LHospital

A *Turbo* C++ project
A basic management system for a general hospital

Individual contributions to the project by:
Anirudh Panigrahi 9151993

Contents

C++ files (.cpp)

1. code/iface3.cpp

```
#include <fstream.h>
#include "base.hpp"
#include "iface.hpp"
#include "hosp.hpp"
#include "emp.hpp"

void interface::employee_management()
{
    const int menu_corner_top_left_y = 5;
    coord c(ui::scr_width * 0.2, menu_corner_top_left_y);
    int ch;
    while(1)
    {
        interface::clear_error();
        box menu(c, ui::scr_width * 0.6, ui::scr_height - 6 );
        menu.settcolor(GREEN);
        menu << ui::centeralign << "Employee Management" << ui::endl << ui::endl;
        menu.settcolor(ui::tcolor);
        menu << "1. View employee data" << ui::endl
             << "2. Add new employee" << ui::endl
             << "3. Remove existing employee" << ui::endl
             << "4. Edit employee data" << ui::endl
             << "5. Pay salary to individual employee" << ui::endl
             << "6. Pay salary to all employees" << ui::endl
             << "7. Back" << ui::endl
             << ui::endl << "Enter your choice: ";
        menu.settcolor_input(YELLOW);
        validate_menu::set_menu_limits(1, 7);
        menu >> validate_menu::input >> ch;
        menu << ui::endl;
        menu.setexit_button("Submit");
        menu.loop();
        menu.hide();
        switch (ch)
        {
            case 1:
            {
                emp_mgmt::view_emp();
                break;
            }
            case 2:
            {
                emp_mgmt::add_emp();
                break;
            }
            case 3:
            {
                emp_mgmt::remove_emp();
                break;
            }
            case 4:
            {
                emp_mgmt::edit_emp();
                break;
```

```
55              }
56              case 5:
57              {
58                  emp_mgmt::pay_emp();
59                  break;
60              }
61              case 6:
62              {
63                  emp_mgmt::pay_all();
64                  break;
65              }
66              case 7:
67              {
68                  return;
69              }
70          }
71      }
72  }
73
74  void interface::employee_screen(unsigned long id)
75  {
76      void * temp = malloc( sizeof(doctor) ); //as doctor has the greatest size
            among employee, doctor, nurse and receptionist classes
77      if(temp == NULL)
78      {
79          interface::log_this("interface::employee_screen() : Not enough memory to
                allocate buffer void * temp = malloc( sizeof(doctor) )");
80          interface::error("Out of memory!! Check log");
81          getch();
82          return;
83      }
84      if(!hospital::get_employee_by_id(id, temp))
85      {
86          interface::error("ID not found or error while reading from file!");
87          getch();
88          free(temp);
89          return;
90      }
91      employee *e = (employee *) temp;
92      const int menu_corner_top_left_y = 5;
93      coord c(ui::scr_width * 0.2, menu_corner_top_left_y);
94      int ch;
95      str heading = "Welcome, ";
96      strcat( heading, e->get_name() );
97      strcat(heading, "!");
98      while(1)
99      {
100         interface::clear_error();
101         box menu(c, ui::scr_width * 0.6, ui::scr_height - 6 );
102         menu.settcolor(GREEN);
103         menu << ui::centeralign << heading << ui::endl << ui::endl;
104         menu.settcolor(ui::tcolor);
105         menu << "1. View profile" << ui::endl
106              << "2. Change login details" << ui::endl
107              << "3. View last 5 transactions" << ui::endl;
108         emp_type type_of_emp = id_to_emp::convert(id);
109         if(type_of_emp == RECEPTIONIST)
110         {
111             menu << "4. Manage patients" << ui::endl
```

```
                << "5. Exit" << ui::endl;
        }
        else
        {
            menu << "4. Exit" << ui::endl;
        }
        menu << ui::endl << "Enter your choice: ";
        menu.settcolor_input(YELLOW);
        if(type_of_emp == RECEPTIONIST)
        {
            validate_menu::set_menu_limits(1, 5);
        }
        else
        {
            validate_menu::set_menu_limits(1, 4);
        }
        menu >> validate_menu::input >> ch;
        menu << ui::endl;
        menu.setexit_button("Submit");
        menu.loop();
        menu.hide();
        switch(ch)
        {
            case 1:
            {
                if( !emp_mgmt::view_emp(id) )
                {
                    interface::error("Failed to display profile!");
                    getch();
                }
                break;
            }
            case 2:
            {
                int ch;
                while(1)
                {
                    box menu3(c, ui::scr_width * 0.6, ui::scr_height - 6);
                    menu3.settcolor(GREEN);
                    menu3 << ui::centeralign << "Change login details" << ui::
                        endl << ui::endl;
                    menu3.settcolor(WHITE);
                    menu3 << "1. Change User ID" << ui::endl
                          << "2. Change Password" << ui::endl
                          << "3. Back" << ui::endl
                          << "Enter your choice: ";
                    menu3.settcolor(ui::tcolor);
                    menu3.settcolor_input(YELLOW);
                    validate_menu::set_menu_limits(1, 3);
                    menu3 >> validate_menu::input >> ch;
                    menu3 << ui::endl;
                    menu3.setexit_button("Submit");
                    menu3.loop();
                    menu3.hide();
                    switch(ch)
                    {
                        case 1:
                        {
                            str new_username;
```

```
170                         box menu4( menu3.getcorner_top_left(), menu3.getwidth
                                (), menu3.getheight() );
171                         menu4.settcolor(GREEN);
172                         menu4 << ui::centeralign << "Change login details" <<
                                 ui::endl << ui::endl;
173                         menu4.settcolor(WHITE);
174                         menu4 << "Change User ID" << ui::endl;
175                         menu4.settcolor(ui::tcolor);
176                         menu4 << "User ID: ";
177                         menu4.setdefault( e->account.get_username() );
178                         menu4.settcolor_input(YELLOW);
179                         menu4 >> new_username;
180                         menu4.setexit_button("Submit");
181                         menu4.setback_func(back_func::set_backbit);
182                         menu4.loop();
183                         menu4.hide();
184                         if(back_func::backbit)
185                         {
186                             back_func::backbit = 0;
187                             break;
188                         }
189                         e->account.set_username(new_username);
190                         const int notice_height = 10;
191                         box notice( menu4.getcorner_top_left(), menu4.
                                getwidth(), notice_height );
192                         notice.settcolor(GREEN);
193                         notice << ui::centeralign << "Change login details"
                                << ui::endl << ui::endl;
194                         if( !hospital::write_employee(temp) )
195                         {
196                             notice.settcolor(RED);
197                             notice << "Failed to write new user ID to file!
                                    Check log" << ui::endl;
198                         }
199                         else
200                         {
201                             notice.settcolor(GREEN);
202                             notice << "User ID changed successfully!" << ui::
                                    endl;
203                         }
204                         notice.setexit_button("Back");
205                         notice.loop();
206                         notice.hide();
207                         goto loop_exit;
208                     }
209                     case 2:
210                     {
211                         str curr_pwd, new_pwd;
212                         for(int i = 0; i < 3; ++i)
213                         {
214                             box menu4( menu3.getcorner_top_left(), menu3.
                                    getwidth(), menu3.getheight() );
215                             menu4.settcolor(GREEN);
216                             menu4 << ui::centeralign << "Change login details
                                    " << ui::endl << ui::endl;
217                             menu4.settcolor(WHITE);
218                             menu4 << "Change Password" << ui::endl;
219                             menu4.settcolor(ui::tcolor);
220                             menu4 << "Enter current password: ";
```

```
221                             menu4.settcolor_input(YELLOW);
222                             menu4 >> box::setpassword >> curr_pwd;
223                             menu4.setexit_button("Submit");
224                             menu4.setback_func(back_func::set_backbit);
225                             menu4.loop();
226                             menu4.hide();
227                             if(back_func::backbit)
228                             {
229                                 break;
230                             }
231                             if( e->account.login(curr_pwd) )
232                             {
233                                 interface::clear_error();
234                                 break;
235                             }
236                             interface::error("Invalid password!! Try again...
                                    ");
237                         }
238                         if(back_func::backbit)
239                         {
240                             back_func::backbit = 0;
241                             break;
242                         }
243                         if(i == 3)
244                         {
245                             const int notice_height = 10;
246                             box notice( menu3.getcorner_top_left(), menu3.
                                    getwidth(), notice_height);
247                             notice.settcolor(GREEN);
248                             notice << ui::centeralign << "Change login
                                    details" << ui::endl << ui::endl;
249                             notice.settcolor(RED);
250                             notice << "Since you entered the wrong password
                                    too many times, you have been logged out. "
251                                    << "Hit the button below to exit the
                                        program." << ui::endl << ui::endl;
252                             notice.setexit_button("Exit");
253                             notice.loop();
254                             notice.hide();
255                             free(temp);
256                             return;
257                         }
258                         box menu5( menu3.getcorner_top_left(), menu3.getwidth
                                (), menu3.getheight() );
259                         menu5.settcolor(GREEN);
260                         menu5 << ui::centeralign << "Change login details" <<
                                 ui::endl << ui::endl;
261                         menu5.settcolor(WHITE);
262                         menu5 << "Change Password" << ui::endl;
263                         menu5.settcolor(ui::tcolor);
264                         menu5 << "Enter new password: ";
265                         menu5.settcolor_input(YELLOW);
266                         menu5 >> box::setpassword >> new_pwd;
267                         menu5.setexit_button("Submit");
268                         menu5.setback_func(back_func::set_backbit);
269                         menu5.loop();
270                         menu5.hide();
271                         if(back_func::backbit)
272                         {
```

```
273                                 back_func::backbit = 0;
274                                 break;      //At the "Enter new password" page,
                                        when shift+bkspc is pressed, control will go
                                        back to "Change login details" menu.
