

**SAKURA - HOSTEL MANAGEMENT SYSTEM**

**Software Design Document**

– Ho Chi Minh City, June 2022 –

**Table of Contents**

[I. Overview](#_heading=h.gjdgxs) **13**

[1. Code Packages/Namespaces](#_heading=h.30j0zll) 13

[2. Database Schema](#_heading=h.j7nd1uf4ar90) 15

[II. Code Designs](#_heading=h.3znysh7) **17**

[1. Register an account, register with facebook, gmail](#_heading=h.2s8eyo1) 17

[a. Class Diagram](#_heading=h.49qz82e1ert2) 17

[b. Class Specifications](#_heading=h.3aa2z3j6ce9u) 17

[AccessController](#_heading=h.7twapy3h57v7) 17

[TenantDAO](#_heading=h.525fxijam9zk) 17

[LandlordDAO](#_heading=h.thpl1otzlnt) 18

[AccountDAO](#_heading=h.7ppbxpn4w7r9) 18

[DBUtils](#_heading=h.ypmc45yhltu) 18

[PasswordHash](#_heading=h.329khvkk5zb7) 18

[StringUtils](#_heading=h.pnnvh3tmeq03) 18

[EmailService](#_heading=h.t7ezu8xajlz4) 18

[c. Sequence Diagram](#_heading=h.c0x4f44vtbhm) 18

[d. Database queries](#_heading=h.4yg130jndsxu) 19

[2. Register with gmail/facebook](#_heading=h.ho453af2xql8) 19

[a. Class Diagram](#_heading=h.qr0rufote3x) 19

[b. Class Specifications](#_heading=h.rujl47qjf8p) 19

[AccessController](#_heading=h.aejpupagu1zy) 19

[TenantDAO](#_heading=h.nfi25s3lbtci) 19

[LandlordDAO](#_heading=h.h8gr2sy1b9z0) 20

[AccountDAO](#_heading=h.2e3801y5wv5m) 20

[DBUtils](#_heading=h.5cn0nsgzgvsy) 20

[PasswordHash](#_heading=h.lzcjrvu7mkrb) 20

[StringUtils](#_heading=h.bx7wzvhnqbfd) 20

[EmailService](#_heading=h.rz5dfe4qoe9s) 20

[c. Sequence Diagram](#_heading=h.xw3tf62cyo7s) 20

[d. Database queries](#_heading=h.9kk8zco3a1jo) 21

[3. Login by username, password](#_heading=h.1nqryuqyyaqg) 21

[a. Class Diagram](#_heading=h.dyds1sy91cqh) 21

[b. Class Specifications](#_heading=h.fu9s7beo0cob) 21

[AccessController](#_heading=h.fzu1lbidkkt6) 21

[AccountDAO](#_heading=h.ktg4ueyazokj) 21

[PasswordHash](#_heading=h.3hy6usmom0uc) 22

[DBUtils](#_heading=h.bjxb75t5gpr2) 22

[c. Sequence Diagram](#_heading=h.el00z3lq21mu) 22

[d. Database queries](#_heading=h.pee085l7tvg5) 22

[4. Login by gmail/facebook](#_heading=h.9ps5nkwf62fo) 23

[a. Class Diagram](#_heading=h.yiouk11dmvzn) 23

[b. Class Specifications](#_heading=h.9d7l90qg66qn) 23

[AccessController](#_heading=h.dp53lsswsnfk) 23

[AccountDAO](#_heading=h.5ammwiwz1wqs) 23

[DBUtils](#_heading=h.8c4iz54c7npz) 24

[GoogleUtils](#_heading=h.rm9hgsr8fpgp) 24

[PasswordHash](#_heading=h.7cmp4gwqtxrd) 24

[c. Sequence Diagram](#_heading=h.l2y9sbg8x8rj) 24

[d. Database queries](#_heading=h.ottj56766vm9) 24

[5. Reset password](#_heading=h.v36nr3s0dslb) 25

[a. Class Diagram](#_heading=h.b5karmjbab77) 25

[b. Class Specifications](#_heading=h.xcv365sopguu) 25

[AccessController](#_heading=h.4qvny2gq2jed) 25

[AccountDAO](#_heading=h.g1qk7j6ddix3) 25

[DBUtils](#_heading=h.og5ds8tywt46) 25

[EmailService](#_heading=h.8wdnsvh1glu3) 25

[c. Sequence Diagram](#_heading=h.smlz9ahqgrn7) 26

[d. Database queries](#_heading=h.lpxxvvlsrxwf) 26

[6. Change user profile](#_heading=h.8hjkt27haw54) 27

[a. Class Diagram](#_heading=h.6jtjnpfo1wj4) 27

[b. Class Specifications](#_heading=h.ybu1qisxj00j) 27

[AccessController](#_heading=h.1q00jtfshl49) 27

[TenantDAO](#_heading=h.9rpnifd1lzw) 27

[LandlordDAO](#_heading=h.82qlrrb801aa) 27

[AccountDAO](#_heading=h.wmpq3nu6aaco) 28

[DBUtils](#_heading=h.6rhi3kzatvv7) 28

[c. Sequence Diagram](#_heading=h.u93lnif4gx72) 28

[d. Database queries](#_heading=h.jbbie6qzsbky) 28

[7. Change password](#_heading=h.nonqkvxtr5h1) 29

[a. Class Diagram](#_heading=h.1w1cc6cnscm5) 29

[b. Class Specifications](#_heading=h.s31fumngimo2) 29

[AccessController](#_heading=h.pwnkmzomlsd8) 29

[AccountDAO](#_heading=h.152piig0gcxy) 29

[DBUtils](#_heading=h.a54ki8gsobqu) 29

[c. Sequence Diagram](#_heading=h.5rc4lknvpepj) 30

[d. Database queries](#_heading=h.36h857aj920o) 30

[8. View hostel list, favorite hostel list (tenant feature)](#_heading=h.etquqfvyx8bq) 31

[a. Class Diagram](#_heading=h.wy5yf7okxz6h) 31

[b. Class Specifications](#_heading=h.zf8h1dlzqqlv) 31

[AccessController](#_heading=h.b5o7afxno7du) 31

[HostelDAO](#_heading=h.xq58rrksac9) 31

[ProvinceDAO](#_heading=h.pevzss9tdr77) 31

[DistrictDAO](#_heading=h.pnjw5pudcw2d) 31

[DBUtils](#_heading=h.4fsljcgejl4z) 31

[c. Sequence Diagram](#_heading=h.z20kugt50360) 32

[d. Database queries](#_heading=h.ym2f5kxaub02) 32

[9. View hostel detail, hostel’s reviews/feedback (tenant feature)](#_heading=h.wz0bez34h9l8) 33

[a. Class Diagram](#_heading=h.m32078s2ijtb) 33

[b. Class Specifications](#_heading=h.rus1vhwvog9v) 33

[HostelController](#_heading=h.uwqbyzdo82cd) 33

[HostelDAO](#_heading=h.sslv9vw9jutl) 33

[FeedbackDAO](#_heading=h.9dettwmul7cz) 33

[DBUtils](#_heading=h.ifdd6byfqbhq) 34

[c. Sequence Diagram](#_heading=h.jwd1i5fds4du) 34

[d. Database queries](#_heading=h.i10s64cfqer1) 34

[10. View room list of a hostel (tenant feature)](#_heading=h.maqbjq2q4kis) 35

[a. Class Diagram](#_heading=h.91tfzf8164wn) 35

[b. Class Specifications](#_heading=h.lo31a56759dh) 35

[AccessController](#_heading=h.2mhlrvwv23n4) 35

[HostelDAO](#_heading=h.hj3rdlf5vdmr) 35

[DBUtils](#_heading=h.3oqff096c735) 35

[c. Sequence Diagram](#_heading=h.punvuvd20b1m) 35

[d. Database queries](#_heading=h.hfdzlgnyttk7) 36

[11. Add new feedback, edit feedback of a hostel (tenant feature)](#_heading=h.8yd99hd6ldup) 36

[a. Class Diagram](#_heading=h.dzzhgju08efu) 36

[b. Class Specifications](#_heading=h.w4k7systuev4) 36

[HostelDAO](#_heading=h.kwlc7jysfww5) 36

[FeedbackDAO](#_heading=h.dm5dy05jxwa8) 36

[ProvinceDAO](#_heading=h.eq66eudoa0cg) 37

[FavoriteHostelDAO](#_heading=h.tpa1agrwjbz7) 37

[DBUtils](#_heading=h.y8jzcvxdtl4a) 37

[c. Sequence Diagram](#_heading=h.b3mad1ub2hma) 37

[d. Database queries](#_heading=h.arono518329b) 37

[12. Toggle a favorite hostel (tenant feature)](#_heading=h.u2b8dewe61cx) 38

[a. Class Diagram](#_heading=h.qfyzi9czs6pm) 38

[b. Class Specifications](#_heading=h.hqy8ak8skjmh) 38

[AccessController](#_heading=h.j1d25ldolqj6) 38

[FavoriteHostelDAO](#_heading=h.vnzmxec9fwou) 38

[DBUtils](#_heading=h.j9z3j5smj0cf) 38

[c. Sequence Diagram](#_heading=h.3suc3eyohfr9) 39

[d. Database queries](#_heading=h.y4nbw44vqu0a) 39

[13. Rent a room of a hostel](#_heading=h.fgbyecm9msoh) 39

[a. Class Diagram](#_heading=h.wrv859cx7pu) 39

[b. Class Specifications](#_heading=h.ub3g167u0lc0) 40

[HostelController](#_heading=h.3k1wdg92nur6) 40

[RoomDAO](#_heading=h.mwlbe9glstbj) 40

[NotificationDAO](#_heading=h.6unxxhtpn9h1) 40

[BookingRequestDAO](#_heading=h.4mwhycgmlyht) 40

[DBUtils](#_heading=h.5qbldbmzrhbg) 40

[c. Sequence Diagram](#_heading=h.di9mgwmytu1d) 40

[d. Database queries](#_heading=h.8z289m6lft1y) 40

[14. View monthly invoice list (Tenant)](#_heading=h.clqjzrk0cui3) 41

[a. Class Diagram](#_heading=h.vxp0a3cr7daw) 41

[b. Class Specifications](#_heading=h.9ektjedxbjh8) 41

[InvoiceController](#_heading=h.66xdhwmf09wa) 41

[ContractDAO](#_heading=h.jlsr888qfccz) 41

[InvoiceDAO](#_heading=h.mikc097ejwtm) 41

[DBUtils (Database)](#_heading=h.i5h0i5sc3fp8) 41

[c. Sequence Diagram](#_heading=h.fbqrnu6ww9ak) 42

[d. Database queries](#_heading=h.1xgowwciqai2) 42

[15. View monthly invoice detail (Tenant feature)](#_heading=h.9gybb86b6zzb) 42

[a. Class Diagram](#_heading=h.z35r9wmpdgf) 42

[b. Class Specifications](#_heading=h.ocbzj2qd1t6p) 43

[InvoiceController](#_heading=h.94uyw9rz6kkm) 43

[InvoiceDAO](#_heading=h.kk3gsrgvu8ej) 43

[ServiceDAO](#_heading=h.xnkxc8e4ff0h) 43

[DBUtils (Database)](#_heading=h.id0ekkgi0xiy) 43

[c. Sequence Diagram](#_heading=h.rpmqrvpnimud) 43

[d. Database queries](#_heading=h.piyr0h8u3rxy) 43

[16. View contract list(Tenant)](#_heading=h.q3b0ka5bofzw) 44

[a. Class Diagram](#_heading=h.bfs2s1mvfieb) 44

[b. Class Specifications](#_heading=h.jxi3o4ya4t70) 44

[ContractController](#_heading=h.oxl94udi9f49) 44

[ContractDAO](#_heading=h.jm7ex7seyam3) 44

[DBUtils](#_heading=h.z18ulbproy09) 44

[c. Sequence Diagram](#_heading=h.1rjx60lgth87) 45

[d. Database queries](#_heading=h.uh9vkofxspj8) 45

[17. Add hostel (Landlord feature)](#_heading=h.h2fl9w5fs9kq) 45

[a. Class Diagram](#_heading=h.b9wr5fj5ud3c) 45

[b. Class Specifications](#_heading=h.ql4g58ijz8q) 46

[HostelController](#_heading=h.thlga8a6apvm) 46

[HostelDAO](#_heading=h.cewxlon5e5ka) 46

[DBUtils](#_heading=h.f03vwrsdhht) 46

[c. Sequence Diagram](#_heading=h.o1z7blm1mfag) 46

[d. Database queries](#_heading=h.wyj9rajywbcx) 46

[18. Add room type (Landlord feature)](#_heading=h.97lbk7vu9edr) 47

[a. Class Diagram](#_heading=h.cbpn1qttcjx) 47

[b. Class Specifications](#_heading=h.m2la3gl70bls) 47

[RoomController](#_heading=h.menrw1aqjkrk) 47

[RoomDAO](#_heading=h.rssa9okd16xe) 47

[DBUtils](#_heading=h.ly9b18clois1) 47

[c. Sequence Diagram](#_heading=h.mpcb9bwapn4l) 48

[d. Database queries](#_heading=h.li8q6damf68k) 48

[19. Add room (Landlord feature)](#_heading=h.175myetajqup) 49

[a. Class Diagram](#_heading=h.lx2b1obxez0o) 49

[b. Class Specifications](#_heading=h.3myuq21w9f8t) 49

[RoomController](#_heading=h.egs3hjbmj8k9) 49

[RoomDAO](#_heading=h.bijw6ouk2vhv) 49

[DBUtils](#_heading=h.uohklm6spf6o) 49

[c. Sequence Diagram](#_heading=h.bg1ondloc7kn) 49

[d. Database queries](#_heading=h.bnajarhr0hpq) 50

[20. Add contract (Landlord feature)](#_heading=h.av2vks3x2szd) 50

[a. Class Diagram](#_heading=h.agkj5dqvs23) 50

[b. Class Specifications](#_heading=h.1rpn79kxa1de) 51

[ContractController](#_heading=h.1digjvg2jiat) 51

[ContractDAO](#_heading=h.rb21d2ws0kic) 51

[DBUtils](#_heading=h.jhdta49na2em) 51

[c. Sequence Diagram](#_heading=h.23riyt92izcp) 51

[d. Database queries](#_heading=h.i3pwh1vlf9jw) 51

[21. Add invoice (Landlord feature)](#_heading=h.hfsinxnrwdh9) 52

[a. Class Diagram](#_heading=h.m0qm4snfhrbh) 52

[b. Class Specifications](#_heading=h.9i3ge9ez0mrg) 52

[InvoiceController](#_heading=h.v5mvpz4u34lb) 52

[InvoiceDAO](#_heading=h.djccqjd8y4b0) 52

[DBUtils](#_heading=h.n3hef4f95bk4) 52

[c. Sequence Diagram](#_heading=h.qqtp49t2pz2x) 53

[d. Database queries](#_heading=h.rtvq1zo4uqw) 53

[22. View room type detail (Landlord feature)](#_heading=h.bkampcp061qg) 54

[a. Class Diagram](#_heading=h.pgam08q9pe1b) 54

[b. Class Specifications](#_heading=h.7mrtutpjwssl) 54

[RoomController](#_heading=h.2n0i10kkj9oq) 54

[RoomDAO](#_heading=h.ypxsbp919tju) 54

[DBUtils](#_heading=h.b9633mrc3xno) 54

[c. Sequence Diagram](#_heading=h.af1ukinp79uv) 54

[d. Database queries](#_heading=h.viqjop6s6u20) 55

[23. View room list (Landlord feature)](#_heading=h.thfawefv01qa) 55

[a. Class Diagram](#_heading=h.83zvpn5dac9m) 55

[b. Class Specifications](#_heading=h.brrw67mlwnj3) 56

[RoomController](#_heading=h.7jruvrcs3jt3) 56

[RoomDAO](#_heading=h.cj95nrtayy3g) 56

[DBUtils](#_heading=h.hkftb48e737) 56

[c. Sequence Diagram](#_heading=h.5f3o4y3cnxz4) 56

[d. Database queries](#_heading=h.lkivmqmpkce8) 56

[24. View room detail (Landlord feature)](#_heading=h.mq6kd04bi8nr) 57

[a. Class Diagram](#_heading=h.ksfuxz3w2u72) 57

[b. Class Specifications](#_heading=h.svte9tta15gg) 57

[RoomController](#_heading=h.r7r0w381jv67) 57

[RoomDAO](#_heading=h.3hm4vw1dkiqy) 57

[DBUtils](#_heading=h.hqwdrrjek9j2) 57

[c. Sequence Diagram](#_heading=h.8s7p16bd5iui) 57

[d. Database queries](#_heading=h.7yy6h4o6xpdt) 58

[25. View contract list (Landlord feature)](#_heading=h.8ajivghe3hhc) 58

[a. Class Diagram](#_heading=h.ofjgmp9082ov) 58

[b. Class Specifications](#_heading=h.pribv53hgfq2) 59

[ContractController](#_heading=h.lb3izlz4stfp) 59

[ContractDAO](#_heading=h.xsiudrv5nul4) 59

[DBUtils](#_heading=h.ujljbrmrv6jm) 59

[c. Sequence Diagram](#_heading=h.d666fx2u3yum) 59

[d. Database queries](#_heading=h.7miny4d4ia5a) 59

[26. View contract detail (Landlord feature)](#_heading=h.a4jypins2cil) 60

[a. Class Diagram](#_heading=h.m1eckzew0ax1) 60

[b. Class Specifications](#_heading=h.txkq0ovzx0s) 60

[ContractController](#_heading=h.51h7wqxv3ql2) 60

[HostelDAO](#_heading=h.veqf3ecqk1vg) 60

[DBUtils](#_heading=h.854p5n5cnur6) 60

[c. Sequence Diagram](#_heading=h.vzy16nhogem) 60

[d. Database queries](#_heading=h.4f2987smd0wj) 61

[27. View invoice list (Landlord feature)](#_heading=h.314x44jrpwlf) 61

[a. Class Diagram](#_heading=h.d8fnay99e38m) 61

[b. Class Specifications](#_heading=h.s7oaomuqk7h) 62

[HostelContontroller](#_heading=h.qudn7cdfpnor) 62

[HostelDAO](#_heading=h.d3obvr9enioj) 62

[DBUtils](#_heading=h.9j4kn5gljsrk) 62

[c. Sequence Diagram](#_heading=h.b5rl7l2piwhu) 62

[d. Database queries](#_heading=h.xhwz09arvk3g) 62

[28. View invoice detail (Landlord feature)](#_heading=h.jt72r8dg0a8g) 63

[a. Class Diagram](#_heading=h.mzbkl7bjdhgk) 63

[b. Class Specifications](#_heading=h.7phz4p4h36e3) 63

[HostelContontroller](#_heading=h.vih8pfwrthn) 63

[HostelDAO](#_heading=h.33y75mjaa376) 63

[DBUtils](#_heading=h.r5ltvqlw3x58) 63

[c. Sequence Diagram](#_heading=h.gxs8o2bj6yvh) 63

[d. Database queries](#_heading=h.tfmgnpn3rq3j) 64

[29. Edit a payment invoice (Landlord feature)](#_heading=h.2ah82qh2zy78) 64

[a. Class Diagram](#_heading=h.b87xsvquogrz) 64

[b. Class Specifications](#_heading=h.habrxqdz1ouu) 65

[InvoiceController](#_heading=h.gpz1ylb6lh93) 65

[InvoiceDAO](#_heading=h.5o38thyembvd) 65

[ServiceDAO](#_heading=h.t81xkzwaw4wy) 65

[DBUtils (Database)](#_heading=h.xxl7r02cq0pt) 65

[30. Statistic (Landlord feature)](#_heading=h.ckl3dkxv7pwx) 66

[a. Class Diagram](#_heading=h.kr6z5lnait8l) 66

[b. Class Specifications](#_heading=h.c5w12infbp59) 66

[HostelContontroller](#_heading=h.24lrougfxz56) 66

[HostelDAO](#_heading=h.yj87yu33r4x8) 66

[InvoiceDAO](#_heading=h.gpr3n87ta6u1) 66

[DBUtils](#_heading=h.x5b0xfaowq4s) 66

[c. Sequence Diagram](#_heading=h.y8uv7pttf2by) 67

[d. Database queries](#_heading=h.jvi7kglw8x4x) 67

[31. Activate/deactivate tenant account(admin feature)](#_heading=h.k32srrh457ql) 67

