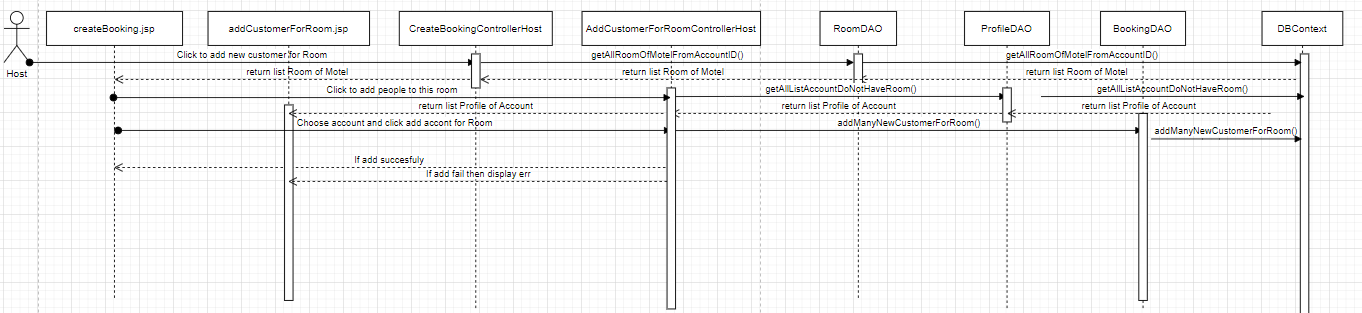
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | addManyNewCustomerForRoom | Add new account for room |

#### ProfileDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllListAccountDoNotHaveRoom | Get the list of information of accounts that are not in the room |

### 

### c. Sequence Diagram(s)



### d. Database queries

* getAllRoomOfMotelFromAccountID:

select Boo.BookingID, Mot.motelID, Mot.motelName, Roo.roomID, Roo.roomName, Ac.AccountID, Pro.profileID, Pro.fullName, Roo.price, Roo.noteRoom, AcO.startAt, Aco.endAt from Profile as Pro full join Account as Ac on Pro.AccountID = Ac.AccountID

full join AccountOfBooking as AcO on Ac.AccountID = AcO.AccountID

full join Booking as Boo on AcO.BookingID = Boo.BookingID

full join Room as Roo on Boo.RoomID = Roo.roomID

full join Motel as Mot on Roo.motelID = Mot.motelID

where Mot.motelID = ?

* addManyCustomerForRoom():

insert into AccountOfBooking(AccountID,BookingID,startAt,endAt) values (?,?,getDate(),getDate()+30)

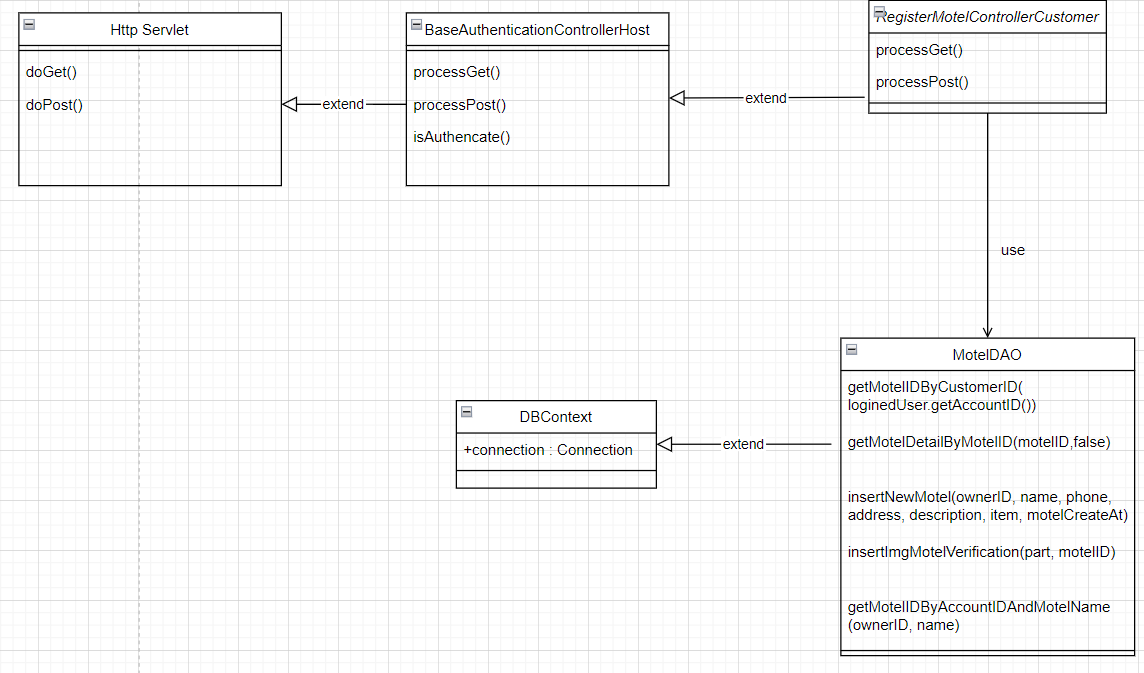
* getAllListAccountDoNotHaveRoom():

select a.AccountID, a.userName, p.fullName, p.phone, p.[address], p.email, p.DOB, p.Gender from Account a inner join [Profile] p on a.AccountID = p.AccountID

where a.AccountID != 1 and a.AccountID != ? and a.AccountID not in (?,?,?)