275                             }
276                             e->account = userid( e->account.get_username(),
                                    new_pwd );
277                             const int notice2_height = 13;
278                             box notice2( menu3.getcorner_top_left(), menu3.
                                    getwidth(), notice2_height );
279                             notice2.settcolor(GREEN);
280                             notice2 << ui::centeralign << "Change login details"
                                    << ui::endl << ui::endl;
281                             if( !hospital::write_employee(temp) )
282                             {
283                                 notice2.settcolor(RED);
284                                 notice2 << "Failed to write new password to file!
                                        Check log" << ui::endl;
285                             }
286                             else
287                             {
288                                 notice2.settcolor(GREEN);
289                                 notice2 << "Password changed successfully!" << ui
                                        ::endl;
290                             }
291                             notice2.settcolor(ui::tcolor);
292                             notice2 << "Please logout and login again by exiting
                                    the program and restarting it." << ui::endl
293                                 << "Press the button below to exit the
                                        program." << ui::endl;
294                             notice2.setexit_button("Exit");
295                             notice2.loop();
296                             notice2.hide();
297                             free(temp);
298                             return;
299                         }
300                         case 3:
301                         {
302                             goto loop_exit;
303                         }
304                     }
305                 }
306                 loop_exit:
307                 break;
308             }
309             case 3:
310             {
311                 transaction * t = e->get_last_5_transactions();
312                 if( t == NULL )
313                 {
314                     interface::error("Error while reading or writing to file!
                            Check log");
315                     getch();
316                     break;
317                 }
318                 coord c2(1, 4);
319                 box menu2(c2, (ui::scr_width / 2), ui::scr_height - 5);
320                 box sidemenu(( c2 + coord((ui::scr_width / 2) - 1, 0)), (ui::
                        scr_width / 2) + 1, ui::scr_height - 5);
```

```
321                 menu2.f << ( ui::top | ui::left ) << (char)204
322                         << ( ui::bottom | ui::left ) << (char)204
323                         << ( ui::top | ui::right ) << (char)203
324                         << ( ui::bottom | ui::right ) << (char)202;
325                 menu2.f.display();
326                 sidemenu.f << ( ui::top | ui::left ) << (char)203
327                            << ( ui::bottom | ui::left ) << (char)202
328                            << ( ui::top | ui::right ) << (char)185
329                            << ( ui::bottom | ui::right ) << (char)185;
330                 sidemenu.f.display();
331                 menu2.settcolor(GREEN);
332                 menu2 << ui::centeralign << "View last 5 transactions" << ui::
                        endl << ui::endl;
333                 menu2.settcolor(ui::tcolor);
334                 for(int i = 0; i < 5; ++i)
335                 {
336                     if( t[i].amount == 0 && !strcmp(t[i].reason, "NA") &&
337                         t[i]._date.day == 0 && t[i]._date.month == 0 && t[i].
                                _date.year == 0
338                         && t[i]._time.hour == 25 && t[i]._time.minute == 0 && t[i
                                ]._time.second == 0 )
339                     {
340                         break;
341                     }
342                     if(i < 3)
343                     {
344                         menu2 << i + 1 << ". " << t[i]._date << ", " << t[i].
                                _time << ui::endl
345                               << "Amount: " << t[i].amount << ui::endl
346                               << "Reason: " << t[i].reason << ui::endl;
347                     }
348                     else
349                     {
350                         sidemenu << i + 1 << ". " << t[i]._date << ", " << t[i].
                                _time << ui::endl
351                                  << "Amount: " << t[i].amount << ui::endl
352                                  << "Reason: " << t[i].reason << ui::endl;
353                     }
354                 }
355                 free(t);
356                 if(i <= 3)
357                 {
358                     menu2.setexit_button("Back");
359                     menu2.loop();
360                 }
361                 else
362                 {
363                     sidemenu.setexit_button("Back");
364                     sidemenu.loop();
365                 }
366                 menu2.hide();
367                 sidemenu.hide();
368                 window.f.display();
369                 break;
370             }
371             case 4:
372             {
373                 if(type_of_emp == RECEPTIONIST)
374                 {
```

```
                    interface::patient_management();
                    break;
                }
                else
                {
                    free(temp);
                    return;
                }
            }
            case 5:
            {
                free(temp);
                return;
            }
        }
    }
 }

emp_mgmt::emp_mgmt()
{}

void emp_mgmt::view_emp()
{
    const int menu2_height = 10;
    box menu2( coord(ui::scr_width * 0.2, 5), ui::scr_width * 0.6, menu2_height);
    menu2.settcolor(GREEN);
    menu2 << ui::centeralign << "Employee Management" << ui::endl << ui::endl;
    menu2.settcolor(WHITE);
    menu2 << "View employee data" << ui::endl;
    menu2.settcolor(ui::tcolor);
    menu2 << "Enter employee's id: ";
    unsigned long id;
    menu2.settcolor_input(YELLOW);
    menu2 >> id;
    menu2 << ui::endl;
    menu2.setexit_button("Submit");
    menu2.setback_func(back_func::set_backbit);
    menu2.loop();
    menu2.hide();
    if(back_func::backbit)
    {
        back_func::backbit = 0;
        return;
    }
    view_emp(id);
}

int emp_mgmt::view_emp(unsigned long id)
{
    void * temp = malloc( sizeof(doctor) ); //as doctor has the greatest size
        among employee, doctor, nurse and receptionist classes
    if(temp == NULL)
    {
        interface::log_this("emp_mgmt::view_emp(int) : Not enough memory to
            allocate buffer void * temp = malloc( sizeof(doctor) )");
        interface::error("Out of memory!! Check log");
        getch();
        return 0;
    }
```

```
    if(!hospital::get_employee_by_id(id, temp))
    {
        interface::error("ID not found or error while reading from file!");
        getch();
        free(temp);
        return 0;
    }
    employee *e = (employee *) temp;
    box menu3( coord(ui::scr_width * 0.2, 5), ui::scr_width * 0.6, ui::scr_height
        - 6 );
    menu3.settcolor(GREEN);
    menu3 << ui::centeralign << "Employee Management" << ui::endl << ui::endl;
    menu3.settcolor(WHITE);
    menu3 << "Employee Details: " << ui::endl;
    menu3.settcolor(ui::tcolor);
    menu3 << "ID: " << e->get_id() << ui::endl;
    menu3 << "Name: " << e->get_name() << ui::endl;
    menu3 << "Age: " << e->get_age() << ui::endl;
    menu3 << "Sex: " << (sex)e->get_sex() << ui::endl;
    menu3 << "Date of Birth: " << e->get_dob() << ui::endl;
    menu3 << "Address: " << e->get_address() << ui::endl;
    menu3 << "Phone no.: " << e->get_phone() << ui::endl;
    menu3 << "Salary: " << e->get_salary() << ui::endl;
    menu3 << "Shift timings: Starts - " << e->get_shift(START) << ui::endl;
    menu3 << "---------------: Ends - " << e->get_shift(END) <<ui::endl;
    switch( id_to_emp::convert( e->get_id() ) )
    {
        case INVALID:   //Test this case, menu3.hide() not working properly
        {
            menu3.clear();
            int menu3_height = 9;
            menu3.setheight(menu3_height);
            menu3.settcolor(GREEN);
            menu3 << ui::centeralign << "Employee Management" << ui::endl << ui::
                endl;
            menu3.settcolor(WHITE);
            menu3 << "Employee Details: " << ui::endl;
            menu3.settcolor(RED);
            menu3 << "Invalid ID!!" << id_to_emp::convert( e->get_id() );
            menu3.settcolor(ui::tcolor);
            menu3.setexit_button("Back");
            menu3.loop();
            menu3.hide();
            break;
        }
        case OTHERS:
        case RECEPTIONIST:  //there are no extra data members in class
            receptionist
        {
            menu3.setexit_button("Back");
            menu3.loop();  //   menu3.clear();  int w = window.getwidth(), m =
                menu3.getwidth(); menu3<<w<<' '<<m;  getch();
            menu3.hide();
            break;
        }
        case DOCTOR:
        {
            doctor *d = (doctor *)temp;
            menu3.hide();
```

```
487             menu3.setcorner_top_left( coord( 1, menu3.getcorner_top_left().y ) );
488             menu3.display();
489             menu3.f << ( ui::top | ui::left ) << (char)204
490                     << ( ui::bottom | ui::left ) << (char)204;
491             menu3.f.display();
492             box sidemenu( menu3.getcorner_top_left() + coord( menu3.getwidth() -
                    1, 0 ), ( ui::scr_width - menu3.getwidth() + 1 ),  menu3.getheight
                    () );