[a. Class Diagram](#_heading=h.o7xb4ojswr7u) 67

[b. Class Specifications](#_heading=h.lthlr4opvuic) 68

[TenantController](#_heading=h.mjdt1sszsaxf) 68

[AccountDAO](#_heading=h.n2duniqnrlfq) 68

[NotificationDAO](#_heading=h.6ry7o8rv3q9e) 68

[DBUtils](#_heading=h.1ptzyke8l7g5) 68

[c. Sequence Diagram](#_heading=h.hrxv55ktru5a) 68

[d. Database queries](#_heading=h.864kxd2mvgvh) 68

[32. View landlord list with paging(admin feature)](#_heading=h.n40ccp29hlea) 69

[a. Class Diagram](#_heading=h.r7deb3yn9djg) 69

[b. Class Specifications](#_heading=h.kn6zs1ututos) 69

[LandlordController](#_heading=h.4hf5lxswk0s5) 69

[LandlordDAO](#_heading=h.kxm8jfmc2yo1) 69

[AccountDAO](#_heading=h.o1ihxi5nj7ic) 69

[DBUtils](#_heading=h.mjz30akqu7lm) 69

[c. Sequence Diagram](#_heading=h.m7u2cn877yo4) 70

[d. Database queries](#_heading=h.uqvqtydsx0y2) 70

[33. View landlord detail (admin feature)](#_heading=h.9shernt7mp2k) 70

[a. Class Diagram](#_heading=h.s861s2rr0csy) 70

[b. Class Specifications](#_heading=h.b7t98pec0am) 71

[LandlordController](#_heading=h.eqlszmu2m0au) 71

[LandlordDAO](#_heading=h.nbzqh3vfnukz) 71

[AccountDAO](#_heading=h.evtat8ws9ili) 71

[DBUtils](#_heading=h.78bcdqj4c28i) 71

[c. Sequence Diagram](#_heading=h.8loke81m6g9d) 71

[d. Database queries](#_heading=h.fegsqv4qbyo7) 71

[34. Activate/deactivate landlord account (admin feature)](#_heading=h.vwakgkbordrk) 72

[a. Class Diagram](#_heading=h.mfyw996guzme) 72

[b. Class Specifications](#_heading=h.8rs9d9syex1s) 72

[LandlordController](#_heading=h.gpnm16ivr1nz) 72

[AccountDAO](#_heading=h.67tywxrieeel) 72

[NotificationDAO](#_heading=h.xkd2o689gdtv) 72

[DBUtils](#_heading=h.2ybn5uv7nwe9) 72

[c. Sequence Diagram](#_heading=h.mdlhjloipl8a) 73

[35. View notifications](#_heading=h.hb9m26s37hk) 73

[a. Class Diagram](#_heading=h.ha7g60t4pez0) 73

[b. Class Specifications](#_heading=h.e5gh7gdxrwau) 74

[NotiController](#_heading=h.pxf7u7ug6tuc) 74

[NotiDAO](#_heading=h.dt7fajew2c93) 74

[DBUtils](#_heading=h.t417ny2vs7y1) 74

[c. Sequence Diagram](#_heading=h.ofy9ob1afk6o) 74

[d. Database queries](#_heading=h.z2fxqqxgiu1e) 74

[**III. Database Tables**](#_heading=h.d7rcp99379oy) **75**

[1. Table Account](#_heading=h.3rdcrjn) 75

[2. Table Landlord](#_heading=h.26in1rg) 75

[3. Table Tenant](#_heading=h.3toq4lmojx5u) 75

[4. Table Hostel](#_heading=h.k5nqxjmv9ov5) 75

[5. Table Room](#_heading=h.q9vv1b1bruxk) 76

[6. Table RoomResident](#_heading=h.fotauwkmhw3) 76

[7. Table Feedback](#_heading=h.k0gzqyntko4d) 76

[8. Table Notification](#_heading=h.wtj568ivqwjn) 77

[9. Table Service](#_heading=h.q8qsud787hd0) 77

[10. Table ServiceDetail](#_heading=h.a1ao309z8qel) 77

[11. Table FavoriteHostel](#_heading=h.bip7a9tcwxmw) 77

[12. Table Contract](#_heading=h.3l36dhw43chy) 78

[13. Table Invoice](#_heading=h.kbzb6326z49m) 78

[14. District](#_heading=h.e8544y6o0pa7) 78

[Save district information such as district name and districtId](#_heading=h.isn53nwtuu49) 78

[15. Province](#_heading=h.4f1q9oh2jrar) 78

[16. GoogleAccount](#_heading=h.5femx31so02s) 79

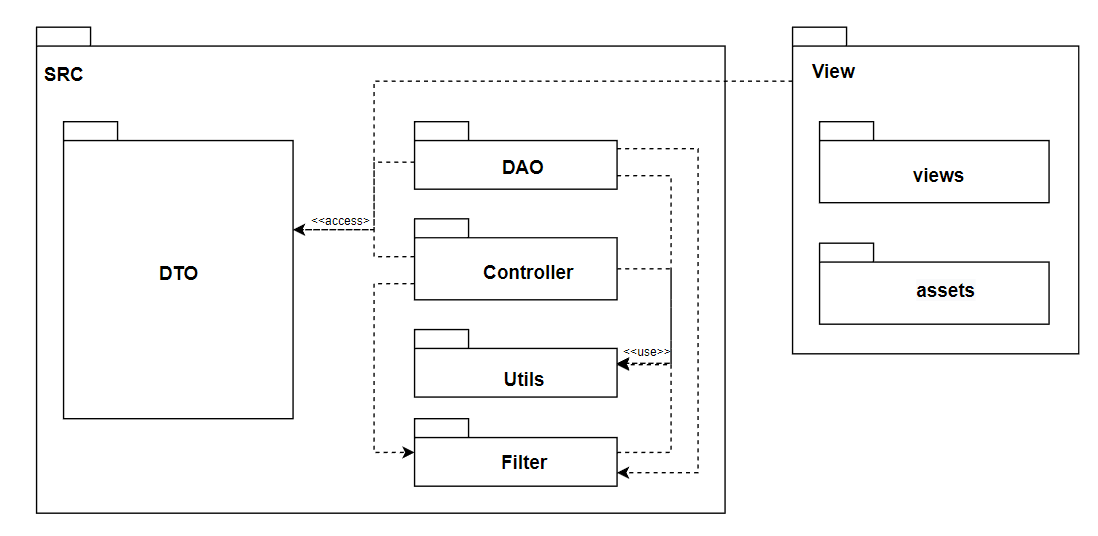
[17. ImageHostel](#_heading=h.qz1es4fs0ry5) 79

[18. RoomType](#_heading=h.5w87rvk3iby8) 79

[19. ImageRoomType](#_heading=h.7g0euwkdgpn0) 79

# I. Overview

## 1. Code Packages/Namespaces



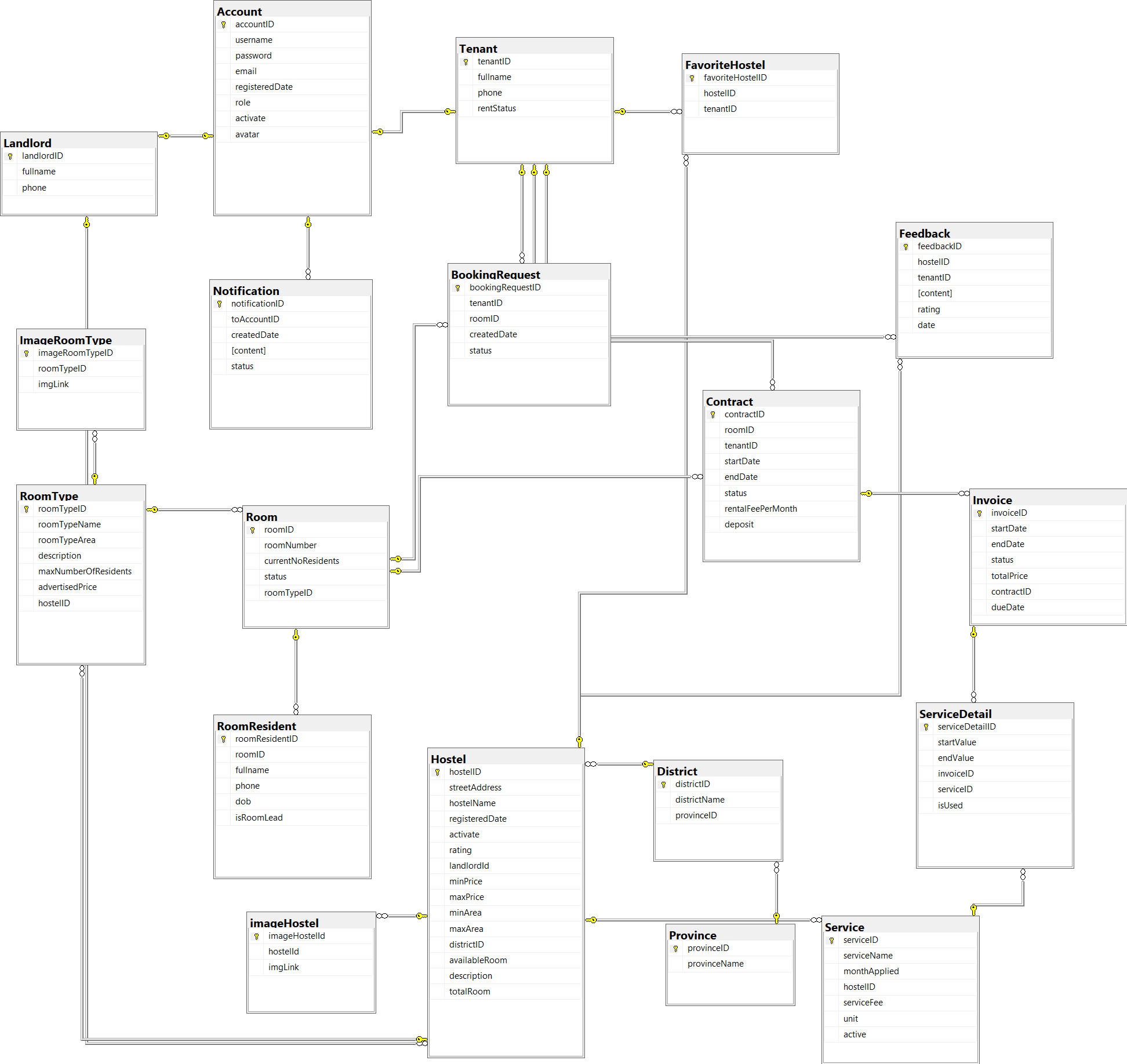
***Package descriptions & package class naming conventions***

| **No** | **Package** | **Description** |
| --- | --- | --- |
| *1* | hostelmanagement.DAO | Each class contains methods used to retrieve data from the database, insert new data or update data in the database by writing SQL queries.  Class naming convention:   * It should start with the uppercase letter. In mixed cases with the first letter of each internal word capitalized * It should be a noun * Use whole words * Ending with “DAO” suffix |
| 2 | hostelmanagement.DTO | Each class corresponds to a table in the database, presenting an entity in the system.  Class naming convention:   * It should start with the uppercase letter. In mixed cases with the first letter of each internal word capitalized * It should be a noun * Use whole words |
| 3 | hostelmanagement.utils | Works as Helper class, contains classes that defines a set of methods that perform common, often reused functions with methods used as a tool to support the process of programming or handling business logic in the system such as database connection, email processing, encrypt data, etc.  Class naming convention:   * It should start with the uppercase letter. In mixed cases with the first letter of each internal word capitalized * It should be nouns with semantic * Recommend naming based on syntax: noun + "Utils" suffix. * Use whole words * e.g: DBUtils.java, EmailService.java, StringUtils.java |
| 4. | hostelmanagement.controller | Responsible for managing the application logic. The controller gets input from the users through the view package, uses the model layer classes DTO, Utils and database access classes to process data, and then returns the results back to the View.  Class naming convention:   * It should start with the uppercase letter. In mixed cases with the first letter of each internal word capitalized * It should be a noun * Use whole words * Ending with “Controller” suffix |
| 5 | hostelmanagement.filter | It is used to intercept the client request and do some pre-processing or post-processing, then perform filtering tasks such as conversion, encryption, input validation, etc.  Class naming convention:   * It should start with the uppercase letter. In mixed cases with the first letter of each internal word capitalized * It should be nouns * Use whole words * Ending with “Filter” suffix |
| 6 | view | Contains the .JSP files showing the user interface of the application  Naming convention:   * Use camelCase for filename. * File extension is ‘.jsp’ |
| 7 | assets | Contains resources associated with showing the user interface of the system, such as images, icons, graphics, fonts, css styling files, javascript files for frontend processing.  Naming convention:   * Use Kebab-case for file name * In lowercase |

## 

## 

## 2. Database Schema



***Table descriptions & package class naming conventions are as below***

| **No** | **Table** | **Description** |
| --- | --- | --- |
| 1 | Account | Contains general information of a user in the system, which 3 roles have in common, such as: username, email, password, phone, etc.   * Primary key: accountID |
| 2 | Tenant | Contains tenant information which will be utilized when renting, managing tenants.   * Primary key: TenanID |
| 3 | Landlord | Contains landlord information   * Primary key: LandlordID |
| 4 | Hostel | Contains an overview of a hostel, such as size, cost, service, image. This information will be used to register for rent, managed by the host and admin. This table does not include detailed information of the rooms.   * Primary key: HostelID * Foreign key: landlordId, wardID |
| 5 | RoomType | A hostel can have many similar rooms in terms of area or price, so it can be collected into a group, we call it ‘type’ here. In a room type, there will be many different rooms.   * Primary key: RoomTypeID * Foreign key: hostelID |
| 6 | Room | Detailed information of a specific hostel room, accompanied by some additional information about the rental status, the number of residents.   * Primary key: RoomID * Foreign key: hostelID |
| 7 | RoomResident | Information about members in a room. Except for the person who has the role of tenant, other members only need to provide their full name and phone number.   * Primary key: RoomResidentID * Foreign key: roomID |
| 8 | Feedback | Contains the reviews, feedbacks of the tenant for the hostel they rent, the guest who look for a room to rent can see these reviews.   * Primary key: FeedbackID * Foreign key: hostelID, tenantID |
| 9 | Announcement | Contains different types of notifications included in the system, such as the announcement from the landlord, notifications from the system, requests from the tenant, etc.   * Primary key: AnnouncementID * Foreign key: tenantID, landlordID |
| 10 | Service | Contains an overview of the services provided by a hostel.   * Primary key: ServiceID * Foreign key: hostelID |
| 11 | ServiceDetail | Detailed information about the services of the hostel, including the service name, fee and unit. This information is updated by the landlord, used to create the monthly invoice for tenants.   * Primary key: ServiceDetailID * Foreign key: invoiceID, serviceID |
| 12 | Contract | Contract information between the landlord and the tenant of the rental room, agreed upon by the two parties and updated by the landlord to the management system. Based on this table, it is possible to know the start and end date of the contract, the rental price, the deposit, etc. The invoices are also calculated based on the contract.   * Primary key: ContractID * Foreign key: roomID, tenantID |
| 13 | Invoice | Contains detailed information about a monthly invoice of a rental room, including rental fee, service fees, and information about payment period.   * Primary key: InvoiceID * Foreign key: roomID |
| 14 | FavoriteHostel | When users click on the favorite button of a hostel, the information of that hostel will be saved in this table, then users can review the list of hostels has been marked.   * Primary key: FavoriteHostelID * Foreign key: hostelID, tenantID |
| 15 | District | List of districts of Ho Chi Minh City and Hanoi.   * Primary key: DistrictID * Foreign key: ProvinceID |
| 16 | Province | Include 2 city HoChiMinh and HaNoi   * Primary key: DistrictID |

# II. Code Designs

## 1. Register an account, register with facebook, gmail

### a. Class Diagram

### b. Class Specifications

#### AccessController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as username, password, email, full name, phone, and roleID. function from AccountDAO, TenantDAO, and LandlordDAO. Create an account with a role as tenant or landlord based on roleID (tenant/landlord object), and save that account to the database. |

#### TenantDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | save() | Input a tenant object from AccessController, execute a query to insert tenant information into the database. Return true if insert successfully. |

#### LandlordDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | save() | Input a landlord object from AccessController, execute a query to insert tenant information into the database. Return true if insert successfully. |

#### AccountDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | save() | Input tenant or landlord information from TenantDAO or LandlordDAO, and execute a query to insert a user into the Account table. If you successfully inserted, return true; otherwise, return false. |
| 02 | findIdByUsername() | Input a username, execute query, and return an account ID corresponding with the username. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection |

#### PasswordHash

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doHashing() | Encrypt user’s password with SHA-1 algorithm |

#### StringUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | randomString() | Create a random verification code to be sent to user's email address. |

#### EmailService

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | sendVerificationEmail() | Input user’s email and verification code, create an email and send to user’s email address. |

### c. Sequence Diagram

### d. Database queries

* *insert into Tenant(tenantID, fullname, phone, rentStatus) values(?, ?, ?, ?)*
* *insert into Landlord(landlordID, fullname, phone) values(?, ?, ?)*
* *insert into Account(username, password, email, registeredDate, role, activate) values(?, ?, ?, ?, ?, ?)*
* *select accountID from Account where username = ?*

## 

## 2. Register with gmail/facebook

### a. Class Diagram

### b. Class Specifications

#### AccessController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as username, password, email, full name, phone, and roleID. function from AccountDAO, TenantDAO, and LandlordDAO. Create an account with a role as tenant or landlord based on roleID (tenant/landlord object), and save that account to the database. |

#### TenantDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | save() | Input a tenant object from AccessController, execute a query to insert tenant information into the database. Return true if insert successfully. |

#### LandlordDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | save() | Input a landlord object from AccessController, execute a query to insert tenant information into the database. Return true if insert successfully. |

#### AccountDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | save() | Input tenant or landlord information from TenantDAO or LandlordDAO, and execute a query to insert a user into the Account table. If you successfully inserted, return true; otherwise, return false. |
| 02 | findIdByUsername() | Input a username, execute query, and return an account ID corresponding with the username. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection |

#### PasswordHash

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doHashing() | Encrypt user’s password with SHA-1 algorithm |

#### StringUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | randomString() | Create a random verification code to be sent to user's email address. |

#### EmailService

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | sendVerificationEmail() | Input user’s email and verification code, create an email and send to user’s email address. |

### c. Sequence Diagram

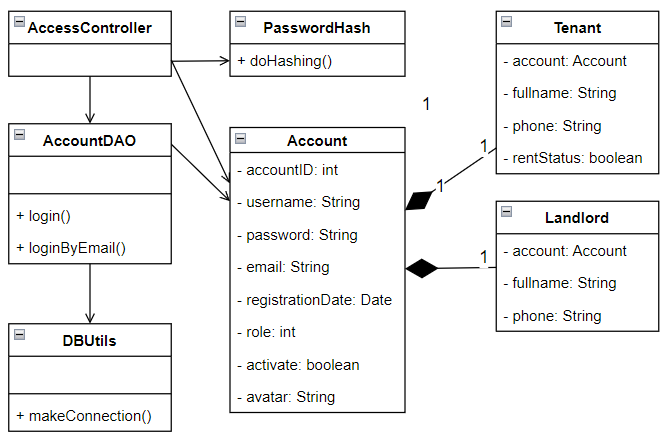
### d. Database queries

* *insert into Tenant(tenantID, fullname, phone, rentStatus) values(?, ?, ?, ?)*
* *insert into Landlord(landlordID, fullname, phone) values(?, ?, ?)*
* *insert into Account(username, password, email, registeredDate, role, activate) values(?, ?, ?, ?, ?, ?)*
* *select accountID from Account where username = ?*

## 

## 3. Login by username, password

### a. Class Diagram



### b. Class Specifications

#### AccessController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get username and password parameters from request, execute query to verify whether username, password are valid or not. Create a session if verify successfully, otherwise display a error message to the user. |

#### AccountDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | login() | Input username and password read from user input parameters, make DB connection, query and compare with username, password saved in database. If username, password exist in database and compare result is equal, return with an associated account. Otherwise return null. |