## 100. Register New Motel by Customer

### a. Class Diagram



### b. Class Specifications

**BaseAuthenticationControllerAdmin**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | isAuthenticated | Get the role of the account in session and check if the role matches or not |

**ManageMotelControllerHost**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processGet | navigate to register-motel-customer.jsp page |
| 02 | processPost | add new motel into database |

#### 

#### MotelDAO Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getMotelIDByAccountIDAndMotelName(ownerID, name) | get motelID of motel base on name and account id |
| 02 | insertNewMotel(ownerID, name, phone, address, description, item, motelCreateAt) | add new motel into database |
| 03 | insertImgMotelVerification(part, motelID) | add image verification into database |
| 04 | getMotelIDByCustomerID | get motel id of by customer id |
| 05 | getMotelIDByAccountIDAndMotelName(ownerID, name) | get motel id by motelName and account id |

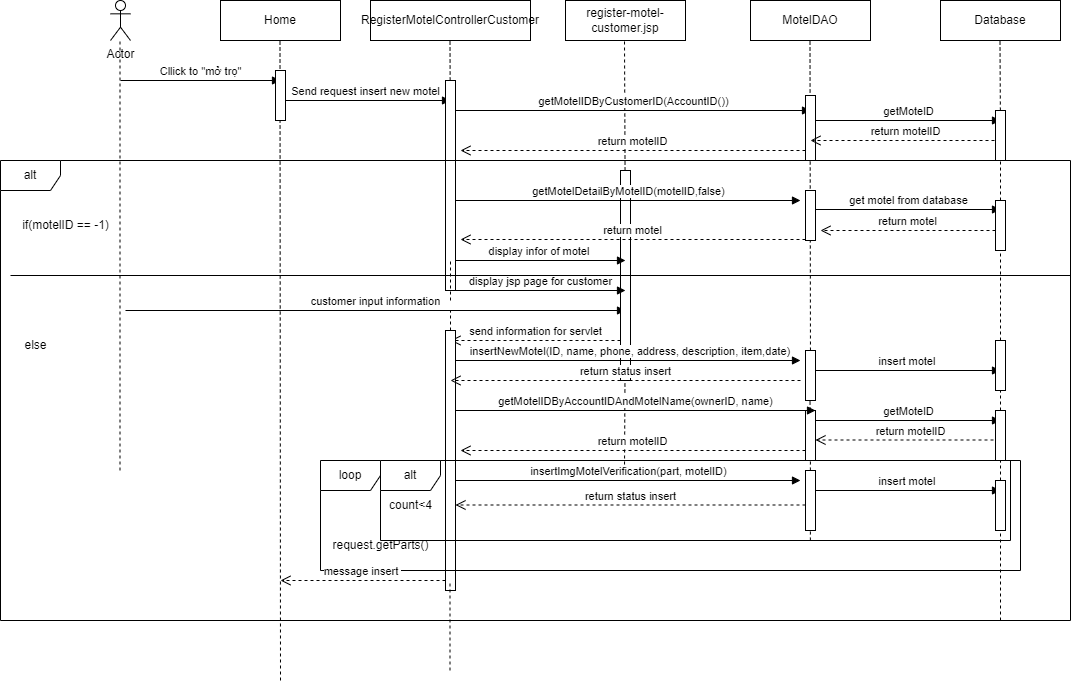
#### 

#### DBContext Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | DBContext | Make a connection to the database |

### 

### c. Sequence Diagram(s)



### d. Database queries

* select m.motelID from Motel as m where m.AccountID = ? and motelStatus = 0
* select m.motelID, m.accountID,(select p.fullName from [Profile] as p where p.AccountID = m.AccountID), m.motelName, (select count(\*) from Room as r where r.motelID = m.motelID), m.motelAddress, m.motelPhone, m.motelStatus, (select MAX(r.price) from Room as r where r.motelID = m.motelID), (select MIN(r.price) from Room as r where r.motelID = m.motelID), m.motelDescription,m.motelImage, img.imageLink from Motel as m left join MotelImage as img on img.motelID = m.motelID where m.motelID = ?
* insert into Motel(AccountID,motelName,motelAddress,motelPhone,motelStatus,motelDescription,motelCreateAt,motelImage,isDeleteMotel) values(?,?,?,?,0,?, motelCreateAt,?,0)
* insert into MotelImage(motelID,imageLink) values(?,?)
* select m.motelID from Motel as m where m.AccountID = ? and m.motelName = ?

## 3. Manager Request By Host

### a. Class Diagram

### b. Class Specifications

### c. Sequence Diagram(s)

### d. Database queries

## 