Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
MoE, UGC & AICTE Approved
NAAC A++ Accredited

An internship report submitted by

STUDENTS NAME - Reg.No URK21CS3007
Reg.No URK21CO2002

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

under the supervision of

Dr.Rajeswari

DIVISION OF COMPUTER SCIENCE AND ENGINEERING
KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec-3 of the UGC Act, 1956)
Karunya Nagar, Coimbatore - 641 114. INDIA

Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES

(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)

MoE, UGC & AICTE Approved

NAAC A++ Accredited

DIVISION OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the report entitled, "Design And Implementation Of Any Time Electricity Bill Payment Machine Controller " is a bonafide record of Internship work done at Intel Unnati Training Program during the academic year 2022-2023 by

JOSEPH STEPHY J (URK21CS3007)
SAHAYA BERDIN P (URK21C02002)

in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering of Karunya Institute of Technology and Sciences.

Guide Signature

Dr.Rajeswari

ACKNOWLEDGEMENT

First and foremost, We praise and thank ALMIGTHY GOD whose blessings have bestowed in us the will power and confidence to carry out our internship.

We are grateful to our beloved founders **Late. Dr. D.G.S. Dhinakaran, C.A.I.I.B, Ph.D** and **Dr. Paul Dhinakaran, M.B.A, Ph.D**, for their love and always remembering us in their prayers.

We entend Our tanks to **Dr. Prince Arulraj, M.E., Ph.D., Ph.D.,** our honorable vice chancellor, **Dr. E. J. James, Ph.D.,** and **Dr. Ridling Margaret Waller, Ph.D.,** our honorable Pro-Vice Chancellor(s) and **Dr. R. Elijah Blessing, Ph.D.,** our respected Registrar for giving me this opportunity to do the internship.

We would like to thank **Dr. Ciza Thomas, M.E., Ph.D.,** Dean, School of Engineering and Technology for her direction and invaluable support to complete the same.

We would like to place my heart-felt thanks and gratitude to **Dr. J. Immanuel John Raja, M.E., Ph.D.,** Head of the Division, Computer Science and Engineering for his encouragement and guidance.

We feel it a pleasure to be indebted to, **Dr. Rajeswari**, Designation, Division of CSE & **Mr.k.Padmanaban** Intel Designation for their invaluable support, advice and encouragement.

We also thank all the staff members of the School of CSE for extending their helping hands to make this in Internship a successful one.

We would also like to thank all my friends and my parents who have prayed and helped us during the Internship.

ABSTRACT

Electricity consumers are often faced with the problem of inaccuracy and delay in monthly billing due to some drawbacks. Thus, it is essential to have an efficient system for such purposes via electronic platform with consideration to proximity. The proposed system automates the conventional process of paying electricity bill by visiting the Electricity Board which is tiresome and time consuming. It is also designed to automate the electricity bill calculation and payment for user convenience. The system is developed with VHDL as the base programming language which can be used as a language to to write text models that describe a logic circuit The system would be having two logins: the administrative and user login. The administrator can view the user's account details and can add the customer's information of consuming units of energy of the current month in their account. The Admin must feed the system with the electricity usage data into respective user's account. The system then calculates the electricity bill for every user and updates the information into their account every month. Users can then view their electricity bill and pay before the month end.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Electricity Billing System is a software-based application.

i. This project aims at serving the department of electricity by computerizing the billing system.

ii. It mainly focuses on the calculation of units consumed during the specified time and the money to be charged by the electricity offices.

iii. This computerized system will make the overall billing system easy, accessible, comfortable, and effective for consumers.

To design the billing system more service oriented and simple, the following features have been implemented in the project. The application has high speed of performance with accuracy and efficiency.

The software provides facility of data sharing, it does not require any staff as in the conventional system. Once it is installed on the system only the meter readings are to be given by the admin where customer can view all details, it has the provision of security restriction.

The electricity billing software calculates the units consumed by the customer and makes bills, it requires small storage for installation and functioning. There is provision for debugging if any problem is encountered in the system.

The system excludes the need of maintaining paper electricity bill, administrator does not have to keep amanual track of the users, users can pay the amount without visiting the office. Thus, it saves human efforts and resources.

1.1 Preamble

We, the owners of our project, respect all customers and make them happy with our service.

The main aim of our project is to satisfy customer by saving their time by payment process, maintaining records, and allowing the customer to view his/her records and permitting them to update their details.

The firm handles all the work manually, which is very tedious and mismatched.

The objectives of our project are as follows:

- ❖ To keep the information of customer.
- ❖ To keep the information of consuming unit energy of current month.
- ❖ To keep the information of consuming unit energy of previous month.
- ❖ To calculate the units consumed every month regularly.
- ❖ To generate the bills adding penalty and rent.
- ❖ To save the time by implementing payment process online.

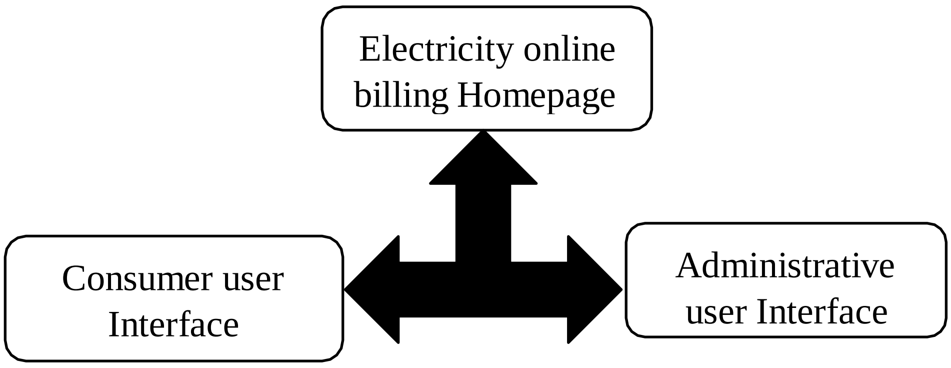

Electricity online billing Homepage
Consumer user Interface
Administrative user Interface

Fig 1.1.1: Block diagram showing the proposed online Electricity billing system.