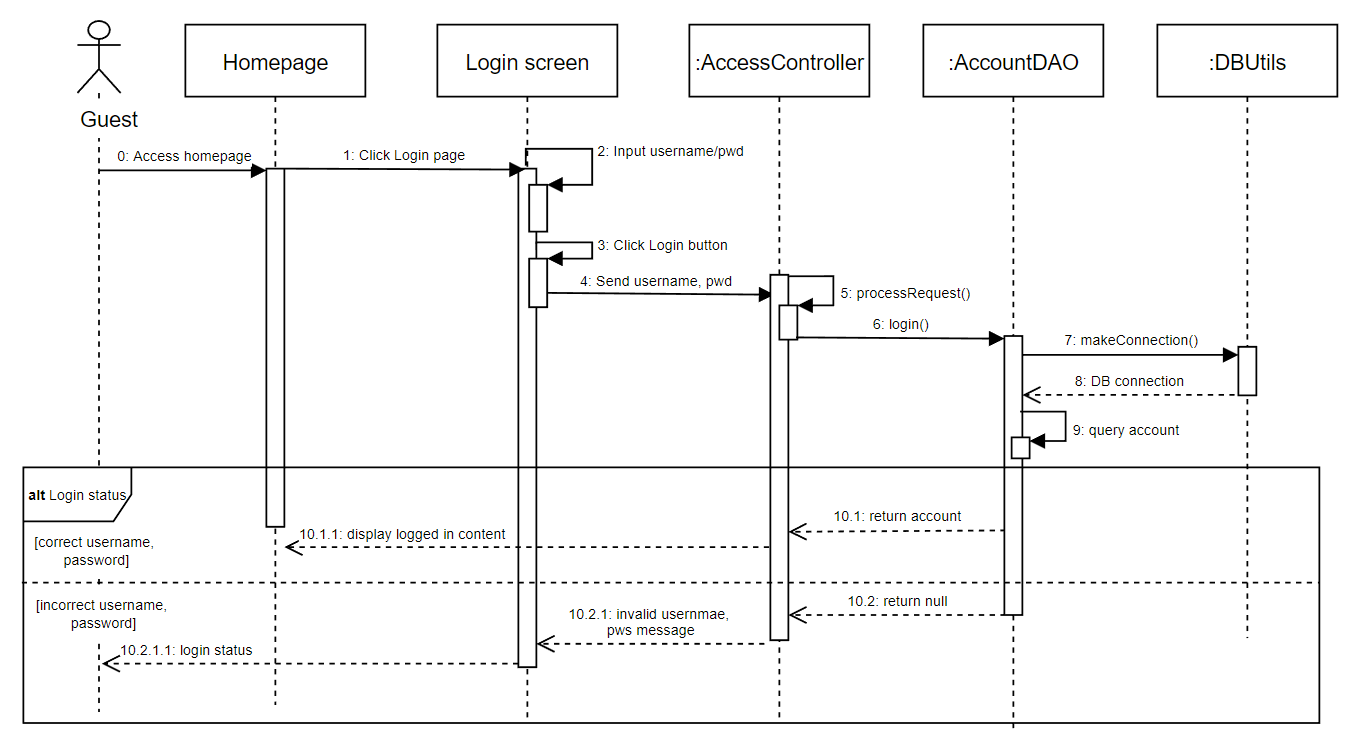
#### PasswordHash

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doHashing() | Encrypt user’s password with SHA-1 algorithm |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection |

### c. Sequence Diagram

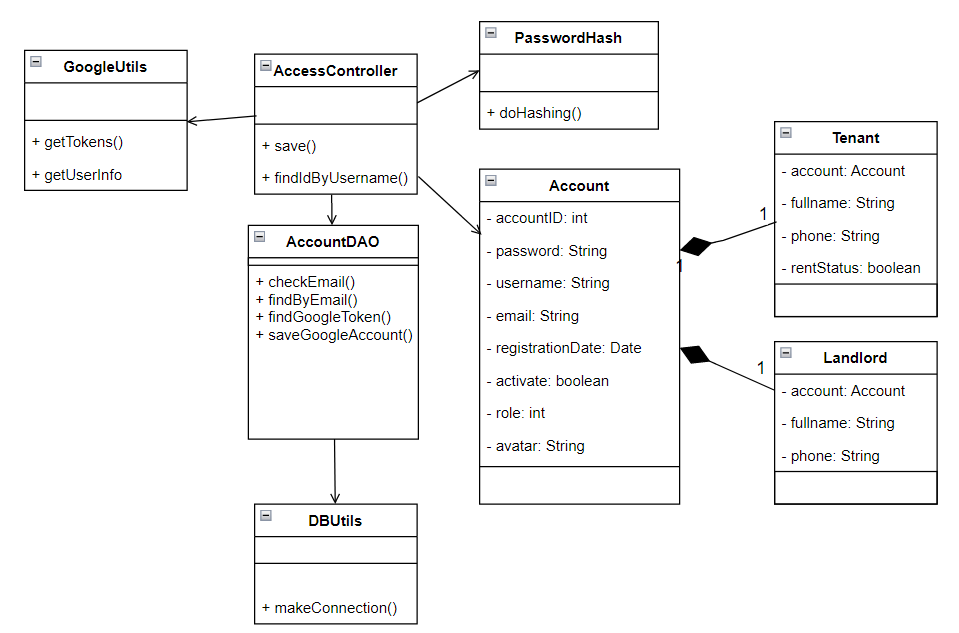


### d. Database queries

* *select \* from Account where username = ? and password = ?*
* *select \* from Account where email = ? and password = ?*

## 4. Login by gmail/facebook

### a. Class Diagram

**

### b. Class Specifications

#### AccessController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get facebook/google ID password parameters from request, execute query to verify whether ID is valid or not. Create a session if verify successfully, otherwise display a error message to the user. |

#### AccountDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | checkEmail() | Input email read from user input parameters, make DB connection, query and compare with email saved in the database. If email exists in the database, return true, otherwise return false. |
| 02 | findByEmail() | Input email read from user input parameters, make DB connection, query and compare with email saved in the database. If email exists in the database, return an account linked with that email, otherwise return null. |
| 03 | findGoogleToken() | Input google token read from Google login parameters, make DB connection, query and compare with token saved in the database. If a token exists in the database, return true, otherwise return false. |
| 04 | saveGoogleAccount() | Input google token read from Google login parameters, make DB connection, and update the token in the database. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection |

#### GoogleUtils

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getToken() | Input google code read from Google login parameters and use that code to get token and return it. |
| 02 | getUserInfo() | Input google token get |

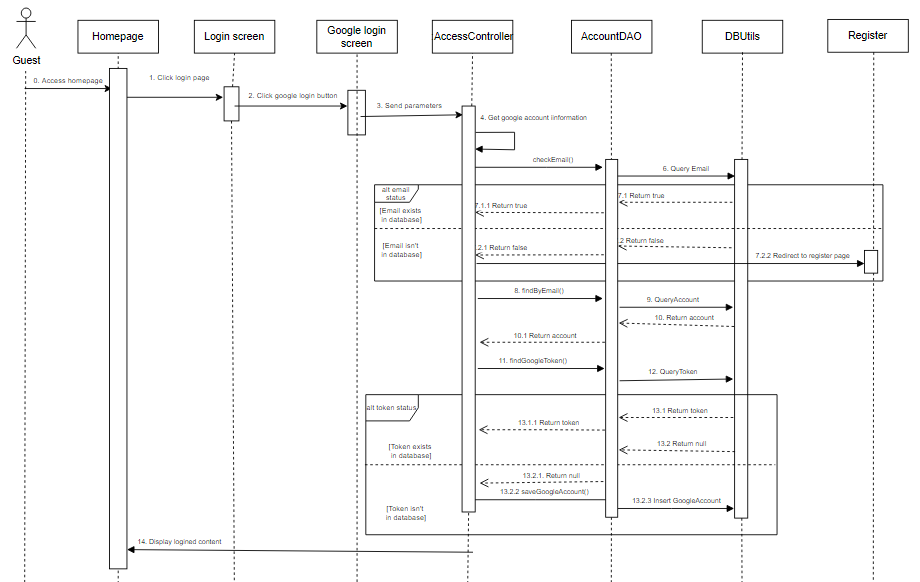
#### PasswordHash

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doHashing() | Encrypt user’s password with SHA-1 algorithm |

#### 

### c. Sequence Diagram



### d. Database queries

* *select \* from Account where email = ?*
* *insert into GoogleAccount (googleID, googleToken) values (?, ?)*

## 5. Reset password

### a. Class Diagram

### b. Class Specifications

#### AccessController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get email parameter from request, generate a new random password for account associated with that email. |

#### AccountDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByEmail() | Input email, execute query to find account corresponding to that email in database. Returns the associated account or null if can find any matching account. |
| 02 | changePassword() | Void method. Input accountID and new random password for the user, update user’s password in database. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection |

#### EmailService

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | sendResetPasswordEmail() | Input email and the new password, then send the user a message with the new password that was generated for him or her so that they may use it to log in to the system. |

### 

### c. Sequence Diagram

### d. Database queries

* *select \* from Account where email = ?*
* *update Account set password = ? where accountID = ?*

## 6. Change user profile

### a. Class Diagram

### b. Class Specifications

#### AccessController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as username, password, email, full name, phone, and roleID. function from AccountDAO, TenantDAO, and LandlordDAO. Create an account with a tenant or landlord role based on roleID (tenant/landlord object), and save that account to the database. |

#### TenantDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | save() | Input a tenant object from AccessController, execute a query to insert tenant information into the database. Return true if insert successfully. |

#### LandlordDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | save() | Input a landlord object from AccessController, execute a query to insert tenant information into the database. Return true if insert successfully. |

#### AccountDAO

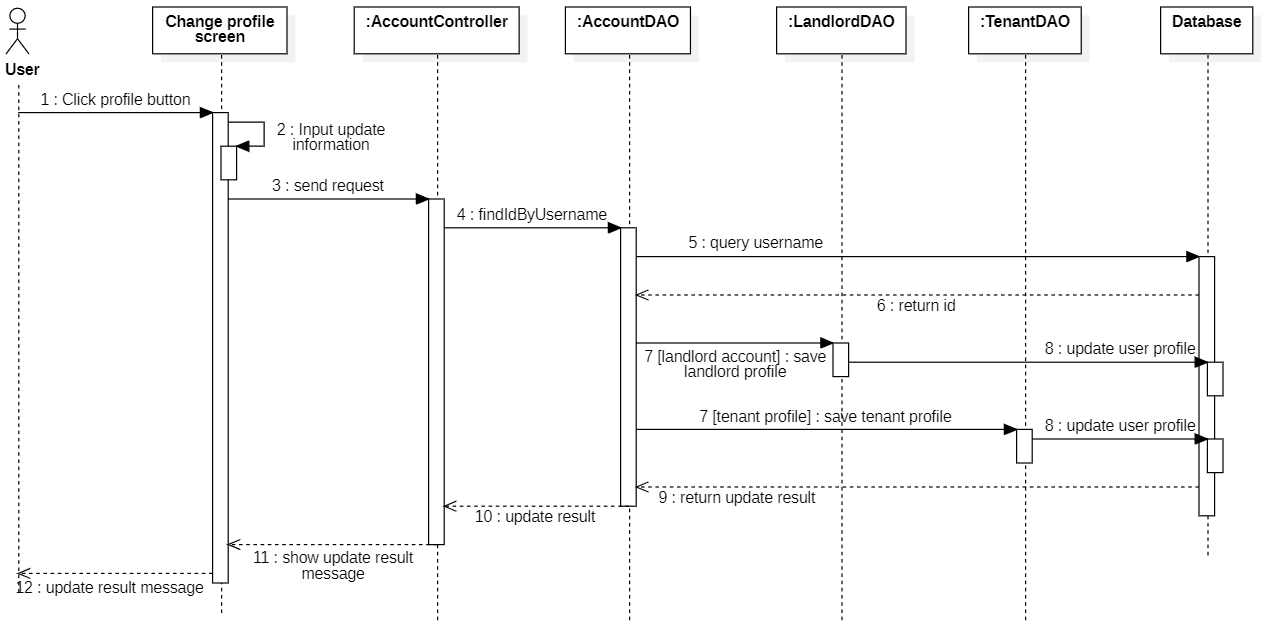
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | save() | Input tenant or landlord information from TenantDAO or LandlordDAO, and execute a query to insert a user into the Account table. If you successfully inserted, return true; otherwise, return false. |
| 02 | findIdByUsername() | Input a username, execute query, and return an account ID corresponding with the username. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection |

### 

### c. Sequence Diagram



### d. Database queries

* *insert into Tenant(tenantID, fullname, phone, rentStatus) values(?, ?, ?, ?)*
* *insert into Landlord(landlordID, fullname, phone) values(?, ?, ?)*
* *insert into Account(username, password, email, registeredDate, role, activate) values(?, ?, ?, ?, ?, ?)*
* *select accountID from Account where username = ?*

## 7. Change password

### a. Class Diagram

### b. Class Specifications

#### AccessController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as username, password, email, full name, phone, and roleID. function from AccountDAO, TenantDAO, and LandlordDAO. Create an account with a role as tenant or landlord based on roleID (tenant/landlord object), and save that account to the database. |

#### AccountDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | changePassword() | Void method. Input accountID and new password of the user, update user’s password in database. Return true if update successfully. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### 

### c. Sequence Diagram

### 

### d. Database queries

* *update Account set password = ? where accountID = ?*

## 8. View hostel list, favorite hostel list (tenant feature)

### a. Class Diagram

### b. Class Specifications

#### AccessController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as paging, favorite, keyword, province, district to filter hostels as users’ will. Get lists of hostels to show users the hostels that are in the database. |

#### HostelDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findFavoriteList() | Input the account id of the user, execute query and return a list of favorite activating hostels |
| 02 | findAll() | No input, execute query and return full list of activating hostel |

#### ProvinceDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findById() | Input the province id, execute query and return a province |

#### DistrictDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findById() | Input the district id, execute query and return a district |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | Make a connection to the database |

#### 

### c. Sequence Diagram

### 

### 

### 

### 

### 

### d. Database queries

* *insert into Tenant(tenantID, fullname, phone, rentStatus) values(?, ?, ?, ?)*
* *insert into Landlord(landlordID, fullname, phone) values(?, ?, ?)*
* *insert into Account(username, password, email, registeredDate, role, activate) values(?, ?, ?, ?, ?, ?)*
* *select accountID from Account where username = ?*

## 9. View hostel detail, hostel’s reviews/feedback (tenant feature)

### a. Class Diagram

### b. Class Specifications

#### HostelController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as hostelID, paging,message, filterStar to filter and show the hostel detail. Get the hostel detail like name, address, min price, max price, review, landlord profile |

#### HostelDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findById() | Input the account id of the user, execute query and return a hostel |
| 02 | findOutstandingHostels() | No input, execute query and return a list of high-rated hostels. |
| 03 | updateRating() | Input the hostel id and a new rating number, execute update and return a boolean |

#### FeedbackDAO

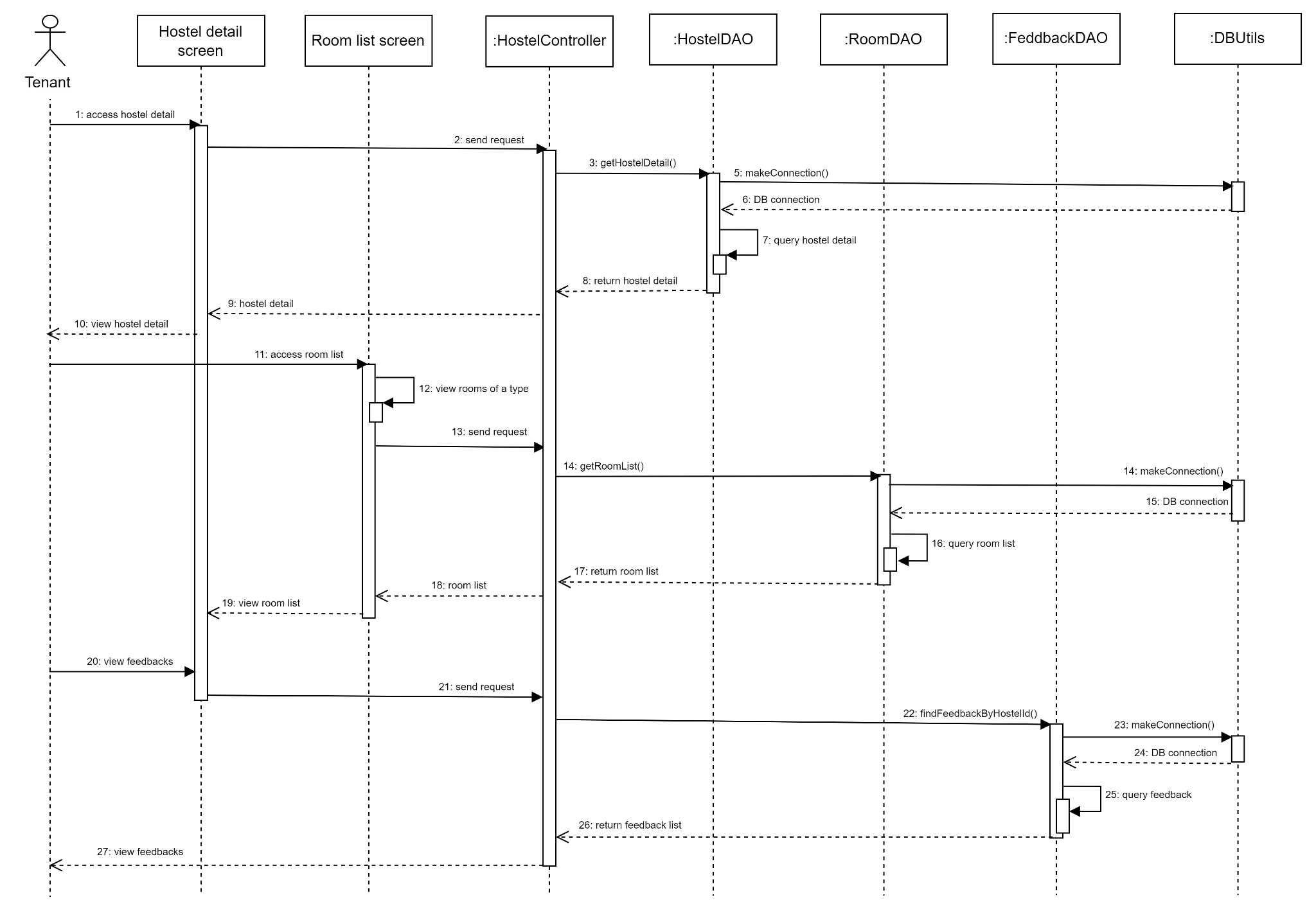
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByHostelId() | Input the hostel id, execute query and return a list of reviews |
| 02 | add() | Input account id, hostel id, feedback content and rating, execute update and return a boolean |
| 03 | update() | Input the account id, hostel id, feedback content, and rating, execute update and return a boolean |
| 04 | findByHostelTenant() | Input hostelId and accountId, execute query and return a true if tenant has accountId id feedbacked hostel has hostelId else return false |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | Make a connection to the database |

#### 

### c. Sequence Diagram



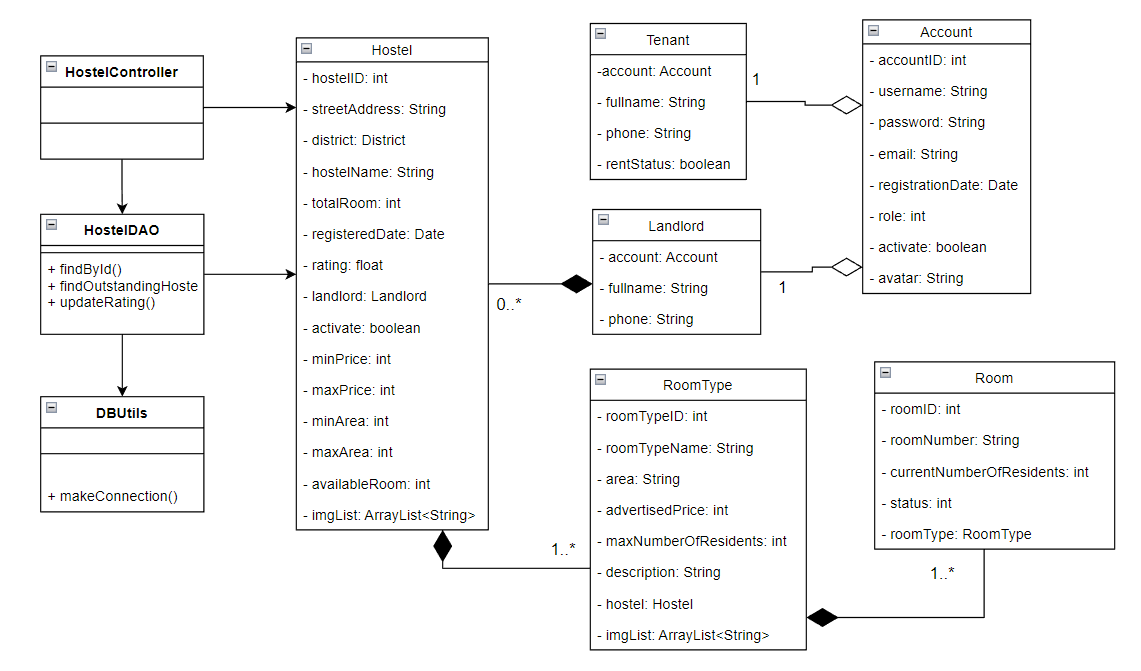
### 

### d. Database queries

* *select TOP(4) \* from Hostel order by rating desc*
* *select \* from Hostel where hostelID = ?*
* *Select \* from Province*
* *select feedbackID, TenantID, content, rating, date from feedback where hostelID = ? order by date DESC*
* *select feedbackID, content, rating, date from feedback where hostelID = ? and tenantID = ?*
* *insert into Feedback(tenantID, hostelId, content, rating) values (?, ?, ?, ?)*
* *update Feedback set content = ?, rating = ?, date = ? where tenantID = ? AND hostelId = ?*
* *update hostel set rating = ? where hostelID = ?*
* *select \* from FavoriteHostel where hostelID = ? and tenantID = ?*
* *Select \* from dist*