493             sidemenu.f << ( ui::top | ui::left ) << (char)203
494                        << ( ui::bottom | ui::left ) << (char)202
495                        << ( ui::top | ui::right ) << (char)185
496                        << ( ui::bottom | ui::right ) << (char)185;
497             sidemenu.f.display();
498             sidemenu << "Speciality(s)" << ui::endl;
499             for(int i = 0; i < 2 && d->get_speciality()[i] <= GEN; ++i)
500             {
501                 sidemenu << i + 1 << ". " << (body_parts)d->get_speciality()[i]
                        << ui::endl;
502             }
503             if(!i)
504             {
505                 sidemenu << "None" << ui::endl;
506             }
507             sidemenu << "Patients currently under care:" << ui::endl;
508             for(i = 0; d->get_patients()[i] && i < 10; ++i)
509             {
510                 sidemenu << i + 1 << ". " << hospital::get_patient_by_id( d->
                        get_patients()[i] ).get_name() << ui::endl;
511             }
512             if(!i)
513             {
514                 sidemenu << "None" << ui::endl;
515             }
516             sidemenu.setexit_button("Back");
517             sidemenu.loop();
518             menu3.hide();
519             sidemenu.hide();
520             window.f.display();
521             break;
522         }
523         case NURSE:
524         {
525             nurse *n = (nurse *)temp;
526             menu3.hide();
527             menu3.setcorner_top_left( coord( 1, menu3.getcorner_top_left().y ) );
528             menu3.display();
529             menu3.f << ( ui::top | ui::left ) << (char)204
530                     << ( ui::bottom | ui::left ) << (char)204;
531             menu3.f.display();
532             box sidemenu( menu3.getcorner_top_left() + coord( menu3.getwidth() -
                    1, 0 ), ( ui::scr_width - menu3.getwidth() + 1 ),  menu3.getheight
                    () );
533             sidemenu.f << ( ui::top | ui::left ) << (char)203
534                        << ( ui::bottom | ui::left ) << (char)202
535                        << ( ui::top | ui::right ) << (char)185
536                        << ( ui::bottom | ui::right ) << (char)185;
537             sidemenu.f.display();
538             sidemenu << "Patients currently under care:" << ui::endl;
539             for(int i = 0; n->get_patients()[i] && i < 5; ++i)
```

```
            {
                sidemenu << i + 1 << ". " << hospital::get_patient_by_id( n->
                    get_patients()[i] ).get_name() << ui::endl;
            }
            if(!i)
            {
                sidemenu << "None" << ui::endl;
            }
            sidemenu.setexit_button("Back");
            sidemenu.loop();
            menu3.hide();
            sidemenu.hide();
            window.f.display();
            break;
        }
    }
    free(temp);
    return 1;
}

void emp_mgmt::add_emp()
{
    int ch;
    str name, dob_str, adr_hno, adr_street, adr_city, adr_dist, adr_state,
        shift_start_str, shift_end_str, uid, pwd;
    unsigned sex_choice;
    Date dob;
    address adr;
    phone phn_no;
    unsigned long salary;
    Time shift_start, shift_end;
    int speciality[2];
    const coord menu2_corner_top_left = coord(ui::scr_width * 0.2, 5);
    const int menu2_width = ui::scr_width * 0.6;
    menu2:
    {
        const int menu2_height = 17;
        box menu2(menu2_corner_top_left, menu2_width, menu2_height);
        menu2.settcolor(GREEN);
        menu2 << ui::centeralign << "Employee Management" << ui::endl << ui::endl
            ;
        menu2.settcolor(WHITE);
        menu2 << "Add new employee" << ui::endl;
        menu2.settcolor(ui::tcolor);
        menu2 << "Step 1: Select employee type" << ui::endl << ui::endl
              << "1. Doctor" << ui::endl
              << "2. Nurse" << ui::endl
              << "3. Receptionist" << ui::endl
              << "4. Others" << ui::endl << ui::endl
              << "Enter your choice: ";
        validate_menu::set_menu_limits(1, 4);
        menu2.settcolor_input(YELLOW);
        menu2 >> validate_menu::input >> ch;
        menu2 << ui::endl;
        menu2.setexit_button("Submit");
        menu2.setback_func(back_func::set_backbit);
        menu2.loop();
        menu2.hide();
        if(back_func::backbit)
```

```
596         {
597             back_func::backbit = 0;
598             return;
599         }
600     }
601     menu3:
602     {
603         box menu3( menu2_corner_top_left, menu2_width, ui::scr_height - 6 );
604         menu3.settcolor(GREEN);
605         menu3 << ui::centeralign << "Employee Management" << ui::endl << ui::endl
                ;
606         menu3.settcolor(WHITE);
607         menu3 << "Add new employee" << ui::endl;
608         menu3.settcolor(ui::tcolor);
609         menu3 << "Step 2: Add employee details" << ui::endl << ui::endl;
610         menu3.settcolor_input(YELLOW);
611         menu3 << "Name: ";
612         menu3 >> name;
613         menu3 << "Sex: 1. Male | 2. Female | 3. Transsexual" << ui::endl
614               << "----- Enter your choice: ";
615         validate_menu::set_menu_limits(1, 3);
616         menu3 >> validate_menu::input >> (int)sex_choice;
617         menu3 << "Date of Birth(DD/MM/YYYY): ";
618         menu3 >> hospital::date_validity >> dob_str;
619         menu3 << "Address: " << ui::endl;
620         menu3 << (char)26 << "House no.: ";
621         menu3 >> adr_hno;
622         menu3 << (char)26 << "Street: ";
623         menu3 >> adr_street;
624         menu3 << (char)26 << "City: ";
625         menu3 >> adr_city;
626         menu3 << (char)26 << "District: ";
627         menu3 >> adr_dist;
628         menu3 << (char)26 << "State: ";
629         menu3 >> adr_state;
630         menu3 << "Phone no.: ";
631         menu3 >> phn_no;
632         menu3 << "Salary: ";
633         menu3 >> salary;
634         menu3 << "Shift timings: Starts - (HH:MM:SS)";
635         menu3 >> hospital::time_validity >> shift_start_str;
636         menu3 << "---------------: Ends - (HH:MM:SS)";
637         menu3 >> hospital::time_validity >> shift_end_str;
638         menu3.setexit_button("Submit");
639         menu3.setback_func(back_func::set_backbit);
640         menu3.loop();
641         menu3.hide();
642         if(back_func::backbit)
643         {
644             back_func::backbit = 0;
645             goto menu2;
646         }
647         --sex_choice;
648         dob = hospital::str_to_date(dob_str);
649         adr = address(adr_hno, adr_street, adr_city, adr_dist, adr_state);
650         shift_start = hospital::str_to_time(shift_start_str);
651         shift_end = hospital::str_to_time(shift_end_str);
652     }
653     menu4:
```

```
654
655     if(ch != 4)
656     {
657         box menu4( menu2_corner_top_left, menu2_width, ui::scr_height - 6 );
658         menu4.settcolor(GREEN);
659         menu4 << ui::centeralign << "Employee Management" << ui::endl << ui::endl
                ;
660         menu4.settcolor(WHITE);
661         menu4 << "Add new employee" << ui::endl;
662         menu4.settcolor(ui::tcolor);
663         menu4.settcolor_input(YELLOW);
664         menu4 << "Step 3: Add login details" << ui::endl << ui::endl;
665         menu4 << "User ID: ";
666         menu4 >> uid;
667         menu4 << "Password: ";
668         menu4 >> box::setpassword >> pwd;
669         menu4 << ui::endl;
670         menu4.setexit_button("Submit");
671         menu4.setback_func(back_func::set_backbit);
672         menu4.loop();
673         menu4.hide();
674     }
675     if(back_func::backbit)
676     {
677         back_func::backbit = 0;
678         goto menu3;
679     }
680     if(ch == 1)
681     {
682         coord c(1, 4);
683         box menu5(c, (ui::scr_width / 2), ui::scr_height - 5);
684         box inp_box(( c + coord((ui::scr_width / 2) - 1, 0)), (ui::scr_width / 2)
                 + 1, ui::scr_height - 5);
685         menu5.f << ( ui::top | ui::left ) << (char)204
686                 << ( ui::bottom | ui::left ) << (char)204
687                 << ( ui::top | ui::right ) << (char)203
688                 << ( ui::bottom | ui::right ) << (char)202;
689         menu5.f.display();
690         inp_box.f << ( ui::top | ui::left ) << (char)203
691               << ( ui::bottom | ui::left ) << (char)202
692               << ( ui::top | ui::right ) << (char)185
693               << ( ui::bottom | ui::right ) << (char)185;
694         inp_box.f.display();
695         menu5 << ui::centeralign << "Employee Management" << ui::endl << ui::endl
                ;
696         menu5.settcolor(WHITE);
697         menu5 << "Add new employee" << ui::endl;
698         menu5.settcolor(ui::tcolor);
699         menu5 << "Step 4: Add doctor details" << ui::endl << ui::endl;
700         menu5 << "Specialization of doctor (max 2)" << ui::endl
701               << "Choose from the following list: " << ui::endl;
702         for(int i = 0; i <= GEN; ++i)
703         {
704             if(i <= 8)
705             {
706                 menu5 << i << ". " << (body_parts)i << ui::endl;
707             }
708             else
709             {
```

```
710                 inp_box << i << ". " << (body_parts)i << ui::endl;
711             }
712         }
713         inp_box.settcolor_input(YELLOW);
714         inp_box << "Enter the number corresponding to the required entry in the 2
                fields below" << ui::endl;
715         validate_menu::set_menu_limits(BRAIN, GEN);
716         inp_box << (char)26;    inp_box >> validate_menu::input >> speciality[0];
717         inp_box << (char)26;    inp_box >> validate_menu::input >> speciality[1];
718         inp_box << ui::endl;
719         inp_box.setexit_button("Submit");
720         inp_box.setback_func(back_func::set_backbit);
721         inp_box.loop();
722         menu5.hide();
723         inp_box.hide();
724         window.f.display();
725     }
726     if(back_func::backbit)
727     {
