Chapter 4: IMPLEMENTATION

4.1 Model WATERFALL MODEL

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall Model - Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.

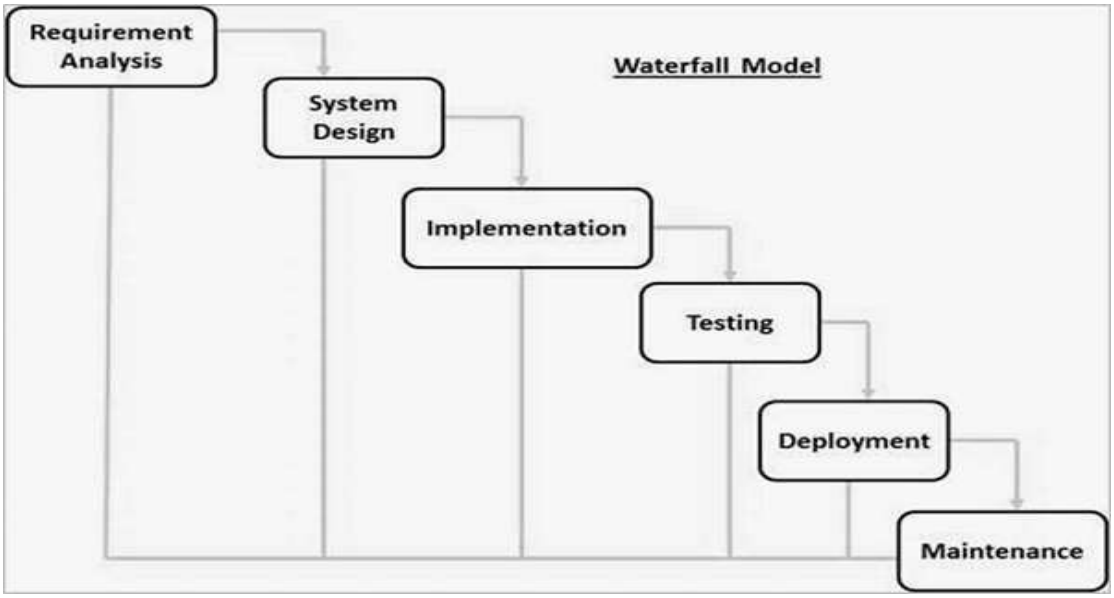

FIGURE. 4.1.1 WATERFALL MODEL

The sequential phases in Waterfall model are −

- Requirement Gathering and analysis − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- System Design − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- Implementation − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- Integration and Testing − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- Deployment of system − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- Maintenance − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

Waterfall Model -Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.

- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model - Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows −

- Simple and easy to understand and use.
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Waterfall Model - Disadvantages

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows −

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.

- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

4.2 Data Flow Diagram

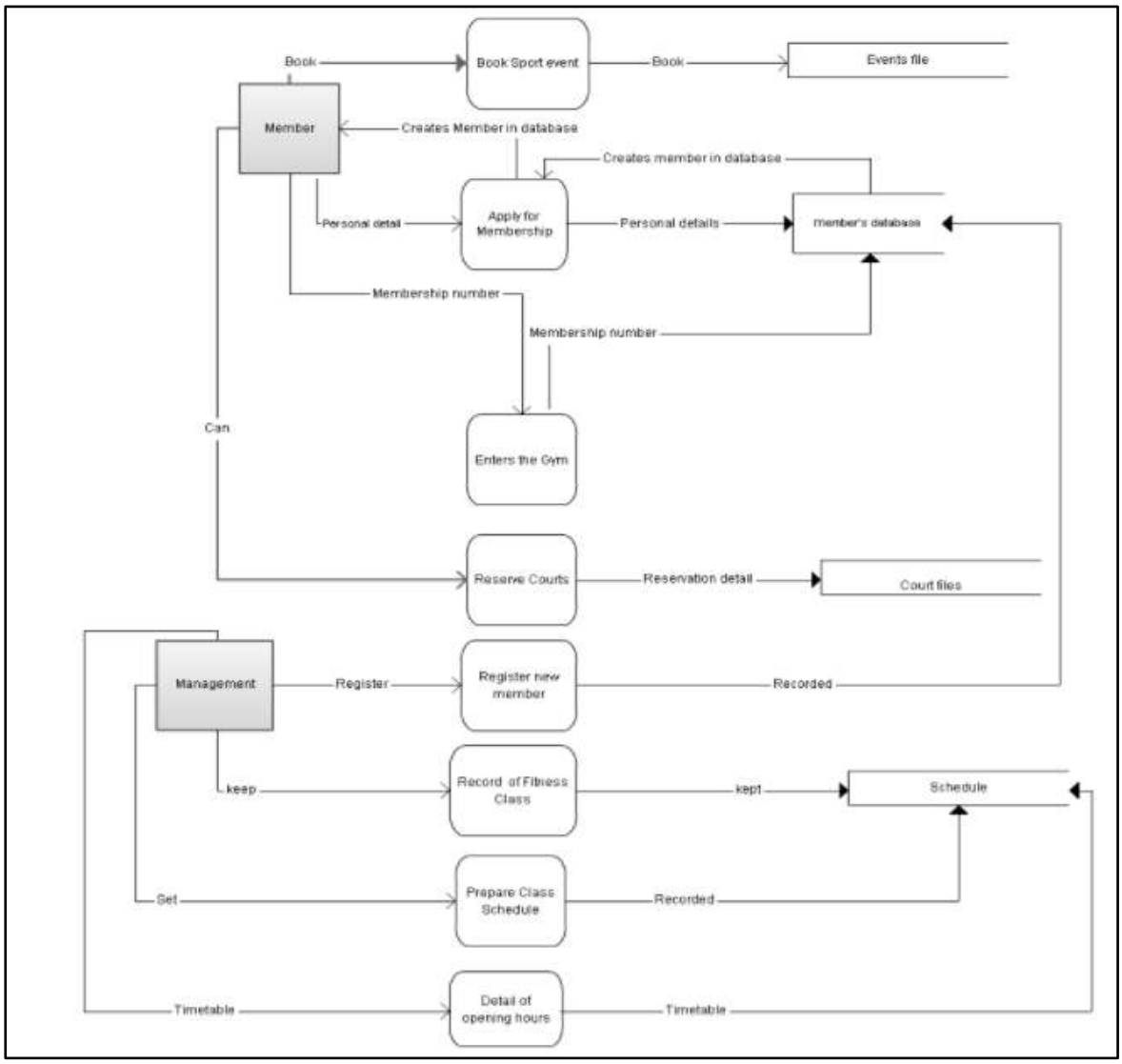

FIGURE 4.2.1 FIRST LEVEL DFD

This level one DFD explains about the feature when user registers on our App. Users can upload their own project when logged in. Users can search and add new projects on our App. When user adds new project admin approval is required.