1.2 Problem Statement

The manual system is suffering from a series of drawbacks. Since whole of the bills is to be maintained with hands the process of keeping and maintaining the information is very tedious and lengthy to customer. It is very time consuming and laborious process because, staff need to be visited the customers place every month to give the bills and to receive the payments. For this reason, we have provided features Present system is partially automated(computerized), existing system is quite laborious as one must enter same information at different places.so by this the users can pay using credit and debit card also and torn and wet cash are not accepted in this machine.

1.3 Proposed Solution

- This project system excludes the need of maintaining paper electricity bill as all the electricity bill records are managed electronically.
- Administrator doesn't have to keep a manual track of the users. The system automatically calculates fine.
- Users don't have to keep a manual track of the users. The system automatically calculates fine.
- There is no need of delivery boy for delivery bills to user's place.

Thus, it saves human efforts and resources.

2.ANALYSIS AND SYSTEM REQUIREMENT

2.1 Existing and Proposed System

The conventional system of electricity billing is not so effective; one staff must visit each customer's house to note the meter readings and collect the data. Then, another staff must compute the consumed units and calculate the money to be paid. Again, the bills prepared are to be delivered to customers. Finally, individual customer must go to electricity office to pay their dues.

Hence, the conventional electricity billing system is uneconomical, requires many staffs to do simple jobs and is a lengthy process overall. In order to solve this lengthy process of billing, a web based computerized system is essential. This proposed electricity billing system project overcomes all these drawbacks with the features. It is beneficial to both consumers and the company which provides electricity.

With the new system, there is reduction in the number of staffs to be employed by the company. The working speed and performance of the software is faster with high performance which saves time. Furthermore, there is very little chance of miscalculation and being corrupted by the staffs.

2.2 Feasibility Study:

Feasibility study is the phase in which the analyst checks that the candidate system is feasible for the organization or not. This entails identification, description & evaluation of the system. Feasibility study is done to select the best system that meets the performance requirement.

If the feasibility study is to serve as a decision document, it must answer key questions.

1. Is there a new and better way to do the job that will benefit the user?
2. What are the costs and savings of the alternatives?
3. What is recommended?

The most successful system projects are not necessarily the biggest or most visible in the business but rather those truly meet user's expectations.

Feasibility considerations

Three key considerations are involved in the feasibility study. They are as follows:-

Economic Feasibility:

Economic analysis is the most frequently used method for evaluating the effectiveness of the candidate system.

We analyse the candidate system (computerized system) is feasible as than the manual system because it saves the money, time and manpower. It also feasible according to cost benefits analysis

Technical Feasibility:

Technical feasibility centers around the technology used. It means the candidate system is technically feasible i.e. it don't have any technical fault and work properly in the given environment. Our system is technically feasible; it is providing us required output.

Behavioral Feasibility:

Behavioral feasibility is the analysis of behavior of the candidate system. In this we analyse that the candidate system is working properly or not. If working than it communicating proper with the environment or not. All this matters are analysed and a good candidate system is prepared. Due to the change of system what is the change in behaviour of the users, this factors are also analysed.

2.3SYSTEM DEVELOPMENT ENVIRONMENT

System development environment shows the hardware and software requirement, which is necessary for developing the software. Necessary software and hardware requirement, which are necessary for making this software are as follows:

2.3.1 : Hardware Requirements:

- Hardware specification: Intel Pentium Processor
- 32 MB RAM or Higher
- 1.2 GB Hard Disk or Greater
- Video Display Unit
- Keyboard

- Mouse

2.3.2 : Software Requirements:

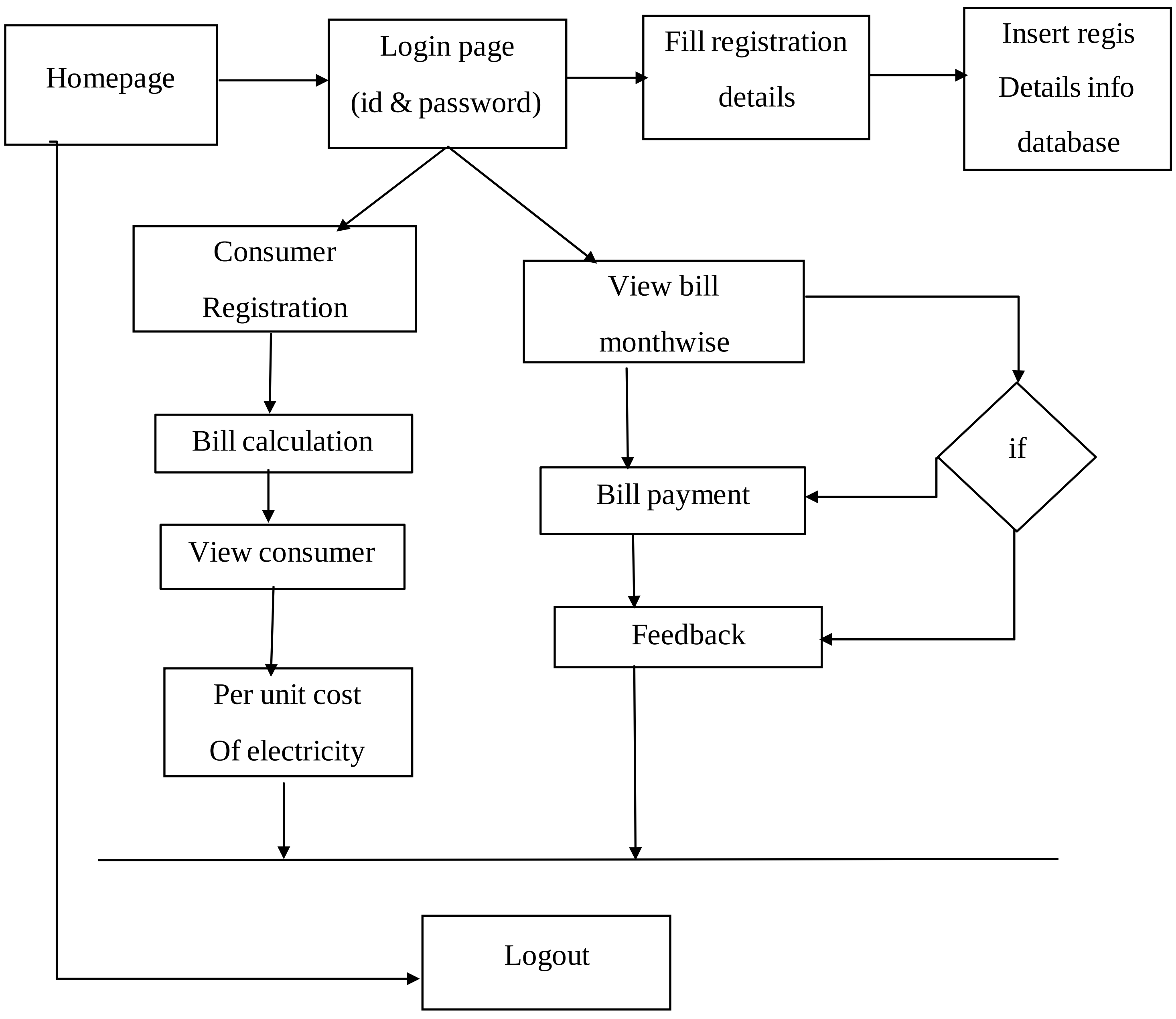
- Operating System: Windows 10
- Software: Quartus Prime Lite Edison
- Front End: VHDL