728         back_func::backbit = 0;
729         goto menu4;
730     }
731     void * temp = NULL;
732     unsigned long id;
733     switch (ch)
734     {
735         case 1:
736         {
737             doctor x(name, sex_choice, dob, adr, phn_no, salary, shift_start,
                    shift_end, speciality[0], speciality[1], uid, pwd);
738             temp = &x;
739             id = x.get_id();
740             break;
741         }
742         case 2:
743         {
744             nurse x(name, sex_choice, dob, adr, phn_no, salary, shift_start,
                    shift_end, uid, pwd);
745             temp = &x;
746             id = x.get_id();
747             break;
748         }
749         case 3:
750         {
751             receptionist x(name, sex_choice, dob, adr, phn_no, salary,
                    shift_start, shift_end, uid, pwd);
752             temp = &x;
753             id = x.get_id();
754             break;
755         }
756         case 4:
757         {
758             employee x(name, sex_choice, dob, adr, phn_no, salary, shift_start,
                    shift_end);
759             temp = &x;
760             id = x.get_id();
761             break;
762         }
763     }
```

```
764     const int notice_height = 12;
765     box notice( menu2_corner_top_left, menu2_width, notice_height );
766     notice.settcolor(GREEN);
767     notice << ui::centeralign << "Employee Management" << ui::endl << ui::endl;
768     if(!hospital::write_employee(temp))
769     {
770         notice.settcolor(RED);
771         notice << "Employee addition unsuccessful!!";
772         notice.setexit_button("Exit");
773         notice.loop();
774         notice.hide();
775         return;
776     }
777     notice << "Employee added successfully!!" << ui::endl;
778     notice.settcolor(WHITE);
779     notice << "Hit the button below to display the details you entered: " << ui::
            endl;
780     notice.settcolor(ui::tcolor);
781     notice << ui::endl;
782     notice.setexit_button("View employee...");
783     notice.loop();
784     notice.hide();
785     view_emp(id);
786 }
787
788 void emp_mgmt::remove_emp()
789 {
790     const coord menu2_corner_top_left = coord(ui::scr_width * 0.2, 5);
791     const int menu2_width = ui::scr_width * 0.6;
792     unsigned long id;
793     char ch;
794     menu2:
795     {
796         const int menu2_height = 10;
797         box menu2(menu2_corner_top_left, menu2_width, menu2_height);
798         menu2.settcolor(GREEN);
799         menu2 << ui::centeralign << "Employee Management" << ui::endl << ui::endl
                ;
800         menu2.settcolor(WHITE);
801         menu2 << "Remove existing employee" << ui::endl;
802         menu2.settcolor(ui::tcolor);
803         menu2 << "Enter employee's id: ";
804         menu2.settcolor_input(YELLOW);
805         menu2 >> id;
806         menu2 << ui::endl;
807         menu2.setexit_button("Submit");
808         menu2.setback_func(back_func::set_backbit);
809         menu2.loop();
810         menu2.hide();
811     }
812     if(back_func::backbit)
813     {
814         back_func::backbit = 0;
815         return;
816     }
817     notice:
818     {
819         const int notice_height = 14;
820         box notice(menu2_corner_top_left, menu2_width, notice_height);
```

```
821         notice.settcolor(GREEN);
822         notice << ui::centeralign << "Employee Management" << ui::endl << ui::
                endl;
823         notice.settcolor(WHITE);
824         notice << "Hit the button below to display the details of the employee
                you want to remove: " << ui::endl;
825         notice.settcolor(ui::tcolor);
826         notice << ui::endl;
827         notice.setexit_button("View employee...");
828         notice.setback_func(back_func::set_backbit);
829         notice.loop();
830         notice.hide();
831     }
832     if(back_func::backbit)
833     {
834         back_func::backbit = 0;
835         goto menu2;
836     }
837     if( !view_emp(id) )
838     {
839         return;
840     }
841     notice2:
842     {
843         const int notice2_height = 14;
844         box notice2( menu2_corner_top_left, menu2_width, notice2_height );
845         notice2.settcolor(GREEN);
846         notice2 << ui::centeralign << "Employee Management" << ui::endl << ui::
                endl;
847         notice2.settcolor(WHITE);
848         notice2 << "Are you sure you want to remove this employee?(y/n): " << ui
                ::endl;
849         notice2.settcolor_input(YELLOW);
850         notice2 >> ch;
851         notice2.settcolor(ui::tcolor);
852         notice2 << ui::endl;
853         notice2.setexit_button("Submit");
854         notice2.setback_func(back_func::set_backbit);
855         notice2.loop();
856         notice2.hide();
857     }
858     if(back_func::backbit)
859     {
860         back_func::backbit = 0;
861         goto notice;
862     }
863     if(ch == 'n' || ch == 'N')
864     {
865         return;
866     }
867     const int notice3_height = 14;
868     box notice3( menu2_corner_top_left, menu2_width, notice3_height );
869     notice3.settcolor(GREEN);
870     notice3 << ui::centeralign << "Employee Management" << ui::endl << ui::endl;
871     notice3.settcolor(RED);
872     str path;
873     switch(id_to_emp::convert(id))
874     {
875         case INVALID:
```

```
876             interface::log_this("emp_mgmt::remove_emp() : No file with zero id
                    exists\nFunction aborted");
877             notice3 << "Invalid ID supplied!! Check log" << ui::endl;
878             notice3.setexit_button("Back");
879             notice3.loop();
880             notice3.hide();
881             return;
882         case OTHERS:
883             sprintf(path, "employee/%lu", id);
884             break;
885         case DOCTOR:
886             mkdir("employee/doctor");
887             sprintf(path, "employee/doctor/%lu", id);
888             break;
889         case NURSE:
890             mkdir("employee/nurse");
891             sprintf(path, "employee/nurse/%lu", id);
892             break;
893         case RECEPTIONIST:
894             mkdir("employee/receptionist");
895             sprintf(path, "employee/receptionist/%lu", id);
896             break;
897     }
898     int remove_status;
899     str file;
900     strcpy(file, path);
901     strcat(file, "/base.dat");
902     if( remove(file) == -1)
903     {
904         str log_str;
905         sprintf(log_str, "emp_mgmt::remove_emp() : Failed to delete base.dat file
                    of id %lu\nFunction aborted", id);
906         interface::log_this(log_str);
907         notice3 << "Failed to delete file of employee!!" << ui::endl;
908         notice3.setexit_button("Back");
909         notice3.loop();
910         notice3.hide();
911         return;
912     }
913     if( rmdir(path) == -1)
914     {
915         str log_str;
916         sprintf(log_str, "emp_mgmt::remove_emp() : Failed to delete folder of id
                %lu", id);
917         interface::log_this(log_str);
918     }
919     notice3.settcolor(GREEN);
920     notice3 << "Employee deletion successful!!" << ui::endl;
921     notice3.setexit_button("Back");
922     notice3.loop();
923     notice3.hide();
924 }
925
926 void emp_mgmt::edit_emp()
927 {
928     void * temp = malloc( sizeof(doctor) ); //as doctor has the greatest size
            among employee, doctor, nurse and receptionist classes
929     if(temp == NULL)
930     {
```

```
931         interface::log_this("emp_mgmt::edit_emp() : Not enough memory to allocate
                buffer void * temp = malloc( sizeof(doctor) )");
932         interface::error("Out of memory!! Check log");
933         getch();
934         return;
935     }
936     str name, dob_str, adr_hno, adr_street, adr_city, adr_dist, adr_state,
            shift_start_str, shift_end_str, uid, pwd, default_dob_str,
            default_shift_str;
937     unsigned sex_choice;
938     Date dob;
939     address adr;
940     phone phn_no;
941     unsigned long salary, id;
942     Time shift_start, shift_end;
943     const coord menu2_corner_top_left(ui::scr_width * 0.2, 5);
944     const int menu2_width = ui::scr_width * 0.6;
945     menu2:
946     {
947         const int menu2_height = 10;
948         box menu2(menu2_corner_top_left, menu2_width, menu2_height);
949         menu2.settcolor(GREEN);
950         menu2 << ui::centeralign << "Employee Management" << ui::endl << ui::endl
                ;
951         menu2.settcolor(WHITE);
952         menu2 << "Edit employee data" << ui::endl;
953         menu2.settcolor(ui::tcolor);
954         menu2 << "Step 1: Enter employee's id: ";
955         menu2.settcolor_input(YELLOW);
956         menu2 >> id;
957         menu2 << ui::endl;
958         menu2.setexit_button("Submit");
959         menu2.setback_func(back_func::set_backbit);
960         menu2.loop();
961         menu2.hide();
962     }
963     if(back_func::backbit)
964     {
965         back_func::backbit = 0;
966         free(temp);
967         return;
968     }
969     if(!hospital::get_employee_by_id(id, temp))
970     {
971         interface::error("ID not found or error while reading from file!");
972         getch();
973         free(temp);
974         return;
975     }
976     notice:
977     {
978         const int notice_height = 14;
979         box notice(menu2_corner_top_left, menu2_width, notice_height);
980         notice.settcolor(GREEN);
981         notice << ui::centeralign << "Employee Management" << ui::endl << ui::
                endl;
982         notice.settcolor(WHITE);
983         notice << "Details of the employee will now be shown with the existing
                data filled. "
```

```
984                 << "Change the data fields that you require to change, and leave
                        the other data fields as they are. "
985                 << "When you are finished, press Submit to submit the new details.