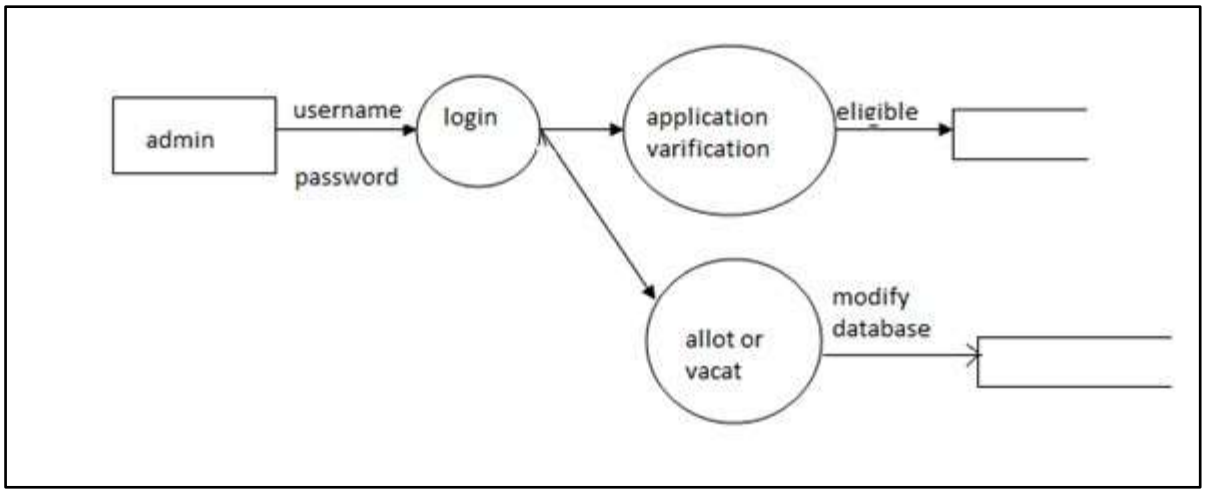

FIGURE 4.2.2 SECOND LEVEL DFD

This level two DFD explains detailed procedure of how users enters their information and gets login in our app with the help of username and password allotted to them.