3.SYSTEM DESIGN AND MODELLING

3.1 System Architecture

The system architecture gives the overview of the organizational system that shows the system boundaries, external entities that interact with the system, and the major information that flows between the entities and the system.

Fig 3.1.1 System Architecture of system



Homepage
Login page (id & password)
Fill registration details
Insert regis Details info database
Consumer Registration
View bill monthwise
Bill calculation
if
Bill payment
View consumer
Feedback
Per unit cost Of electricity
Logout

3.2 Preliminary Design

System design is an abstract representation of a system component and their relationship and which describe the aggregated functionally and performance of the system. It is also the plan or blueprint for how to obtain answer to the question being asked. The design specifies various type of approach.

3.2.1 Entity – Relationship Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

There are two reasons to create a database diagram. You're either designing a new schema or you need to document our existing structure.

If you have an existing database you need to document, you create a database diagram using data directly from your database. You can export your data base structure as a CSV file (there are some scripts on how to do this here), then have a program generate the ERD automatically.

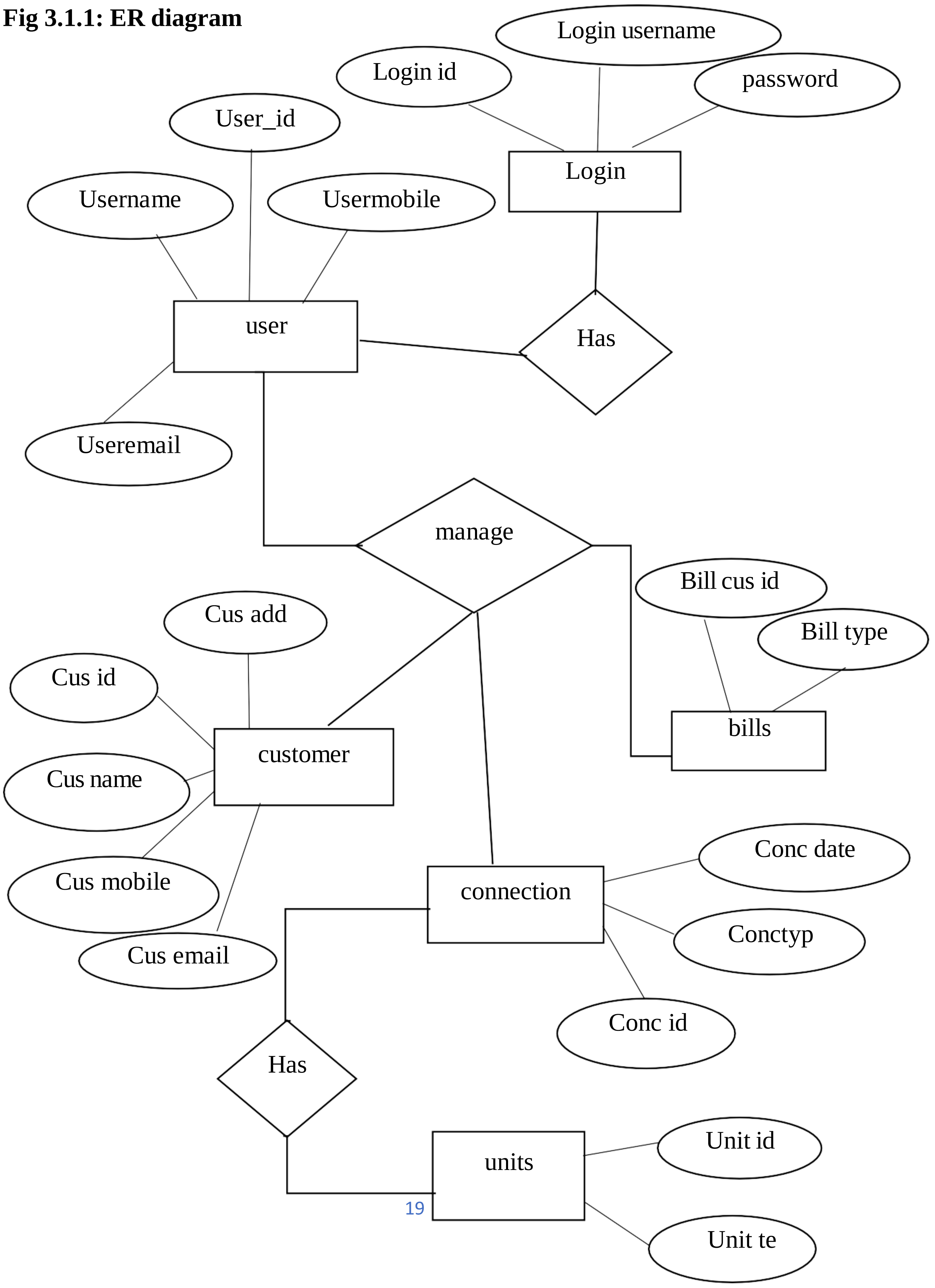
An ER diagram is a means of visualizing how the information a system produces is related. There are five main components of an ERD:

- ❖ Entities, which are represented by rectangles. An entity is an object or concept about which you want to store information.
- ❖ A weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.
- ❖ Actions, which are represented by diamond shapes, show how two entites share information in the database.
- ❖ In some case, entities can be self-linked. For example, employees can supervise other employees.