                        " << ui::endl;
986         notice.settcolor(ui::tcolor);
987         notice << ui::endl;
988         notice.setexit_button("View employee...");
989         notice.setback_func(back_func::set_backbit);
990         notice.loop();
991         notice.hide();
992     }
993     if(back_func::backbit)
994     {
995         back_func::backbit = 0;
996         goto menu2;
997     }
998     employee *e = (employee *) temp;
999     menu3:
1000    {
1001        const int menu3_height = 18;
1002        box menu3( menu2_corner_top_left, menu2_width, menu3_height );
1003        menu3.settcolor(GREEN);
1004        menu3 << ui::centeralign << "Employee Management" << ui::endl << ui::endl
                ;
1005        menu3.settcolor(WHITE);
1006        menu3 << "Edit employee data" << ui::endl;
1007        menu3.settcolor(ui::tcolor);
1008        menu3 << "Step 2: Edit employee details" << ui::endl << ui::endl;
1009
1010        menu3.settcolor_input(YELLOW);
1011        menu3 << "Name: ";
1012        menu3.setdefault( e->get_name() );
1013        menu3 >> name;
1014        menu3 << "Sex: 1. Male | 2. Female | 3. Transsexual" << ui::endl
1015              << "---- Enter your choice: ";
1016        validate_menu::set_menu_limits(1, 3);
1017        menu3.setdefault( e->get_sex() + 1 );
1018        menu3 >> validate_menu::input >> (int)sex_choice;
1019        menu3 << "Date of Birth(DD/MM/YYYY): ";
1020        sprintf(default_dob_str, "%u/%u/%u", e->get_dob().day, e->get_dob().month
                , e->get_dob().year);
1021        menu3.setdefault( default_dob_str );
1022        menu3 >> hospital::date_validity >> dob_str;
1023        menu3 << "Address: " << ui::endl;
1024        menu3 << (char)26 << "House no.: ";
1025        menu3.setdefault( e->get_address().house_no );
1026        menu3 >> adr_hno;
1027        menu3 << (char)26 << "Street: ";
1028        menu3.setdefault( e->get_address().street );
1029        menu3 >> adr_street;
1030        menu3 << (char)26 << "City: ";
1031        menu3.setdefault( e->get_address().city );
1032        menu3 >> adr_city;
1033        menu3 << (char)26 << "District: ";
1034        menu3.setdefault( e->get_address().district );
1035        menu3 >> adr_dist;
1036        menu3 << (char)26 << "State: ";
1037        menu3.setdefault( e->get_address().state );
1038        menu3 >> adr_state;
```

```
1039            menu3 << "Phone no.: ";
1040            menu3.setdefault( e->get_phone() );
1041            menu3 >> phn_no;
1042            menu3 << "Salary: ";
1043            menu3.setdefault( e->get_salary() );
1044            menu3 >> salary;
1045            menu3 << "Shift timings: Starts - (HH:MM:SS)";
1046            sprintf(default_shift_str, "%u:%u:%u", e->get_shift(START).hour, e->
                    get_shift(START).minute, e->get_shift(START).second );
1047            menu3.setdefault( default_shift_str );
1048            menu3 >> hospital::time_validity >> shift_start_str;
1049            menu3 << "---------------: Ends - (HH:MM:SS)";
1050            sprintf(default_shift_str, "%u:%u:%u", e->get_shift(END).hour, e->
                    get_shift(END).minute, e->get_shift(END).second );
1051            menu3.setdefault( default_shift_str );
1052            menu3 >> hospital::time_validity >> shift_end_str;
1053            menu3.setexit_button("Submit");
1054            menu3.setback_func(back_func::set_backbit);
1055            menu3.loop();
1056            menu3.hide();
1057        }
1058        if(back_func::backbit)
1059        {
1060            back_func::backbit = 0;
1061            goto notice;
1062        }
1063        --sex_choice;
1064        dob = hospital::str_to_date(dob_str);
1065        adr = address(adr_hno, adr_street, adr_city, adr_dist, adr_state);
1066        shift_start = hospital::str_to_time(shift_start_str);
1067        shift_end = hospital::str_to_time(shift_end_str);
1068        e->set_name(name);
1069        e->set_sex(sex_choice);
1070        e->set_dob(dob);
1071        e->set_address(adr);
1072        e->set_phone(phn_no);
1073        e->set_salary(salary);
1074        e->set_shift(START, shift_start);
1075        e->set_shift(END, shift_end);
1076        if(id_to_emp::convert(id) == DOCTOR)
1077        {
1078            coord c(1, 4);
1079            doctor *d = (doctor *)temp;
1080            box menu4(c, (ui::scr_width / 2), ui::scr_height - 5);
1081            box inp_box(( c + coord((ui::scr_width / 2) - 1, 0)), (ui::scr_width / 2)
                     + 1, ui::scr_height - 5);
1082            menu4.f << ( ui::top | ui::left ) << (char)204
1083                    << ( ui::bottom | ui::left ) << (char)204
1084                    << ( ui::top | ui::right ) << (char)203
1085                    << ( ui::bottom | ui::right ) << (char)202;
1086            menu4.f.display();
1087            inp_box.f << ( ui::top | ui::left ) << (char)203
1088                  << ( ui::bottom | ui::left ) << (char)202
1089                  << ( ui::top | ui::right ) << (char)185
1090                  << ( ui::bottom | ui::right ) << (char)185;
1091            inp_box.f.display();
1092            menu4 << ui::centeralign << "Employee Management" << ui::endl << ui::endl
                    ;
1093            menu4.settcolor(WHITE);
```

```
1094        menu4 << "Edit employee data" << ui::endl;
1095        menu4.settcolor(ui::tcolor);
1096        menu4 << "Step 3: Edit doctor details" << ui::endl << ui::endl;
1097        int speciality[2];
1098        menu4 << "Specialization of doctor (max 2)" << ui::endl
1099              << "Choose from the following list: " << ui::endl;
1100        for(int i = 0; i <= GEN; ++i)
1101        {
1102            if(i <= 8)
1103            {
1104                menu4 << i << ". " << (body_parts)i << ui::endl;
1105            }
1106            else
1107            {
1108                inp_box << i << ". " << (body_parts)i << ui::endl;
1109            }
1110        }
1111        inp_box.settcolor_input(YELLOW);
1112        inp_box << "Enter the number corresponding to the required entry in the 2
                fields below" << ui::endl;
1113        validate_menu::set_menu_limits(BRAIN, GEN);
1114        inp_box << (char)26;    inp_box.setdefault(d->get_speciality()[0]);
               inp_box >> validate_menu::input >> speciality[0];
1115        inp_box << (char)26;    inp_box.setdefault(d->get_speciality()[1]);
               inp_box >> validate_menu::input >> speciality[1];
1116        inp_box << ui::endl;
1117        inp_box.setexit_button("Submit");
1118        inp_box.setback_func(back_func::set_backbit);
1119        inp_box.loop();
1120        menu4.hide();
1121        inp_box.hide();
1122        window.f.display();
1123        d->set_speciality(speciality);
1124    }
1125    if(back_func::backbit)
1126    {
1127        back_func::backbit = 0;
1128        goto menu3;
1129    }
1130    const int notice2_height = 12;
1131    box notice2(menu2_corner_top_left, menu2_width, notice2_height);
1132    notice2.settcolor(GREEN);
1133    notice2 << ui::centeralign << "Employee Management" << ui::endl << ui::endl;
1134    if(!hospital::write_employee(temp))
1135    {
1136        notice2.settcolor(RED);
1137        notice2 << "Employee edit unsuccessful!!";
1138        notice2.setexit_button("Exit");
1139        notice2.loop();
1140        notice2.hide();
1141        free(temp);
1142        return;
1143    }
1144    notice2 << "Employee edited successfully!!" << ui::endl;
1145    notice2.settcolor(WHITE);
1146    notice2 << "Hit the button below to display the details you entered: " << ui
           ::endl;
1147    notice2.settcolor(ui::tcolor);
1148    notice2 << ui::endl;
```

```
1149     notice2.setexit_button("View employee...");
1150     notice2.loop();
1151     notice2.hide();
1152     view_emp(id);
1153     free(temp);
1154 }
1155
1156 void emp_mgmt::pay_emp()
1157 {
1158     unsigned long id;
1159     char ch;
1160     const coord menu2_corner_top_left = coord(ui::scr_width * 0.2, 5);
1161     const int menu2_width = ui::scr_width * 0.6;
1162     const int menu2_height = 10;
1163     menu2:
1164     {
1165         box menu2(menu2_corner_top_left, menu2_width, menu2_height);
1166         menu2.settcolor(GREEN);
1167         menu2 << ui::centeralign << "Employee Management" << ui::endl << ui::endl
                ;
1168         menu2.settcolor(WHITE);
1169         menu2 << "Pay salary to individual employee" << ui::endl;
1170         menu2.settcolor(ui::tcolor);
1171         menu2 << "Enter employee's id: ";
1172         menu2.settcolor_input(YELLOW);
1173         menu2 >> id;
1174         menu2 << ui::endl;
1175         menu2.setexit_button("Submit");
1176         menu2.setback_func(back_func::set_backbit);
1177         menu2.loop();
1178         menu2.hide();