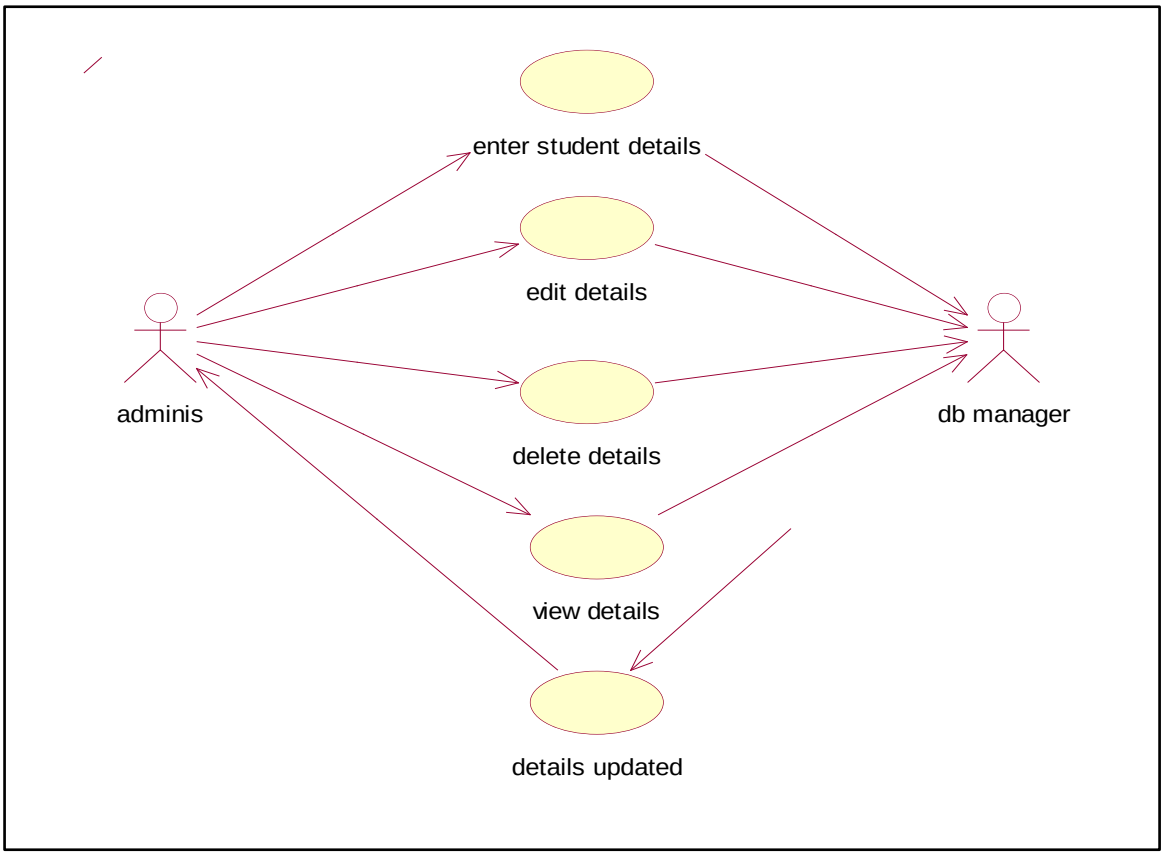

FIGURE 4.2.3 USE CASE DIAGRAM

This use case diagram explains about the administrator functionalities and its benefits. It also gives information about the users who are registered on the app, When administrator accesses the content of App administrator has to login on the App. Administrator can enable and disable users and also projects hosted on the App. Users can search the projects and add their own projects on the App this use case explains about how users can do this.