## 10. View room list of a hostel (tenant feature)

### a. Class Diagram



### b. Class Specifications

#### AccessController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as hostelID, paging,message, filterStar to filter and show the hostel detail. Get the hostel detail like name, address, min price, max price, review, landlord profile |

#### HostelDAO

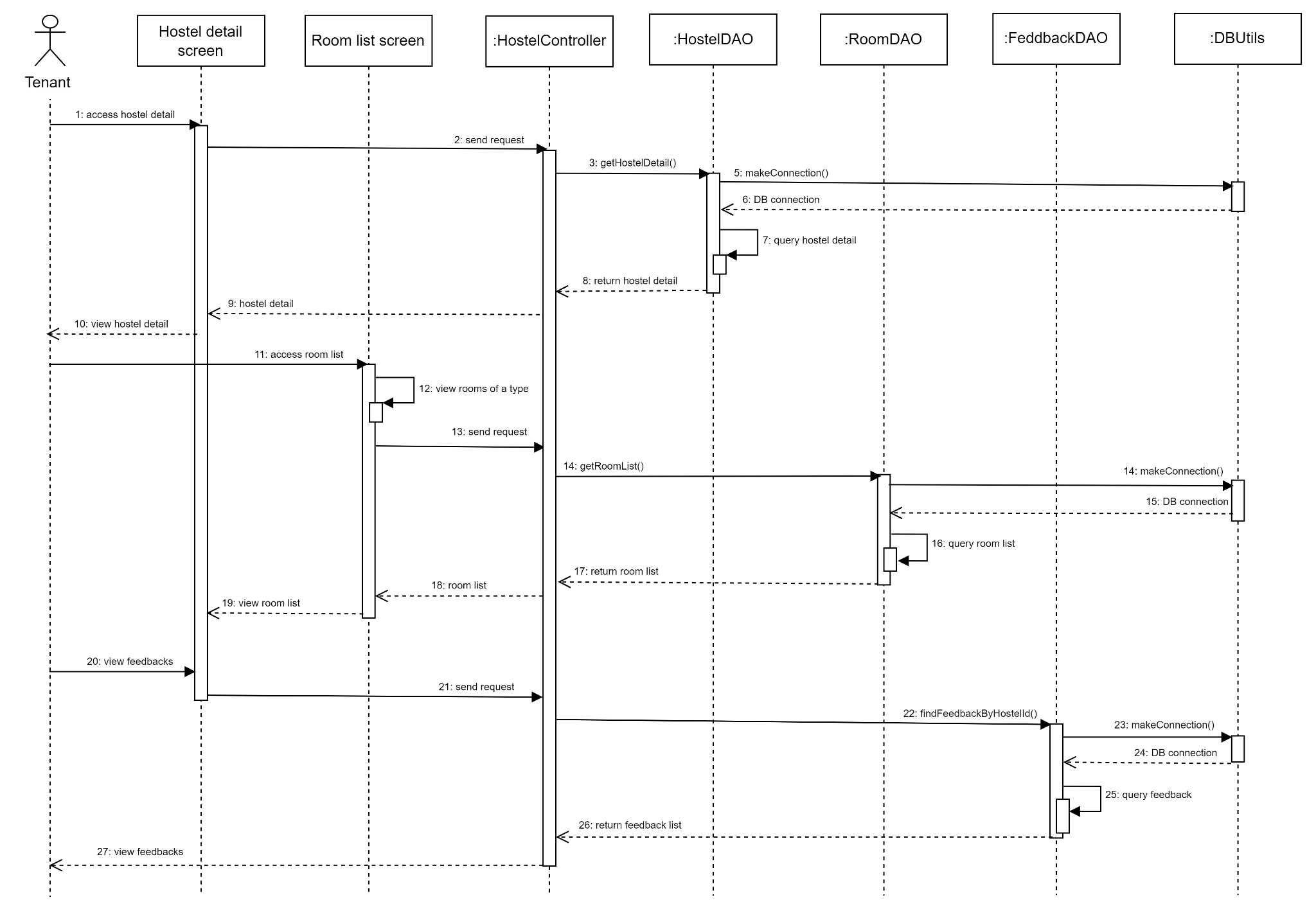
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findById() | Input the account id of the user, execute query and return a hostel |
| 02 | findOutstandingHostels() | No input, execute query and return a list of high-rated hostels. |
| 03 | updateRating() | Input the hostel id and a new rating number, execute update and return a boolean |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | Make a connection to the database |

#### 

### c. Sequence Diagram



### 

### d. Database queries

* *select TOP(4) \* from Hostel order by rating desc*
* *select \* from Hostel where hostelID = ?*
* *update hostel set rating = ? where hostelID = ?*
* *select \* from FavoriteHostel where hostelID = ? and tenantID = ?*

## 11. Add new feedback, edit feedback of a hostel (tenant feature)

### a. Class Diagram

### b. Class Specifications

#### 

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as hostelID, paging,message, filterStar to filter and show the hostel detail. Get the hostel detail like name, address, min price, max price, review, landlord profile |

#### HostelDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findById() | Input the account id of the user, execute query and return a hostel |
| 02 | findOutstandingHostels() | No input, execute query and return a list of high-rated hostels. |
| 03 | updateRating() | Input the hostel id and a new rating number, execute update and return a boolean |

#### FeedbackDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByHostelId() | Input the hostel id, execute query and return a list of reviews |
| 02 | add() | Input account id, hostel id, feedback content and rating, execute update and return a boolean |
| 03 | update() | Input the account id, hostel id, feedback content, and rating, execute update and return a boolean |
| 04 | findByHostelTenant() | Input hostel id and account id, execute query and return a feedback |

#### ProvinceDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findAll() | No input, execute query and return a list of provinces |

#### FavoriteHostelDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByHostelTenant() | Input the hostelID and accountID, execute query and return a boolean |

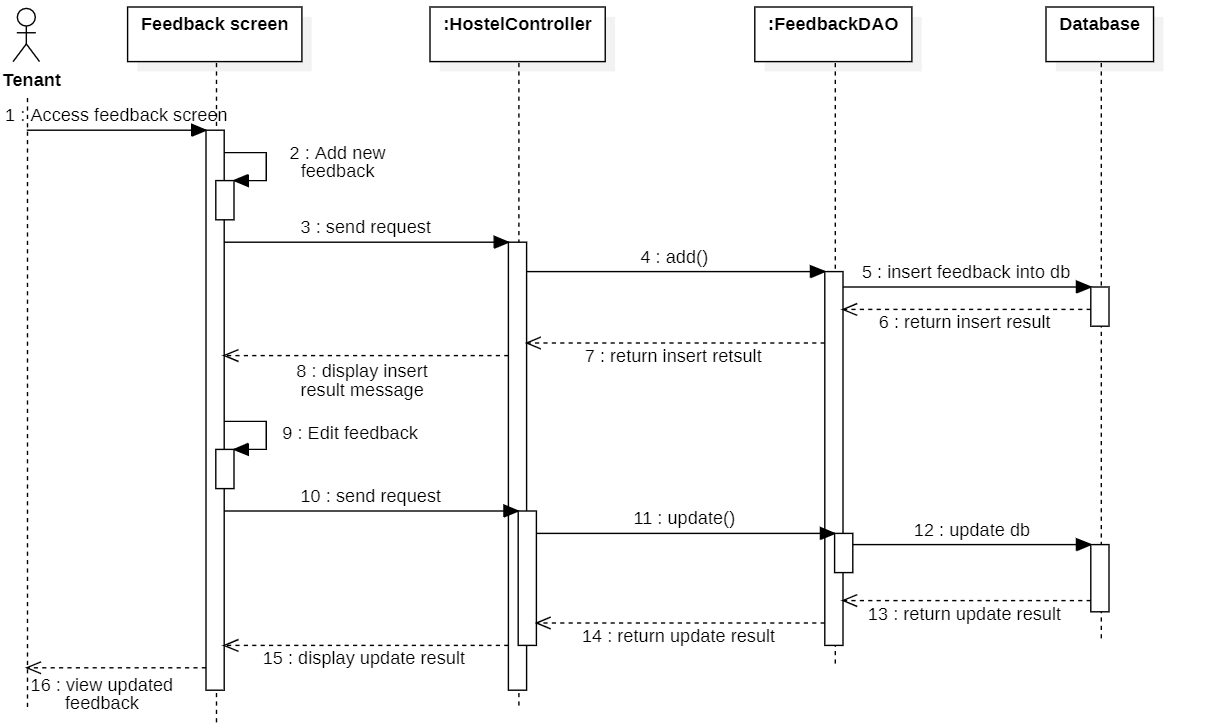
#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | Make a connection to the database |

#### 

### 

### c. Sequence Diagram

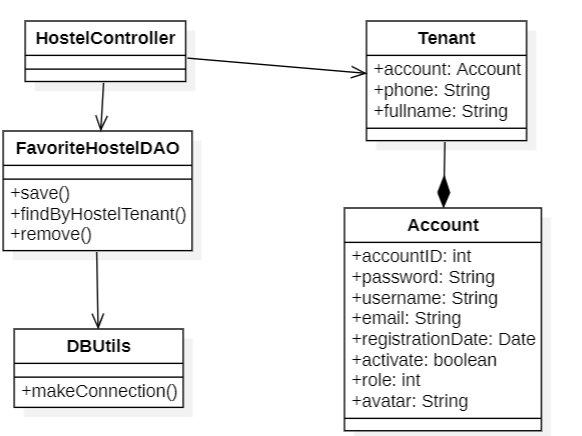


### d. Database queries

* *select TOP(4) \* from Hostel order by rating desc*
* *select \* from Hostel where hostelID = ?*
* *update hostel set rating = ? where hostelID = ?*
* *select \* from FavoriteHostel where hostelID = ? and tenantID = ?*

## 12. Toggle a favorite hostel (tenant feature)

### a. Class Diagram



### b. Class Specifications

#### AccessController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as hostelID, paging,message, filterStar to filter and show the hostel detail. Get the hostel detail like name, address, min price, max price, review, landlord profile |

#### FavoriteHostelDAO

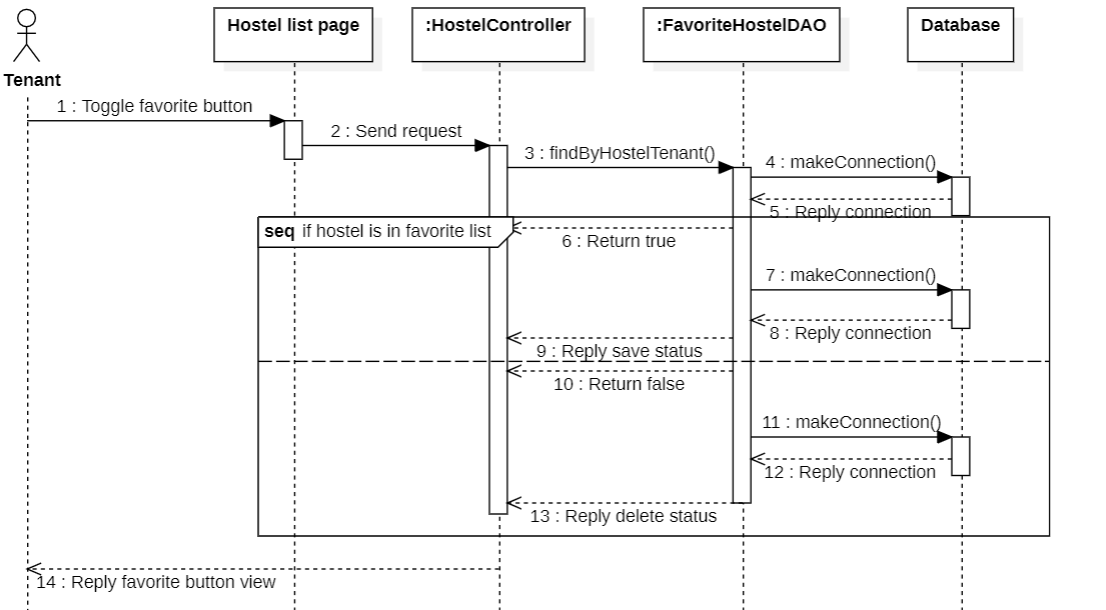
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByHostelTenant() | Input the hostelID and accountID, execute query and return a boolean |
| 02 | save() | Input hostelID and tenantID, execute update and return a boolean |
| 03 | remove() | Input hostelID and tenantID, execute remove and return a boolean |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | Make a connection to the database |

#### 

### c. Sequence Diagram



### d. Database queries

* *select \* from FavoriteHostel where hostelID = ? and tenantID = ?*
* *insert into FavoriteHostel(hostelID, tenantID) values(?, ?)*
* *delete from FavoriteHostel where hostelID = ? and tenantID = ?*

## 13. Rent a room of a hostel

### a. Class Diagram

### b. Class Specifications

#### HostelController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get rental request from tenant, save it to database and save two notifications sent to landlord and tenant |

#### RoomDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findById() | Input roomID, execute query and return a room |

#### NotificationDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | saveNotification() | Input Notification, execute insert and return a boolean |

#### BookingRequestDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | saveBookingRequest() | InputBooking Request, execute insert and return a boolean |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | Make a connection to the database |

#### 

#### 

### c. Sequence Diagram

### d. Database queries

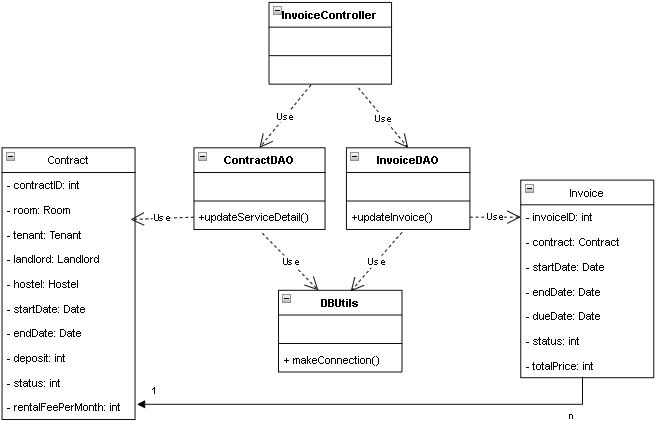
*Select \* from Room where roomID = ?*

*Insert into Notification (toAccountID, createdDate, content, status) values (?, ?, ?, ?)*

*Insert into BookingRequest (tenantID, roomID, createdDate, status) values (?, ?, ?, ?)*

## 14. View monthly invoice list (Tenant)

### a. Class Diagram



### b. Class Specifications

#### InvoiceController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as invoiceId, set request attribute, process after receiving request from user and call functions at InvoiceDAO. |

#### ContractDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByTenant() | returns a list of Contract objects. Parameter is a Tenant object. |

#### InvoiceDAO

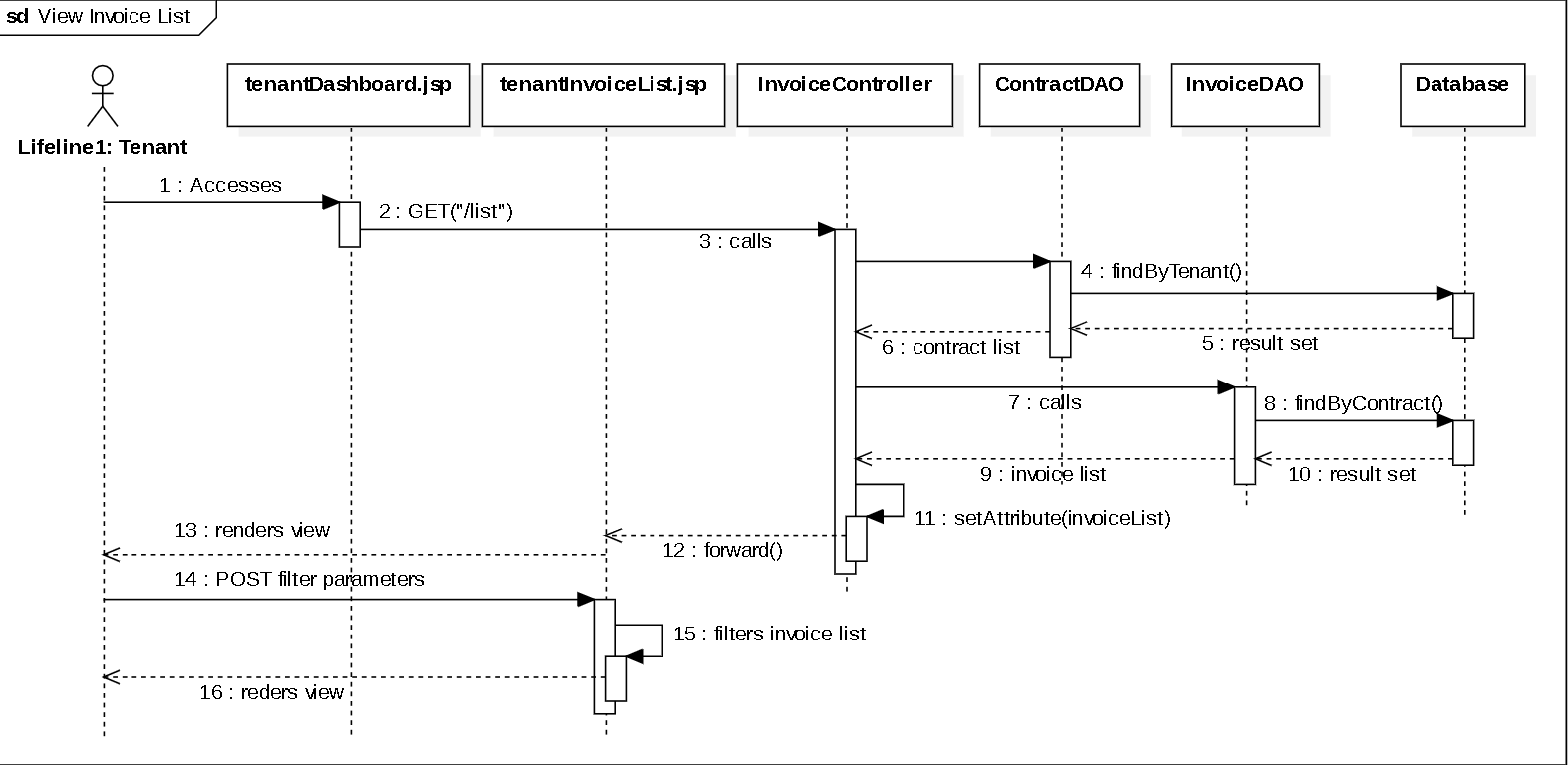
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByContract() | returns a list of Invoice objects. Parameter is a Contract object. |

#### DBUtils (Database)

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | No input, void function, connect to the database |