1179     }
1180     if(back_func::backbit)
1181     {
1182         back_func::backbit = 0;
1183         return;
1184     }
1185     notice:
1186     {
1187         const int notice_height = 14;
1188         box notice(menu2_corner_top_left, menu2_width, notice_height);
1189         notice.settcolor(GREEN);
1190         notice << ui::centeralign << "Employee Management" << ui::endl << ui::
                endl;
1191         notice.settcolor(WHITE);
1192         notice << "Hit the button below to display the details of the employee
                you want to pay salary to: " << ui::endl;
1193         notice.settcolor(ui::tcolor);
1194         notice << ui::endl;
1195         notice.setexit_button("View employee...");
1196         notice.setback_func(back_func::set_backbit);
1197         notice.loop();
1198         notice.hide();
1199     }
1200     if(back_func::backbit)
1201     {
1202         back_func::backbit = 0;
1203         goto menu2;
1204     }
```

```
1205     if( !view_emp(id) )
1206     {
1207         return;
1208     }
1209     {
1210         const int notice2_height = 14;
1211         box notice2( menu2_corner_top_left, menu2_width, notice2_height );
1212         notice2.settcolor(GREEN);
1213         notice2 << ui::centeralign << "Employee Management" << ui::endl << ui::
                endl;
1214         notice2.settcolor(WHITE);
1215         notice2 << "Are you sure you want to pay salary to this employee?(y/n): "
                 << ui::endl;
1216         notice2.settcolor_input(YELLOW);
1217         notice2 >> ch;
1218         notice2.settcolor(ui::tcolor);
1219         notice2 << ui::endl;
1220         notice2.setexit_button("Submit");
1221         notice2.setback_func(back_func::set_backbit);
1222         notice2.loop();
1223         notice2.hide();
1224     }
1225     if(back_func::backbit)
1226     {
1227         back_func::backbit = 0;
1228         goto notice;
1229     }
1230     if(ch == 'n' || ch == 'N')
1231     {
1232         return;
1233     }
1234     const int notice3_height = 14;
1235     box notice3( menu2_corner_top_left, menu2_width, notice3_height );
1236     notice3.settcolor(GREEN);
1237     notice3 << ui::centeralign << "Employee Management" << ui::endl << ui::endl;
1238     notice3.settcolor(RED);
1239     if( !hospital::pay_salary(id, system::get_date(), system::get_time()) )
1240     {
1241         notice3 << "Failed to pay salary to the employee! Check log";
1242         notice3.setexit_button("Back");
1243         notice3.loop();
1244         notice3.hide();
1245         return;
1246     }
1247     notice3.settcolor(GREEN);
1248     notice3 << "Pay salary successful!!" << ui::endl;
1249     notice3.setexit_button("Back");
1250     notice3.loop();
1251     notice3.hide();
1252 }
1253
1254 void emp_mgmt::pay_all()
1255 {
1256     char ch;
1257     const int menu2_height = 11;
1258     box menu2(coord(ui::scr_width * 0.2, 5), ui::scr_width * 0.6, menu2_height);
1259     menu2.settcolor(GREEN);
1260     menu2 << ui::centeralign << "Employee Management" << ui::endl << ui::endl;
1261     menu2.settcolor(WHITE);
```

```
1262     menu2 << "Pay salary to all employees" << ui::endl;
1263     menu2.settcolor(ui::tcolor);
1264     menu2 << "Are you sure you want to pay salary to all employees?(y/n): ";
1265     menu2.settcolor_input(YELLOW);
1266     menu2 >> ch;
1267     menu2 << ui::endl;
1268     menu2.setexit_button("Submit");
1269     menu2.loop();
1270     menu2.hide();
1271     if(ch == 'n' || ch == 'N')
1272     {
1273         return;
1274     }
1275     const int notice_height = 10;
1276     box notice( menu2.getcorner_top_left(), menu2.getwidth(), notice_height );
            notice.hide();
1277     box notice2( notice.getcorner_top_left(), notice.getwidth(), notice.getheight
            () );
1278     notice2.settcolor(GREEN);
1279     notice2 << ui::centeralign << "Employee Management" << ui::endl << ui::endl;
1280     notice2.hide(); notice.display();
1281     notice.settcolor(GREEN);
1282     notice << ui::centeralign << "Employee Management" << ui::endl << ui::endl;
1283     notice.settcolor(ui::tcolor);
1284     notice << "Pay all salaries in progress..." << ui::endl;
1285     if( !hospital::pay_all_salaries() )
1286     {
1287         notice.hide();
1288         notice2.settcolor(RED);
1289         notice2 << "Failed to pay salary to all employees! Check log";
1290         notice2.setexit_button("Back");
1291         notice2.loop();
1292         notice2.hide();
1293         return;
1294     }
1295     notice.hide();  notice2.display();
1296     notice2 << "Pay all salaries successful!!" << ui::endl;
1297     notice2.setexit_button("Back");
1298     notice2.loop();
1299     notice2.hide();
1300 }
```

2. code/HOSP.CPP

```
1
2  int hospital::get_employee_by_id(unsigned long ID, void * target)
3  {
4      if(target == NULL)
5      {
6          interface::log_this("hospital::get_employee_by_id() : NULL pointer
               supplied to function\nFunction aborted");
7          return 0;
8      }
9      str temp;
10     int size_of_target;
11     switch(id_to_emp::convert(ID))
12     {
13         case INVALID:
```

```
14              interface::log_this("hospital::get_employee_by_id() : Invalid id
                    supplied to function\nFunction aborted");
15              return 0;
16          case OTHERS:
17              sprintf(temp, "employee/%lu/base.dat", ID);
18              size_of_target = sizeof(employee);
19              break;
20          case DOCTOR:
21              sprintf(temp, "employee/doctor/%lu/base.dat", ID);
22              size_of_target = sizeof(doctor);
23              break;
24          case NURSE:
25              sprintf(temp, "employee/nurse/%lu/base.dat", ID);
26              size_of_target = sizeof(nurse);
27              break;
28          case RECEPTIONIST:
29              sprintf(temp, "employee/receptionist/%lu/base.dat", ID);
30              size_of_target = sizeof(receptionist);
31              break;
32      }
33      int i = hospital::read_from( ID, (char*) target, size_of_target, temp );
34      if(!i)
35      {
36          target = NULL;
37          return 0;
38      }
39      return 1;
40  }
41
42  int hospital::write_employee(void * a)
43  {
44      if(a == NULL)
45      {
46          interface::log_this("hospital::write_employee() : NULL pointer supplied
                to function\nFunction aborted");
47          return 0;
48      }
49      mkdir("employee");
50      str temp;
51      int size_of_target;
52      employee *x = (employee *) a;
53      const unsigned long ID = x->get_id();
54      switch(id_to_emp::convert(ID))
55      {
56          case INVALID:
57              interface::log_this("hospital::write_employee() : Object with ID zero
                    cannot be written to file\nFunction aborted");
58              return 0;
59          case OTHERS:
60              sprintf(temp, "employee/%lu", ID);
61              size_of_target = sizeof(employee);
62              break;
63          case DOCTOR:
64              mkdir("employee/doctor");
65              sprintf(temp, "employee/doctor/%lu", ID);
66              size_of_target = sizeof(doctor);
67              break;
68          case NURSE:
69              mkdir("employee/nurse");
```

```
70              sprintf(temp, "employee/nurse/%lu", ID);
71              size_of_target = sizeof(nurse);
72              break;
73          case RECEPTIONIST:
74              mkdir("employee/receptionist");
75              sprintf(temp, "employee/receptionist/%lu", ID);
76              size_of_target = sizeof(receptionist);
77              break;
78      }
79      mkdir(temp);
80      strcat(temp, "/base.dat");
81      ofstream fout ( temp , ios::out | ios::binary);
82      if(!fout)
83      {
84          interface::log_this("hospital::write_employee() : Employee data file
                could not be created or accessed\nFunction aborted");
85          return 0;
86      }
87      fout.write( (char *) a , size_of_target );
88      if(fout.fail())
89      {
90          interface::log_this("hospital::write_employee() : Error while writing to
                file (fout.fail())\nFunction aborted");
91          return 0;
92      }
93      return 1;
94  }
95
96  int hospital::pay_salary(unsigned long id, Date d1, Time t1)
97  {
98      void * e = malloc( sizeof(doctor) );
99      if(e == NULL)
100     {
101         interface::log_this("hospital::pay_salary() : Not enough memory to
                allocate buffer void * temp = malloc( sizeof(doctor) )");
102         interface::error("Out of memory!! Check log");
103         getch();
104         return 0;
105     }
106     str temp;
107     switch(id_to_emp::convert(id))
108     {
109         case INVALID:
110             interface::log_this("hospital::pay_salary() : Invalid id supplied to