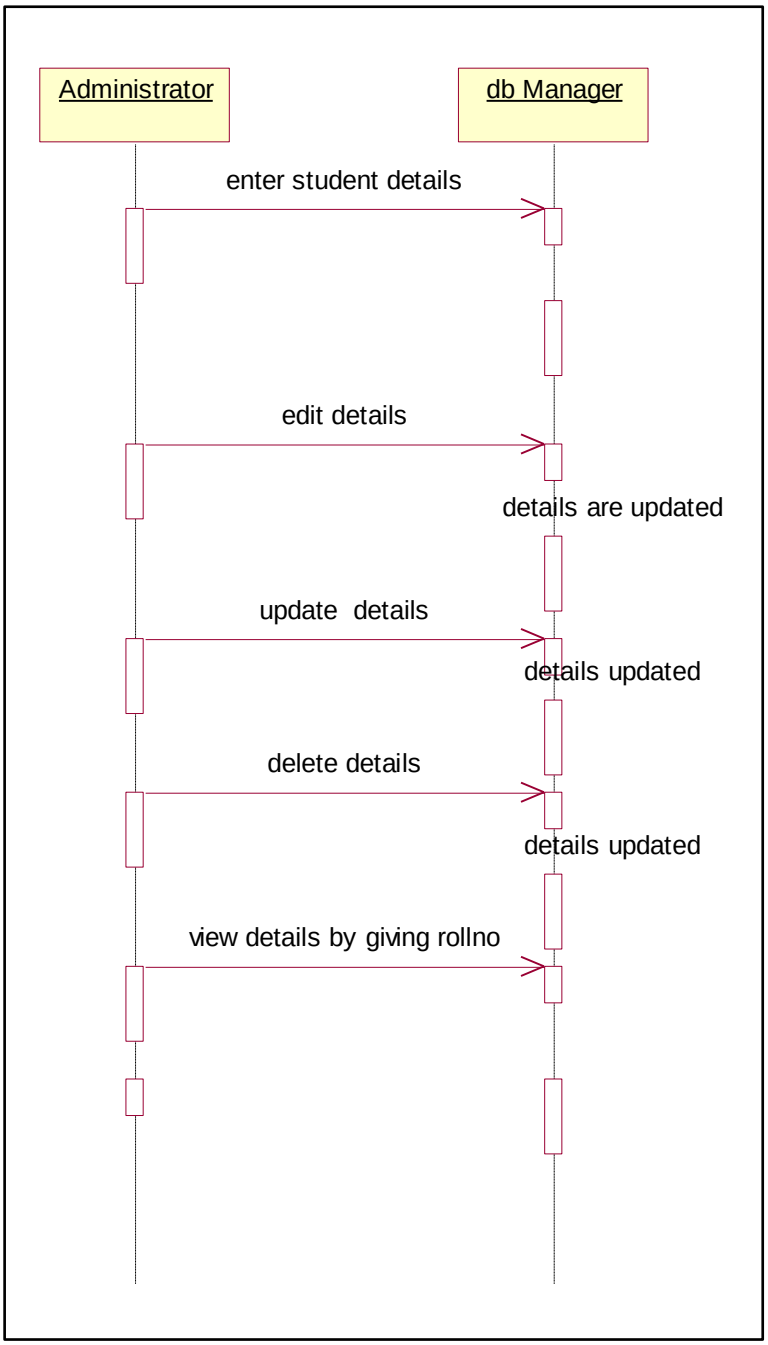

FIGURE 4.2.4 SEQUENCE DIAGRAM FOR LOGIN

This sequence diagram explains about how user logs in on our App. First users have to register, then they are asked for email id and password. When login is not valid then it sends a sorry message box on the interface and if it succeeds then it shows project interface.

4.4 USER INTERFACE DESIGN

1. HOME PAGE

SCREEN 4.4.1 HOME PAGE SNAP SHOT

This above snapshot 4.4.1 explains about the source code of home page, were user can register, add, update, or delete as shown in the above snapshot.

SOURCE CODE

```
package com.example.owner.gym_app;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;

public class HomePage extends AppCompatActivity {
    Button workout,diet,packages,supplents,contact,notification,add;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home_page);

    //update this info
        workout= findViewById(R.id.btnworkout);
        add=(Button) findViewById(R.id.addcustomer);
        packages=(Button) findViewById(R.id.btnpayments);
        supplents=(Button) findViewById(R.id.btnupdate);
        contact=(Button) findViewById(R.id.btndelete);
        notification=(Button) findViewById(R.id.btnnotify);


        add.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
            Intent intent=new Intent(HomePage.this,AddCustomer.class);
            startActivity(intent);
        }
    });

    notification.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent=new Intent();
            PendingIntent pendingIntent=PendingIntent.getActivity(HomePage.this,0,intent,0);
            Notification noti=new Notification.Builder(HomePage.this)
                    .setTicker("Ticker Title")
                    .setContentText("Content Title")
                    .setContentText("Content Text")
                    //.setSmallIcon(R.drawable.ic_launcher)
                    .setContentIntent(pendingIntent).getNotification();

            noti.flags=Notification.FLAG_AUTO_CANCEL;
            NotificationManager
nm=(NotificationManager)getSystemService(NOTIFICATION_SERVICE);
            nm.notify(0,noti);


        }
    });


    workout.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startActivity(new Intent(HomePage.this, Register.class));
        }
    });
    }
```

2. Login

SCREEN 4.4.2 SNAP SHOT FOR LOGIN SESSION

This snapshot explains about the functionality of how admin login is performed. When user id and password are matched with that of administrator credentials then administrator is directed on the respective page.

SOURCE CODE:

```
package com.example.owner.gym_app;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class Login extends AppCompatActivity {
   Button b1,b2;
   EditText e1,e2;
   DatabaseHelper db;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_login);
      db=new DatabaseHelper(this);
      e1=(EditText)findViewById(R.id.editText);
      e2=(EditText)findViewById(R.id.editText2);
      b1=(Button)findViewById(R.id.button);
      b2=(Button)findViewById(R.id.register);

      b1.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            String email=e1.getText().toString();
            String password=e2.getText().toString();
            Boolean chkemailpass=db.emailpassword(email,password);
            if(chkemailpass==true){
```

```
                Toast.makeText(getApplicationContext(),"login
successful",Toast.LENGTH_LONG).show();
                Intent g=new Intent(Login.this,HomePage.class);
                startActivity(g);

            }

            else{
                Toast.makeText(getApplicationContext(),"Wrong email or
password",Toast.LENGTH_LONG).show();



            }
        }
    });
    b2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent i=new Intent(Login.this,Register.class);
            startActivity(i);
        }
    });
  }
}
```