- Attributes, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity.
- A multivalued attribute can have more than one vale. For example, an employee entity can have multiple skill values.
- A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's annual salary.
- Connecting lines, solid lines that connect attributes to show the relationships of the entities in diagram.
- Cardinality specifies how many instances of an entity relate to one instance of another entity. Ordinality is also closely linked to cardinality

Figure 3.2.1 describes the ER diagram of Electricity Billing System. It has 5 entities namely login, customer, bill, and meter info. The entities have attributes which are primary and foreign and attributes. The primary attributes are underlined.

Fig 3.1.1: ER diagram

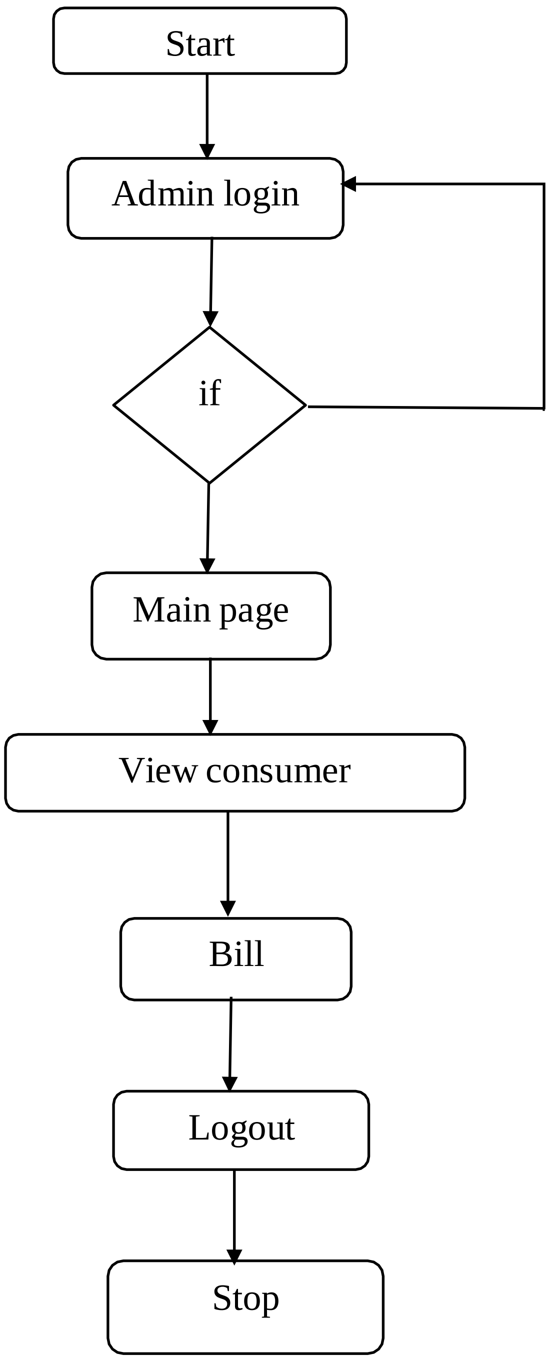


4. IMPLIMENTATION

4.1 Implementation of operations

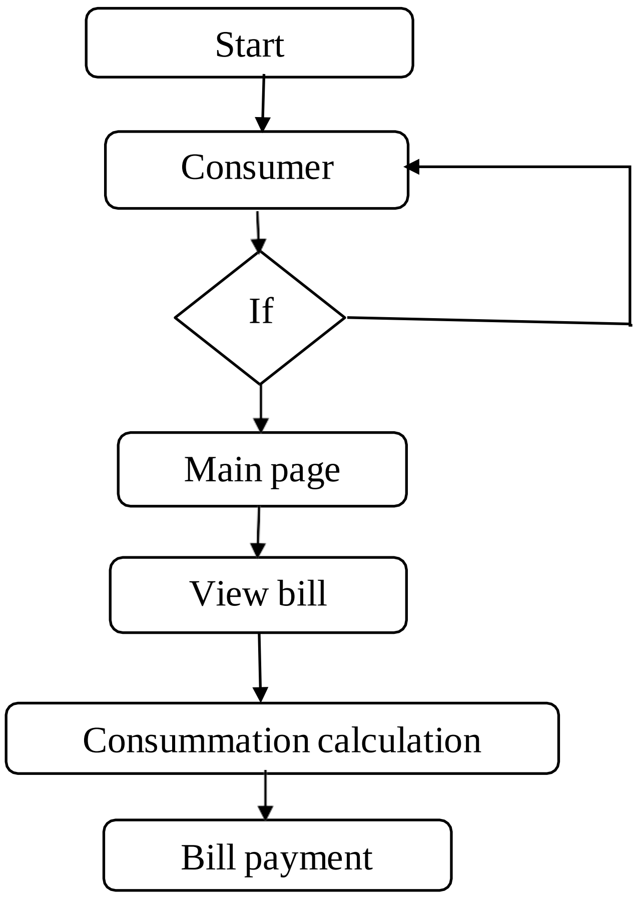
- **Adding customer:**Here admin can add new customer list who started using electricity bill system.
- **Searching deposit details:** Here admin can search according to meter number and month to view deposit details.
- **Viewing Details:** Here admin and user can view customer details and about details.
- **Updating customer:** Here customer can update their details by using meter no of the customer.
- **Delete customer:** Here admin can delete details based on meter number.

Fig 4.1.1: Activity Flow chart of Administrator



.

Fig 4.1.2 Activity Flow chart of consumer



4.2 DFD(Data Flow Diagram)

They are the versatile diagramming tools used for structured system analysis. They are specifically used for process modelling which involves graphically representing the function or process, which captures, manipulate, store, and distribute data between a system and its environment and between components within a system. Basically the tool we use in this for drawing diagram is draw.io.