#### 

### c. Sequence Diagram

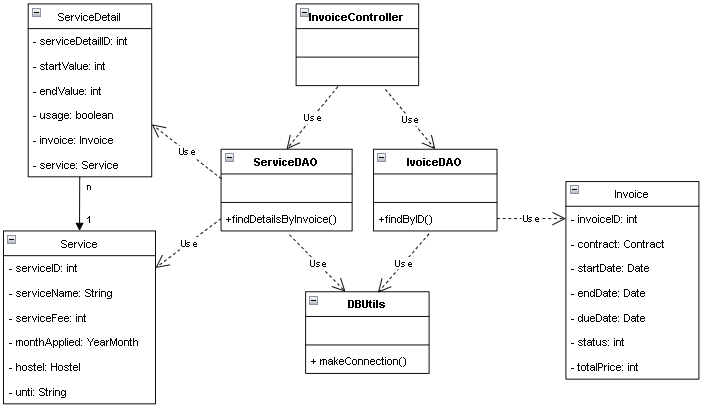


### d. Database queries

* *select \* from Contract where tenantID = ? order by startDate desc*
* *select \* from Invoice where contractID = ?*

## 15. View monthly invoice detail (Tenant feature)

### a. Class Diagram



### b. Class Specifications

#### InvoiceController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters, set request attribute, process after receiving request from user and call functions at InvoiceDAO, serviceDAO. |

#### InvoiceDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByID() | returns an Invoice object. Parameter is an invoice ID |

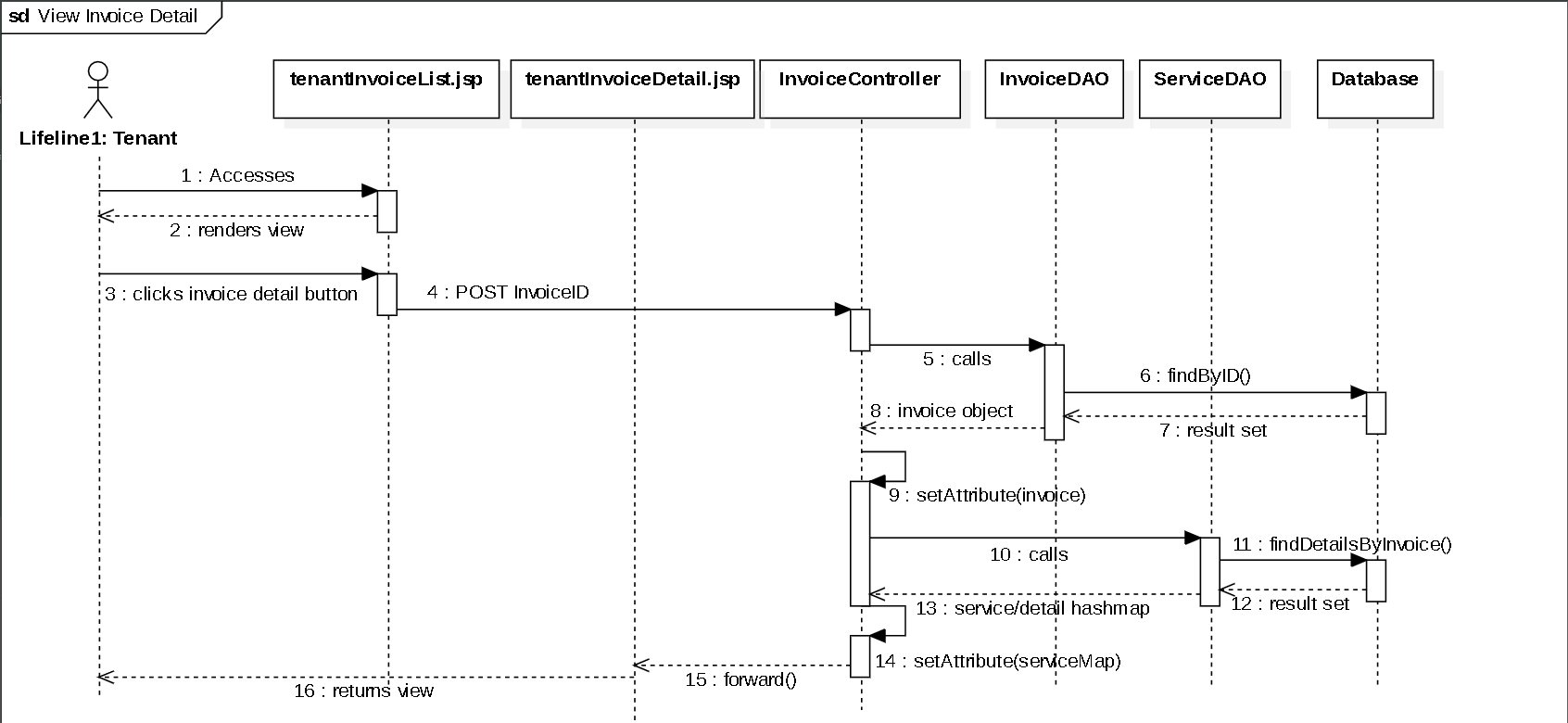
#### ServiceDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findDetailsByInvoice() | returns a hashmap of <Service object, ServiceDetail object>. Parameter is an Invoice object. |

#### DBUtils (Database)

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### c. Sequence Diagram

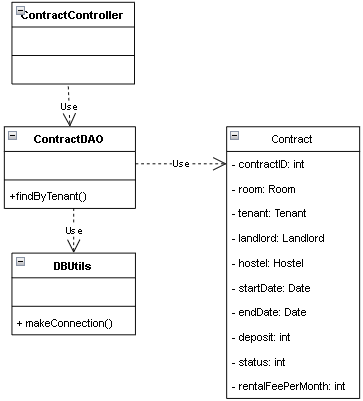


### d. Database queries

* *select \* from Invoice where invoiceID = ?*
* *select \* from ServiceDetail where invoiceID = ?*

## 16. View contract list(Tenant)

### a. Class Diagram



### b. Class Specifications

#### ContractController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as TenantID, set request attribute, process after receiving request from user and call functions at contractDAO. |

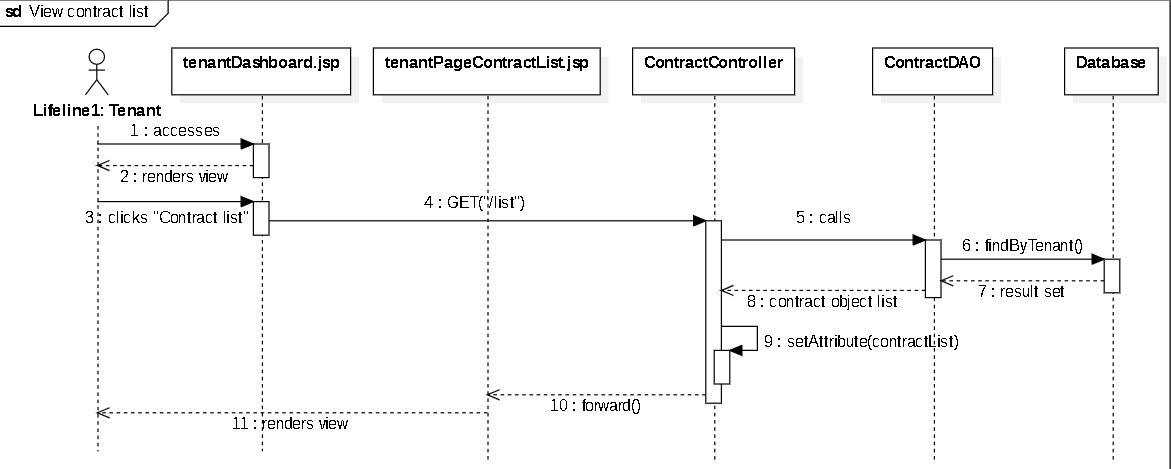
#### ContractDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByTenant() | returns list of contract with a specific tenant id. parameter is a tenant object |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### c. Sequence Diagram

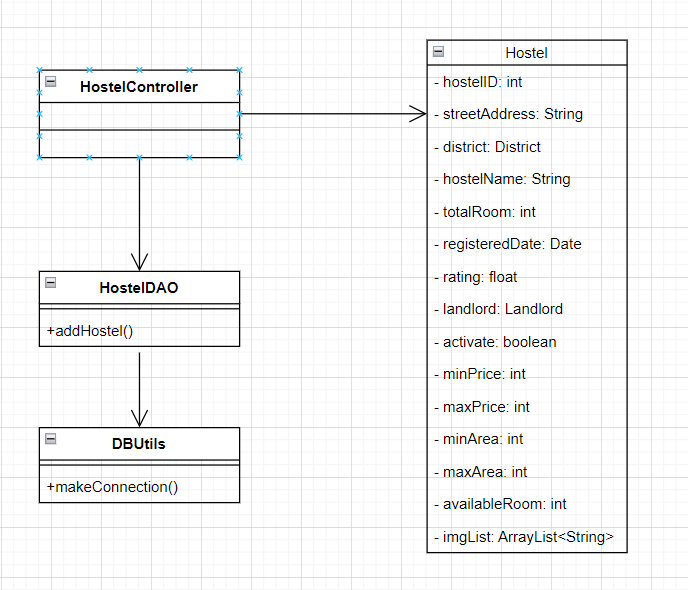
****

### d. Database queries

* *select \* from Contract where tenantID = ? order by startDate desc*

## 17. Add hostel (Landlord feature)

### a. Class Diagram

**

### b. Class Specifications

#### HostelController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as hostelId, set request attribute, process after receiving request from user and call functions at HostelDAO. |

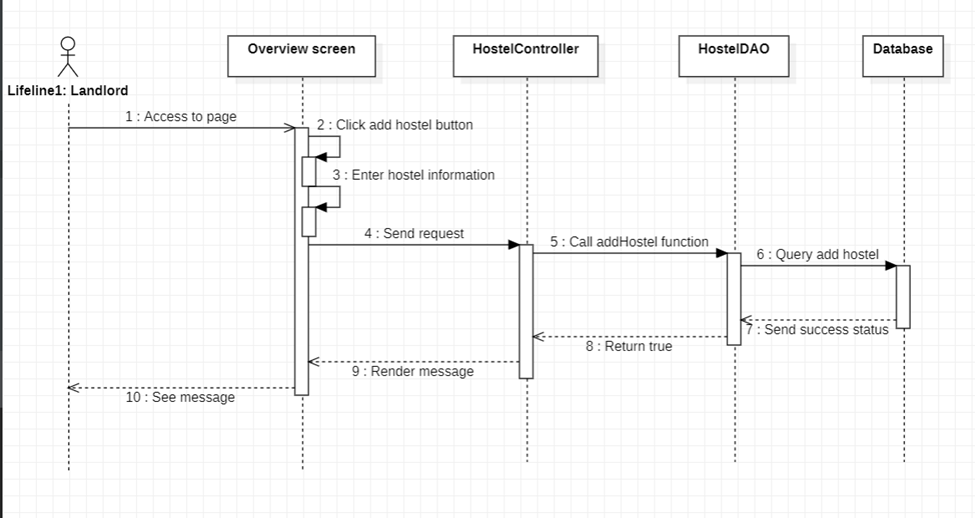
#### HostelDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | addHostel() | Returns true if add hostel successfully else returns false. input is hostel and landlordId. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### c. Sequence Diagram



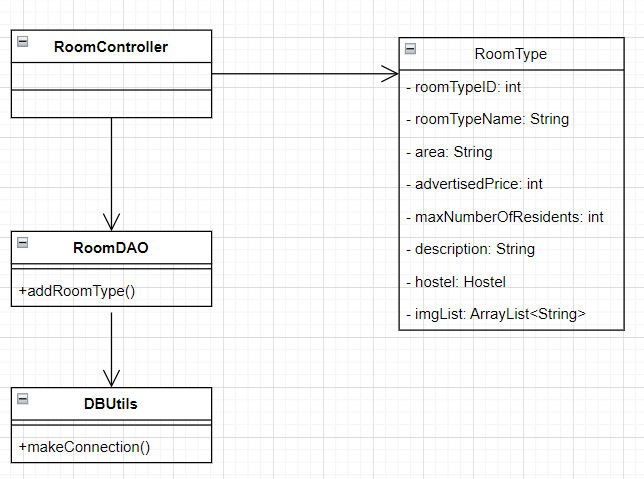
### d. Database queries

* *INSERT INTO Hostel(streetAddress, districtId, hostelName, totalRoom, landlordId, availableRoom, imgHostelId) VALUES(?, ?, ?, ?, ?, ?, ?)*

#### 

## 18. Add room type (Landlord feature)

### a. Class Diagram



### b. Class Specifications

#### RoomController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as roomid, roomtypeId set request attribute, process after receiving request from user and call functions at RoomDAO. |

#### RoomDAO

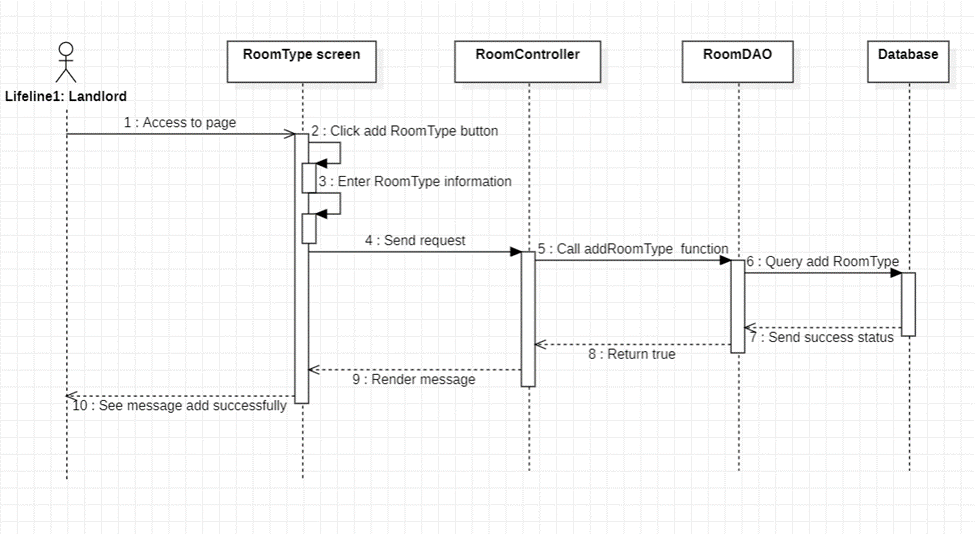
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | addRoomType() | Returns true if add room type successfully else returns false. Input is hostelId and roomType. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### 

### c. Sequence Diagram

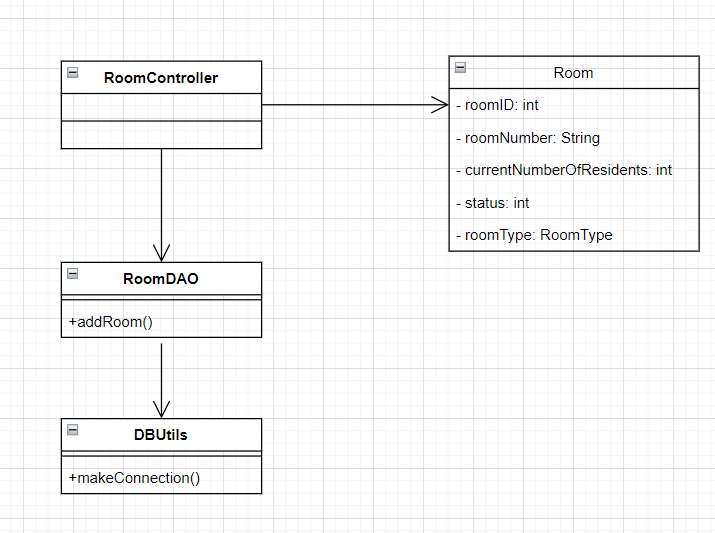


### d. Database queries

* *INSERT INTO RoomType(roomTypeName, area, advertisedPrice, maxNumberOfResidents, description, hostelId, imgRoomTypeId) VALUES(?, ?, ?, ?, ?, ?, ?,?)*

## 19. Add room (Landlord feature)

### a. Class Diagram



### b. Class Specifications

#### RoomController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as roomId, roomTypeId, set request attribute, process after receiving request from user and call functions at RoomDAO. |

#### RoomDAO

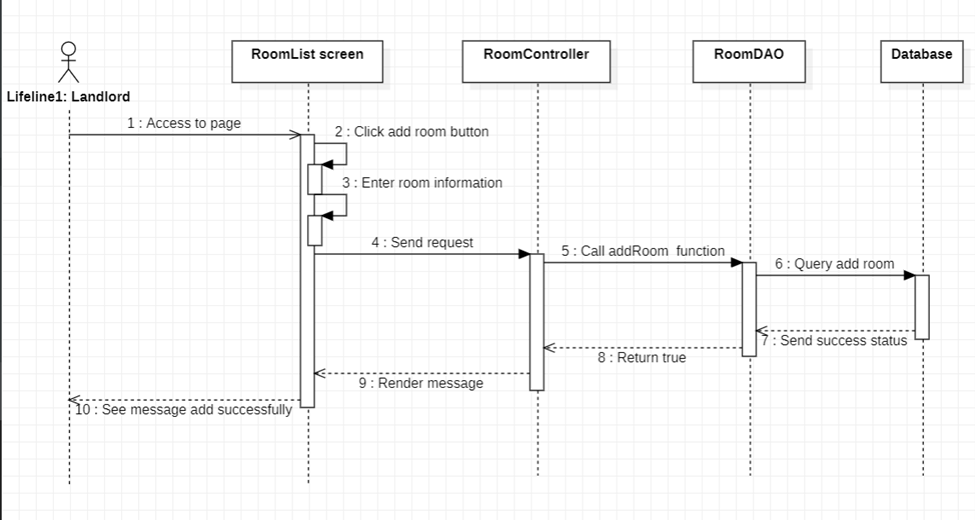
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | addRoom() | Returns true if add room successfully else returns false. input is room. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### 

### c. Sequence Diagram

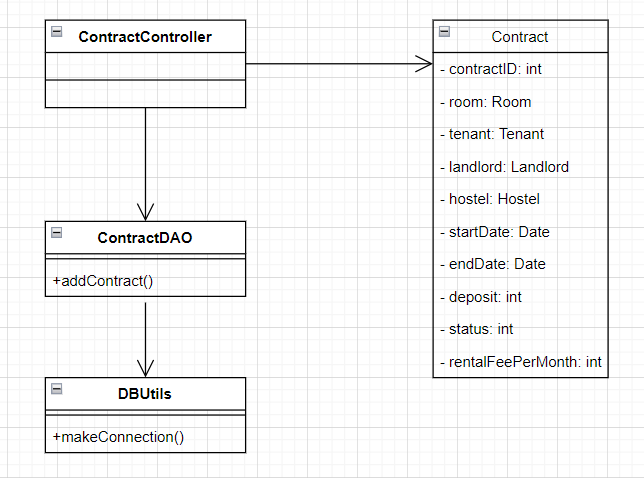


### d. Database queries

* *INSERT INTO Room(roomNumber, currentNumberOfResidents, status, roomType) VALUES(?,?,?,?)*

## 20. Add contract (Landlord feature)

### a. Class Diagram



### b. Class Specifications

#### ContractController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as contractId, set request attribute, process after receiving request from user and call functions at ContractDAO. |

#### ContractDAO

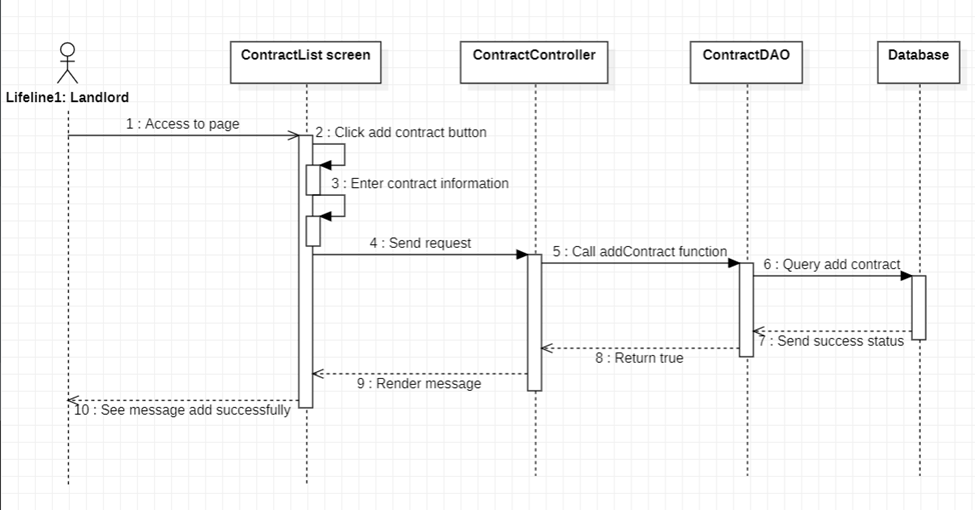
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | addContract() | Returns true if add contract successfully else returns false. input is contract. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### 