                    function\nFunction aborted");
111             return 0;
112         case OTHERS:
113             sprintf(temp, "employee/%lu/trans.dat", id);
114             break;
115         case DOCTOR:
116             sprintf(temp, "employee/doctor/%lu/trans.dat", id);
117             break;
118         case NURSE:
119             sprintf(temp, "employee/nurse/%lu/trans.dat", id);
120             break;
121         case RECEPTIONIST:
122             sprintf(temp, "employee/receptionist/%lu/trans.dat", id);
123             break;
124     }
```

```
125     if(!hospital::get_employee_by_id(id, e))
126     {
127         interface::log_this("hospital::pay_salary() : Employee not found or error
                while reading file\nFunction aborted");
128         free(e);
129         return 0;
130     }
131     unsigned long inp1;
132     char inp2[100] = "Salary paid to ";
133     employee * emp = (employee *)e;
134     inp1 = emp->get_salary();
135     strcat(inp2, emp->get_name());
136     transaction t = hospital::deduct_money(inp1, inp2, d1, t1);
137     free(e);
138
139     ofstream fout ( temp ,ios::binary | ios::app );
140     if(!fout)
141     {
142         interface::log_this("hospital::pay_salary() : Employee data file could
               not be created or accessed\nFunction aborted");
143         return 0;
144     }
145     fout.write((char *) &t, sizeof(transaction));
146     if(fout.fail())
147     {
148         interface::log_this("hospital::pay_salary() : Error while writing to file
                (fout.fail())\nFunction aborted");
149         return 0;
150     }
151     return 1;
152 }
153
154 int hospital::pay_all_salaries()
155 {
156     Date d1 = system::get_date();
157     Time t1 = system::get_time();
158     unsigned long max_id;
159     ifstream fin;
160     fin.open("employee/max_id.dat", ios::binary);
161     if(!fin)
162     {
163         interface::log_this("hospital::pay_all_salaries() : No employees found or
                cannot access file max_id.dat\nFunction aborted");
164         return 0;
165     }
166     else
167     {
168         fin.read((char *) &max_id, sizeof(unsigned long));
169         if(fin.fail())
170         {
171             interface::log_this("hospital::pay_all_salaries() : Error while
                   reading file max_id.dat(fin.fail())\nFunction aborted");
172             return 0;
173         }
174         if(!employee::get_generate_id_status())
175         {   //if generate_id_status is zero, then no id is generated after max_id
                + 1
176             //Thus, the following loop should run max_id + 1 times
177             ++max_id;
```

```
178         }
179         for(unsigned long i = 1; i <= max_id; ++i)
180         {
181             int a = hospital::pay_salary(i, d1, t1);
182             if(!a)
183             {
184                 str log_msg;
185                 sprintf(log_msg, "hospital::pay_all_salaries() : Failed to pay
                        salary of id %lu...\nSkipped", i);
186                 interface::log_this(log_msg);
187             }
188         }
189     }
190     return 1;
191 }
192
193 int hospital::time_validity(const char * inp_time)
194 {
195     return time_validity( str_to_time(inp_time) );
196 }
197
198 int hospital::time_validity(Time t)
199 {
200     if( t.hour > 24 || t.minute > 59 || t.second > 59)
201     {
202         return 0;
203     }
204     return 1;
205 }
206
207 Time hospital::str_to_time(const char * inp_time)
208 {
209 ///////In this function invalid time(25:00:00) is returned if time is in
        incorrect format//////////
210     char inp[3][3] = {"25", "0", "0"};
211     int inp_x = 0, inp_y = 0;
212     Time null(25, 0, 0);
213     if( strlen(inp_time) > 8 || strlen(inp_time) < 5 || inp_time[strlen(inp_time)
            - 1] == ':')
214     {
215         return null;
216     }
217     for(int i = 0; i < strlen(inp_time); ++i)
218     {
219         if(inp_time[i] == ':' && inp_y != 0)
220         {
221             inp[inp_x][inp_y] = '\0';
222             ++inp_x;
223             inp_y = 0;
224             continue;
225         }
226         else if( (inp_y == 0 && inp_time[i] == ':') || inp_y > 1
227                 || (inp_time[i] < '0' || inp_time[i] > '9') )
228         {
229             return null;
230         }
231         inp[inp_x][inp_y] = inp_time[i];
232         ++inp_y;
233     }
```

```
234     char *endptr;
235     null.hour = (unsigned int) strtol(inp[0], &endptr, 10);
236     null.minute = (unsigned int) strtol(inp[1], &endptr, 10);
237     null.second = (unsigned int) strtol(inp[2], &endptr, 10);
238     return null;
239 }
```

3. code/iface.cpp

```
1  int interface::validate_menu::input(const char * ch)
2  {
3      char *endptr;
4      int a = (int) strtol(ch, &endptr, 10);
5      if(!validation::vint(ch) || a < lowest_choice || a > greatest_choice)
6      {
7          return 0;
8      }
9      else
10     {
11         return 1;
12     }
13 }
14
15 void interface::validate_menu::set_menu_limits(int a, int b)
16 {
17     lowest_choice = a;
18     greatest_choice = b;
19 }
20
21 int interface::validate_menu::lowest_choice = 0;
22 int interface::validate_menu::greatest_choice = 0;
23
24 int interface::back_func::set_backbit()
25 {
26     backbit = 1;
27     return 1;
28 }
29
30 int interface::back_func::backbit = 0;
31
32 int interface::log_this(char * message)
33 {
34     Date dnow = system::get_date();
35     Time tnow = system::get_time();
36     char text[300];
37     sprintf(text, "$ [%u-%u-%u %u:%u:%u +0530]: ", dnow.day, dnow.month, dnow.
           year, tnow.hour, tnow.minute, tnow.second);
38     strcat(text, message);
39     ofstream fout;
40     fout.open("log.txt", ios::out | ios::app);
41     if(!fout)
42         return 0;
43     fout << text << endl;
44     if(fout.fail())
45         return 0;
46     fout.close();
47     return 1;
48 }
```

```

interface::interface(){}
```

4. code/EMP.CPP

```
#include "hosp.hpp"
#include "iface.hpp"
#include "emp.hpp"
#include "base.hpp"
#include <fstream.h>

//////////////////////////////////
/// Function definitions for class employee

int employee::generate_id()
{
    mkdir("employee");
    unsigned long max_id;
    ifstream fin;
    fin.open("employee/max_id.dat", ios::binary);
    if(!fin)
    {
        interface::log_this("employee::generate_id() : File max_id.dat not found
            or error while loading file\nmax_id will be set to zero");
        max_id = 0;
    }
    else
    {
        fin.read((char *) &max_id, sizeof(unsigned long));
        if(fin.fail())
        {
            interface::log_this("employee::generate_id() : Error while reading
                from file max_id.dat (fin.fail())\nFunction aborted");
            id = 0;
            return 0;
        }
    }
    fin.close();
    ++max_id;
    id = max_id;
    ofstream fout;
    fout.open("employee/max_id.dat", ios::binary);
    fout.write((char *) &max_id, sizeof(unsigned long));
    if(fout.fail())
    {
        interface::log_this("employee::generate_id() : Error while writing to
            file max_id.dat (fout.fail())\nFunction aborted");
        return 0;
    }
    else
        return 1;
}

int employee::generate_id_status = 1;

employee::employee(str inp1, int inp2, Date inp3, address inp4, phone inp5,
    unsigned long inp6, Time inp7, Time inp8, str inp9, str inp10) : person(inp1,
    inp2, inp3, inp4, inp5), account(inp9, inp10)
```

```
{
    if(!generate_id_status)
    {
        interface::error("ID cannot be generated for this employee. Check log");
        interface::log_this("employee::employee() : ID generation using
            generate_id() unsuccessful as generate_id_status is set to zero.\nThis
             is because some error was encountered during the last ID generation")
            ;
    }
    else
    {
        employee::generate_id_status = generate_id();
        id_to_emp i1(id, OTHERS);
        if(!i1.status)
        {
            interface::error("ID not generated properly for this employee. Check
                log");
            interface::log_this("employee::employee() : i1.status was set to zero
                , i.e id_list.dat doesn't have a record of the employee's id");
        }
        salary = inp6;
        shift_start = inp7;
        shift_end = inp8;
    }
}

employee::employee() : person()
{
    id = 0;
}

int employee::get_age()
{
    ////////////////Updating age to present age//////////
    set_dob(dob);       //This function is used here to invoke calc_age() in it
        only(because calc_age is directly not accessible)
    void * temp = malloc( sizeof(doctor) );
    if(temp != NULL && hospital::get_employee_by_id(id, temp))          //if
        employee's file exists on disk
    {