Fig 4.2.1 zero level DFD

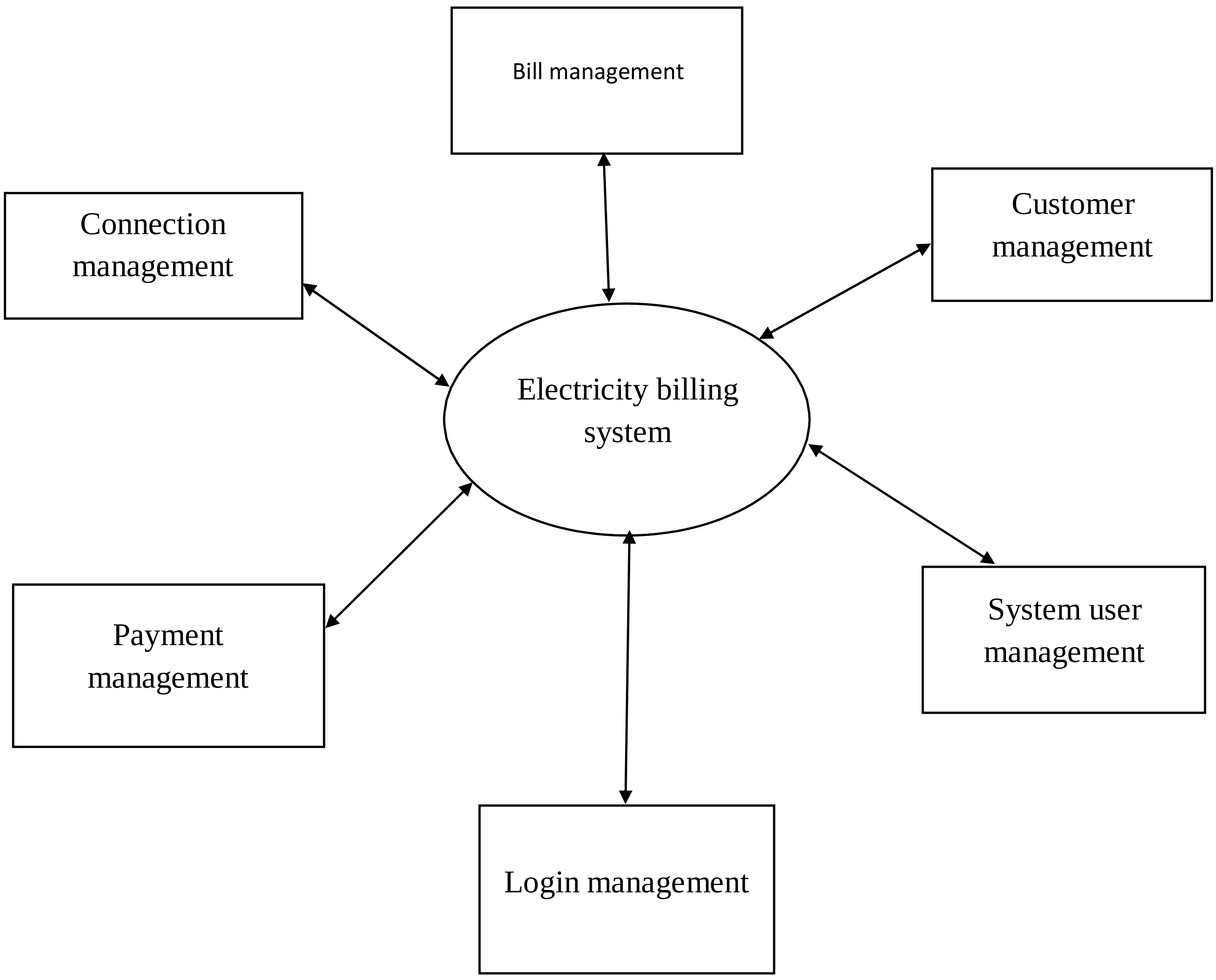


Fig 4.2.2 First level DFD

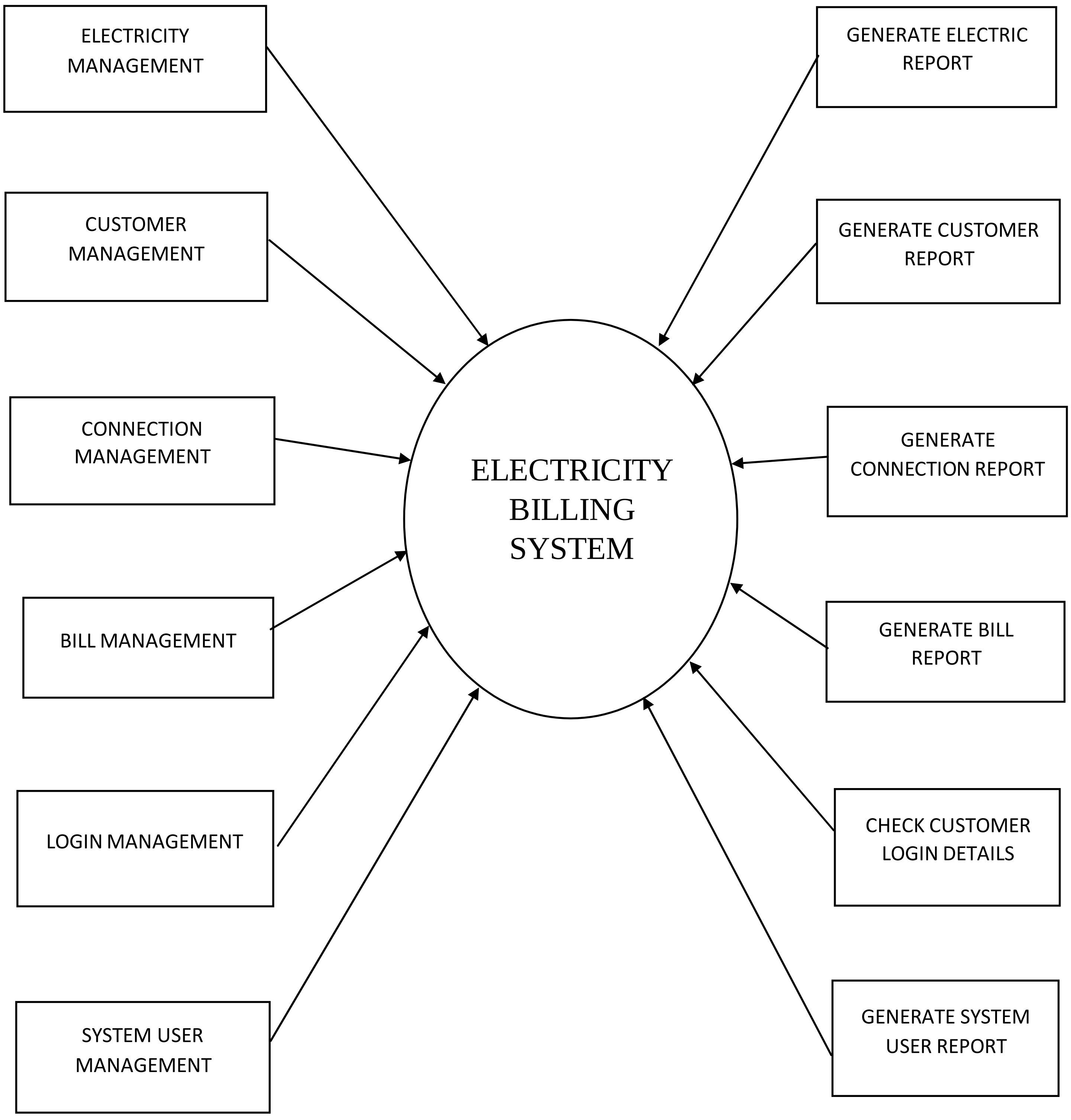
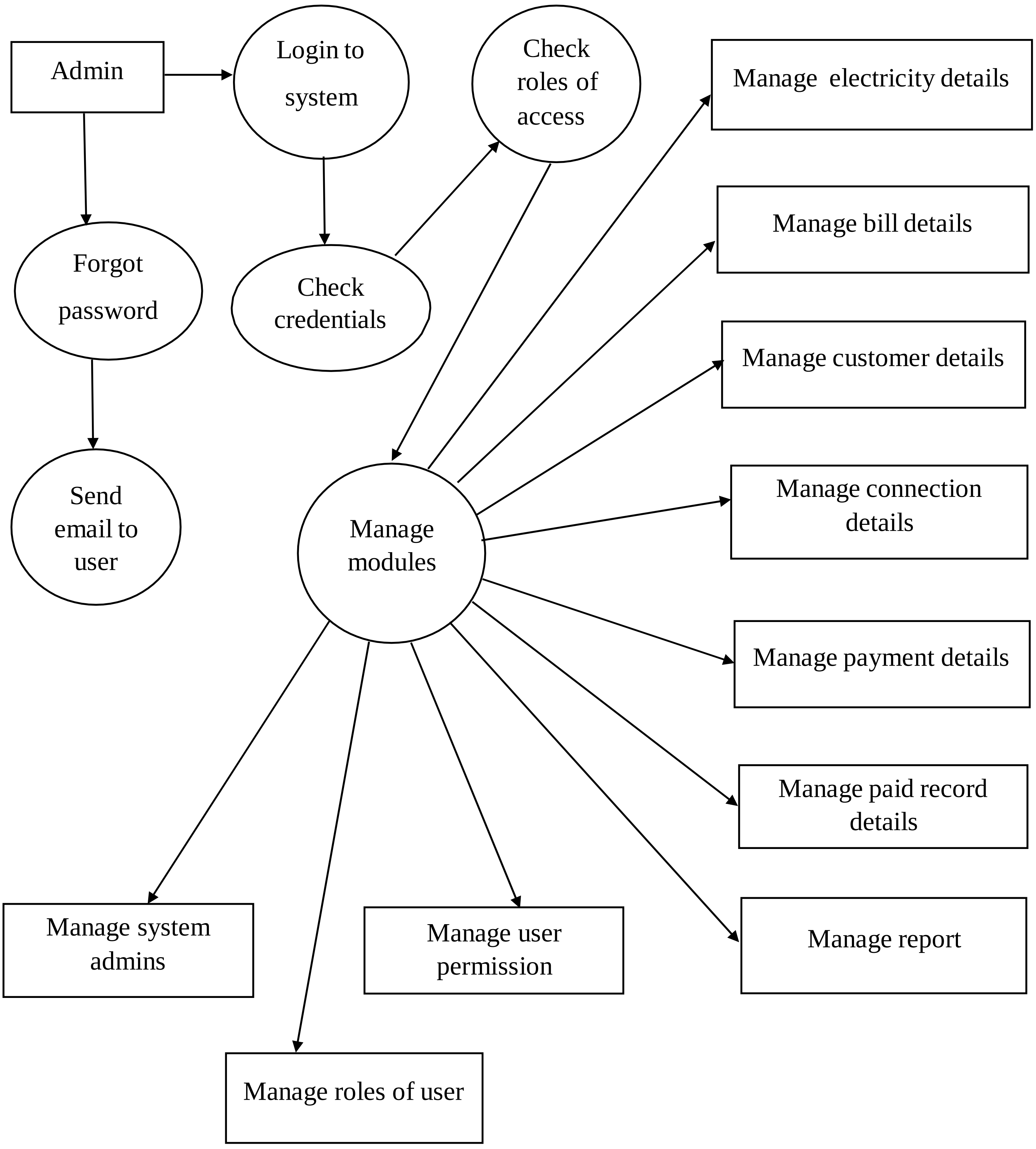


Fig 4.2.3 Second level DFD



CHAPTER 5

TESTING

This chapter gives the outline of all the testing methods that are carried out to get a bug free application.

5.1 Testing process

Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, compiles with the standards, that it was designed to. Testing process involves building of test cases, against which, the product has to be tested. In some cases, test cases are done based on the system requirements specified for the product/software, which is to be developed.

5.2 Testing objectives

The main objectives of testing process are as follows:

• Testing is a process of executing a program with the intent of finding an error.

• A good test case is one that has high probability of finding an as yet undiscovered error.

• A successful test is one that uncovers an as yet undiscovered error.