### c. Sequence Diagram

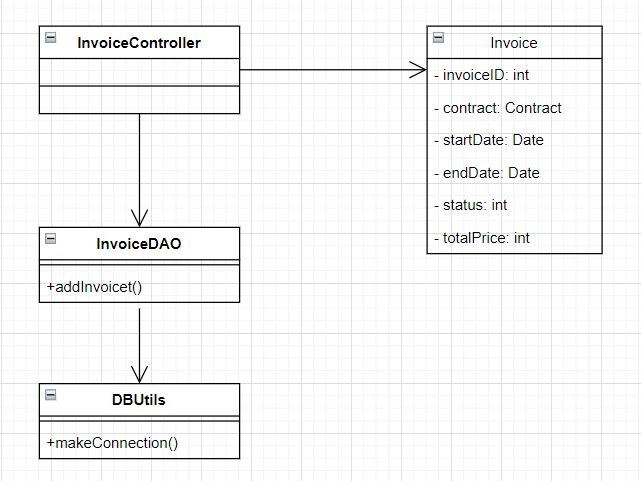


### d. Database queries

* *INSERT INTO Contract(roomId, tenantId, startDate, endDate, deposit, status, rentalFeePerMonth) VALUES(?, ?, ?, ?, ?, ?, ?)*

## 21. Add invoice (Landlord feature)

### a. Class Diagram



### b. Class Specifications

#### InvoiceController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as invoiceId, set request attribute, process after receiving request from user and call functions at InvoiceDAO. |

#### InvoiceDAO

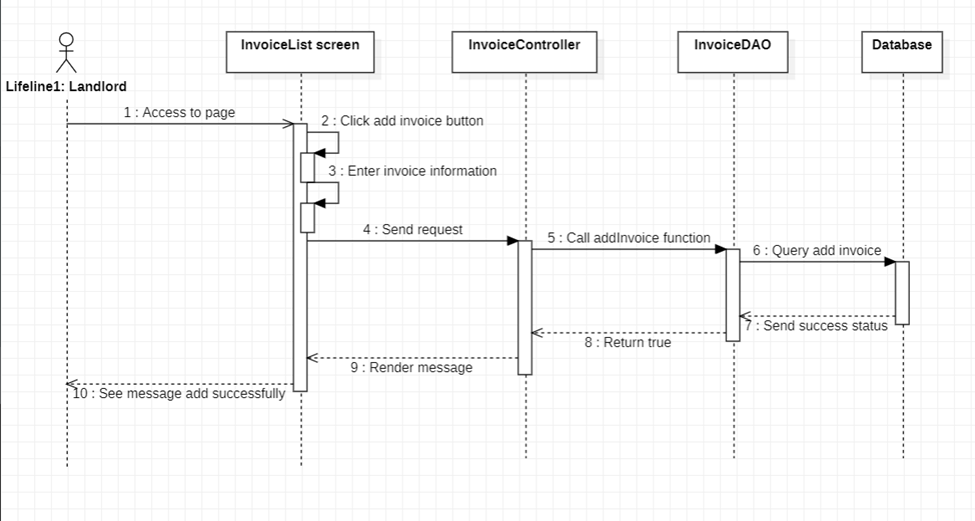
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | addInvoice() | Returns true if add invoice successfully else returns false. input is invoice. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### 

### c. Sequence Diagram

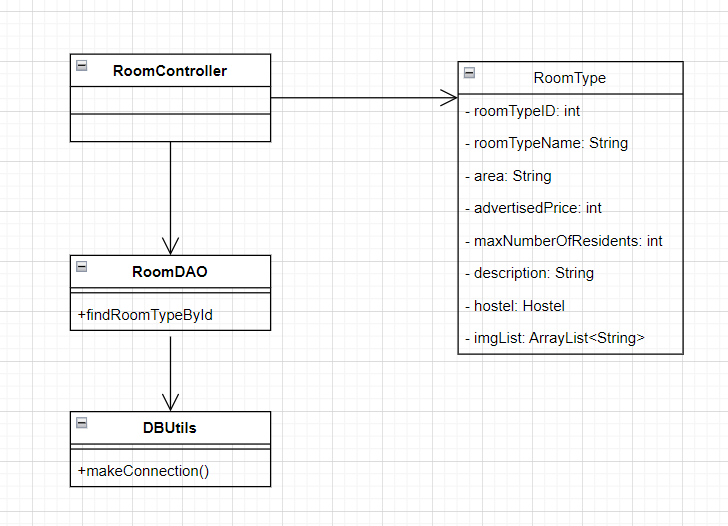


### d. Database queries

* *INSERT INTO Invoice(contractId, startDate, endDate, status, totalPrice) VALUES(?, ?, ?, ?, ?)*

## 22. View room type detail (Landlord feature)

### a. Class Diagram



### b. Class Specifications

#### RoomController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as roomId, roomTypeId, set request attribute, process after receiving request from user and call functions at RoomDAO. |

#### RoomDAO

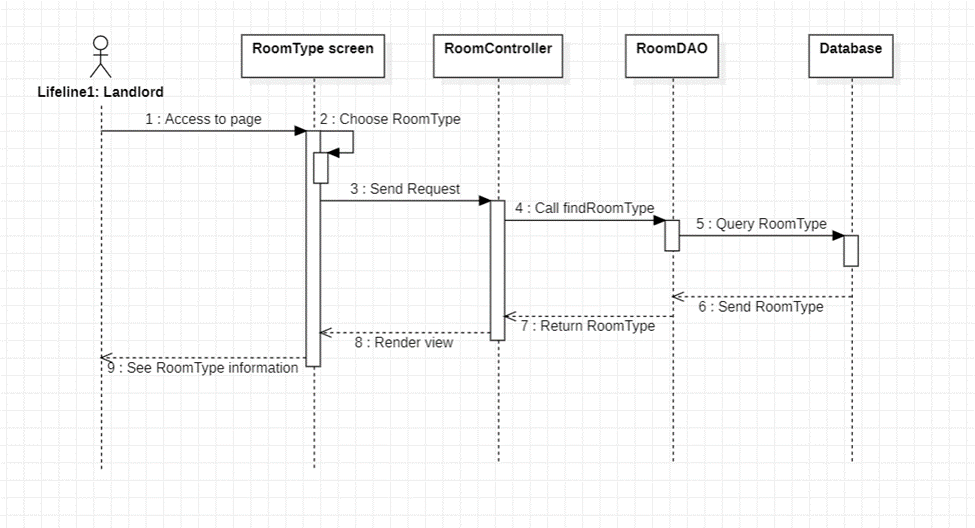
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findRoomTypeById | Returns the Room Type. Input id of roomtype. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### 

### c. Sequence Diagram

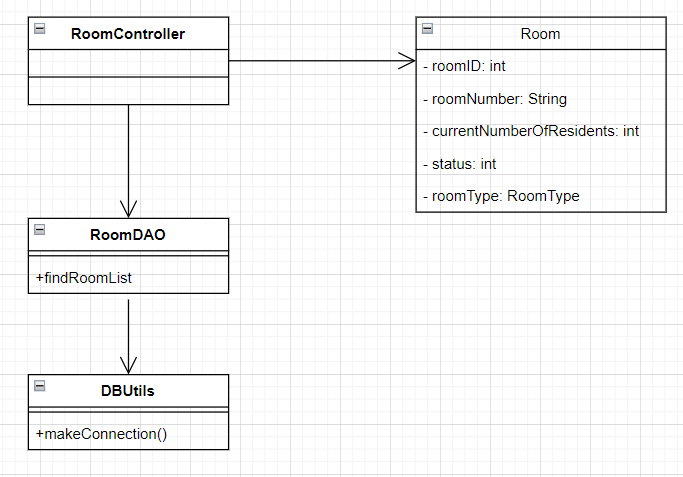


### d. Database queries

* *SELECT \* FROM RoomType WHERE roomTypeId = ?*

## 23. View room list (Landlord feature)

### a. Class Diagram



### b. Class Specifications

#### RoomController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as roomId, roomTypeId, set request attribute, process after receiving request from user and call functions at RoomDAO. |

#### RoomDAO

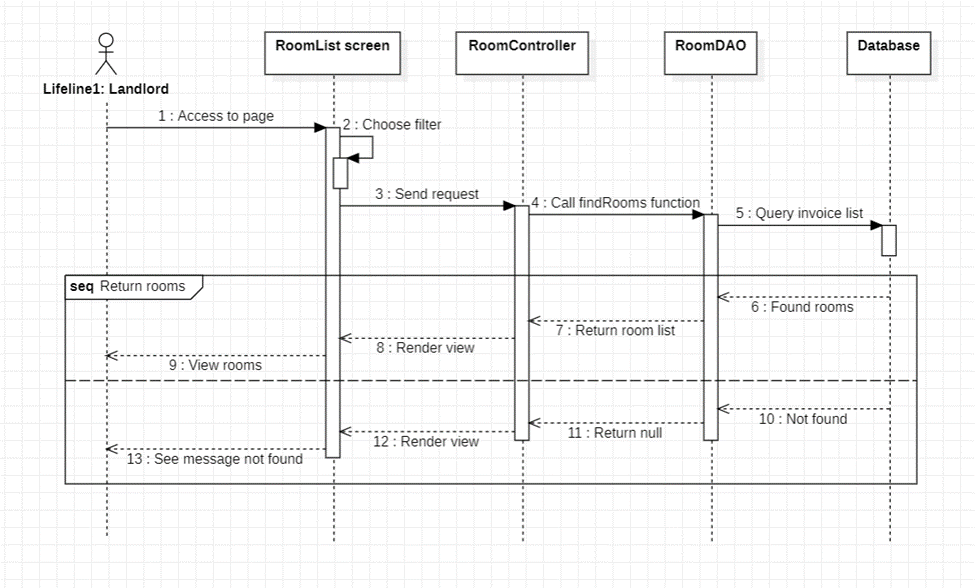
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findRoomList() | Returns the list of rooms. Input is landlordId. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### 

### c. Sequence Diagram

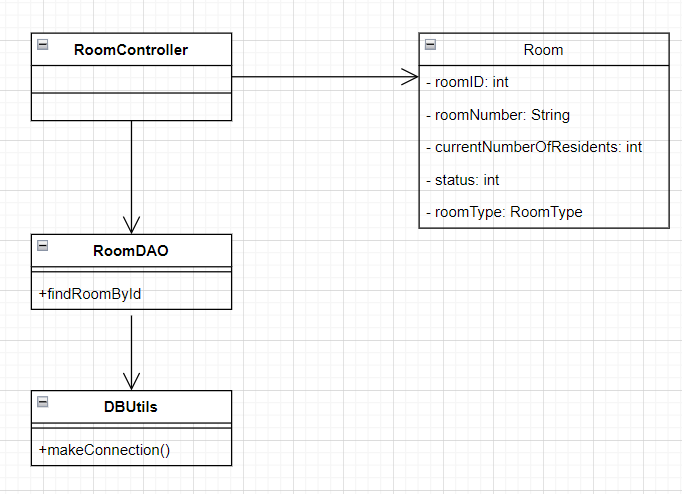


### d. Database queries

* *SELECT \* FROM Room Where hostelId = ?*

## 24. View room detail (Landlord feature)

### a. Class Diagram



### b. Class Specifications

#### RoomController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as roomId, set request attribute, process after receiving request from user and call functions at RoomDAO. |

#### RoomDAO

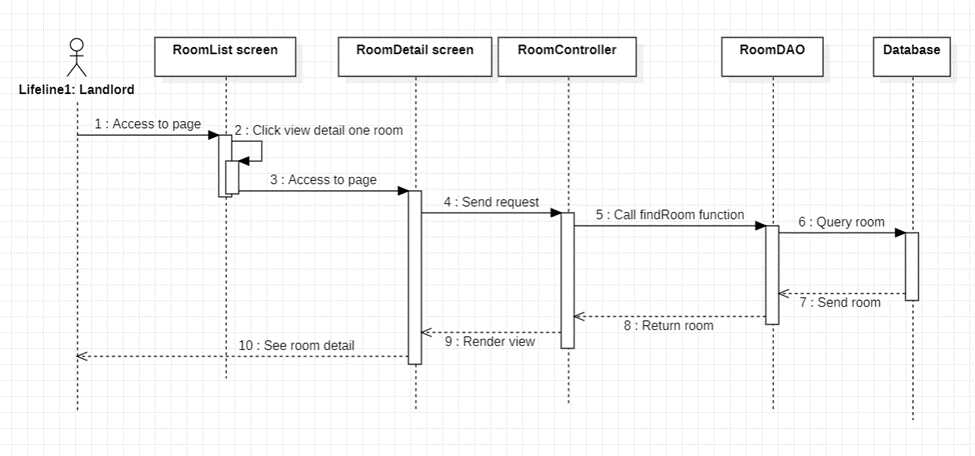
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findRoomByIdl() | Returns the detailed room. input is roomId. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### 

### c. Sequence Diagram

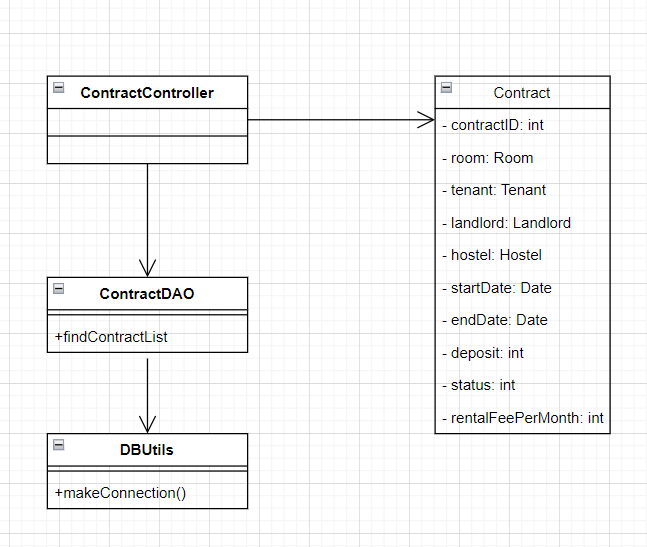


### d. Database queries

* *SELECT \* FROM Room Where roomId= ?*

## 25. View contract list (Landlord feature)

### a. Class Diagram



### b. Class Specifications

#### ContractController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as contractId, set request attribute, process after receiving request from user and call functions at ContractDAO. |

#### ContractDAO

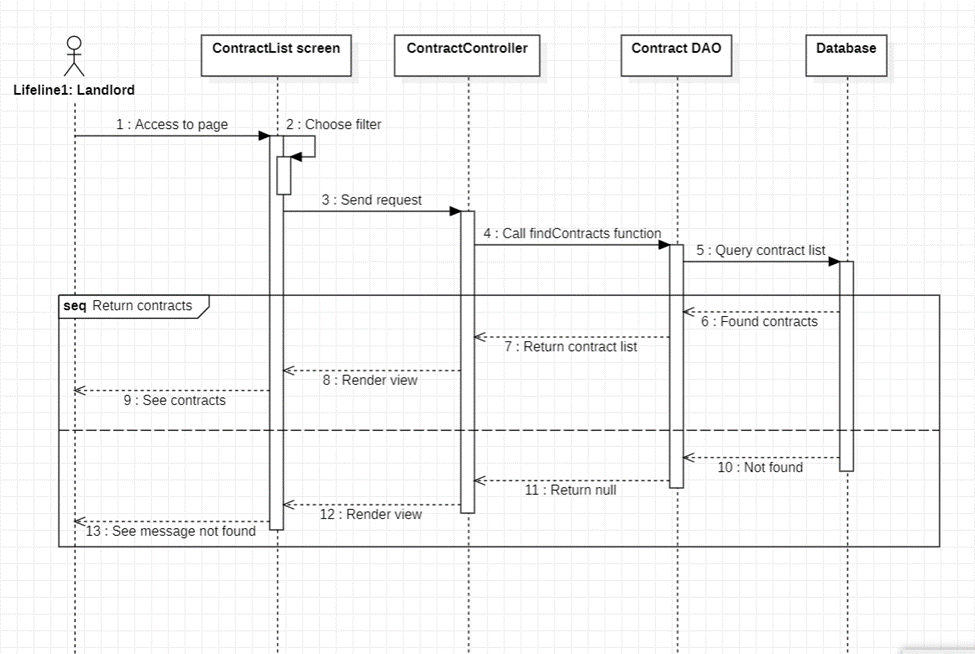
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findContractList() | Returns the contract list . Input is roomId. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### 

### c. Sequence Diagram

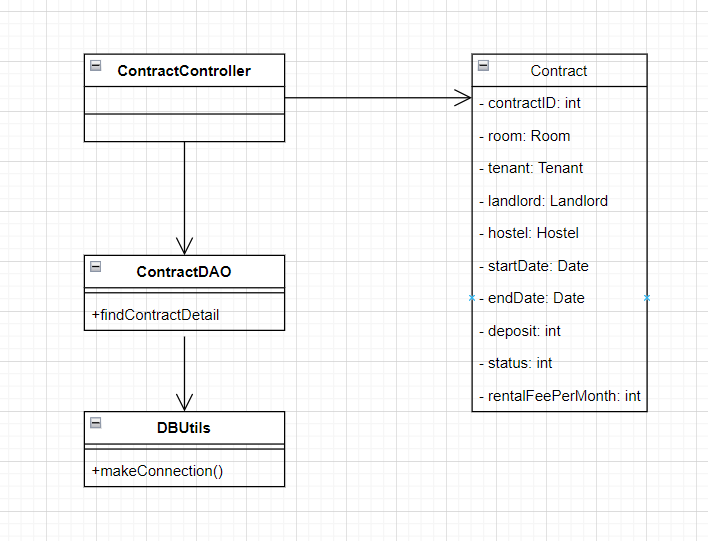


### d. Database queries

* *SELECT \* FROM Contract Where roomId= ?*

## 26. View contract detail (Landlord feature)

### a. Class Diagram



### b. Class Specifications

#### ContractController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as contractId, set request attribute, process after receiving request from user and call functions at ContractDAO. |

#### HostelDAO

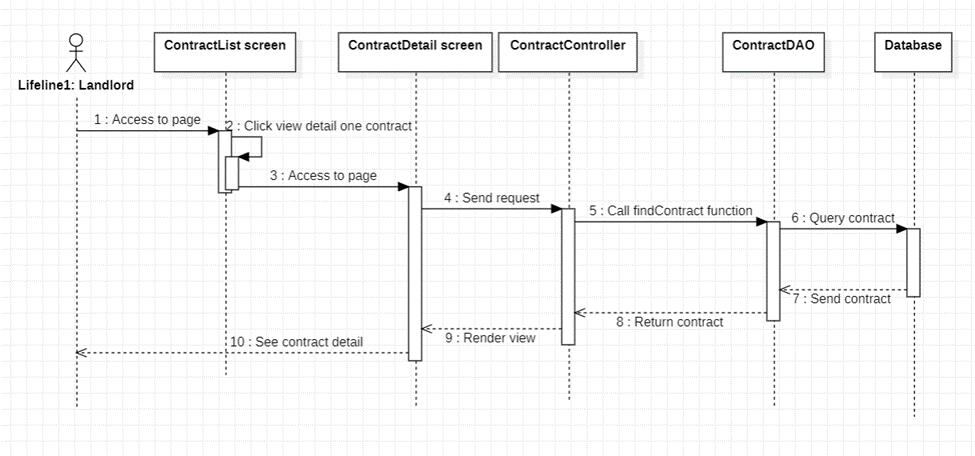
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findContract() | Returns the detailed contract. Input is hostel and contractId. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### 