        hospital::write_employee( this );               //overwrite that file
    }
    free(temp);
    return age;
}

unsigned long employee::get_salary(){
    return salary;
}

void employee::set_salary(unsigned long inp)
{
    salary = inp;
}

Time employee::get_shift(int inp){
    switch(inp){
        case START:
            return shift_start;
```

```
101         case END:
102             return shift_end;
103         default:
104             return Time(0,0,0);
105     }
106 }
107
108 void employee::set_shift(int inp1, Time inp2)
109 {
110     switch (inp1)
111     {
112         case START:
113             shift_start = inp2;
114             return;
115         case END:
116             shift_end = inp2;
117             return;
118         default:
119             return;
120     }
121 }
122
123 unsigned long employee::get_id()
124 {
125     return id;
126 }
127
128 transaction * employee::get_last_5_transactions()
129 {
130     transaction * t = (transaction *)malloc(5 * sizeof(transaction));
131     if(t == NULL)
132     {
133         interface::log_this("employee::get_last_5_transactions() :Not enough
                memory to allocate buffer void * temp = malloc( sizeof(doctor) )\
                nFunction aborted");
134         return NULL;
135     }
136     for(int i = 0; i < 5; ++i)
137     {
138         t[i] = transaction();
139     }
140     str temp;
141     switch( id_to_emp::convert(id) )
142     {
143         case INVALID:
144         {
145             char log_msg[300];
146             sprintf(log_msg, "employee::get_last_5_transactions() : The object
                    has invalid id (%lu)\nFunction aborted", id);
147             interface::log_this(log_msg);
148             free(t);
149             return NULL;
150         }
151         case DOCTOR:
152         {
153             sprintf(temp, "employee/doctor/%lu/trans.dat", id);
154             break;
155         }
156         case NURSE:
```

```
157         {
158             sprintf(temp, "employee/nurse/%lu/trans.dat", id);
159             break;
160         }
161         case RECEPTIONIST:
162         {
163             sprintf(temp, "employee/receptionist/%lu/trans.dat", id);
164             break;
165         }
166         case OTHERS:
167         {
168             sprintf(temp, "employee/%lu/trans.dat", id);
169             break;
170         }
171     }
172     ifstream fin ( temp ,ios::binary | ios::in | ios::nocreate | ios::ate);
173     if(!fin)
174     {
175         char log_msg[300];
176         sprintf(log_msg, "employee::get_last_5_transactions() : Failed to open
                file trans.dat for id %lu\nFunction aborted", id);
177         interface::log_this(log_msg);
178         free(t);
179         return NULL;
180     }
181     int max_i, size_of_file = fin.tellg();
182     if( size_of_file >= ( 5 * sizeof(transaction) ) )
183     {
184         const int a = (-5) * sizeof(transaction);
185         fin.seekg(a, ios::end);
186         max_i = 5;
187     }
188     else
189     {
190         fin.seekg(0, ios::beg);
191         max_i = (int)( size_of_file / sizeof(transaction) );
192     }
193     for(i = 0; i < max_i && !fin.eof(); ++i)
194     {
195         fin.read((char *) (t+i), sizeof(transaction));
196         if(fin.fail())
197         {
198             char log_msg[300];
199             sprintf(log_msg, "employee::get_last_5_transactions() : Failed to
                    read file trans.dat for id %lu(loop failed at i = %i)\nFunction
                    aborted", id, i);
200             interface::log_this(log_msg);
201             free(t);
202             return NULL;
203         }
204     }
205     fin.close();
206     return t;
207 }
208
209 int employee::get_generate_id_status()
210 {
211     return generate_id_status;
212 }
```

```
213
214 ////////////////////////////////////////
215 //// Doctor, Nurse and Receptionist class member defs
216
217 doctor::doctor(str inp1, int inp2, Date inp3, address inp4, phone inp5, unsigned
        long inp6, Time inp7, Time inp8, int inp10, int inp11, str inp12, str inp13) :
         employee(inp1, inp2, inp3, inp4, inp5, inp6, inp7, inp8, inp12, inp13)
218 {
219     id_to_emp i1(get_id(), DOCTOR);
220     if(!i1.status)
221     {
222         interface::error("ID not generated properly for this employee. Check log"
                );
223         interface::log_this("doctor::doctor() : i1.status was set to zero, i.e
                id_list.dat doesn't have a record of the employee's id");
224     }
225     speciality[0] = inp10;
226     speciality[1] = inp11;
227
228     for(int i = 0; i < 10; i++){
229         patients[i] = 0;
230     }
231 }
232
233 doctor::doctor() : employee()
234 {
235     speciality[0] = speciality[1] = GEN + 1;    //storing an invalid value in
            speciality
236     for(int i = 0; i < 10; ++i)
237     {
238         patients[i] = 0;
239     }
240 }
241
242 int * doctor::get_speciality()
243 {
244     return speciality;
245 }
246
247 long * doctor::get_patients()
248 {
249     return patients;
250 }
251
252 void doctor::set_speciality(int inp[2])
253 {
254     speciality[0] = inp[0];
255     speciality[1] = inp[1];
256 }
257
258 void doctor::set_patients(long inp[10])
259 {
260     for(int i = 0; i < 10; ++i)
261     {
262         patients[i] = inp[i];
263     }
264 }
265
266 nurse::nurse(str inp1, int inp2, Date inp3, address inp4, phone inp5, unsigned
```

```
    long inp6, Time inp7, Time inp8, str inp10, str inp11) : employee(inp1, inp2,
    inp3, inp4, inp5, inp6, inp7, inp8, inp10, inp11)
{
    id_to_emp i1(get_id(), NURSE);
    if(!i1.status)
    {
        interface::error("ID not generated properly for this employee. Check log"
            );
        interface::log_this("nurse::nurse() : i1.status was set to zero, i.e
            id_list.dat doesn't have a record of the employee's id");
    }
    for(int i = 0; i < 5; i++){
        patients[i] = 0;
    }
}

nurse::nurse() : employee()
{
    for(int i = 0; i < 5; ++i)
    {
        patients[i] = 0;
    }
}

long * nurse::get_patients()
{
    return patients;
}

void nurse::set_patients(long inp[5])
{
    for(int i = 0; i < 5; ++i)
    {
        patients[i] = inp[i];
    }
}

receptionist::receptionist(str inp1, int inp2, Date inp3, address inp4, phone
    inp5, unsigned long inp6, Time inp7, Time inp8, str inp10, str inp11) :
    employee(inp1, inp2, inp3, inp4, inp5, inp6, inp7, inp8, inp10, inp11)
{
    id_to_emp i1(get_id(), RECEPTIONIST);
    if(!i1.status)
    {
        interface::error("ID not generated properly for this employee. Check log"
            );
        interface::log_this("receptionist::receptionist() : i1.status was set to
            zero, i.e id_list.dat doesn't have a record of the employee's id");
    }
}

receptionist::receptionist() : employee()
{}


////////////////////////////////
/// Function definitions for class id_to_emp

id_to_emp::id_to_emp(unsigned long inp1, int inp2)
```

```
318 {
319     status = 0;
320     id = inp1;
321     if(!id)
322     {
323         employee_type = INVALID;
324     }
325     else
326     {
327         employee_type = inp2;
328     }
329     mkdir("employee");
330     ofstream fout;
331     fout.open("employee/id_list.dat", ios::binary | ios::ate);
332     if(!fout)
333     {
334         interface::log_this("id_to_emp::id_to_emp() : File id_list.dat couldn't
                be opened...\nFunction aborted");
335     }
336     else
337     {
338         fout.seekp(id * sizeof(id_to_emp), ios::beg);
339         fout.write((char *) this, sizeof(id_to_emp));
340         if(fout.fail())
341         {
342             interface::log_this("id_to_emp::id_to_emp() : Error while writing to
                    file id_list.dat (fout.fail())\nFunction aborted");
343         }
344         else
345         {
346             status = 1;
347         }
348     }
349 }
350
351 id_to_emp::id_to_emp()
352 {
353     id = employee_type = status = 0;
354 }
355
356 int id_to_emp::convert(unsigned long ID)
357 {
358     id_to_emp a;
359     ifstream fin;
360     fin.open("employee/id_list.dat", ios::binary);
361     if(!fin)
362     {
363         interface::log_this("id_to_emp::convert() : File id_list.dat not found!!"
                );
364         return INVALID;
365     }
366     fin.seekg( (ID * sizeof(id_to_emp)) );
367     fin.read((char *) &a, sizeof(id_to_emp));
368     if(fin.fail())
369     {
370         interface::log_this("id_to_emp::convert() : Error while reading from file
                 id_list.dat (fin.fail())");
371         return INVALID;
372     }
```

```
373     fin.close();
374     if(a.id != ID)
375     {
376         interface::log_this("id_to_emp::convert() : (For dev only)Error in the
                code... Recheck it!!");
377         return INVALID;
378     }
379     return a.employee_type;
380 }
```