5.3 Levels of Testing

Different levels of testing are used in the testing process; each level of testing aims to test different aspects of the system. The basic levels are unit testing, integration testing, system testing and acceptance testing.

5.3.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design the module. The software built, is a collection of individual modules. In this kind of testing exact flow of control for each module was verified. With detailed design consideration used as a guide, important control paths are tested to uncover errors within the boundary of the module.

Table 5.1: Negative test case for phone number insertion

Function name	Input	Expected error	Error	Resolved
Input phone number	98977	Phone is invalid	Length of phone number is not equal to 10	Consume()
Input phone number	9877avg	Phone number is invalid	Alphabets are being taken as input for phone number	

Table 5.2: Positive test case for phone number insertion

Function name	Input	Expected error	Error	Resolved
Input phone number	8690345678	Expected output is seen		

Table 5.3: Negative test case for email insertion

Function name	Input	Expected error	Error	Resolved
Input email	Sail.in	Email is invalid	Email is not in a format given	Consumer()

Table 5.4: Positive test case for email insertion

Function name	Input	Expected error	Error	Resolved
Input email	Akil23@gmail.com	Expected output is seen		

Table 5.5: Negative test case for customer name insertion

Function name	**Input**	**Expected output**	**Error**	**Resolved**
Input customer name	Sana123	Name is invalid	Number are being taken as input	Consume()

Table 5.6: Positive test case for customer name insertion

Function name	**Input**	**Expected name**	**Error**	**Resolved**
Input customer name	Pooja	Expected output		

5.3.2 Integration testing

The second level of testing is called integration testing. In this, many class-tested modules are combined into subsystems, which are then tested. The goal here is to see if all the modules can be integrated properly. We have been identified and debugged.

Table 5.7: Test case on basis of generation of bill

Function name	**Input**	**Expected error**	**Error**	**Resolved**
Negative searching of total bill	1234(meter no) January(month)	Details seen but not total bill	Output not seen	Consume ()
Positive searching of total bill	1234(meter no) January(month)	Must display full generated bill		

5.3.3 System testing

Here the entire application is tested. The reference document for this process is the requirement document, and the goal is to see IF the application meets its requirements. Each module and component of ethereal was thoroughly tested to remove bugs througha system testing strategy. Test cases were generated for all possible input sequences and the output was verified for its correctness.

Table 5.8: Test cases for the project

Steps	Action	Expected error
Step 1 Choice	The screen appears when the users run the program. 1. If admin login 2. If customer login	A page with different menus appears. 1. Admin panel opens and 2. Customer panel opens.
Step 2	The screen appears when the admin logs in and selects any one of the menus from the click of the mouse	A window for adding new customer, Calculate bill, etc
Selection 1	• New customer • Customer details • Delete customer	
Step 2.1	The screen appears when the customer login and selects any one of the menus from the click of the mouse	A window for generating bill, update Customer details, view details, generating bill
Selection 2	• Update details • View details	
Selection 2a	• Generate bills	
Selection 2b	• Pay bills • Bill details	

6.CONCLUSION

Usability testing was part of the post implementation review and performance evaluation for the Electricity Online Bill Payment System, in order to ensure that the intended users of the newly developed system can carry out the intended task effectively using real data so as to ascertain the acceptance of the system and operational efficiency. It caters for consumers' bills and also enables the administrator to generate monthly reports. It is possible for the administrator to know the consumers have made payment in respect of their bills for the current month, thereby improving the billing accuracy, reduce the consumption and workload on the Electricity Board employees or designated staff., increase the velocity of electricity distribution, connection, tariff scheduling and eliminates variation in bills based on market demand. The conceptual framework allows necessary adjustments and enhancement maintenance to integrate future demands according to the technological or environmental changes with time. It manages the consumers' data and validates their input with immediate notification centralized in Electricity Board offices across the nation.

6.1 Future Analysis:

We have left all the options open so that if there is any other future requirement in the system by the user for the enhancement of the system then it is possible to implement them. We hope that the project will serve its purpose for which it is develop there by underlining success of process