### c. Sequence Diagram

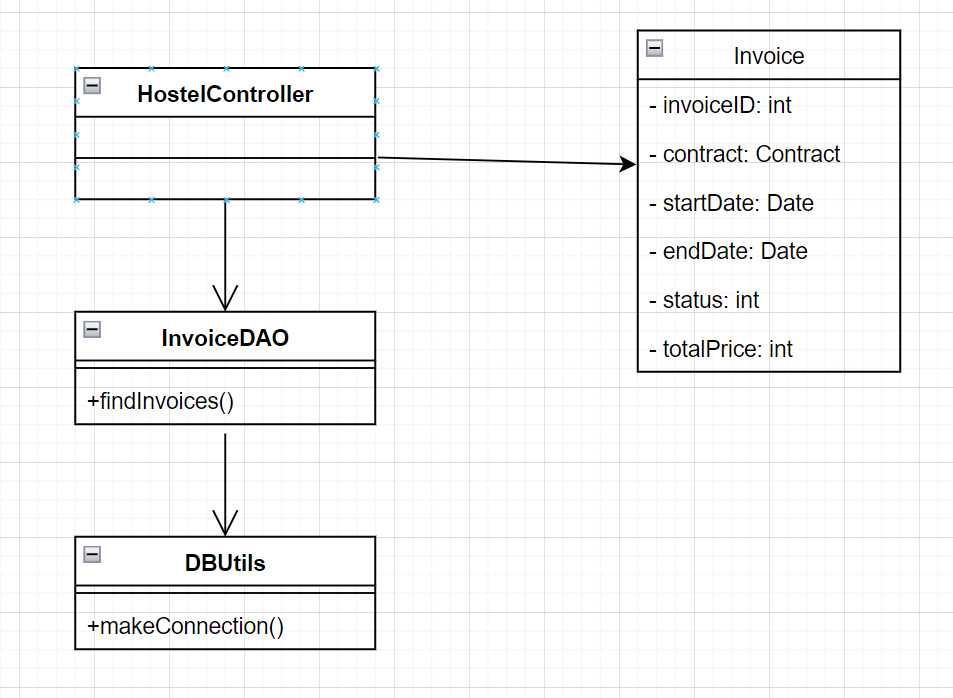


### d. Database queries

* *SELECT \* FROM Contract Where contractId = ? and hostelId = ?*

## 27. View invoice list (Landlord feature)

### a. Class Diagram



### b. Class Specifications

#### HostelContontroller

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as invoiceId, set request attribute, process after receiving request from user and call functions at InvoiceDAO. |

#### HostelDAO

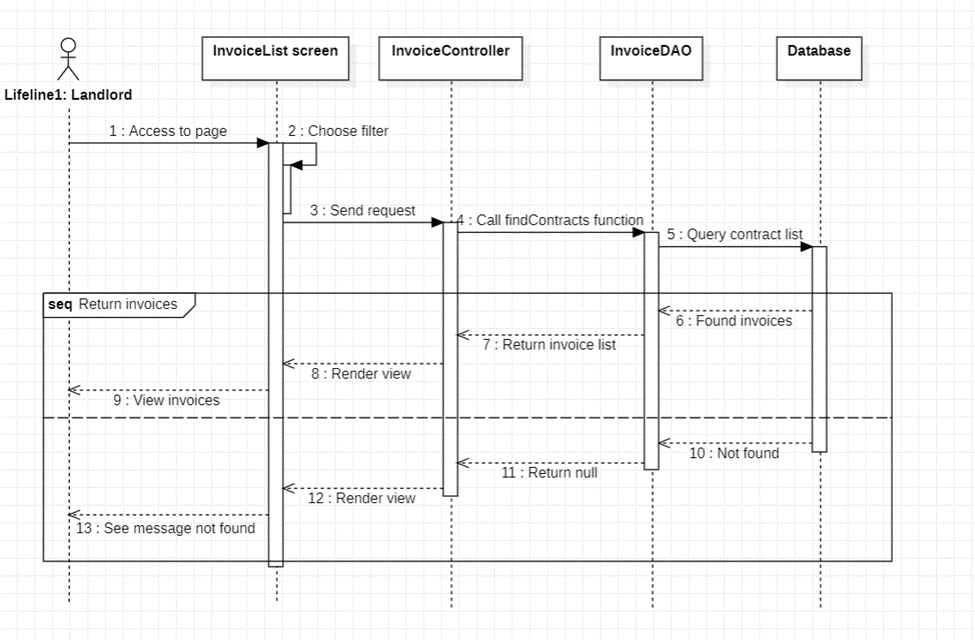
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findInvoices() | Returns invoice list. input is roomId. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### 

### c. Sequence Diagram

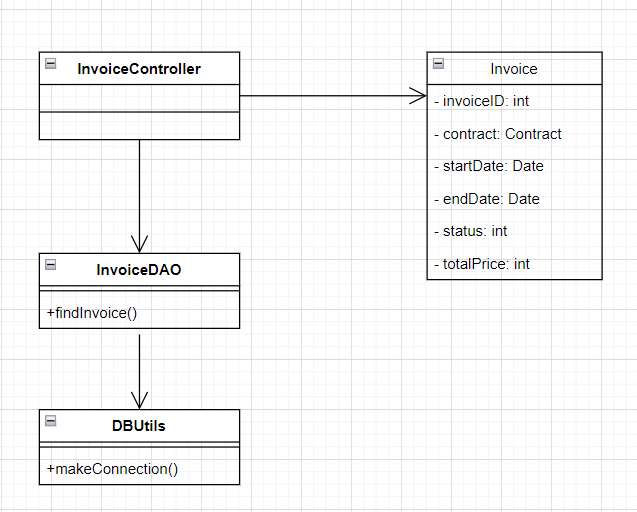


### d. Database queries

* *SELECT \* FROM Invoices Where invoiceId = ?*

## 28. View invoice detail (Landlord feature)

### a. Class Diagram



### b. Class Specifications

#### HostelContontroller

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as hostelId, set request attribute, process after receiving request from user and call functions at DAO. |

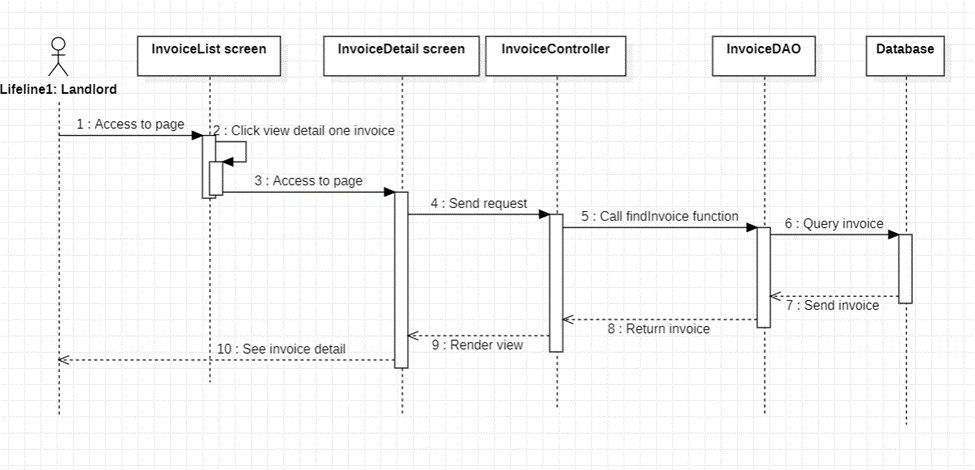
#### HostelDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findInvoice() | Returns the invoicel. input is invoiceId. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### c. Sequence Diagram

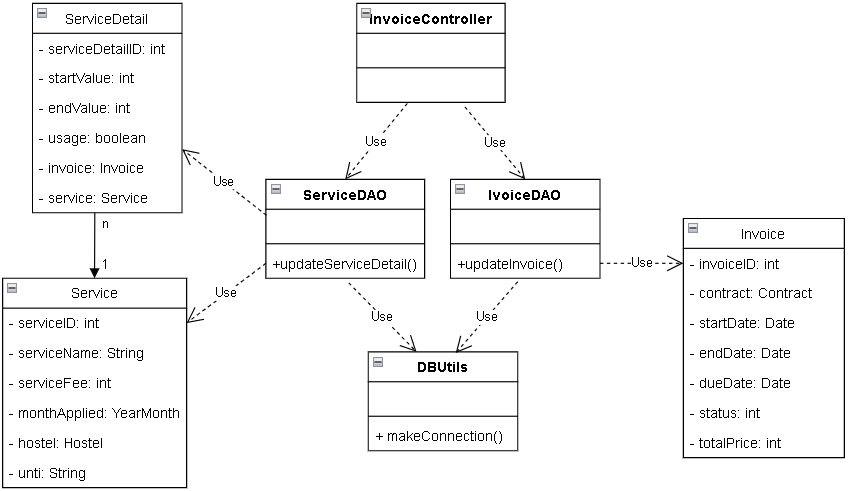


### d. Database queries

* *SELECT \* FROM Invoice Where invoiceId= ?*

## 29. Edit a payment invoice (Landlord feature)

### a. Class Diagram



### 

### b. Class Specifications

#### InvoiceController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters, set request attribute, process after receiving request from user and call functions at InvoiceDAO. |

#### InvoiceDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateInvoice() | update table Invoice; parameter is an Invoice object |

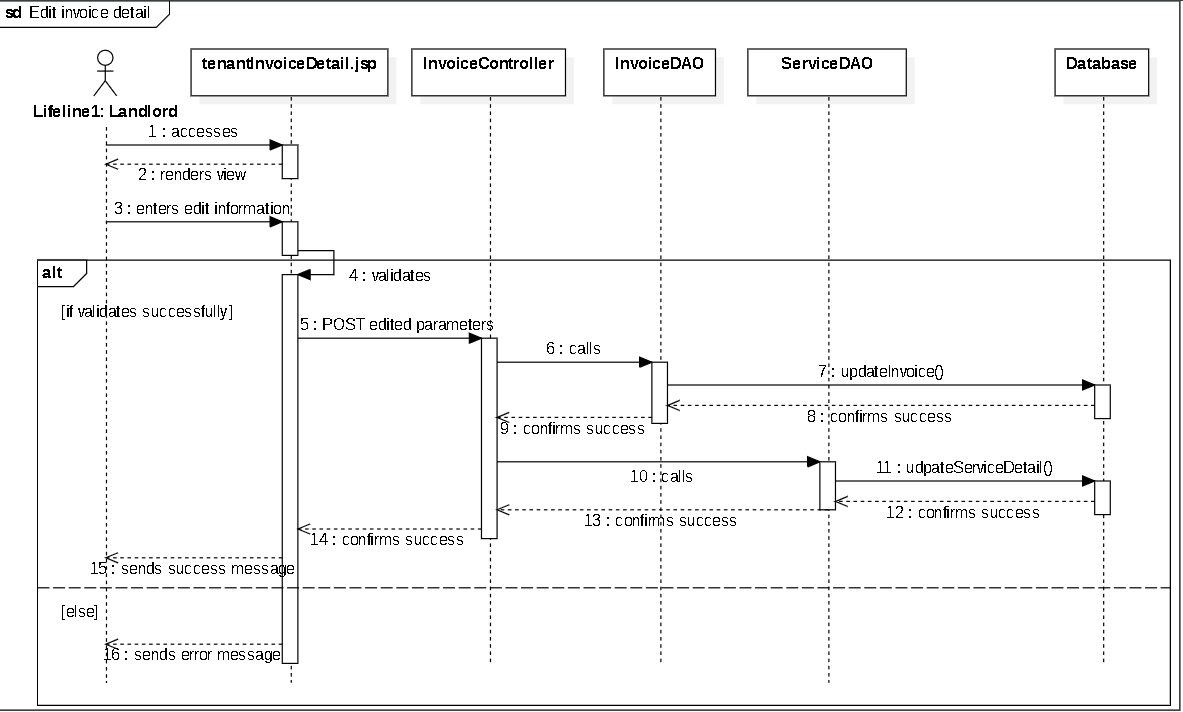
#### ServiceDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | updateServiceDetail() | update table ServiceDetail; parameters is ServiceDetail object |

#### DBUtils (Database)

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

**c. Sequence Diagram**

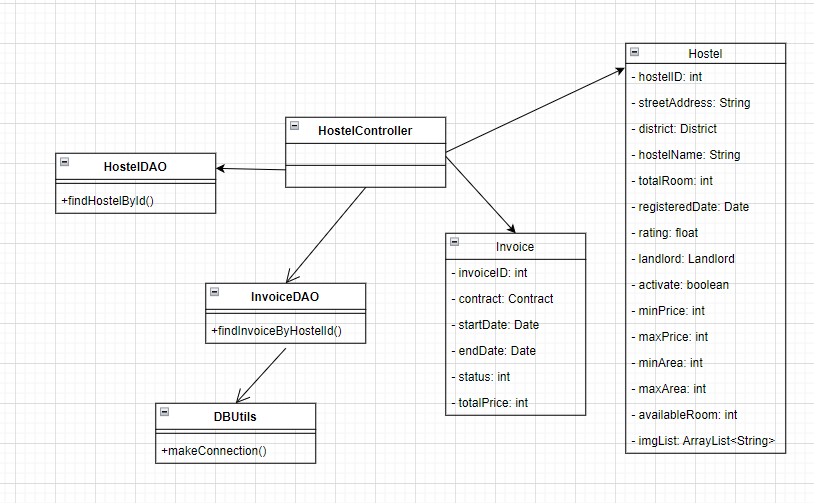
**  
d. Database queries**

* *update Invoice set startDate = ?, endDate = ?, status = ?, totalPrice = ?, dueDate = ? where invoiceID = ?*
* *update ServiceDetail set startValue = ?, endValue = ? where serviceDetailId*

## 

## 30**. Statistic** (Landlord feature)

### a. Class Diagram



### b. Class Specifications

#### HostelContontroller

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters such as hostelId, set request attribute, process after receiving request from user and call functions at HostelDAO or InvoiceDAO. |

#### HostelDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findHostelById | Returns the hostel. Input is hostelId. |

#### InvoiceDAO

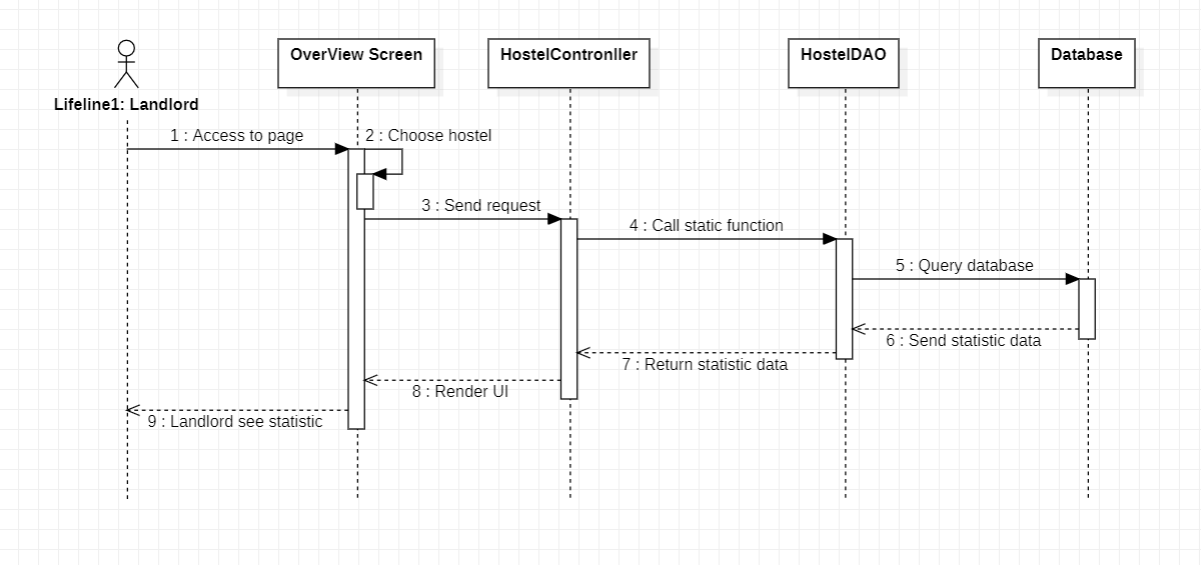
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findInvoiceByHostelId | Returns the invoice list. Input is hostelId. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | There is no input. Create connection with database, return a connection. |

### 

### c. Sequence Diagram

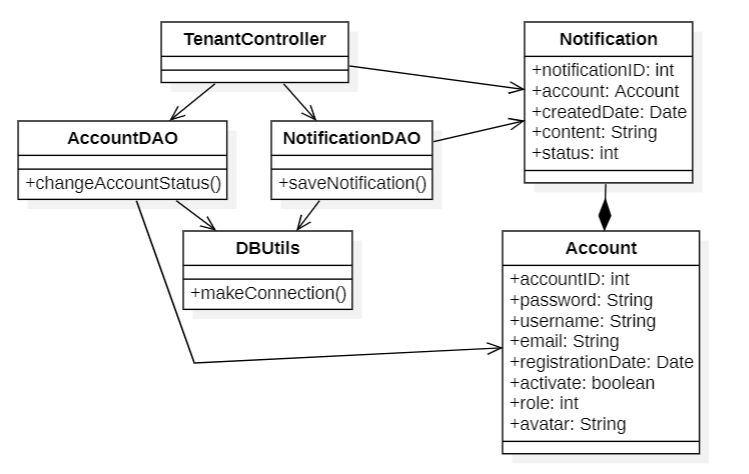


### d. **Database queries**

* *SELECT \* FROM Hostel WHERE landlordID = ?*
* *SELECT \* FROM Invoice WHERE contractID = ?*

## 31. Activate/deactivate tenant account(admin feature)

### a. Class Diagram



### b. Class Specifications

#### TenantController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Change tenant status and send a notification to admin account |

#### AccountDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | changeAccountStatus | Input account, updating account status and return a boolean |

#### NotificationDAO

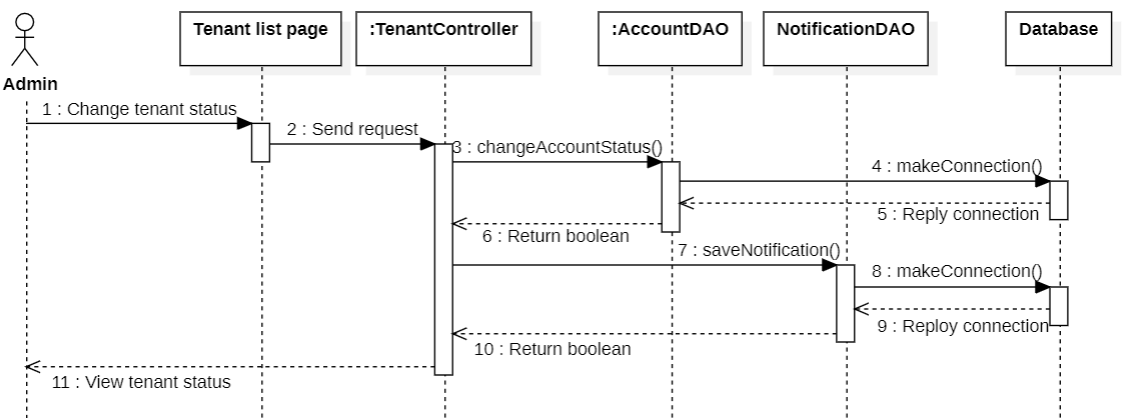
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | saveNotification() | Input a notification, insert notification and return a boolean |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | No input, void function, connect to the database |

#### 

### c. Sequence Diagram

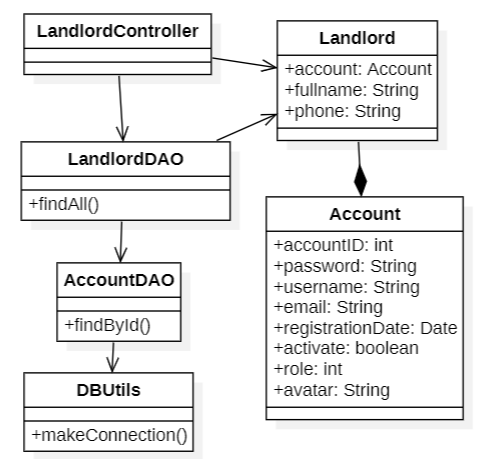


### d. Database queries

* *Update Account set status = 0 where accountID = ?*
* *Insert into Notification (toAccountID, createdDate, content, status) values (?, ?, ?, ?)*

## 32. View landlord list with paging(admin feature)

### a. Class Diagram



### 

### 

### 

### 

### b. Class Specifications

#### LandlordController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get full landlord list from database |

#### LandlordDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findAll() | No input, execute query and return a list of landlords |

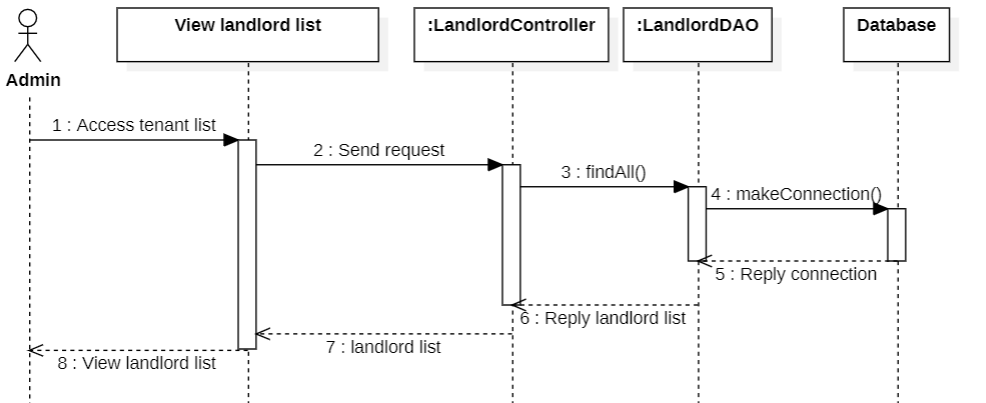
#### AccountDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findById() | Input accountId, execute query and return an account |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | No input, void function, connect to the database |

### c. Sequence Diagram

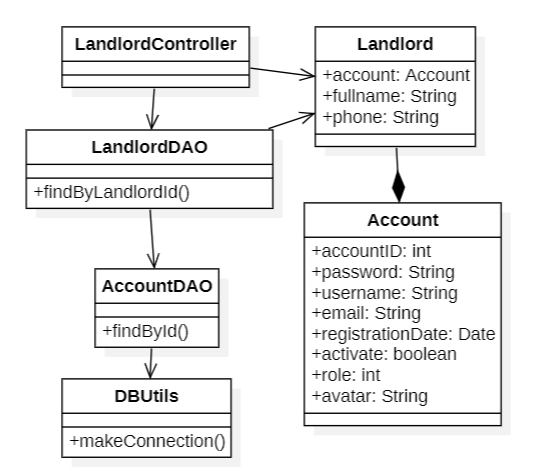


### d. Database queries

* *Select \* from Landlord*
* *Select \* from Account where AccountID = ?*

## 33. View landlord detail (admin feature)

### a. Class Diagram



### b. Class Specifications

#### LandlordController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get a landlord from database based on landlord id parameters |

#### LandlordDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findByLandlordId() | Input accountId, execute query and return a landlord |

#### AccountDAO

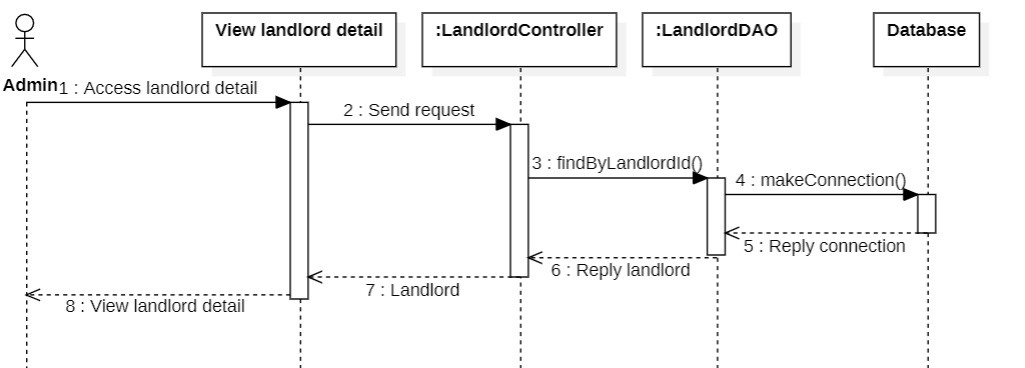
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findById() | Input accountId, execute query and return an account |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | No input, void function, connect to the database |

#### 

### c. Sequence Diagram

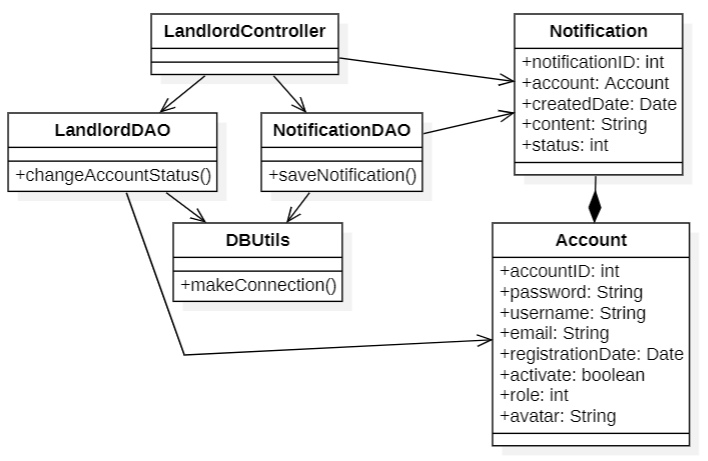


### d. Database queries

* *Select \* from Landlord where landlordId = ?*
* *Select \* from Account where accountId = ?*

## 34. Activate/deactivate landlord account (admin feature)

### a. Class Diagram



### b. Class Specifications

#### LandlordController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Change landlord status and send a notification to admin account |

#### AccountDAO

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | changeAccountStatus | Input account, updating account status and return a boolean |

#### NotificationDAO

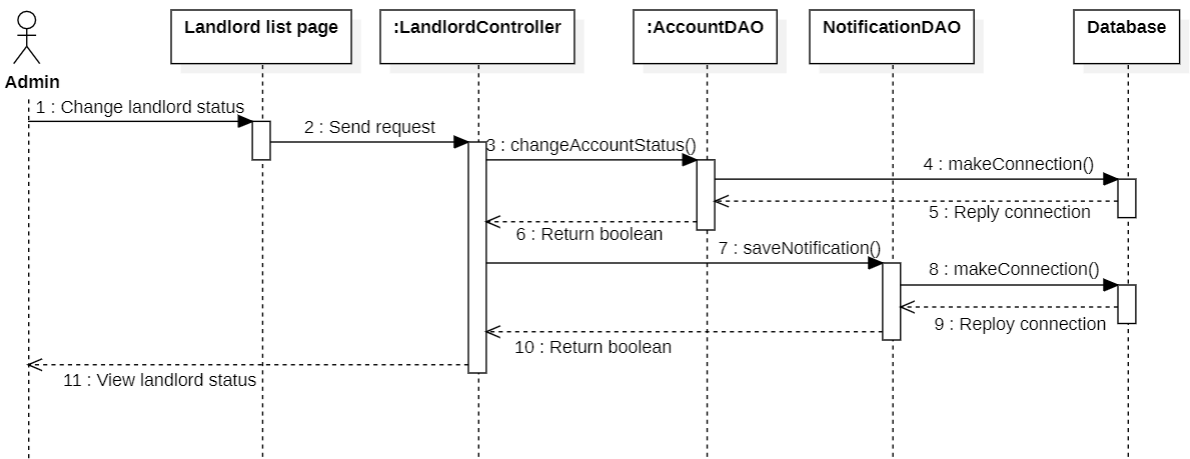
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | saveNotification() | Input a notification, insert notification and return a boolean |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | No input, void function, connect to the database |

#### 

### c. Sequence Diagram

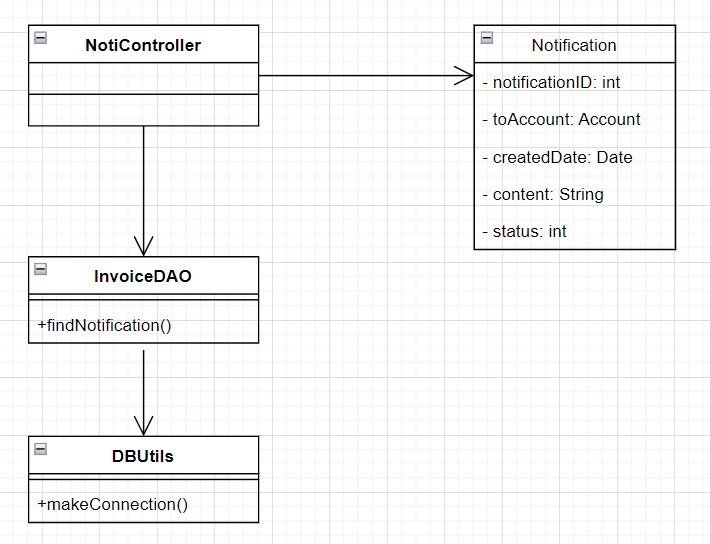


### **d. Database queries**

*Insert into Notification (toAccountID, createdDate, content, status) values (?, ?, ?, ?)*

## 35. View notifications

### a. Class Diagram



### b. Class Specifications

#### NotiController

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | processRequest() | Get request parameters, set request attribute, process after receiving request from user and call functions at NotiDAO. |

#### NotiDAO

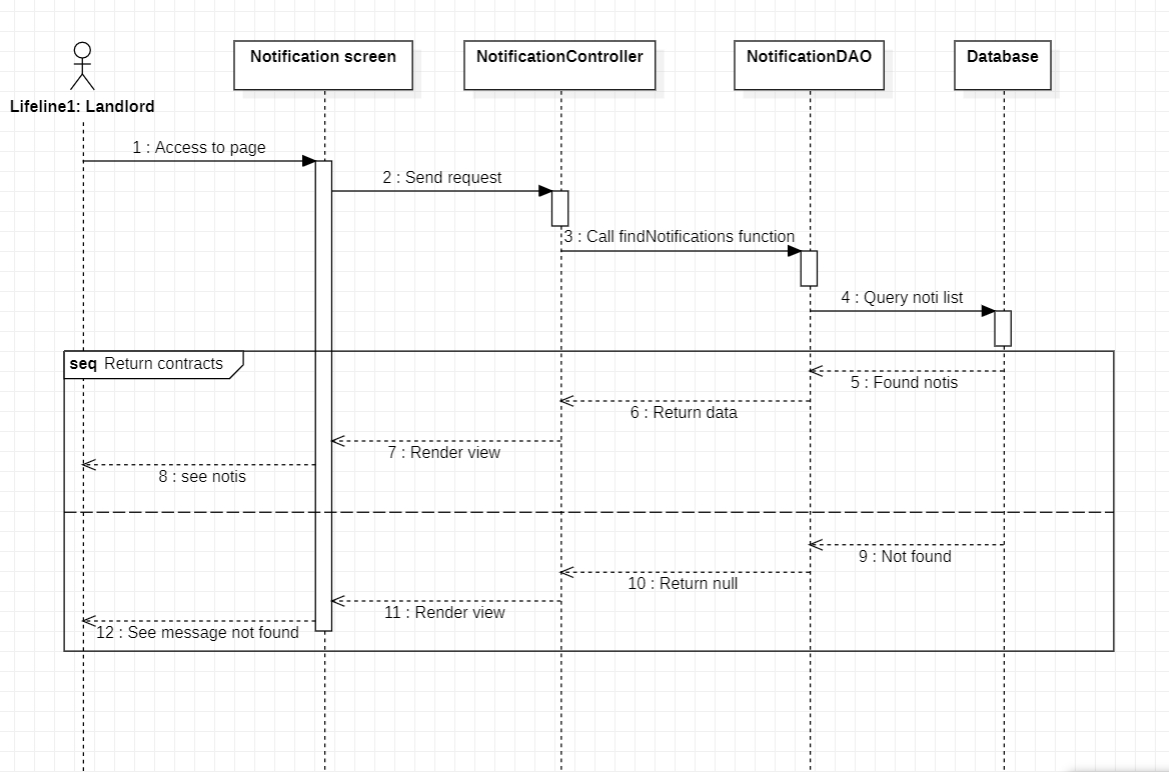
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | findNotification() | Returns detailed notification. Input is landlordId. |

#### DBUtils

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | makeConnection() | No input, void function, connect to the database |

#### 

### c. Sequence Diagram



.

### d. Database queries

* *SELECT \* FROM Notifications WHERE landlordId = ?*.

# III. Database Tables

## 1. Table Account

Generic information about a system user, such as username, email, password, phone, etc., that is shared by the three roles.

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | accountId | int |  | x | x | x |  |
| 2 | username | varchar(30) | 30 | x | x |  |  |
| 3 | password | varchar(15) | 15 |  | x |  |  |
| 4 | email | varchar(30) | 30 |  |  |  |  |
| 5 | registeredDate | Date |  |  | x |  |  |
| 6 | role | tinyint |  |  | x |  |  |
| 7 | activate | bit |  |  | x |  |  |
| 8 | avatar | varchar(MAX) |  |  |  |  |  |

## 2. Table Landlord

Detailed information about a landlord, such as landlordId, full name and phone

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | landlordId | int |  | x | x | x |  |
| 2 | fullname | nvarchar(30) | 30 |  |  |  |  |
| 3 | phone | varchar(15) | 15 | x |  |  |  |

## 

## 3. Table Tenant

Detailed information about a tenant, such as landlordId, full name and phone

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | tenantId | int |  | x | x | x |  |
| 2 | fullname | nvarchar(30) | 30 |  |  |  |  |
| 3 | phone | varchar(10) | 10 | x |  |  |  |
| 4 | rentStatus | bit |  |  | x |  |  |

## 

## 4. Table Hostel

Contains an overview of a hostel, such as size, cost, service, image. This information will be used to register for rent, managed by the host and admin. This table does not include detailed information of the rooms.

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | hostelId | int |  | x | x | x |  |
| 2 | streetAddress | nvarchar(30) | 30 |  | x |  |  |
| 3 | address | nvarchar(255) | 255 | x | x |  |  |
| 4 | hostelName | nvarchar(30) | 30 |  | x |  |  |
| 5 | totalRoom | int |  |  | x |  |  |
| 6 | registeredDate | date |  |  | x |  |  |
| 7 | activate | bit |  |  | x |  |  |
| 8 | rating | float |  |  |  |  |  |
| 9 | landlordId | int |  |  | x | x |  |
| 10 | description | nvarchar(MAX) |  |  |  |  |  |
| 11 | images | nvarchar(MAX) |  |  | x |  |  |
| 12 | minPrice | int |  |  | x |  |  |
| 13 | maxPrice | int |  |  | x |  |  |
| 14 | minArea | int |  |  | x |  |  |
| 15 | maxArea | int |  |  | x |  |  |
| 16 | districtID | int |  |  | x | x |  |
| 17 | availableRoom | int |  |  | x |  |  |

## 

## 5. Table Room

Detailed information about a certain room, along with some further details on the state of the rental, number of residents.

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | roomId | int |  | x | x | x |  |
| 2 | roomTypeId | int |  |  | x | x |  |
| 3 | roomNumber | nvarchar(255) | 255 |  | x |  |  |
| 4 | currentNoResidents | int |  |  | x |  |  |
| 5 | status | int |  |  | x |  |  |

## 

## 6. Table RoomResident

The hostel offers a variety of services, including the name of the service, the fee for that service, and the unit size for that service. This information is from the landlord, and is used to create the monthly rent invoice for tenants.

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | roomResidentId | int |  | x | x | x |  |
| 2 | roomId | int |  |  | x | x |  |
| 3 | fullname | nvarchar(30) | 30 |  | x |  |  |
| 4 | phone | char(10) | 10 |  | x |  |  |
| 5 | dob | date |  |  | x |  |  |
| 6 | isRoomLead | bit |  |  | x |  |  |

## 7. Table Feedback

Store ratings and comments made by tenants for the hostel they rent; visitors looking to book a room may read these reviews.

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | feedbackId | int |  | x | x | x |  |
| 2 | hostelId | int |  |  | x | x |  |
| 3 | tenantId | int |  |  | x | x |  |
| 4 | content | nvarchar(MAX) |  |  | x |  |  |
| 5 | rating | tinyint |  |  | x |  |  |
| 6 | date | date |  |  | x |  |  |

## 8. Table Notification

Various system notifications, including requests from tenants, system notifications, and announcements from the landlord, among others.

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | notificationId | int |  | x | x | x |  |
| 2 | toAccountId | int |  |  | x | x |  |
| 3 | createdDate | date |  |  | x |  |  |
| 4 | content | nvarchar(MAX) | 4000 |  | x |  |  |
| 5 | status | int |  |  | x |  |  |

## 9. Table Service

Includes a summary of the services offered by a hostel.

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | serviceId | int |  | x | x | x |  |
| 2 | serviceName | nvarchar(30) | 30 |  | x |  |  |
| 3 | serviceFee | int |  |  | x |  |  |
| 4 | monthApplied | date |  |  | x |  |  |
| 5 | hostelId | int |  |  | x | x |  |
| 6 | unit | nvarchar(20) | 20 |  | x |  |  |
| 7 | active | bit |  |  | x |  |  |

## 10. Table ServiceDetail

Information in-depth about the hostel's services, including the name of the service, the cost, and the unit. The landlord updates this data, which is used to produce the renters' monthly invoice.

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | serviceDetailId | int |  | x | x | x |  |
| 2 | startValue | real |  |  |  |  |  |
| 3 | endValue | real |  |  |  |  |  |
| 4 | quantity | real |  |  | x |  |  |
| 5 | invoiceId | int |  |  | x | x |  |
| 6 | serviceId | int |  |  | x | x |  |

## 11. Table FavoriteHostel

Users can review the list of hostels that have been designated as favorites by clicking on the favorite button for a hostel, which saves the hostel's information in this table.

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | favoriteHostelID | int |  | x | x | x |  |
| 2 | hostelID | int |  |  | x | x |  |
| 3 | tenantID | int |  |  | x | x |  |

## 12. Table Contract

Information about the rental room contract between the landlord and tenant. This table may be used to determine the contract's start and end dates as well as the rental cost and deposit. The contract is used to calculate the invoices as well (rental fee).

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | contractID | int |  | x | x | x |  |
| 2 | roomID | int |  |  | x | x |  |
| 3 | tenantID | int |  |  | x | x |  |
| 4 | startDate | date |  |  | x |  |  |
| 5 | endDate | date |  |  |  |  |  |
| 6 | deposit | int |  |  | x |  |  |
| 7 | status | tinyint |  |  | x |  |  |
| 8 | rentalFeePerMonth | int |  |  | x |  |  |

## 13. Table Invoice

Details about a monthly rental room invoice, including the rental cost, service charges, and information about the payment schedule.

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | invoiceID | int |  | x | x | x |  |
| 2 | contractID | int |  |  | x | x |  |
| 3 | startDate | date |  |  | x |  |  |
| 4 | endDate | date |  |  | x |  |  |
| 5 | status | tinyint |  |  | x |  |  |
| 6 | totalPrice | int |  |  | x |  |  |
| 7 | dueDate | date |  |  |  |  |  |

## 14. District

## Save district information such as district name and districtId

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | districtID | int |  | x | x | x |  |
| 2 | districtName | nvarchar(50) | 50 |  | x |  |  |
| 3 | provinceID | int |  |  | x | x |  |

## 

## 15. Province

Save province information such as province name and provinceId

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | provinceID | int |  | x | x | x |  |
| 2 | provinceName | nvarchar(50) | 50 |  | x |  |  |

## 

## 16. GoogleAccount

Save google token in database

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | googleID | int |  | x | x | x |  |
| 2 | googleToken | nvarchar(50) | 50 | x | x | x |  |

## 

## 17. ImageHostel

Save images of hostel with imageHostelTypeID, hostelID, imgLink as information

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | imageHostelID | int |  | x | x | x |  |
| 2 | hostelID | int |  |  | x | x |  |
| 3 | imgLink | varchar(255) | 255 |  | x |  |  |

## 

## 18. RoomType

In order to bring together identical rooms in a hostel that are similar in size or cost, we'll name them "types." There will be a variety of rooms within a room type.

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | roomTypeId | int |  | x | x | x |  |
| 2 | roomTypeName | nvarchar(255) | 255 |  | x |  |  |
| 3 | roomTypeArea | int |  |  | x |  |  |
| 4 | description | nvarchar(255) | 255 |  | x |  |  |
| 5 | maxNumberOfResidents | int |  |  | x |  |  |
| 6 | advertisedPrice | int |  |  | x |  |  |
| 7 | hostelID | int |  |  | x | x |  |

## 19. ImageRoomType

Save images of room type with imageRoomTypeID, roomTypeID, imgLink as information

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | imageRoomTypelID | int |  | x | x | x |  |
| 2 | roomTypeID | int |  |  | x | x |  |
| 3 | imgLink | varchar(255) | 255 |  | x |  |  |

## 