Appendix

Source code 1:

```
module ATP_Machine_Electricity_Bill_Payment (
   input wire clk,
   input wire reset,
   input wire card_inserted,     // Card insertion detection
   input wire [7:0] card_data,   // Card data (e.g., customer ID)
   input wire [3:0] pin,         // PIN for authorization
```

```
    input wire payment_1000,      // Payment in Rs 1000
    input wire payment_500,       // Payment in Rs 500
    input wire payment_100,       // Payment in Rs 100
    input wire payment_50,        // Payment in Rs 50
    output reg [7:0] display,     // Display output
    output reg payment_success,  // Payment success flag
    output reg payment_fail,     // Payment failure flag
    output reg payment_timeout   // Payment timeout flag
);
    // Internal signal declarations
    reg [3:0] state;
    reg authorized;
    reg payment_completed;
    reg excess_payment;
    reg [12:0] counter;
    reg [7:0] bill_amount;
    reg [7:0] payment_amount;
    reg [7:0] remaining_amount;
    reg [7:0] history_amount;

    // Constants
    localparam [7:0] BILL_AMOUNT = 8'hF4; // Total bill amount in
cents
    localparam [12:0] TIMEOUT_COUNTER = 13'd39062; //
Timeout duration in clock cycles

    // Clock process
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            state <= 4'b0000;
            authorized <= 1'b0;
            payment_completed <= 1'b0;
            excess_payment <= 1'b0;
            counter <= 13'd0;
            bill_amount <= 8'd0;
            payment_amount <= 8'd0;
            remaining_amount <= 8'd0;
            history_amount <= 8'd0;
        end else begin
            case (state)
              4'b0000: begin // Idle state
                if (card_inserted) begin
                  state <= 4'b0010; // Transition to data entry state
                end
              end
              4'b0010: begin // Data entry state
                if (1'b1) begin
                  state <= 4'b0011; // Transition to validation state
                end
              end
              4'b0011: begin // Validation state
                if (1'b1) begin
                  state <= 4'b0100; // Transition to bill process state
                  bill_amount <= BILL_AMOUNT;
                  remaining_amount <= bill_amount;
                end
              end
              4'b0100: begin // Bill process state
                if (1'b1) begin
                  state <= 4'b0101; // Transition to payment by cash
state
                  payment_amount <= 8'd0;
                  end
                end
                4'b0101: begin // Payment by cash state
                  if (payment_1000 || payment_500 || payment_100 ||
payment_50) begin
            payment_amount <= payment_amount +
(payment_1000 * 1000) + (payment_500 * 500) + (payment_100 *
100) + (payment_50 * 50);
            remaining_amount <= remaining_amount -
payment_amount;
                  end
                  if (remaining_amount == 0) begin
            payment_completed <= 1'b1;
                  end
                  if (1'b1) begin
            state <= 4'b0110; // Transition to acknowledge state
                  end
                end
                4'b0110: begin // Acknowledge state
                  if (1'b1) begin
            state <= 4'b0111; // Transition to transaction process
state
                  end
                end
                4'b0111: begin // Transaction process state
                  counter <= counter + 1;
                  if (counter == TIMEOUT_COUNTER) begin
            state <= 4'b1101; // Transition to timeout state
            payment_timeout <= 1'b1; // Set timeout flag
                  end else if (payment_completed) begin
            state <= 4'b1000; // Transition to payment confirm
```

```
state
          end else begin
            state <= 4'b1110; // Transition to fail state
          end
        end
        4'b1000: begin // Payment confirm state
          if (excess_payment) begin
            state <= 4'b1001; // Transition to reduce if excess
state
          end else begin
            state <= 4'b1011; // Transition to receipt state
          end
        end
        4'b1001: begin // Reduce if excess state
          if (1'b1) begin
            state <= 4'b1010; // Transition to reduction process
state
          end
        end
        4'b1010: begin // Reduction process state
          if (1'b1) begin
            state <= 4'b1011; // Transition to receipt state
          end
        end
        4'b1011: begin // Receipt state
          if (1'b1) begin
            state <= 4'b1100; // Transition to history state
          end
        end
        4'b1100: begin // History state
          if (1'b1) begin
```

```
                state <= 4'b0000; // Transition back to idle state
            end
        end
        4'b1101: begin // Timeout state
            if (1'b1) begin
                state <= 4'b0000; // Transition back to idle state
            end
        end
        4'b1110: begin // Fail state
            if (1'b1) begin
                state <= 4'b0000; // Transition back to idle state
            end
        end
        default: state <= 4'b0000; // Default case to handle any
unexpected state
      endcase
    end
  end

  // Display output assignment
  always @* begin
    case (state)
      4'b0010, 4'b0100: display = card_data; // Display card data
during data entry and bill process states
      4'b0101: display = remaining_amount; // Display remaining
amount during payment by cash state
      4'b0110: display = bill_amount; // Display bill amount during
acknowledge state
      4'b1000, 4'b1001, 4'b1010: display = payment_amount; //
Display current payment amount during payment confirm and
reduction process states
```

```
            4'b1011: display = {excess_payment, payment_amount}; // Display excess payment and payment amounts during receipt state
            4'b1100: display = history_amount; // Display history amount during history state
            default: display = 8'd0; // Display 0 in other states
        endcase
    end

    // Payment success and failure flag assignment
    always @* begin
        case (state)
            4'b1000, 4'b1011: begin
                payment_success = 1'b1;
                payment_fail = 1'b0;
            end
            4'b1110: begin
                payment_success = 1'b0;
                payment_fail = 1'b1;
            end
            default: begin
                payment_success = 1'b0;
                payment_fail = 1'b0;
            end
        endcase
    end

endmodule
```

Source code2:

```
module ATP_Machine_Electricity_Bill_Payment_TB;
    // Inputs
    reg clk;
    reg reset;
    reg card_inserted;
    reg [7:0] card_data;
    reg [3:0] pin;
    reg payment_1000;
    reg payment_500;
    reg payment_100;
    reg payment_50;

    // Outputs
    wire [7:0] display;
    wire payment_success;
    wire payment_fail;
    wire payment_timeout;

    // Instantiate the ATP Machine module
    ATP_Machine_Electricity_Bill_Payment ATP_Machine (
        .clk(clk),
        .reset(reset),
        .card_inserted(card_inserted),
        .card_data(card_data),
        .pin(pin),
        .payment_1000(payment_1000),
        .payment_500(payment_500),
        .payment_100(payment_100),
        .payment_50(payment_50),
        .display(display),
        .payment_success(payment_success),
        .payment_fail(payment_fail),
        .payment_timeout(payment_timeout)
    );

    // Clock generation
    always begin
        #5 clk = ~clk;
    end

    // Initialize inputs
    initial begin
        clk = 0;
        reset = 1;
        card_inserted = 0;
        card_data = 0;
        pin = 0;
        payment_1000 = 0;
        payment_500 = 0;
        payment_100 = 0;
        payment_50 = 0;

        #10 reset = 0; // Deassert reset after 10 time units

        // Scenario 1: Successful payment
        #20 card_inserted = 1;
        #5 card_data = 8'hAB;
        #5 card_inserted = 0;
        #10 pin = 4'b1010; // Correct PIN
        #5 payment_1000 = 1; // Rs 1000 payment
```

```
        #5 payment_500 = 1; // Rs 500 payment
        #5 payment_100 = 1; // Rs 100 payment
        #5 payment_50 = 1; // Rs 50 payment
        #50;
        #5; // Add any additional waiting time for observation

        // Scenario 2: Failed payment
        #20 card_inserted = 1;
        #5 card_data = 8'hCD;
        #5 card_inserted = 0;
        #10 pin = 4'b0101; // Incorrect PIN
        #5 payment_1000 = 1; // Rs 1000 payment
        #5 payment_500 = 1; // Rs 500 payment
        #5 payment_100 = 1; // Rs 100 payment
        #5 payment_50 = 1; // Rs 50 payment
        #50;
        #5; // Add any additional waiting time for observation

        // Scenario 3: Timeout
        #20 card_inserted = 1;
        #5 card_data = 8'hEF;
        #5 card_inserted = 0;
        #10 pin = 4'b1011; // Correct PIN
        #50; // Wait for timeout
        #5; // Add any additional waiting time for observation

        // End simulation
        #10 $finish;
    end
endmodule
```